GCC: Compiler Hotness

Audience: This is a basic high-level introduction to the GNU Compiler Collection (gcc) for those developers largely unfamiliar with it.

GCC and This Class

You don't need a deep knowledge of gcc's features and command line arguments for this class. We take care of almost all of that for you. If and when you get to the point of working extensively with gcc directly—or more likely configuring build scripts—see the gcc references linked in the final section of this document.

Verify Your Installation

Though you need little working knowledge of gcc for this class, one thing you should be able to do is verify its installation. This is done simply with:

```
gcc --version
```

If this command fails, gcc is not installed or you have a path problem.

What Is GCC?

C evolved as a system-level programming language in parallel with Unix at AT&T's Bell Labs in the late 60's and early 70's. The language was not officially standardized until the late 80's and early 90's. The GNU Compiler Collection (gcc) was begun in the early 80's as a key, free component of the larger GNU Operating System. In fact, GNU as a full operating system is only just now coming to fruition, but pieces of the larger project have been readily adopted within Linux for quite some time. Originally gcc built executable binaries from only C source code. GCC now supports several languages through different front ends.

GCC is a pipeline of processing layers executed in multiple passes. Each layer and pass is responsible for transforming source code through various intermediate forms into machine code executable on a target platform. It's this structure that allows GCC to work with more languages than only C and has allowed it to be ported to so many different targets and platforms.

Availability

It's safe to assume gcc is available to create native executables for every computer on earth of desktop class or greater. It's also available for many embedded platforms as well. In embedded development, gcc most often takes the form of a cross-compiler able to create binaries for your embedded processor through your desktop machine.

Of course, many other implementations of the C language exist. The gcc implementation is the most widely available today, and it's free.

A Word on Unity and CMock

Unity itself along with the code CMock generates build with pretty much any compiler on any platform you can throw them against. Many iterations of development, bug fixing, crafting of conditional compilation options, and testing with a range of tool chains has refined the code to be highly portable.

Linux

In Linux, you already have gcc and a command line terminal to run it. Command line switches allow you to create binaries for a limited set of targets other than the host processor.

OS X

If you're using OS X, gcc is part of the Unix underbelly of the operating system accessible through the terminal. (Actually, the latest versions of OS X have gcc symlinked to clang, another powerful open source compiler. Clang is the new hotness. In this case, clang acts almost identically to gcc, so the differences aren't worth discussing here). Command line switches allow you to create binaries for a limited set of targets other than the host processor.

Windows

Unlike other platforms, Windows does not come pre-installed with gcc. In Windows world, Microsoft's Visual Studio is the vendor environment for Windows development.

However, gcc is available within command line development environments available for Windows. We recommend MinGW [download] because it's free, works well, and is lightweight and self-contained. The better known option of this sort is Cygwin. Cygwin is a big package that attempts to provide an entire Unix-like environment for Windows. Like MinGW, Cygwin is also free, but it's generally a larger time investment than MinGW. Cygwin also requires dancing around requirements like dependencies on `cygwin.dll` for binaries produced by gcc (MinGW produced binaries have no such complications). If you know what you're doing Cygwin is a fine option.

Incidentally, though we recommend MinGW, you can, of course, use Unity and CMock with Microsoft's Visual Studio or any other native C compiler.

Other Platforms and Embedded Targets

Search the web for “gcc” together with your specific target and cross your fingers. You’ll probably get lucky. You’ll likely find a cross compiler variant of gcc that runs on Linux or Windows and builds a binary for your target processor / platform.

References & Resources

- Simple gcc tutorial
- Introduction to gcc, a manual
- C Frequently Asked Questions
- Dennis Ritchie’s recollections: *The Development of the C Language*