C++ Name Lookup

Bringing name lookup into the modern age

Nathan Sidwell
Facebook
nathans@fb.com

$O(N^2)$,
$O(N^2)$ everywhere

- How we got here
- Overload sets
- Namespaces
- Classes

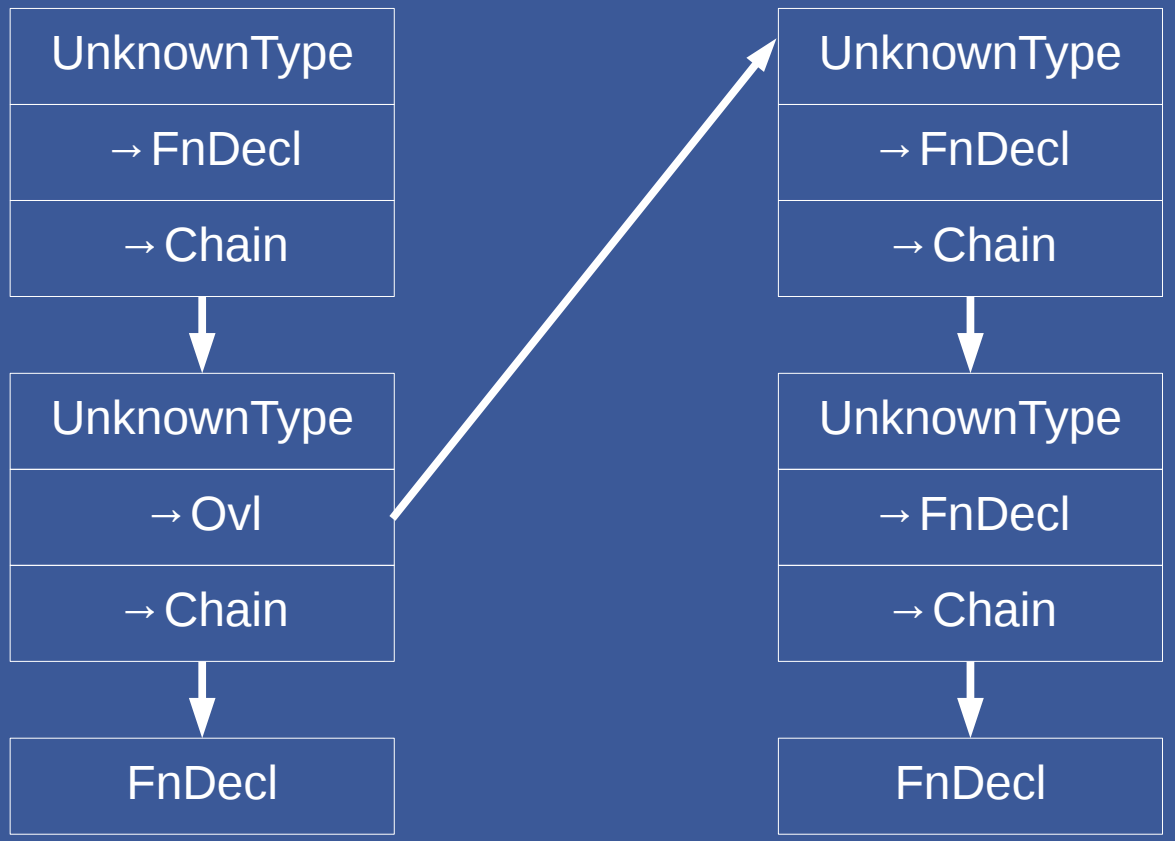
History

- C++ FE cloned from C FE
  - Predates Namespaces
  - Predates ADL (Koenig)
  - Predates STL
  - Predates template meta-programming
  - Predates modules

- name-lookup.c
  - 34 `.*push.*` functions

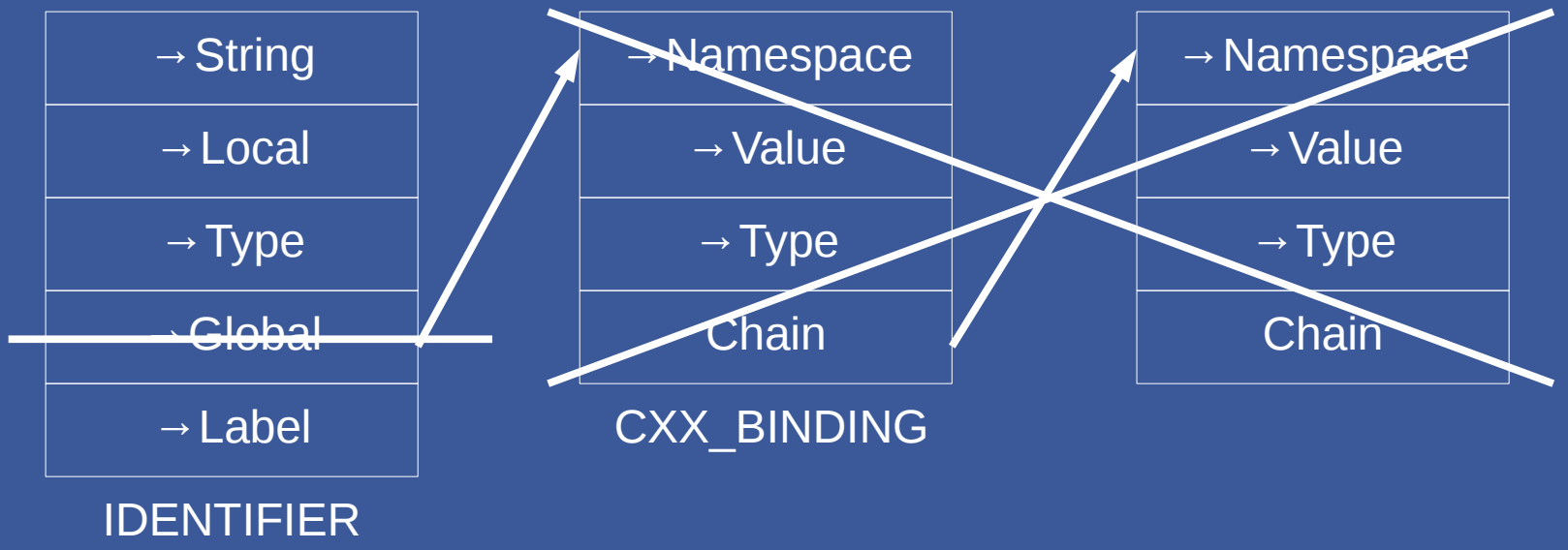
Overload Sets

- ~~tree fns = …;~~
  ~~for (; fns; fns = OVL_NEXT (fns)) {~~
  ~~tree fn = OVL_CURRENT (fns);~~
  ~~…~~
  ~~}~~

- ```
  tree fns = …;
  for (lkp_iterator iter (fns);
       iter; ++iter) {
    tree fn = *iter;
    …
  }
  ```
  ```

Lookups



- 1D: ovl_iterator – a binding value
- 2D: lkp_iterator – a lookup result

Hidden Functions

- Undeclared builtins
- Friend injection
  - ```
    class Foo {
      friend void Frob (Foo *);
    }
    ```
- ~~`tree remove_hidden (tree fns);`~~
- Sort overloads, hidden functions first
  - `tree ovl_insert (tree fn, tree ovl);`
  - `bool ovl_iterator::hidden_p ();`
  - `tree ovl_skip_hidden (tree ovl);`
    ```

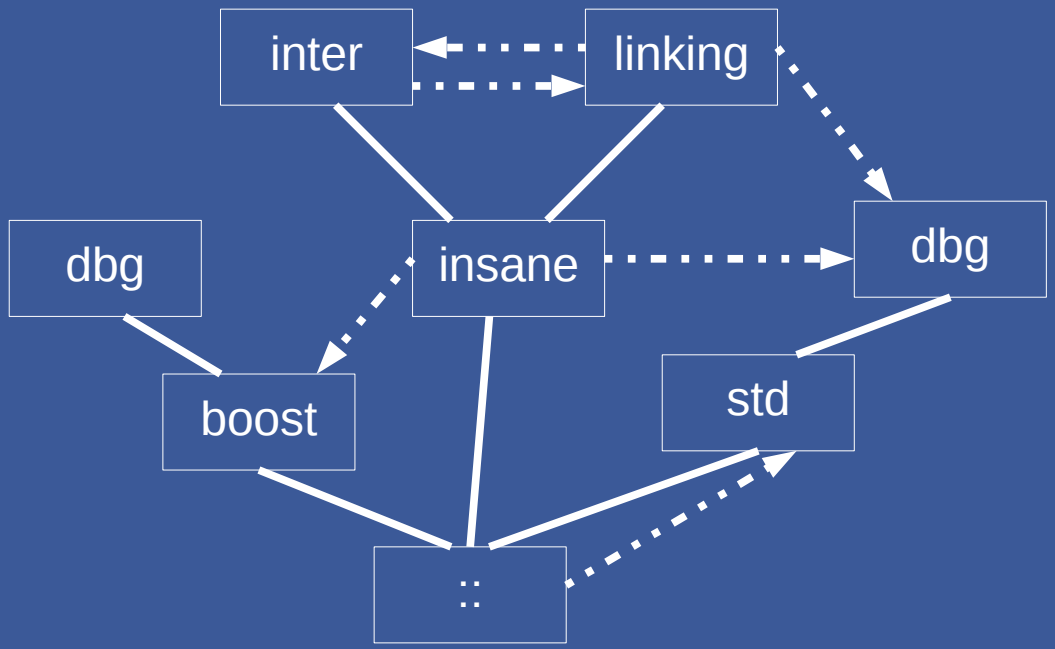
Scopes

Namespace Bindings



- ~~IDENTIFIER_NAMESPACE_BINDINGS(IDENT)~~
- `hash_map<lang_identifier *, tree> *bindings;`
- `NAMESPACE_BINDINGS(NS)`

Namespace Name Lookup



- `using namespace over_there;`

Qualified Namespace Lookup

```
Foo::Bar
```

```
Look in Foo
   Found?  Done.
Else recursively follow using directives
   Merge results
```

- A flood fill algorithm
  - ~~vec<tree, va_gc> *seen;~~
  - LOOKUP_SEEN_P(N)
  - LOOKUP_FOUND_P(N)

Unqualified Namespace Lookup

Recursively add every using directive to queue

Look in current scope

Look in every queued namespace where current scope is ancestor of using src & dst

Found? Merge results, done.

Else goto parent scope and repeat

- ancestor_p (here, src, dst)
- visited_p (here)

Ancestor Namespace

- Was $O(N^2)$ in time
  - Precomputed using directive graph - $O(N^2)$ in space
- Add `SCOPE_DEPTH(N)`
  - `base.u.bits.address_space`
- Again,
  - ~~`vec<tree, va_gc> *seen;`~~
  - `LOOKUP_SEEN_P(N)`
  - `LOOKUP_FOUND_P(N)`

ADL

```
Foo (a, b, c);
```

- Look in namespaces of types of arguments
- Again
  - ~~`vec<tree, va_gc> *seen;`~~
  - `LOOKUP_SEEN_P(N)`
  - `LOOKUP_FOUND_P(N)`
- Can cause recursive lookup

```
TPL<int> *ptr = nullptr;
Foo (ptr);
```

  - Detect and save/restore global state

Class Scope

TYPE_METHODS

- Chain of member functions
- `TYPE_FIELDS` chains non-function members
  - Let’s just put them all there

- Freed up slot available in `RECORD_TYPE`

CLASSTYPE_METHOD_VEC

- Sorted vector of overloaded member functions
  - Special slots for ctors, dtor, conv-ops
- Give ctors & dtor fixed internal name
  - Lookup by ctor_identifier/dtor_identifier
- Put all conv-ops on one slot
  - Prepend dummy decl, lookup by conv_op_identifier
  - Deal with overload properties afterwards

CLASSTYPE_SORTED_FIELDS

- Sorted vector of non-function members
  - Built after struct is complete
- ~~CLASSTYPE_SORTED_FIELDS(KLASS)~~
- `hash_map<lang_identifier *, tree> *bindings;`
- `CLASSTYPE_BINDINGS(KLASS)`
- Populate during definition
- Merge with CLASSTYPE_METHOD_VEC

CLASSTYPE_SORTED_FIELDS

gcc: internal compiler error: Segmentation fault signal terminated program cc1plus
Please submit a full bug report,
with preprocessed source if appropriate.
See <https://gcc.gnu.org/bugs/> for instructions.

Instead ...

- When completing a struct, TYPE_FIELDS ...
  - ~~Create CLASSTYPE_SORTED_FIELDS~~
  - Add them to CLASSTYPE_METHOD_VEC
  - Dedup as needed

- Declare victory!


facebook
INFRASTRUCTURE