GNU Tools Cauldron 2012

Compile/Assemble/Link/Debug/Repeat

July 9 to 11, 2012, Prague, Czech Republic

Jan Hubička (ed.)
GNU Tools Cauldron 2012 is organized jointly by

- IÚUK (Computer Science Institute, Charles University),
- CE-ITI (Institute for Theoretical Computer Science),
- Google.

Sponsors: Google, Red Hat, SUSE, AdaCore, IBM.
Organizing committee: Jan Hubička, Diego Novillo, Ian Lance Taylor.
Preface

The purpose of this workshop is to gather all GNU tools developers in order to discuss current and future work and coordinate efforts. There will be presentations on ongoing projects and developer tutorials.
Schedule
The schedule is subject to change. Up to date information will be on display in the front of room S9 and on the GNU Cauldron web page http://gcc.gnu.org/wiki/cauldron2012.

We have two rooms in the first floor available to the event (S9 and S10). While the main track schedule is full, everyone is welcome to announce event in room S10 anytime during the gathering.

Sunday, July 8

We are pleased to invite all attendees to the opening reception, sponsored by Red Hat. The reception will take place in the Refectory (1st floor) 19:00–22:00.

Monday, July 9

<table>
<thead>
<tr>
<th>Time</th>
<th>BoF/Talk/Event</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:30</td>
<td>Breakfast (morning coffee)</td>
<td>Corridor</td>
</tr>
<tr>
<td>10:00</td>
<td>Keynote — Free software, GNU and GCC — Richard Stallman</td>
<td>S9</td>
</tr>
<tr>
<td>11:00</td>
<td>Free Software: A viable model for Commercial Success</td>
<td>S9</td>
</tr>
<tr>
<td>12:00</td>
<td>Lunch (provided)</td>
<td>Corridor</td>
</tr>
<tr>
<td>13:00</td>
<td>Status of High-Level Loop Optimizations in GCC</td>
<td>S9</td>
</tr>
<tr>
<td>14:00</td>
<td>Fission</td>
<td>S9</td>
</tr>
<tr>
<td>15:00</td>
<td>GNU Binutils Porting Guide For A New Processor Arch.</td>
<td>S9</td>
</tr>
<tr>
<td>15:45</td>
<td>Straight-line strength reduction in GCC</td>
<td>S10</td>
</tr>
<tr>
<td>16:15</td>
<td>Coffee Break</td>
<td>Corridor</td>
</tr>
<tr>
<td>16:15</td>
<td>An implementation of predicated value numbering</td>
<td>S9</td>
</tr>
<tr>
<td>17:15</td>
<td>The Quest for Cheaper Variable Tracking in GCC</td>
<td>S9</td>
</tr>
<tr>
<td>18:15</td>
<td>GDB vs. MPI (Message Passing Interface)</td>
<td>S9</td>
</tr>
<tr>
<td></td>
<td>Identifying compiler options to minimize energy consumption</td>
<td>S10</td>
</tr>
</tbody>
</table>
## Tuesday, July 9

<table>
<thead>
<tr>
<th>Time</th>
<th>BoF/Talk/Event</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td><strong>Breakfast (morning coffee)</strong></td>
<td>Corridor</td>
</tr>
<tr>
<td>9:30</td>
<td>11:15 GDB BoFs</td>
<td>S9</td>
</tr>
<tr>
<td>11:30</td>
<td>Finding races and memory errors with GCC instrumentation</td>
<td>S9</td>
</tr>
<tr>
<td></td>
<td>The Benefit of GCC’s open structure on instrumentation in the HPC area</td>
<td>S10</td>
</tr>
<tr>
<td>12:15</td>
<td><strong>Lunch (provided)</strong></td>
<td>Corridor</td>
</tr>
<tr>
<td>13:30</td>
<td>Improving Function Pointer Security</td>
<td>S9</td>
</tr>
<tr>
<td></td>
<td>for Virtual Method Dispatches</td>
<td></td>
</tr>
<tr>
<td></td>
<td>StarPU’s C Extensions for Hybrid CPU/GPU Task Programming, or, An Experience</td>
<td>S10</td>
</tr>
<tr>
<td></td>
<td>in Turning a Clumsy API Into Language Extensions</td>
<td></td>
</tr>
<tr>
<td>14:30</td>
<td>Supporting Parallel Component Debugging</td>
<td>S9</td>
</tr>
<tr>
<td></td>
<td>Using the GDB Python Interface</td>
<td></td>
</tr>
<tr>
<td></td>
<td>C++ Conversion BoF</td>
<td>S10</td>
</tr>
<tr>
<td>15:15</td>
<td><strong>Coffee Break</strong></td>
<td>Corridor</td>
</tr>
<tr>
<td>15:45</td>
<td>Reducing DWARF debuginfo size</td>
<td>S9</td>
</tr>
<tr>
<td>16:45</td>
<td>What’s New in C++11</td>
<td>S9</td>
</tr>
<tr>
<td>17:45</td>
<td>The Cilk Plus Implementation on GCC</td>
<td>S9</td>
</tr>
<tr>
<td></td>
<td>Link-time optimization BoF</td>
<td>S10</td>
</tr>
<tr>
<td>20:00</td>
<td><strong>Social dinner</strong></td>
<td>Refectory</td>
</tr>
</tbody>
</table>

## Wednesday, July 10

<table>
<thead>
<tr>
<th>Time</th>
<th>BoF/Talk/Event</th>
<th>Room</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:00</td>
<td><strong>Breakfast (morning coffee)</strong></td>
<td>Corridor</td>
</tr>
<tr>
<td>9:30</td>
<td>10:15 Control-flow preservation in GCC for safety-critical uses</td>
<td>S9</td>
</tr>
<tr>
<td>10:00</td>
<td>11:15 Compiler Optimizations for Dynamic Scripting Language Interpreters and</td>
<td>S9</td>
</tr>
<tr>
<td></td>
<td>JITs</td>
<td></td>
</tr>
<tr>
<td>11:30</td>
<td>The Local Register Allocator Project</td>
<td>S9</td>
</tr>
<tr>
<td>12:15</td>
<td><strong>Lunch (provided)</strong></td>
<td>Corridor</td>
</tr>
<tr>
<td>13:30</td>
<td>GCC Doc Futures</td>
<td>S9</td>
</tr>
<tr>
<td></td>
<td>BoF: A native 32-bit psABI for x86-64</td>
<td>S10</td>
</tr>
<tr>
<td>14:30</td>
<td>15:15 Pre-Parsed Headers</td>
<td>S9</td>
</tr>
<tr>
<td>15:15</td>
<td><strong>Coffee Break</strong></td>
<td>Corridor</td>
</tr>
<tr>
<td>15:45</td>
<td>G++ diagnostics: present and (near) future</td>
<td>S9</td>
</tr>
<tr>
<td>16:45</td>
<td>17:30 New programming abstractions for concurrency</td>
<td>S9</td>
</tr>
</tbody>
</table>
Abstracts
Monday, July 9

Keynote Presentation — Free software, GNU and GCC

Presenter: Richard Stallman
Room S9, 10:00–10:45

The free software movement’s goals, and how GNU and GCC are part of achieving them.

Free Software: A viable model for Commercial Success

Presenter: Robert Dewar
Room S9, 11:00–11:45

This talk will discuss our experience at AdaCore, one of only a handful of 100% Free Software companies. All of our commercial products are licensed under the GPL and other Free Software Licenses. People often assume that there is a conflict between the use of such licenses and the needs of a commercial software company. Our experience at AdaCore shows that on the contrary, the Free Software model can be very successful both for us as a company and for our customers. We think this model can be used in many other circumstances, and want to encourage free software enthusiasts to consider this model in other circumstances.

Status of High-Level Loop Optimizations in GCC

Presenter: Richard Guenther
Room S9, 13:00–13:45

We will present the state of high-level loop optimizations in GCC and discuss a viable path forward. We will put forward several options as a basis for discussion.

Fission

Presenter: Cary Coutant
Room S9, 14:00-14:45

Fission is about improving debugger usability and link-time performance. We’ve designed DWARF extensions that allow us to split the bulk of the debug information from the object files, allowing us to substantially reduce total link time and the size of the linked binary. In addition, because the input files to the linker are significantly smaller, bandwidth needed
for a distributed build system is also reduced. The final executable will contain an index of the debug information, allowing the debugger to locate the debug information on demand, so that debugger start-up time can also be reduced. A full description of the project is on the GCC wiki: http://gcc.gnu.org/wiki/DebugFission.

GNU Binutils Porting Guide For A New Processor Architecture

Presenter: M R Swami Reddy
Room S9, 15:00–15:45

The paper intends to help software engineers who want to port the binutils tools to a new hardware architecture for the first time. Although the GNU binutils project includes a 100-page document to its internals, this paper is aimed primarily at those wishing to develop/port GNU binutils itself for the first time. This paper discuss the porting process step by step along with a example, where ever possible. "We also discuss the organization of the GNU binutils source code, and include a sample build and test process.

Straight-line strength reduction in GCC

Presenter: Bill Schmidt
Room S10, 15:00–15:45

GCC has long lacked a strength reduction capability outside of loops. Previous attempts to address this within existing frameworks, such as partial redundancy elimination, have not been successful. A primary reason for this is that these frameworks process individual expressions independently. For strength reduction, a determination of profitability often requires examining chains of related strength reduction candidates. This short presentation will demonstrate the issues involved and outline a new SSA dominator-based proposal for efficiently performing non-loop strength reduction.

An implementation of predicated value numbering

Presenter: Michael Matz
Room S9, 16:15–17:00

A presentation of ongoing work to replace the current value numberer with one capable of dealing with predicates (i.e., value numbers depending on conditions).
The Quest for Cheaper Variable Tracking in GCC

Presenter: Alexandre Oliva
Room S9, 17:15–18:00

GCC’s variable tracking pass got visibly more expensive with the introduction of VTA, Variable Tracking at Assignments. The pass scans each basic block for insns relevant for variable location debug information generation, propagates locations and values across basic blocks with global dataflow analysis, and finally generates notes with location or value expressions for variables.

The last part has been recently improved from an algorithm whose worst case was exponential to one that is linear on the variable/value equivalence graph size. The other parts have gained some memory savings by keeping global equivalences in a global data structure rather than in per-block equivalence sets, but there’s need and room for performance improvements, particularly in the confluence operation in dataflow analysis.

The goal of this session is to present the current inner workings of the variable tracking pass, including the recent changes and existing plans, then opening for discussion, requests and suggestions of further improvements.

GDB vs. MPI (Message Passing Interface)

Presenters: Joachim Protze and Andreas Knüpfer
Room S9, 18:15–19:00

The MPI (Message Passing Interface) standard is the one established method to achieve highest scale parallelism on today’s biggest supercomputers. There are many implementations including free ones. Yet the standard makes life for debuggers pretty difficult.

The MPI API hides away all sorts of management information in handles to give maximum flexibility to implementors. Unfortunately, this includes data type information of all messages. Therefore, debuggers are pretty much unable to show the contents of messages that are exchanged between parallel processes.

We implemented a solution for GDB using two stages: one to collect data type information from the MPI API and a GDB plugin to print a message’s contents in a correct and convenient way. With this, GDB and MPI work together like they should have in the first place . . . in our opinion.
Identifying compiler options to minimize energy consumption by embedded programs

Presenter: Jeremy Bennett
Room S10, 18:15–19:00

This summer Embecosm will be running a joint project with Bristol University Department of Computer Science to look at the impact of compiler options on energy consumption by programs on embedded processors. Many people have opinions on this, but there is very little hard data. Bristol University’s equipment can measure the power consumed by a processor in great detail and to fine time resolution. We will test a representative range of programs (suggestions will be solicited from the audience) with a wide range of compiler options. We will use a number of different processors (XMOS, ARM) as well as different processors in the same family (ARM). We will also compare GCC to LLVM. The results will be published in an open access journal to provide a baseline data set for future research. One channel we wish to pursue subsequently is use of MILEPOST technology to automatically select the best low energy options when compiling programs. The project, starting on 9 July, will be led by Jeremy Bennett (Embecosm) and Simon Hollis (Bristol University), with the work carried out by James Pallister of Embecosm, who will then return to Bristol University for a 3-year PhD in this field. The purpose of this talk is to solicit views from the wider GCC community at the start of this project, particularly with regard to the features of GCC that are most likely to yield benefits and should thus be explored. We look forward to presenting the results at next year’s meeting.
Tuesday, July 10

GDB BoFS: GDB Roadmap, Towards Multicore GDB, Towards feature parity of GDB remote and native debugging

Room S9, 9:30–11:15

GDB Roadmap

Presenter: Jan Kratochvíl

Status of missing features, status of features being worked on (known to me). Which way to keep unused template methods code separate from the code output. Dynamic types (such as variable length arrays) implementation choices in GDB.

Towards Multicore GDB

Presenters: Stan Shebs

Multicore systems have been around for a while, but the next generation takes it to a whole new level, with high-performance embedded designs consisting of anywhere from 30 to 1,000 cores. GDB needs significant work to be useful in debugging these targets, both in user interface and to improve performance.

The first part of the task is to expand GDB’s vocabulary by formalizing the notion of core as its own first-class object, conceptually similar to a thread but persistent, and by introducing the “process/thread/core set”, by which the user works with groups of threads, cores, etc, rather than just one at a time.

The second part is to partition the debugging workload so that GDB is less of a bottleneck. For instance, we introduce the notion of an agent library that can run on each core and handles some tasks locally, such as testing of a breakpoint condition, only notifying GDB when the condition is true.

This presentation will review the current status of multicore work, and look ahead to additional ideas to facilitate debugging of future multicore systems.

Towards feature parity of GDB remote and native debugging

Presenter: Ulrich Weigand
GDB supports debugging applications running natively on the host systems as well as debugging applications running on a remote system. In the latter case, GDB talks to a remote stub on the target side. If the target system is running Linux, this remote stub is usually gdbserver, which comes as part of the GDB distribution itself. In principle, there should be no difference between debugging a Linux process natively and debugging it remotely via gdbserver. However, for historical reasons, the GDB native Linux target is implemented as a completely separate code base from gdbserver. This has over time led to the unfortunate situation that certain GDB features are in fact only available when debugging natively (and other features are available only when debugging remotely). I’m planning to present an overview of the current state of affairs, including a couple of improvements that were implemented recently. I’m also planning to discuss proposals how to move forward, ideally towards a goal of achieving feature parity between native and remote/gdbserver debugging by actually sharing a single code base.

Finding races and memory errors with GCC instrumentation (Address Sanitizer)

Presenters: Konstantin Serebryany and Dmitriy Vyukov
Room S9, 11:30–12:15

We will present two dynamic testing tools based on compile-time instrumentation. **Address-Sanitizer** (ASan) finds memory bugs, such as use-after-free and out-of-bound accesses to heap, stack and globals. This tool could be seen as a partial replacement for Valgrind and similar tools. The major advantages over Valgrind are the speed (less than 2× slowdown on average) and the ability to handle bugs related to stack and globals. **AddressSanitizer** can also fully replace Mudflap.

**ThreadSanitizer** (TSan) finds data races. It uses the same race detection algorithm as the Valgrind-based TSan, but compile-time instrumentation allows it to be much faster (2×–4× slowdown).

Both tools are implemented using GCC and LLVM infrastructures, so we will provide a comparison between GCC and LLVM from our perspective. We will also share our experience in deploying theses testing tools in large software projects.

More info:

- [http://code.google.com/p/address-sanitizer/](http://code.google.com/p/address-sanitizer/)
The Benefit of GCC’s open structure on instrumentation in the HPC area

Presenters: Johannes Ziegenbalg and Bert Wesarg (Technische Universität Dresden)
Room S10, 11:30–12:15

Function instrumentation is one foundational method of performance data gathering. This data is stored on disc in event trace files to run a performance analysis later on. Unfortunately, automatic instrumentation often results in lots of trace events being generated during the measurement run, especially in high-performance computing applications. This may alter the program behavior due to a large runtime-overhead. Additionally, the trace file becomes too large to be analyzed efficiently. Therefore, instrumentation filtering is inevitable. Though GCC is one of the few compilers which support function instrumentation filtering at compile time without altering the source code, it’s filtering is difficult, if not impossible, to control.

In our presentation, we talk about our achievements using the instrumentation framework InterAspect to generate a GCC plugin which provides better control over the instrumentation.

We also present our plans to reduce the induced overhead by improving the generated code from the function instrumentation.

Improving Function Pointer Security for Virtual Method Dispatches

Presenter: Caroline Tice
Room S9, 13:30–14:15

A common vector of attack in C++ programs is for attackers to make use of use-after-free bugs in the program to overwrite vtable pointers and hijack program execution. If the attacker discovers a use-after-free bug in the program, he waits until the object has been freed, then re-allocates the same memory to be an object of the same size and overwrites the vtable pointer in that object. When the object is accessed by the program (the use-after-free bug), it uses the attacker’s vtable pointer to then go and start executing the attacker’s code.

We are working on an approach to detect when attackers have overwritten vtable pointers, without significant performance penalties, and without changing the C++ ABI.
StarPU’s C Extensions for Hybrid CPU/GPU Task Programming, or, An Experience in Turning a Clumsy API Into Language Extensions

Presenter: Ludovic Courtès
Room S10, 13:30–14:15

StarPU started as a run-time support library for hybrid CPU/GPU task programming, later supplemented by a GCC plug-in. The GCC plug-in allows programmers to annotate C code to describe tasks and their implementations. Each task may have one or more implementations, such as CPU implementations or implementations written in OpenCL.

StarPU’s support library schedules tasks over the available CPU cores and GPUs, and is also responsible for scheduling any data transfers between main memory and GPUs.

This talk will present the rationale for StarPU’s C extensions and describe them. We will then report on our experience turning a C API into convenient language extensions, and discuss this use case for GCC plug-ins.

Supporting Parallel Component Debugging Using the GDB Python Interface

Presenters: Kevin Pouget, Miguel Santana, Jean-François Mehaut and Vania Marangozova-Martin
Room S9, 14:30–15:15

In this presentation, we will introduce the work we have undertaken in a join R&D effort of STMicroelectronics and the Laboratoire d’Informatique de Grenoble on the GDB project.

In the context of parallel and embedded computing, debugging is well-recognized as a complex activity. Nowadays, such applications are not developed anymore from scratch, relying only on the programming language primitives. Instead, they lean upon more advanced programming models allowing an easier expression of parallelism.

Interactive debuggers like GDB evolved from their earlier times when they could only handle machine instructions to support the source languages used by developers to write their applications. We believe that their next evolution could be the support of programming models, which would help the developers to manipulate higher level abstractions like the entities or communication mechanisms defined by the programming model. These abstractions will have the advantage of being closer to the concepts the developer dealt with during development time and they will help her to keep focused on application execution behavior.
Hence, our work consists in improving GDB towards the support of such programming models. On top of GDB’s Python interface, and extending it with contributed patches whenever it was required, we prepared a framework supporting the debugging of an ST home-made embedded component framework for MPSoC systems, running on an x86 simulator. The presentation will detail how we leveraged GDB to gather relevant runtime information about the component framework and the set of new features we developed, along with use-cases about their usage.

**C++ Conversion BoF**

Presenter: Diego Novillo and Lawrence Crowl  
Room S10, 14:30–15:15

**Reducing DWARF debuginfo size**

Presenters: Cary Coutant and Mark Wielaard  
Room S9, 15:45–16:30

Generating, linking, reading and storing DWARF debuginfo take significant resources, time and space. We want to discuss some efforts that have recently been done to reduce some of that in the compiler, linker, package manager and tools, like debuggers, that use the DWARF debug information. We are interested in discussing efforts that worked, the various tradeoffs, efforts that didn’t produce significant results and ideas for future DWARF reduction work and/or standardization.

**What’s New in C++11**

Presenter: Jason Merrill  
Room S9, 16:45–17:30

The second revision of the C++ language standard, C++ 2011, was ratified last year, thirteen years after the first one. In this talk, I will discuss notable additions to the language since C++98/03.

**The Cilk Plus Implementation on GCC**

Presenters: Balaji V. Iyer, Robert Geva and Pablo Halpern  
Room S9, 17:45–18:30
In the current era of multicore processors, it is necessary for programmers to write efficient code to exploit their full capabilities. In this presentation, we address the Intel(r) Cilk(tm) Plus language extension that is implemented in a GCC branch. Cilk Plus is a set of language constructs for C/C++ for data and task parallelism.

The first construct defines three keywords (Cilk_spawn, Cilk_sync and Cilk_for) that can be used on an existing serial program to make it task parallel. The keywords are simple to use, make the program easy to read and provide strong guarantees of serial equivalence. However, they require the help of a runtime whose source is also included in the compiler branch.

The other components in Cilk Plus provide data-parallelism constructs. Array notations aid the compiler to schedule batches of iterations to execute in parallel. If the processor has vectorization support, this construct can assist the compiler to vectorize the code. In addition, there are built-in functions that provide intrinsic operations such as finding maximum/minimum, sum and product of all the array elements. Second, elemental functions provide an option to take a scalar function in standard C and C++ and deploy it on many elements of arrays without prescribing an order of operations among the array elements. This allows the compiler to generate a vector version of the function, which vectorizes across a batch of consecutive calls to the elemental function. Finally, we provide a pragma and a set of clauses called pragma SIMD that allow users to communicate intent for vector execution and certain pertinent information to ease the job of the compiler in generating vector code.

In the first part of the presentation, we explain all the components of Cilk Plus. After this, we walk-through the compiler modifications in GCC and present some of the performance that can be achieved on some common benchmarks that were converted to Cilk Plus. We end the presentation with some future work and optimization opportunities in the compiler.

**Link-time optimization BoF**

Presenter: Jan Hubička  
Room S10, 17:45–18:30

Link-time optimization is getting more mature. It is now possible to build and optimize large applications (such as Mozilla) or Linux kernel. We will quickly review progress over last year, such as introduction of symbol table, stability and performance improvements, debug info fixes. We will discuss more in detail remaining problems that needs to be addressed soon — debug info, scalability, reliability, program partition issues, how to implement more fancy whole program optimizations etc.
The presentation is about the introduction of a `-fpreserve-control-flow` option in GCC, which directs the compiler operations so that the control-flow expressed in a source persists in the generated assembly code.

The interest is twofold:

1. Facilitate source to object traceability analysis, sometimes required as part of safety critical certification processes.

2. Allow non-intrusive coverage analysis using an instrumented execution environment instead of program instrumentation, also attractive in certification contexts. Beyond traceability analysis requirements, control flow preservation is key to support the non-intrusive analysis of coverage criteria that care about source boolean expressions and their operands, such as the “Decision” or “MCDC” coverage metrics in the avionics certification area (DO178-B standard).

The basic idea is to allow inferring which values were taken by boolean operands or expressions from information on the execution flow at the corresponding machine branch points (provided by the instrumented execution environment). Very roughly, we need the relevant branches to remain there and accurate enough debug info to map them to source expressions in presence of arbitrarily complex constructs, which poses a few challenges to solve in the compiler.

At this point, we have a stable implementation in our local GCC 4.5 series, supporting optimizations up to `-O1`. We use this to offer a non-intrusive coverage analysis framework, using valgrind or qemu as virtual execution environments instrumented to produce execution traces.

We are about to port this to GCC 4.7 and would be happy to contribute to mainline after exchanging with other developers on the approach.

The presentation will include an introduction to the

- major motivations for this work
- general implementation scheme
- challenges encountered and resolutions
- current status & results future plans
Compiler Optimizations for Dynamic Scripting Language Interpreters and JITs

Presenter: David Edelsohn
Room S9, 10:30–11:15

Modern Dynamic Scripting Languages such as Python, Ruby and PHP traditionally have been implemented as interpreters written in C. With their increasing usage in web frameworks and cloud computing infrastructure, they frequently are deployed on GNU/Linux systems, which means they are compiled with GCC. This presentation will examine compiler optimizations that can improve the performance of these types of languages. As performance demands of these languages increase, some implementations are turning to JITs and this talk will explore some compiler features that assist JITs for these languages.

The Local Register Allocator Project

Presenter: Vladimir Makarov
Room S9, 11:30–12:15

The Local Register Allocator (LRA) project is focused on replacing famous GCC reload pass. The project history, motivation and goals, and the different approaches considered to LRA implementation are discussed. LRA overview and structure, tasks solved by LRA, the current state of the project including SPEC2000 benchmark results on some major platforms are given.

The future of LRA project and possible new RA optimizations which utilize new CPU features could be implemented on the base LRA are discussed.

GCC Doc Futures

Presenter: Benjamin Kosnik
Room S9, 13:30-14:15

A complete survey of gcc documentation: what exists, in what formats and why, outstanding legal issues in the GPL vs. GFDL war, where documentation is located in the source, install, and website, how documentation is packaged for releases, how patches are tracked for release notes, how porting information and other derived/contributed information can be made part of canonical GCC documentation sources. A method for integrating wiki content into canonical manuals will be proposed. Known issues with the current documentation will be enumerated, and the audience will be queried and otherwise inspired/cajoled into contributing a more complete list of known issues. Priorities will be assigned to this
derived list of known issues, and volunteers solicited to implement solutions in time for the next major GCC release.

A native 32-bit psABI for x86-64

Presenter: H.J. Lu
Room S10, 13:30-14:15

This talk presents the current status of x32 psABI, which brings x86-64 features to 32-bit applications while keeping memory footprint to 32 bits. It will discuss the performance of the new ABI and the challenges it faces.

Pre-Parsed Headers

Presenters: Lawrence Crowl and Diego Novillo
Room S9, 14:30–15:15

In this talk we will discuss the status of the pre-parsed headers (PPH) project. In particular, we will describe implementation challenges, the current state of the PPH branch, lessons learned during implementation and future plans.

G++ diagnostics: present and (near) future

Presenter: Paolo Carlini
Room S9, 15:45–16:30

Between the 4.6 and the 4.7 releases series a lot of work went into the C++ front-end (and the preprocessor) to improve the diagnostics and add new warnings, even without mentioning hundreds of fixes for many new and long standing bugs: eg, `-ftrack-macro-expansion`, PR c++/48934, `-Wdelete-non-virtual-dtor`, `-Wzero-as-null-pointer-constant`. Most definitely, 4.8 will get some form of “caret diagnostics” and more work is ongoing. Still, from many points of view, Clang++ still has an edge, for example ranges, “typedef unwrapping”, spell checker, etc. Which kinds of improvements we would like to see in GCC as soon as possible? Which ones are doable with a moderate effort and which require extended infrastructural work? Which diagnostics we would like to handle differently than Clang?
New programming abstractions for concurrency

Presenter: Torvald Riegel
Room S9, 16:45–17:30

Parallelization is becoming more important than in the past, and for more developers. Parallel code often results in a concurrent execution of parts of the program (i.e., when threads do not execute truly in parallel but have to coordinate or synchronize with each other). Because concurrent code is typically more complex than sequential code, we need to provide programming abstractions that make these tasks easier for programmers.

In this talk, I will first give a brief overview of concurrency and the associated programming challenges, and then describe two programming abstractions that have been recently added to GCC: the C++11/C11 atomics and Transactional Memory. Both are based on the C++11/C11 memory model, which I will also introduce.
Practical info
Wi-Fi

SSID: MS-KONFERENS
Password: Prazdniny2012

Malá Strana (Lesser Town)

Workshop:

1 Workshop site
Malostranské náměstí (Lesser Town Square) 25

All events of GNU Cauldron will be held in this building, at 1st floor.

T1 Tram stop Malostranské náměstí (Lesser Town Square).

Trams 12, 20, 22 and 57 (night tram). One stop from Metro station Malostranská (Metro A).

Accommodation:

A1 Hotel U tří pštrosů
Dražického náměstí 12, Prague 1

A2 Hotel Waldstein
Valdštejnské náměstí 6

A3 Hotel Kampa Garden
U Sovových mlýnů 9

Restaurants and pubs:

R1 Malostranská Beseda

Conveniently located in the front of workshop site with an option to sit outside. It features good Plzeň beer and Czech food. It is split into a restaurant and cafeteria. About 350CZK for dinner.

R2 U Glaubiců
Malostranské náměstí 266/5

Cheap and good Plzeň beer, overpriced food.

R3 Lokál Inn
Míšeňská 66/12
Authentic Czech cuisine in the centre of Prague. Good Plzeň beer, Czech only menu, some food can be scary for foreigners. Not recommended to vegetarians. About 350CZK for dinner.

R4 Konírna
Maltézské náměstí 292/10


R5 Bangkok
Josefská 1.

Thai place, offers cheap lunch menus.

R6 U Hrocha
Thunovská 10

Traditional beer place. Good beer served in 13th century space. No non-smoking area.

R7 U Mecenáše
Malostranské náměstí 261/10,

Good place for a fancy dinner and wine. Try Apollo desert designed for Neil Armstrong.

R8 U Ferdinanda
Karmelitská 18

Czech pub, lunch menus, Ferdinand beer.
Workshop and transport:

1 Workshop site
Malostranské náměstí (Lesser Town Square) 25

The building of IT section of the Mathematical and Physics Faculty of the Charles University is located in a former jesuit College. The College was founded in 1673 and built in Early Baroque style by G. D. Orsiny. The building was finished in 1691 and it was used the highest ranking representatives of the Order. During its history it was also national bank (1925–27, the tresors are still preserved), German Army headquarters in WW2, or archive (located in the Refectory). Remarkable is also the church of St. Nicholas, next to the building. It was founded in 1703 under lead of a great Bohemian Baroque architect Kryštof Dientzenhofer. It is one of the largest and finest baroque churches in the country.

T1 Tram stop Malostranské náměstí (Lesser Town Square). Trams 12, 20, 22 and 57 (night tram).

T2 Tram stop Helichova. Trams 12, 20, 22 and 57 (night tram).

T3 Tram stop Újezd. Trams 12, 20, 22 and 57 (night tram).

T4 Metro and Tram stop Malostranská. Metro A; trams 12, 18, 20, 22 and 57 (night tram).

T5 Metro and Tram stop Staroměstská. Metro A; trams 17, 18, and 53 (night tram).

T6 Tram stop Národní divadlo (National Theater). Trams 6, 9, 18, 22, 53, 57, 58 and 59 (night tram).

T7 Metro and tram stop Národní. Metro B; trams 6, 9, 18, 22, 53, 57, 58, 59.

T8 Metro stop Můstek. Metro A and B.

T9 Metro and tram stop Anděl. Metro B; trams 6, 9, 10, 16, 58, 59.

Accommodation:

A4 Hostel Sokol
Nosticova 465/2

A5 Karolínium
Celetná 14

Main sightseeing spots:

1 Lesser Town (Workshop Site)
Across the Vltava River from the city centre and leading to the castle, this quarter offers beautiful streets and churches (of which St. Nicholas Church, next to workshop site, is the most renowned). The Lennon Wall, which used to be a source of irritation to the communist regime, is also found here, near a Venetian-like canal with water wheel and close to the Charles Bridge.

S1 Prague Castle

This, the biggest ancient castle in the world, according to Guinness World Records, rises like a dream above the city offering beautiful views of the areas below. Also on site is the St. Vitus Cathedral with its lookout tower, the Castle Picture Gallery, several palaces and museums and the beautiful Royal Garden, among others. You can also watch the Presidential Guard, and the changeover of the guards on duty on the hour. A Prague castle ticket is 350 CZK and an audio guide costs a further 350 CZK.

S2 Charles Bridge

Charles Bridge connects Old Town with Lesser Town. Its construction started in the 14th century and it is one of Prague’s most beautiful structures. During the day, it is a bustling place of trade and entertainment, as musicians busk and artists sell their paintings and jewelry.

S3 Old Town Square (Astronomical clock)

Prague’s historic centre includes numerous historic buildings and monuments, most notably the famed Astronomical Clock (Orloj), the pure Gothic Týn Church, the mural-covered Storch building, and the Jan Hus monument. Nearby, the Estate Theatre is a neoclassical theatre where Mozart’s opera Don Giovanni was first performed. Old Town features many historical churches (St. James Church, Church of Our Lady before Týn among others) and some other interesting historical buildings like the Old Town Hall.

S4 Kampa (park and private museum of modern art)

S5 Jewish town (Old Jewish Cemetory)

This historic Jewish ghetto is interesting for its well preserved synagogues. The Old New Synagogue (Czech: Staronová synagoga) is Europe’s oldest active synagogue and it is rumoured to be the resting place of the famed Prague Golem. Another interesting synagogue is the Spanish Synagogue, a highly ornamental building of Moorish style. Other attractions include the Old Jewish Cemetery, which is the oldest in Europe, and Kafka’s house. The Old New Synagogue is NOT a part of the Jewish museum, so if you wish to see everything, it is recommended that you buy a combined pass to all of the Jewish attractions for 480 CZK.

S6 Petřín outlook tower; mirror labyrinth; museum of touristic club and Jára Cimrman; observatory.