Applied Math 205: Configuring C++

Michael S. Emanuel

08-Sep-2021

Choosing a C++ Toolchain for Your System

AM 205 is holding group activities over the next several weeks that will introduce students to programming in C++ and parallel programming. In order to have the best experience at these sessions, the course staff suggest that students who have not already done so install a working C++ compiler and toolchain on their computer system(s). For most AM 205 students, this means installng software on your laptop computer, but some students may have access to multiple systems including a desktop PC at home or in a lab.

In order to run a C++ program, you must first select and install a compiler. The term compiler is accurate but incomplete. When we convert C and C++ source files into executable programs, we rely on a whole suite of programs including

- a C prepreprocessor, e.g. `cpp`[1] which handles `#include` directives and macros
- a C compiler, generically named `cc`, e.g. `gcc`, `clang`
- a C++ compiler, generically named `cxx`, e.g. `g++` or `clang++`
- a linker, e.g. `ld` which combines object files into an executable program
- a build system, e.g. `GNU Make`

Collectively, this suite of programs is often called the **toolchain** in software development.

Each toolchain has its own quirks and incompatibilities with the others. Time spent learning one of them is an investment and we want you to get a high return on it. So before we dive into writing, building and running C++ programs, we must first make an important strategic decision about which toolchain to use. **The AM 205 course staff recommends that students getting started with C++ for this course use the GNU Compiler Collection (gcc) collection.**[2] Why are we making this recommendation? `gcc` is a free open source program that is available on all major computing platforms including Windows, Apple and Linux. It is mature and has a high degree of conformity to the C++20 language standard. Perhaps most importantly, it is the most popular choice in the numerical computing community. In the sections below, I will explain how to install the gcc toolchain on Linux, Windows and Apple Mac systems.

Installing `gcc` and `g++` on an Ubuntu Linux System

Installing gcc is most straightforward on Ubuntu systems and can be accomplished in a few minutes. I will show exactly how to do this for systems using the `apt` package manager such as Ubuntu. The process is analogous for other Linux distros using different package managers like `rpm` and `yum`. First we can start by installing the version of `gcc` that has been incorporated into the main Ubuntu repositories, which is version 10.3.0. We can install it by running this command:[3]

```
$sudo apt install gcc-10 g++-10
```

[1] It's a bit confusing that the name of this tool looks like a C++ source file, but the preprocessor `cpp` is ancient and predates the C++ language.

[2] This advice does not apply to students who are already familiar with another toolchain.

[3] You will need administrative permissions to follow the recipe shown here. This probably applies to a personal computer that you own. If you want to run `gcc` on a system where you are not the administrator, it is very likely that this package is installed. For instance, on some shared computing resources at Harvard, the command `module load gcc` will make gcc available in your session. You can speak to the system administrator or post on the class discussion board if you need help with this.

It turns out that gcc-10 is already a bit old. This is because there is a long lead time before software makes it to the main Ubuntu package management repositories. In order to get version 11, we need to add a different repository to the package manager:

```
$sudo add-apt-repository ppa:ubuntu-toolchain-r/test
```

Once this command runs successfully, we need to update the list of available packages, then install gcc-11:

```
$sudo apt update
$sudo apt install gcc-11 g++-11
```

Now we have two versions of gcc installed on our system, named gcc-10 and gcc-11. We can just type g++-11 every time we want to compile a C++ program, but that gets old in a hurry. The Linux utility update alternatives allows us to dynamically switch which program we get when we type gcc. These two commands will set it up for you:[4]

```
$sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-10 100 \
--slave /usr/bin/g++ g++ /usr/bin/g++-10
$sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-11 110 \
--slave /usr/bin/g++ g++ /usr/bin/g++-11
```

Once we have installed our gcc alternatives, we can review them and set them like this:

```
$update-alternatives --display gcc
$sudo update-alternatives --auto gcc
```

We can confirm that we have the expected version in automatic mode by running this command:

```
$gcc --version
```

The output on my system is:

```
gcc (Ubuntu 11.1.0-1ubuntu1 20.04) 11.1.0
```

Finally, we should install some useful software development tools by running:

```
$sudo apt install build-essential
```

Installing gcc and g++ on a Windows 10 System

Until relatively recently, Windows users were in for a slog if they wanted to run Linux programs. There were options, but they were not attractive. Things have improved considerably since Microsoft embraced open source software and introduced the Windows Subsystem for Linux (WSL). The AM 205 course staff recommends that students with Windows computer systems install WSL 2.

The official instructions from Microsoft to install WSL 2 on Windows 10 systems can be found at https://docs.microsoft.com/en-us/windows/wsl/install-win10. Once WSL 2 is on your computer, you can install Ubuntu 20.04 LTS from the Microsoft App Store. After a restart, you can run the Ubuntu 20.04 app and it will give you a terminal. There are other ways to interact with WSL 2 that are a bit slicker. You can connect to it using the Windows Terminal app or a terminal emulator app such as cmder. You can also do software development on WSL 2 from the VS Code IDE by connecting to it over SSH. This is a built in feature of VS Code that is easy to use. After you have a working installation of Ubuntu in your WSL 2 environment, you can follow the instructions from the previous section on configuring gcc and g++.

Installing gcc and g++ on an Apple Mac System

To get started with software development on your Mac, download the Xcode software development tools package from the App Store. Install the main package as well as the command line tools. Xcode will include a high quality C++ compiler clang++. This is a good choice for many people starting out with C++ development on a Mac. However, for this course, the staff recommends that students use g++ for better compatibility with the larger scientific computing community.

Running Linux applications on an Apple system is quite complicated, and you are faced with a strategic choice about which package management system you want to use for "ports" from Linux to Apple. There

[4] I apologize for the offensively named terminology in the update-alternatives command. The software community is slowly moving forward to more inclusive language, e.g. with the naming of the default branch in git switching over to main, but it's very much a work in progress. The update-alternatives has not yet renamed the option to make a second symlink be dependent on the primary. I would like to see it renamed to –depends myself.

are two main options to consider, Mac Brew and MacPorts. Mac Brew is the more popular choice. But Macports has some important advantages, including that it has more up to date packages, and better security practices (it requires `sudo` to install software instead of a workaround a la homebrew). You can read about them online if you search for "homebrew vs. macports" on Google. While there is no one right answer to this question, the AM 205 course staff recommends that students use MacPorts for the course unless they are already set up on Mac Brew.

You can find detailed instructions to install MacPorts at guide.macports.org. Once you have a working installation of MacPorts, you need to issue just one command in the Mac terminal app:

```
%sudo port install gcc11
```

You can test that you installed `gcc` successfully by running the command `gcc --version`. On my system, this generates six lines of output indicating configuration options including

- Configuration options `prefix` and `with-gxx-include-dir`
- Compiler used to build gcc: `Apple clang version 12.0.5 (clang-1205.0.22.11)`
- Target architecture `Target: x86_64-apple-darwin20.6.0`
- `Thread model: posix`.

If you run into any problems with the installation on a Mac, please post to the Ed discussion board. Either the teaching staff or other students can help you.