Getting Started as a GCC Contributor

David Malcolm <dmalcolm@redhat.com>

FOSDEM 2024

4th February 2024

Overview

- How to build GCC from source
- High-level view of GCC internals
- Tour of how GCC compiles an optimizes a simple C function
- How to add a warning

- Feel free to ask questions
- See also: https://gcc-newbies-guide.readthedocs.io/en/latest/index.html

Red Hat

Objectives

- Get you comfortable at:
  - Building GCC from source
  - Debugging GCC using GDB
- Write your first patch
  - Tweak an existing warning
  - Add a new warning
  - Improve an optimization
  - Fix a bug
- Figure out how/where GCC’s documentation needs to be improved

Building GCC from source

Building GCC from source

- “Build” vs “host” vs “target”
- “Build” machine requirements:
    - Ideally running some form of Linux (with tools such as GCC and GDB already installed)
    - Ideally 10GB of free disk space
        - More is better! Separate development vs testing trees
    - More cores is better
    - GCC compile farm is available if you need a powerful box

Building GCC from source

- Prerequisites (Debian):

```
sudo apt install \
perl gawk binutils gcc-multilib \
python3 python3-pip gzip make tar zstd autoconf automake \
gettext gperf dejagnu autogen guile-3.0 expect tcl flex texinfo \
git diffutils patch git-email
```

Red Hat

Building GCC from source

- Prerequisites (Fedora/Red Hat/etc):

```
sudo dnf install \
  diffutils gawk gcc-c++ gettext git dejagnu \
  make patch texinfo flex
```

Building GCC from source

- Separate build/src subdirectories:

```
gcc-from-git/
gcc-from-git/src/
gcc-from-git/build/
```

Building GCC from source

- Separate build/src subdirectories:

```
mkdir gcc-from-git
cd gcc-from-git
git clone git://gcc.gnu.org/git/gcc.git src
mkdir build
cd build
../src/configure with various args
```

Red Hat

Configuring a minimal build

- ../src/configure \
  --enable-languages=c,c++ \
  --disable-bootstrap \
  --prefix=*some absolute path*
- *make -j4*

What does “gcc foo.c” actually do?

“GCC” vs “gcc” vs “cc1” etc

- “GCC” is the project
- *“gcc” is a (relatively) small “driver” binary*
- *“cc1” is the C compiler, “cc1plus” for C++, etc*
- *DEMO: -v*

GCC’s internal representations

How does GCC represent your code internally?

```
./xgcc -B. \
    -S \
    test.c \
    -O2 \
    -fverbose-asm \
    -fdump-tree-all \
    -fdump-ipa-all \
    -fdump-rtl-all \
    -wrapper gdb,--args
```

How does GCC represent your code internally?

- Tokens (libcpp)
- “Tree”
- GENERIC
- GIMPLE
- GIMPLE with CFG
- GIMPLE-SSA
- RTL, with CFG
- RTL, without CFG
- Callgraph

Debugging GCC

Debugging GCC

- Goals:
  - Identify where a particular diagnostic is being emitted
  - Identify where in the frontend a particular tree node is created/modified
  - Identify where in the middle-end a particular optimization is happening
- DEMO !

Fixing a bug

Fixing a bug

- Goals:
  - Pick a bug marked with the keyword “easyhack”
  - URL: https://tinyurl.com/gcc-easyhacks

More ambitious hacks

More ambitious hacks

- Too much to cover today…
  - Adding a new warning
  - Adding a new optimization
  - Adding a new frontend
- Coming up next: adding a new target

Wrapping up

Wrapping up

- Questions?
    - I’m around here at FOSDEM
- Next steps?
    - Join us on the mailing lists/IRC
- What needs improving in the docs?
- Thanks for coming!

THANKS

Red Hat