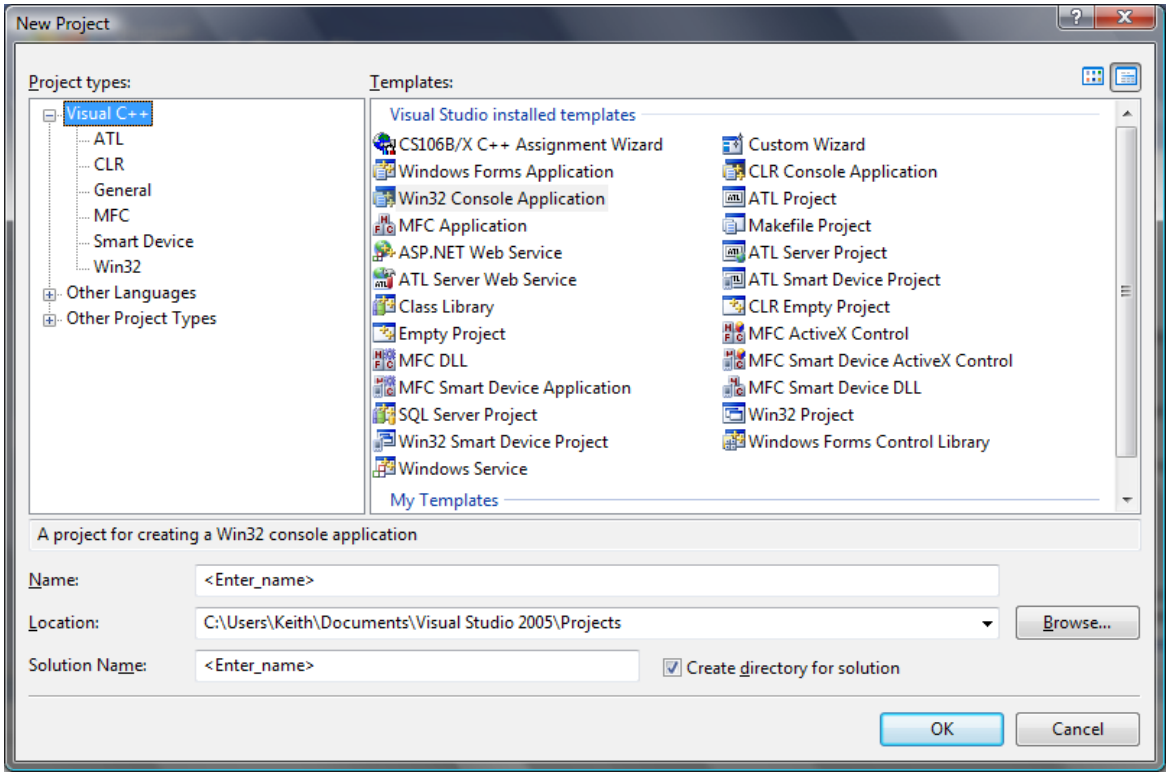
Chapter 1: Getting Started

Every journey begins with a single step, and in ours it's getting to the point where you can compile, link, run, and debug C++ programs. This depends on what operating system you have, so in this section we'll see how to get a C++ project up and running under Windows, Mac OS X, and Linux.

Compiling C++ Programs under Windows

This section assumes that you are using Microsoft Visual Studio 2005 (VS2005). If you are a current CS106B/X student, you can follow the directions on the course website to obtain a copy. Otherwise, be prepared to shell out some cash to get your own copy, though it is definitely a worthwhile investment.* Alternatively, you can download Visual C++ 2008 Express Edition, a free version of Microsoft's development environment sporting a fully-functional C++ compiler. The express edition of Visual C++ lacks support for advanced Windows development, but is otherwise a perfectly fine C++ compiler. You can get Visual C++ 2008 Express Edition from http://www.microsoft.com/express/vc/. With only a few minor changes, the directions for using VS2005 should also apply to Visual C++ 2008 Express Edition, so this section will only cover VS2005.

VS2005 organizes C++ code into "projects," collections of source and header files that will be built into a program. The first step in creating a C++ program is to get an empty C++ project up and running, then to populate it with the necessary files. To begin, open VS2005 and from the **File** menu choose **New > Project...**. You should see a window that looks like this:

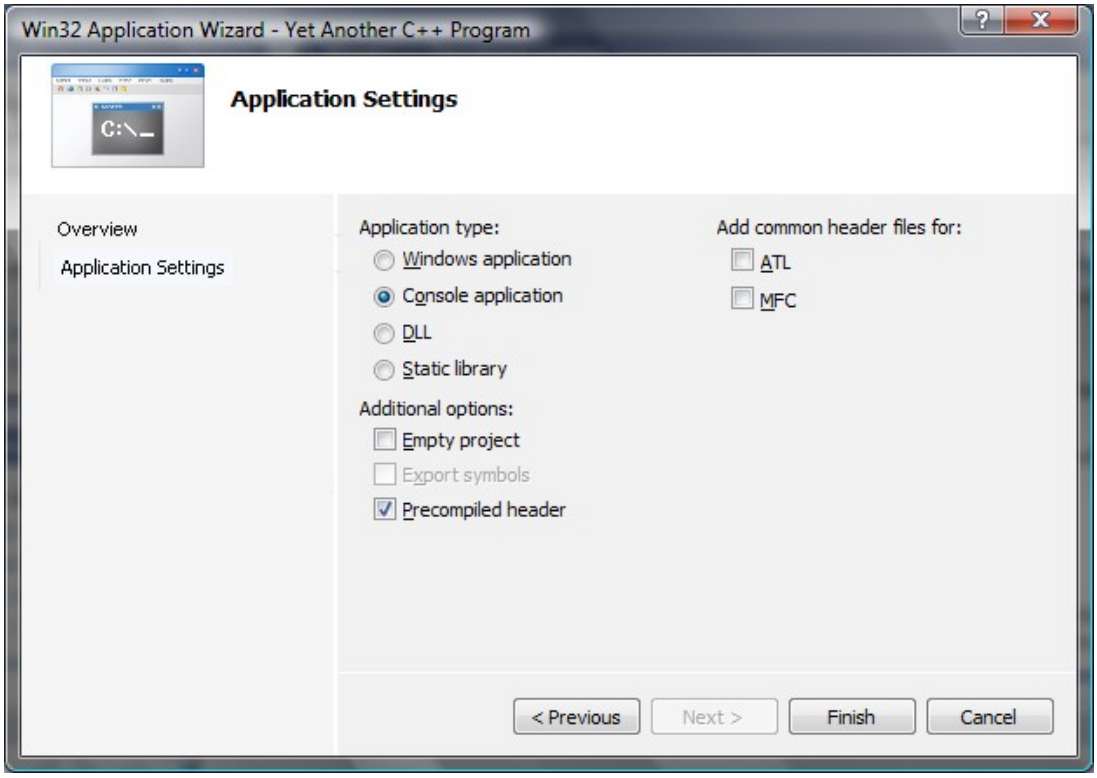


* I first began programming in C++ in 2001 using Microsoft Visual C++ 6.0, which cost roughly eighty dollars. I recently (2008) switched to Visual Studio 2005. This means that the compiler cost just over ten dollars a year. Considering the sheer number of hours I have spent programming, this was probably the best investment I have made.

As you can see, VS2005 has template support for all sorts of different projects, most of which are for Microsoft-specific applications such as dynamic-link libraries (DLLs) or ActiveX controls. We're not particularly interested in most of these choices – we just want a simple C++ program! To create one, find and choose **Win32 Console Application**. Give your project an appropriate name, then click **OK**. You should now see a window that looks like this, which will ask you to configure project settings:

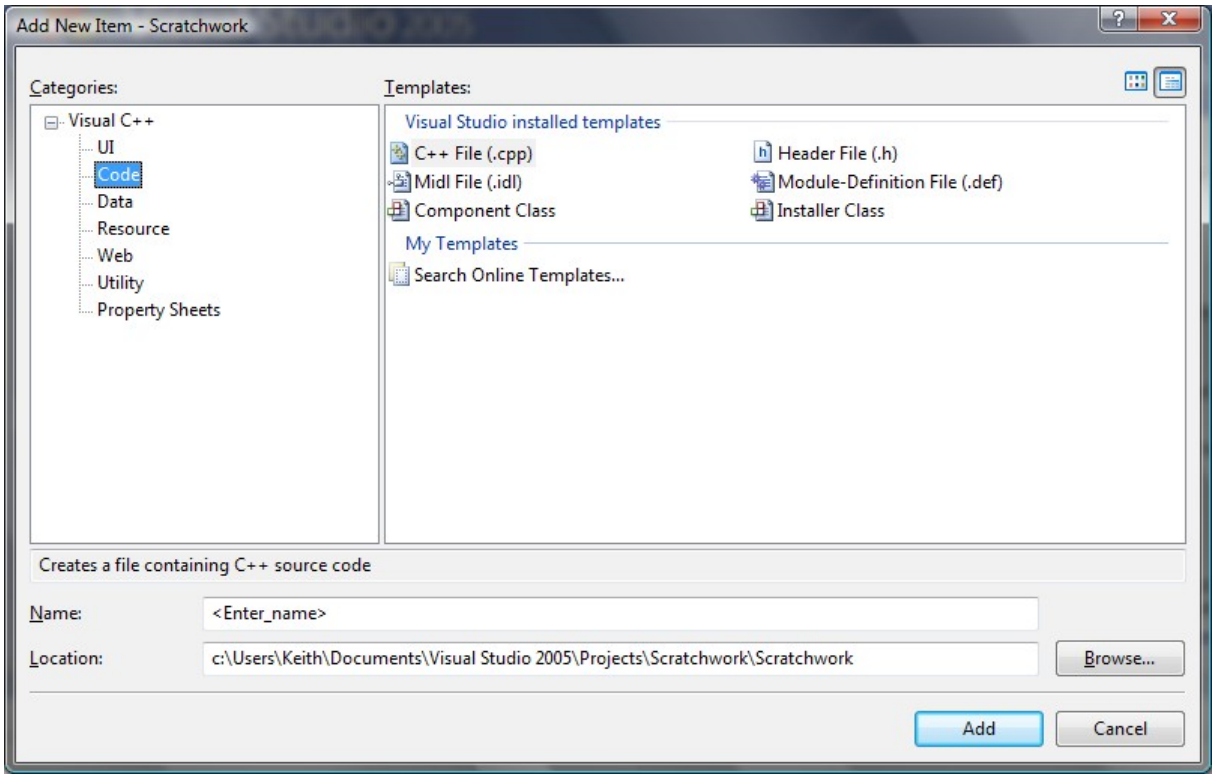
Note that the window title will have the name of the project you entered in the previous step in its title; "Yet Another C++ Program" is a placeholder.

At this point, you **do not** want to click **Finish**. Instead, hit **Next >** and you'll be presented with the following screen:



Keep all of the default settings listed here, but make sure that you check the box marked **Empty Project**. Otherwise VS2005 will give you a project with all sorts of Microsoft-specific features built into it. Once you've checked that box, click **Finish** and you'll have a fully functional (albeit empty) C++ project.

Now, it's time to create and add some source files to this project so that you can enter C++ code. To do this, go to **Project > Add New Item...** (or press CTRL+SHIFT+A). You'll be presented with the following dialog box:



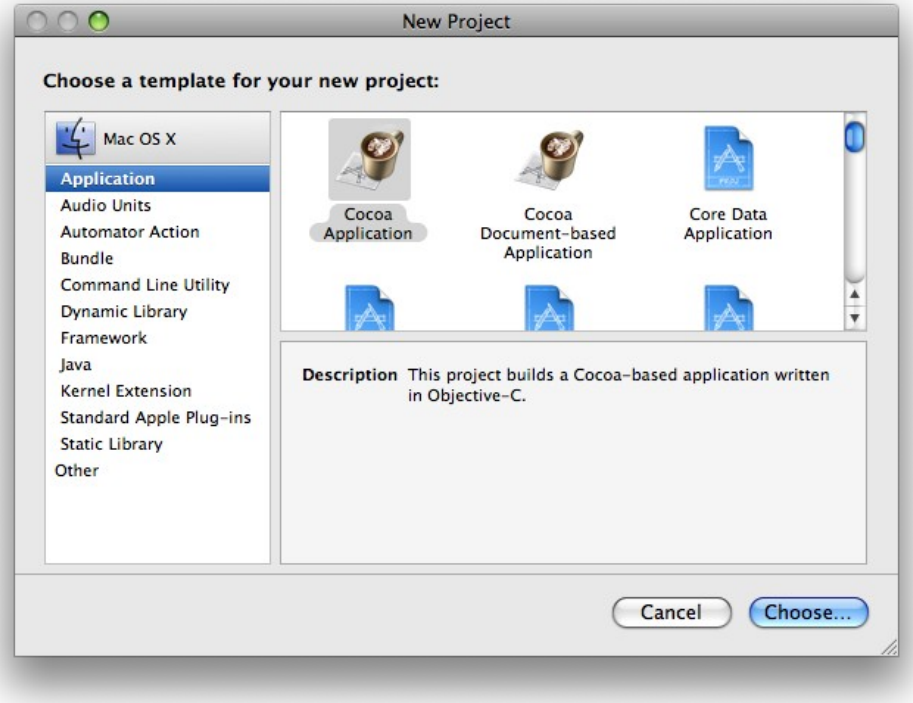
Choose **C++ File (.cpp)** and enter a name for it inside the Name field. VS2005 automatically appends .cpp to the end of the filename, so don't worry about manually entering the extension. Once you're ready, click **Add** and you should have your source file ready to go. Any C++ code you enter in here will be considered by the compiler and built into your final application.

Once you've written the source code, you can compile and run your programs by pressing **F5**, choosing **Debug> Start Debugging**, or clicking the green "play" icon. By default VS2005 will close the console window after your program finishes running, and if you want the window to persist after the program finishes executing you can run the program without debugging by pressing **CTRL+F5** or choosing **Debug > Start Without Debugging**. You should be all set to go!

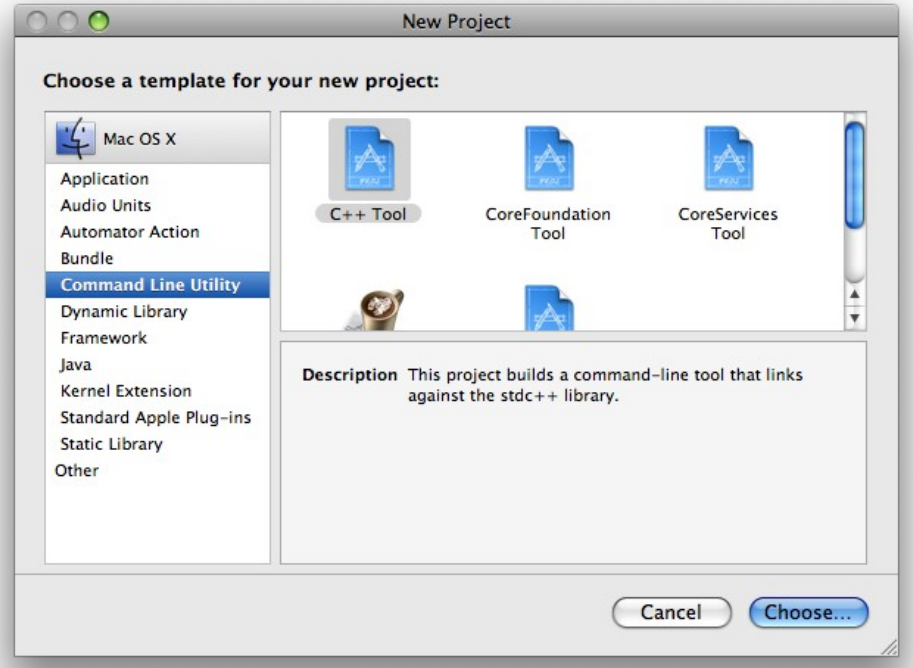
Compiling C++ Programs in Mac OS X

If you're developing C++ programs on Mac OS X, your best option is to use Apple's Xcode development environment. You can download Xcode free of charge from the Apple Developer Connection website at http://developer.apple.com/.

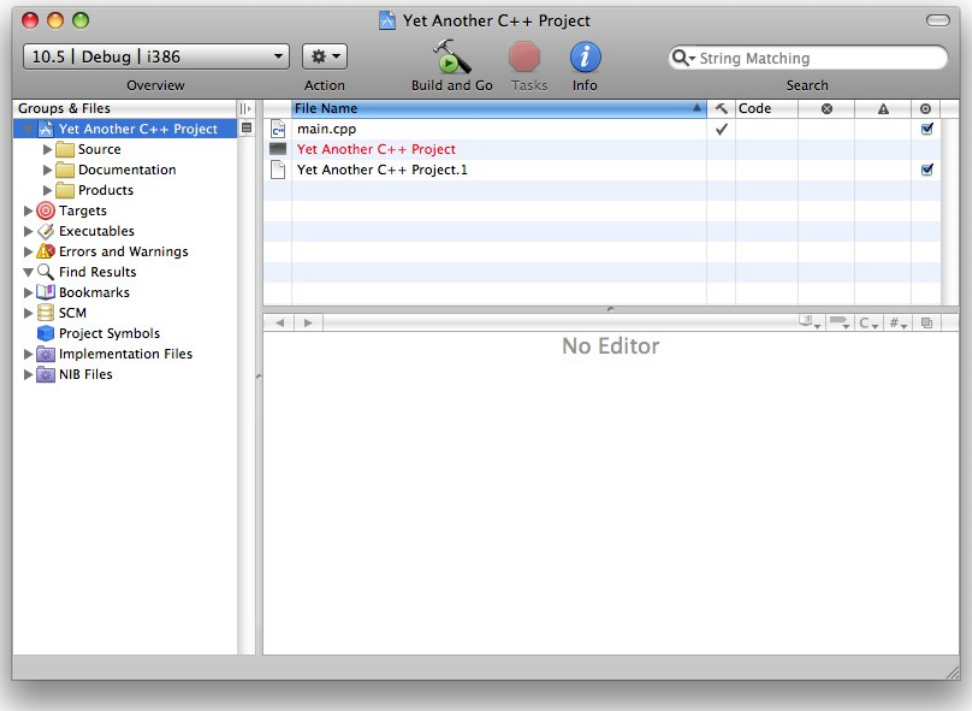
Once you've downloaded and installed Xcode, it's reasonably straightforward to create a new C++ project. Open Xcode. The first time that you run the program you'll get a nice welcome screen, which you're free to peruse but which you can safely dismiss. To create a C++ project, choose **File > New Project...**. You'll be presented with a screen that looks like this:



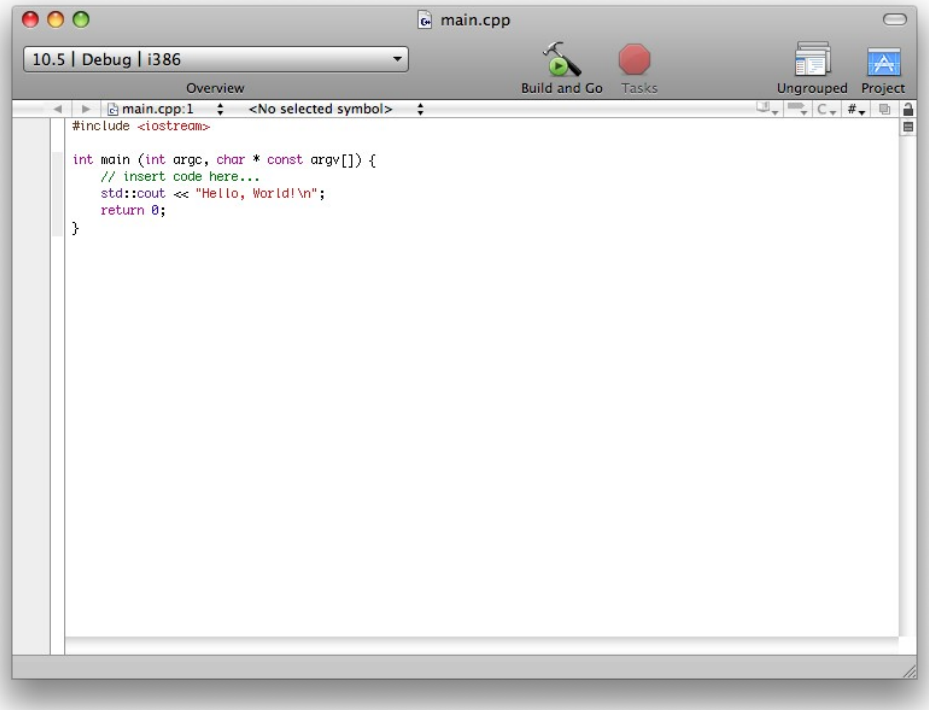
There are a lot of options here, most of which are Apple-specific or use languages other than C++ (such as Java or Objective-C). In the panel on the left side of the screen, choose **Command Line Utility** and you will see the following options:



Select **C++ Tool** and click the **Choose...** button. You'll be prompted for a project name and directory; feel free to choose whatever name and location you'd like. In this example I've used the name "Yet Another C++ Project," though I suggest you pick a more descriptive name. Once you've made your selection, you'll see the project window, which looks like this:

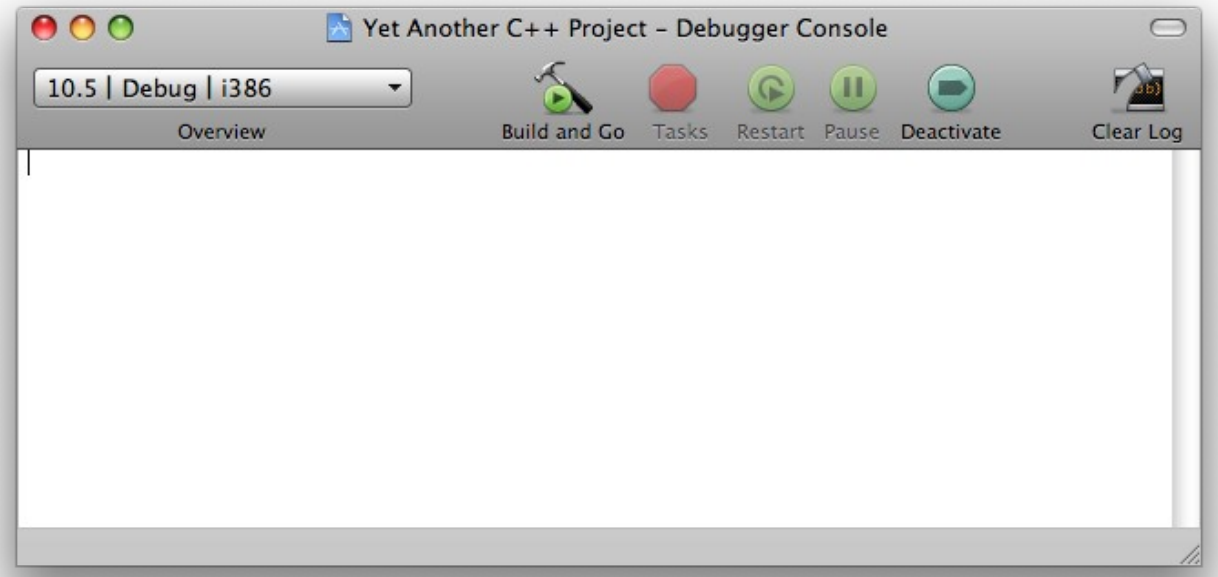


Notice that your project comes prepackaged with a file called `main.cpp`. This is a C++ source file that will be compiled and linked into the final program. By default, it contains a skeleton implementation of the Hello, World! program, as shown here:



Feel free to delete any of the code you see here and rewrite it as you see fit.

Because the program we've just created is a command-line utility, you will need to pull up the console window to see the output from your program. You can do this by choosing **Run > Console** or by pressing ⇧⌘R. Initially the console will be empty, as shown here:



Once you've run your program, the output will be displayed here in the console. You can run the program by clicking the **Build and Go** button (the hammer next to a green circle containing an arrow). That's it! You now have a working C++ project.

If you're interested in compiling programs from the Mac OS X terminal, you might find the following section on Linux development useful.

Compiling C++ Programs under Linux

For those of you using a Linux-based operating system, you're in luck – Linux is extremely developer-friendly and all of the tools you'll need are at your disposal from the command-line.

Unlike the Windows or Mac environments, when compiling code in Linux you won't need to set up a development environment using Visual Studio or Xcode. Instead, you'll just set up a directory where you'll put and edit your C++ files, then will directly invoke the GNU C++ Compiler (`g++`) from the command-line.

If you're using Linux I'll assume that you're already familiar with simple commands like `mkdir` and `chdir` and that you know how to edit and save a text document. When writing C++ source code, you'll probably want to save header files with the .h extension and C++ files with the .cc, .cpp, .C, or .c++ extension. The .cc extension seems to be in vogue these days, though .cpp is also quite popular.

To compile your source code, you can execute `g++` from the command line by typing `g++` and then a list of the files you want to compile. For example, to compile `myfile.cc` and `myotherfile.cc`, you'd type

```
g++ myfile.cc myotherfile.cc
```

By default, this produces a file named `a.out`, which you can execute by entering `./a.out`. If you want to change the name of the program to something else, you can use `g++`'s `-o` switch, which produces an output file of a different name. For example, to create an executable called `myprogram` from the file `myfile.cc`, you could write

```
g++ myfile.cc -o myprogram
```

`g++` has a whole host of other switches (such as `-c` to compile but not link a file), so be sure to consult the `man` pages for more info.

It can get tedious writing out the commands to compile every single file in a project to form a finished executable, so most Linux developers use *makefiles*, scripts which allow you to compile an entire project by typing the `make` command. A full tour of makefiles is far beyond the scope of an introductory C++ text, but fortunately there are many good online tutorials on how to construct a makefile. The full manual for `make` is available online at http://www.gnu.org/software/make/manual/make.html.

Other Development Tools

If you are interested in using other development environments than the ones listed above, you're in luck. There are dozens of IDEs available that work on a wide range of platforms. Here's a small sampling:

- **NetBeans**: The NetBeans IDE supports C++ programming and is highly customizable. It also is completely cross-platform compatible, so you can use it on Windows, Mac OS X, and Linux.

- **MinGW**: MinGW is a port of common GNU tools to Microsoft Windows, so you can use tools like g++ without running Linux. Many large software projects use MinGW as part of their build environment, so you might want to explore what it offers you.

- **Eclipse**: This popular Java IDE can be configured to run as a C++ compiler with a bit of additional effort. If you're using Windows you might need to install some additional software to get this IDE working, but otherwise it should be reasonably straightforward to configure.

- **Sun Studio**: If you're a Linux user and command-line hacking isn't your cup of tea, you might want to consider installing Sun Studio, Sun Microsystem's C++ development environment, which has a wonderful GUI and solid debugging support.

- **Qt Creator**: This Linux-based IDE is designed to build C++ programs using the open-source Qt libraries, but is also an excellent general-purpose C++ IDE. It is a major step above what the terminal and your favorite text editor have to offer, and I highly recommend that you check this program out if you're a Linux junkie.