

# Using The Apple C++ Compiler

## Editing And Compiling In Apple Xcode

This summarizes the steps in creating a console program using Apple's Xcode in OS X 10.4.1 and above, and applies to most other Mac OSs, starting with OS 9.

### ❑ 1. Download and Install:

Xcode is not automatically included in the Apple Mac OS. To see if it is already installed, go to the Terminal application and enter the command:

```
g++
```

If the reply is "application not found", then it is not installed. In this case, go to <http://developer.apple.com>, and click the link to download the current version of **Xcode Tools**. This downloads an installation file named, for example: **xcode\_tools\_2.1.dmg** (for version 2.1). Use this to install Xcode and GNU's g++ applications.

### ❑ 2. Create Files:

Use any text editor with which you are familiar, such as **vi** or **Xcode**. If you do not have a suitable editor, you can download **JNotePad** from the Robert Burns' Class Website (<http://comsc.dvc.edu/rburns>) under the link "*more free downloads*". JNotePad installs as a single JAR file, which you can place on your desktop – double-click it to start the program. (If it does not work, you may need to download and install the Java JRE, version 1.4.2 or above, from <http://java.sun.com>).

### Type the Code for the Program

Using C++ language syntax, type your program into the editor's window. For example, type the following, with no indenting on the first line of coding. Use 2-space indenting on the first indented line. (JNotePad automatically inserts 2 spaces when the TAB key is pressed.) Skip single lines where indicated. There are three different sets of enclosing symbols used -- the less-than and greater-than symbols around **iostream**, the parentheses after **main**, and the curly braces where indicated. Be sure to use upper-case and lower-case lettering where shown.

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello, World" << endl;          (that's end-el, not end-one)
    return 0;
}
```

Save your work to your user folder with either the File->Save or File->Save As menu command, and be sure to include a CPP extension (e.g., **HelloWorld.cpp**).

### ❑ 3. Compile and Run:

To compile, use a command like the following (the flag is "dash-oh", *not* "dash-zero") to create an executable:

```
g++ HelloWorld.cpp -o HelloWorld
```

this is how to specify the name of the *executable*

*To run* the program, enter the name of the executable on the command line. Preceded by dot-slash, e.g.:

```
./HelloWorld
```

#### Advanced Instructions – Beyond the Introduction to Programming

To compile and build projects consisting of *more than one CPP*, list the CPPs separated by spaces, like this:

```
g++ main.cpp Time.cpp -o main
```

this executable's name is "main"

The executable name is the last name in the command – **main** in the above example.

To compile a CPP *without building*, include the `-c` flag -- this produces an object file, "Time.o":

```
g++ -c Time.cpp
```

To build an executable from already-compiled object files, list the object files instead of the CPPs:

```
g++ main.o Time.o -o main
```

When working with multiple CPP files in a single project, it is recommended to compile each CPP separately, using the `-c` flag during development. This makes debugging easier. Once the program is working, and you are making small code adjustments, then you should go back to compiling and building all in one command.

#### Sample Session:

Here is a typical Terminal session:

```
Last login: Sat Aug 13 08:53:55 on ttty2
```

```
Welcome to Darwin!
```

```
Robert-Burns-Computer:~ Robert$ g++ HelloWorld.cpp -o HelloWorld
```

```
Robert-Burns-Computer:~ Robert$ ls -l
```

```
total 2
```

```
-rwxr-xr-x  1 Robert Robert 17472 Aug 13 09:00 HelloWorld
```

```
-rw-r--r--  1 Robert Robert  123 Aug 13 08:57 HelloWorld.cpp
```

```
Robert-Burns-Computer:~ Robert$ ./HelloWorld
```

```
Hello, world
```

```
Robert-Burns-Computer:~ Robert$ exit
```

this is the *compile* command

command to *list* files in folder...

...these are the files

the output

command to *end* the Terminal session

the command to *run* the program is its executable's name, preceded by dot-slash

#### Using the Command Line Buffer – Minimize Typing and Typos

So that you do not have to retype the compile and run commands, use the up and down arrow keys to navigate through previously-typed commands.

The usual sequence is to type the compile and build command, followed by the run command. After that, *up-up* returns to the compile and build command, and *up-up* goes from there to the run command.