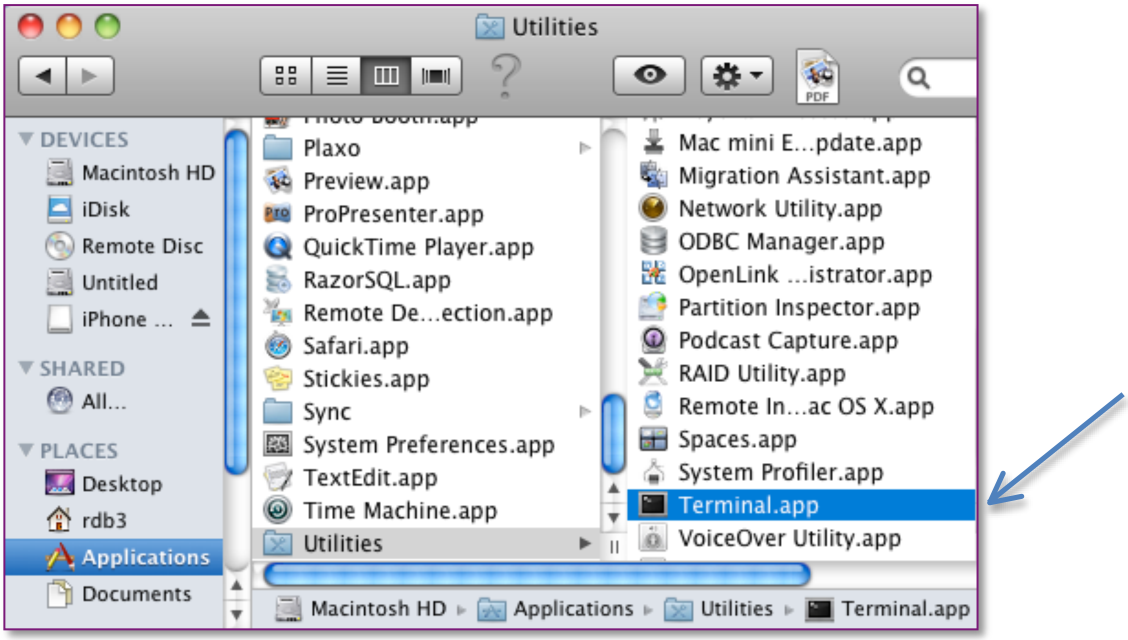
Using The Apple C++ Compiler

Editing And Compiling In Apple Xcode, July 2013 Edition

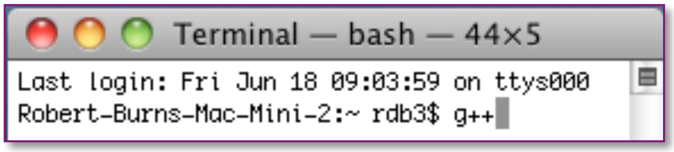
This summarizes the steps in creating a console program using Apple's Xcode in OS X (Snow Leopard), and applies to most other Mac OSs, starting with OS 9.

□ 1. Check for XCode Installation:

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 "compile" command, and see if it works or not. Here's how to start the Terminal app from Finder – go to **Applications**, **Utilities**, and choose **Terminal.app**:

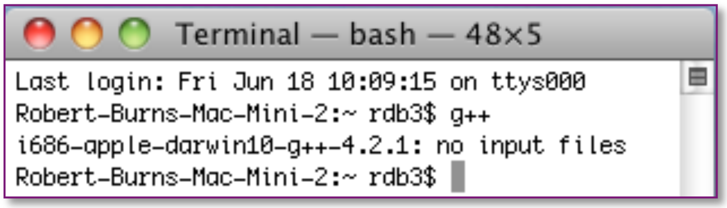


Then in Terminal, type the "compile" command, **g++**, like this, and press ENTER:



```
Last login: Fri Jun 18 09:03:59 on ttys000
Robert-Burns-Mac-Mini-2:~ rdb3$ g++
```

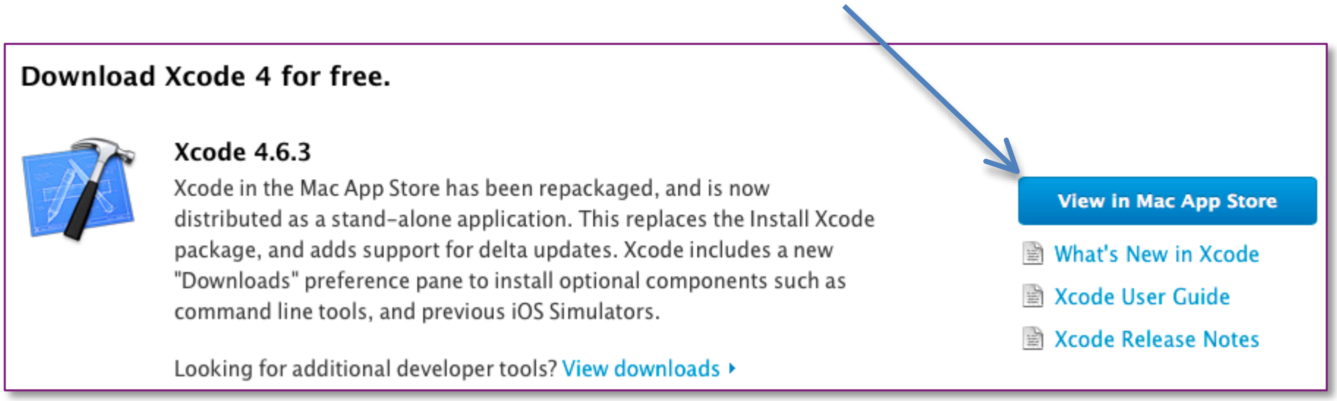
If the reply is "**command not found**", then it is *not* installed. In this case, proceed to step 2. Otherwise, if the reply is "**no input files**", then it *is* installed and you can proceed directly to step 3.



```
Last login: Fri Jun 18 10:09:15 on ttys000
Robert-Burns-Mac-Mini-2:~ rdb3$ g++
i686-apple-darwin10-g++-4.2.1: no input files
Robert-Burns-Mac-Mini-2:~ rdb3$
```

❑ 2. Download and Install XCode:

The Apple App Store has XCode for free. But you need the Lion OS or better. You can find it in the store, or use a browser to go to **http://developer.apple.com/xcode**, and click the “View in Mac App Store” button, located to the right of the “Download Xcode 4 for free” section:

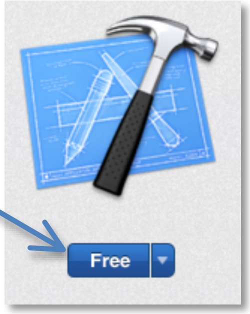


Download Xcode 4 for free.

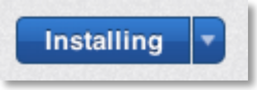
Xcode 4.6.3

Xcode in the Mac App Store has been repackaged, and is now distributed as a stand-alone application. This replaces the Install Xcode package, and adds support for delta updates. Xcode includes a new "Downloads" preference pane to install optional components such as command line tools, and previous iOS Simulators.

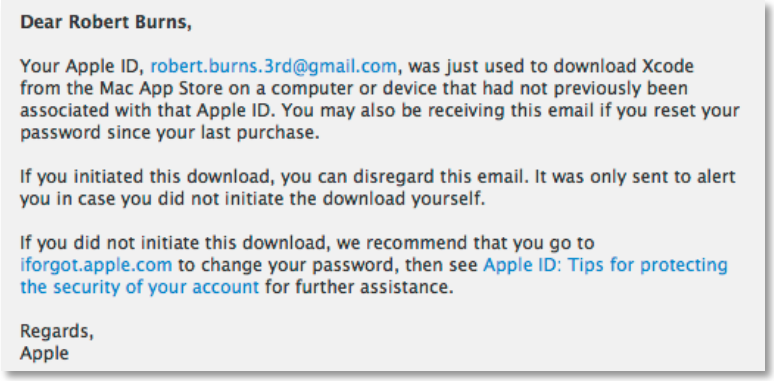
Looking for additional developer tools? View downloads ›



This takes you to the App Store, where you can click the “Free” button. Sign in with your iTunes account, and start the download. It will show this:



…until the download and installation are completed. You may receive an email from Apple while this is going on – you can ignore and delete it. It looks like this:



Dear Robert Burns,

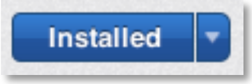
Your Apple ID, robert.burns.3rd@gmail.com, was just used to download Xcode from the Mac App Store on a computer or device that had not previously been associated with that Apple ID. You may also be receiving this email if you reset your password since your last purchase.

If you initiated this download, you can disregard this email. It was only sent to alert you in case you did not initiate the download yourself.

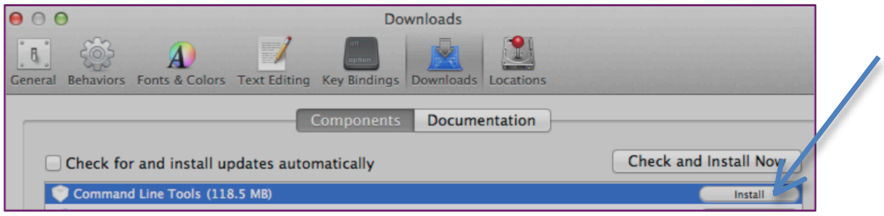
If you did not initiate this download, we recommend that you go to iforgot.apple.com to change your password, then see Apple ID: Tips for protecting the security of your account for further assistance.

Regards,
Apple

When the installation step complete, you should see this beside XCode in the App Store:



Once installed, you will have to *run XCode just once*, to get the "command line tools". Go to XCode's "Preferences", and in the "Downloads" pane, use the "install" button for the "Command Line Tools".



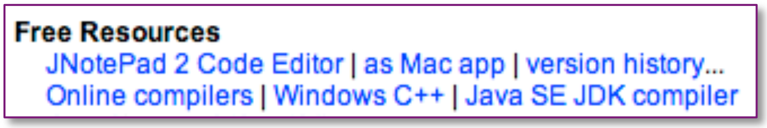
□ Check for and install updates automatically

Command Line Tools (118.5 MB)

Repeat step 1 of this document. It should direct you to step 3 below.

□ 3. Create Files:

You may any text editor with which you are familiar, such as **vi** or **Xcode**. If you do not have a preferred editor, you can download the **JNotePad app** from the Robert Burns' Class Website (**http://cs.dvc.edu**) using then "**as Mac app**" link under the heading "**Free Resources**":



Free Resources
JNotePad 2 Code Editor | as Mac app | version history...
Online compilers | Windows C++ | Java SE JDK compiler

Save it to your Applications folder or top your desktop, as you prefer – double-click it to start the app.

Type the Code for the Program

Type your program in the editor's window (Start with File->New in JNotepad). 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. Note that "`return 0;`" is *not* required in any C++ main function.

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

int main()
{
  cout << "Hello, World" << endl;
}
```

(that's end-el, not end-one)

Save your work to your *user folder* (the one with the house icon in Finder) 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 Using the Terminal App:

In Terminal, 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
```

Sample Session:

```
Last login: Sat Aug 13 08:53:55 on ttyp2
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 command to *run* the program is its executable's name, preceded by dot-slash

the output

command to *end* the Terminal session

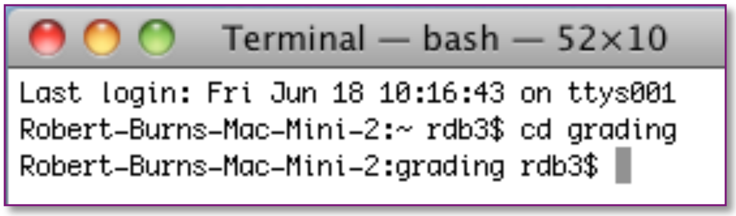
Using the Command Line Buffer ─ Minimize Typing and Typos

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.

Folder Options

If you prefer to use a separate folder for your CPP files, create a folder in your user folder with any name you prefer. Then save your files there. To compile and run, you'll need to enter the "change directory" command in Terminal, something like this, before starting your compile-and-run sessions:



```
Last login: Fri Jun 18 10:16:43 on ttys001
Robert-Burns-Mac-Mini-2:~ rdb3$ cd grading
Robert-Burns-Mac-Mini-2:grading rdb3$ 
```

In the above example, the target folder is named "grading", and all CPP and data files are stored in that folder. Note that the Terminal prompt includes the folder name.

□ 4. Compile and Run – Advanced Instructions for Multi-CPP File Projects:

Advanced programming can involve multiple CPP and H files. The full set of files for a single program consisting of multiple CPP and H files is called a "project". To use command line compiling with projects, refer to the following sample:

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.

Note that no reference is ever made to H files in compile commands – they are attached via **#include "…"** statements in the CPPs, and get compiled that way.