A

An Introduction to GCC

A.1 About GCC

GCC is an acronym for GNU Compiler Collection (http://gcc.gnu.org) which is the de-facto standard compiler generation framework for a number on GNU/Linux and many other variants of Unix/Linux on a wide variety of machines and is one of the most dominant softwares in the free software community.

GCC started as C compiler, and was the acronym for *GNU C Compiler* in the early days. Over the years, it has been continuously upgraded to support a number of back end machines. Similarly, on the front end side, it has grown to support a number of front end languages like C++, Objective C, Java, and FORTRAN to name a few. As a consequence, it has been renamed as *GNU Compiler Collection*.

As of 2008, GCC supports six front end languages, and 34 back end machines. This results in a quite huge code base, and GCC has earned a reputation of being one of the most complex and major free software/open source projects. A rough line count of all the C source files (including header files) of just the compiler code (i.e., only the `gcc` directory) with the major block comments removed is about 1440336 lines. This does not include the code that describes the supported back end machines, as well as code for other purposes like the build system, libraries etc.

GCC generates production quality optimizing compilers from descriptions of target platforms. It supports a wide variety of source languages and target machines (including operating system specific variants) in a ready-to-deploy form. Besides, new machines can be added by describing instruction set architectures and some other information e.g., calling conventions. This retargetability requirement implies that target information is incorporated into the compiler at build time rather than at design time. The GCC sources consist of

- Language dependent front end.
- Language and target independent modules.
- Target architecture specification.

A compiler for specific language-target pair is generated by selecting front end for desired language and generating back end for specified target.

A.2 Building GCC

There are four directories that are useful to describe the user level building of GCC. They are not required to be defined in practice.

- The directory where we have downloaded the compressed sources. We denote this by `$DOWNLOADDIR`
- The directory where we extract the downloaded sources. We denote this by `$GCCHOME`
- The directory where we build the compiler for the chosen source language and target machine. We denote this by `$BUILDDIR`
- The directory where the built compiler is installed for use. We denote this by `$INSTALLDIR`

The GCC build instructions in `$GCCHOME/INSTALL/index.html` recommend the use of a distinct build directory and discourages building GCC in `$GCCHOME`. Any directory with suitable permissions that is different from `$GCCHOME` may be used.

The binaries, libraries, headers and documentation that is built is installed as a directory tree under `$INSTALLDIR`. This is any convenient directory with suitable permissions, and usually distinct from the others. The default is a system wide installation directory e.g., `/usr/local`, but can be specified when GCC is configured for building.

There are four steps to building the compiler.

- change to the `$BUILDDIR`,
- configure the pristine GCC sources,
- build the compiler binaries, libraries etc., and
- install the compiler.

In the description below, unless otherwise stated, we assume a GNU/Linux system running on an i386 with the GNU Bourne Again SHell (`bash`) as the command shell. All commands are issued at the bash shell prompt, and shell commands or scripts are bash scripts.

Configuring GCC

The pristine GCC sources must be informed about some details like the system on which it will eventually run. A shell script called `configure` is used for this. Most pieces of required information have reasonable default values, and the usual way is to simply issue the `configure` command, which uses the defaults. However, specific non default values can be given to the `configure` command through command

line switches. Being a retargetable compiler that supports a number of high level languages (HLLs), the sources need to be informed about the particular source language and the target hardware on which the built compiler is to be used. By default, GCC is configured to build a compiler for the target on which it is being compiled. If a compiler for a specific language is desired, then the switch `--enable-languages` can be used. The install directory defaults to `/usr/local`, but can also be specified using the `--prefix` switch. The configure `--help` command lists out various such options whose details are documented in `$GCCHOME/INSTALL/index.html`.

To build only a C compiler for a i386 for running on a GNU/Linux operating system and `/home/gcc/gcc-trial-install` as the installation directory, we configure as follows:

1. Change to the build directory

```
cd $BUILDDIR
```

2. Specify that we need only the C compiler, to run on an i386 machine running GNU/Linux and `/home/gcc/gcc-trial-install` as the installation directory (each option is shown on a separate line for clarity, but is one single command line)

```
$GCCHOME/configure
--enable-languages=c
--target=i386-linux-gnu
--prefix=/home/gcc/gcc-trial-install
```

The `configure` program makes a number of checks for a successful build and generates a `Makefile` (as `$BUILDDIR/Makefile`) if all checks pass. It is useful to redirect the output of `configure` to some file for later study as follows:

```
$GCCHOME/configure > configure.log 2> configure.errors
```

Compiling GCC

Once the configuration successfully generates the `Makefile`, the GCC source is compiled by issuing the `make` command. The steps are:

1. `cd $BUILDDIR`
2. `make`

Compiling GCC involves building the compiler for each source language, the driver program `gcc`, the associated header files, support libraries, and the documentation. The driver program `gcc` so generated is the command that users use to compile their source programs. The driver takes the user's source file to be compiled and invokes a sequence of programs (the compiler, the assembler and the linker) that generate its binary.

It is useful to redirect the output of `make` to some file for later study as follows:

```
$BUILDDIR/make > make.log 2> make.errors
```

Installing GCC

Final installation installs various components of the compiler like the driver, the compiler, libraries, the documentation etc., in a well-defined directory structure in the `$INSTALLDIR` directory. The following structure is typically used:

- `$INSTALLDIR/bin`: Directory where the various executables are installed.
- `$INSTALLDIR/include`: Directory where the various headers are installed.
- `$INSTALLDIR/lib`: Directory where the various libraries are installed.
- `$INSTALLDIR/man`: Directory where the various online manual pages are installed.
- `$INSTALLDIR/info`: Directory where the various online info pages are installed.

To install the built sources, use the following command:

```
$BUILDDIR/make install
```

It is useful to redirect the output of install to some file for later study as follows:

```
$BUILDDIR/make install > install.log 2> install.errors
```

Downloading and Installing *gdfa*

A patch of GCC 4.3.0 for *gdfa* is available at the following URL. Patches for later versions will be made available on this page whenever possible.

http://www.cse.iitb.ac.in/uday/dfaBook-web

Following steps patch up GCC with *gdfa* code.

1. `cd $GCCHOME`
2. `patch -p0 < patch_file_with_path`

Now GCC can be configured, compiled, and installed.

A.3 Further Readings in GCC

Here we list further resources for learning about GCC.

- GCC Internals
  http://gcc.gnu.org/onlinedocs/gccint.html
  This is the official internals document which exhaustively describes most details and is a part of the documentation distributed with the compiler code.

- GCC Internals documents developed at IIT Bombay
  http://www.cse.iitb.ac.in/grc/
  This is the website of *GCC Resource Center* at IIT Bombay. It hosts the GCC documents developed at IIT Bombay.

- The GCC Wiki
  http://gcc.gnu.org/wiki/
  The official GCC Wiki pages where the various aspects of GCC, including some description of the internals, are being developed by the GCC developers and others.

- The GCC Internals workshop held at IIT Bombay
  http://www.cse.iitb.ac.in/~uday/gcc-workshop/
  This workshop that focused mainly on the machine descriptions was held at IIT Bombay in June 2007. The slides and some associated software is available on the `Downloads` page of the workshop.

- The GCC on Wikipedia
  http://en.wikipedia.org/wiki/GNU_Compiler_Collection

- The GCC Internals on Wikipedia
  http://en.wikibooks.org/wiki/GNU_C_Compiler_Internals