

Programming Design, Spring 2014

Include self-defined header files

TA: Michael Hsu

If you want to compile a C++ program that includes self-defined header files on Mac or other Linux environment. We suggest you the following two ways to do that. One way is using g++ to compile and run programs in terminal (Recommended). Another way is using Xcode to new a project (i.e., a file with filename extension .xcodeproj).

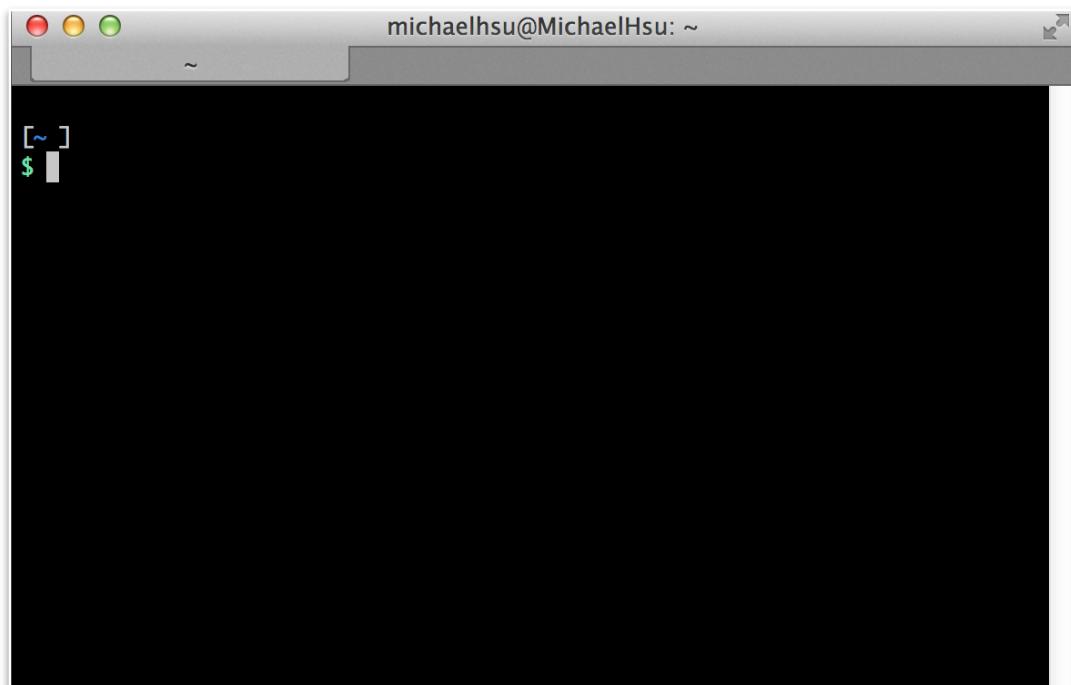
Here, we provide two *cpp* files, and one *header* file to you for demonstrating how you can run it on Mac. Just the same as windows' Dev-C++. Download example code that was mentioned in the class.

Using Commands in Terminal (Recommended)

In Figure 1, find terminal in the Spotlight results, and then click it to open a new Terminal windows (Figure 2).



▲ Figure 1: Search your Terminal on Mac.



▲ Figure 2: Terminal windows view.

Before starting, some basic knowledge you need to know is about unix-like system command-line commands. Here we list some useful commands.

First, we need to change directory (*cd path*) to the right one which your code is available under this folder. We assume that the codes are under the path

~/Downloads/project

And then we need to switch our path to the destination path of *project*.

```
$ cd Downloads/project
```

Then we type the command below to list (*ls*) what files are under the folder *project*.

```
$ ls
```

And, we can compile our C++ codes as below. The two *cpp* files will be linked together automatically (i.e., It links all the object files that are separated by a white space.). Here we will get one executable file (e.g., *run*). *[-o filename]* is an argument of output file name.

```
$ g++ main.cpp myMax.cpp -o run
```

Finally, execute *[./program]* the program, and the results will be printed on the Terminal windows. The above steps are shown in Figure 3.

```
$ ./run
```

The screenshot shows a terminal window with the following session:

```
michaelhsu@MichaelHsu: ~/Downloads/project
..loads/project
[~] $ cd Downloads/project
[~/Downloads/project] $ ls
main.cpp      myMax.cpp      myMax.h
[~/Downloads/project] $ g++ main.cpp myMax.cpp -o run
[~/Downloads/project] $ ls
main.cpp      myMax.cpp      myMax.h      run
[~/Downloads/project] $ ./run
9
```

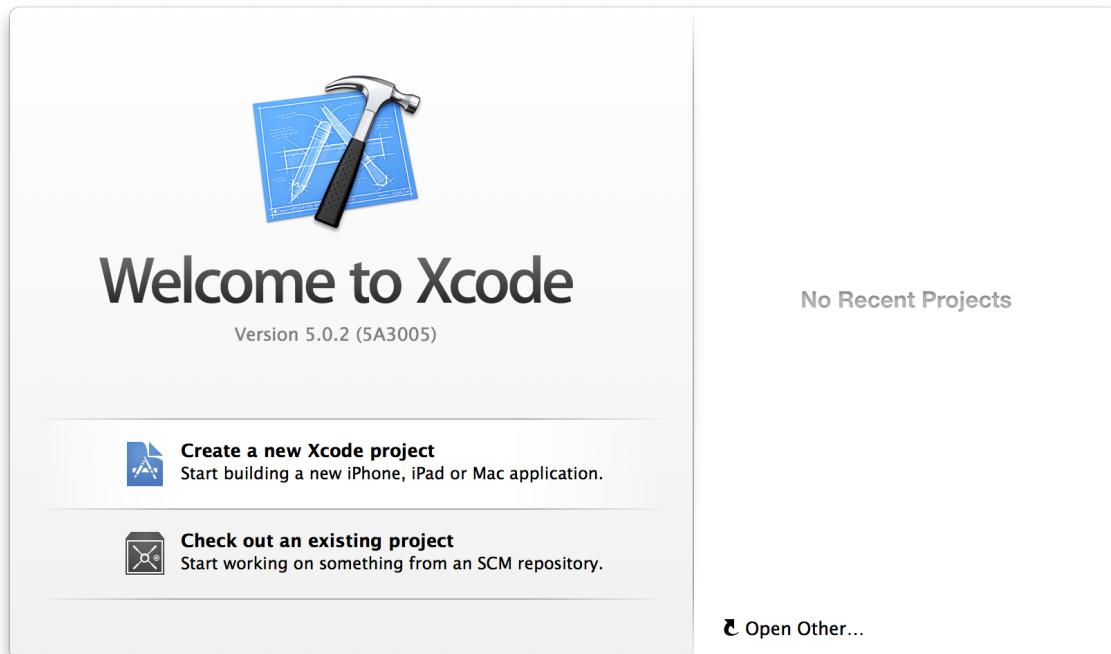
Annotations with callouts explain the steps:

- "Change directory." points to the `cd Downloads/project` command.
- "List files under it." points to the `ls` command showing files: `main.cpp`, `myMax.cpp`, `myMax.h`.
- "Compile the c++ program." points to the `g++ main.cpp myMax.cpp -o run` command.
- "The blue words are current path." points to the blue text in the prompt, indicating the current working directory: `[~/Downloads/project]`.
- "Execute and Results." points to the `./run` command and its output, which is the number `9`.

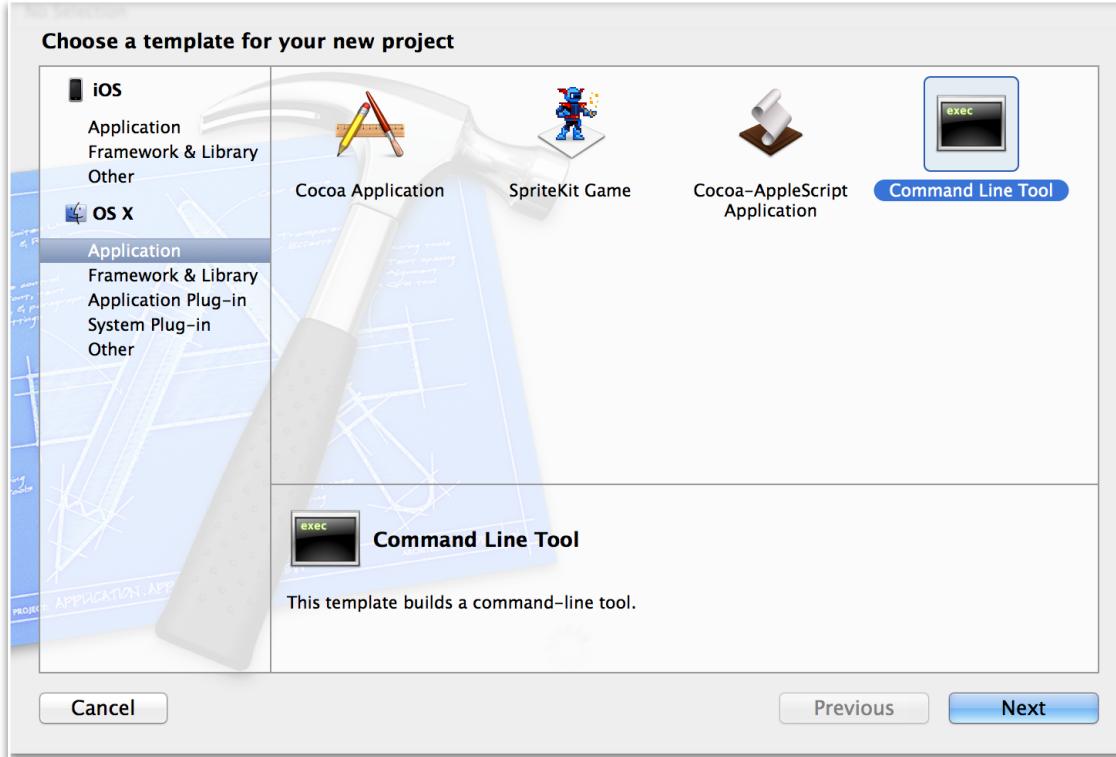
▲ Figure 3: Screenshot of final results.

Using Xcode

We use the Xcode in Mac which is almost the same as windows' Dev-C++. In Figure 4, we create a new Xcode project, and then select the **OSX > Application > Command Line Tool** option (Figure 5).



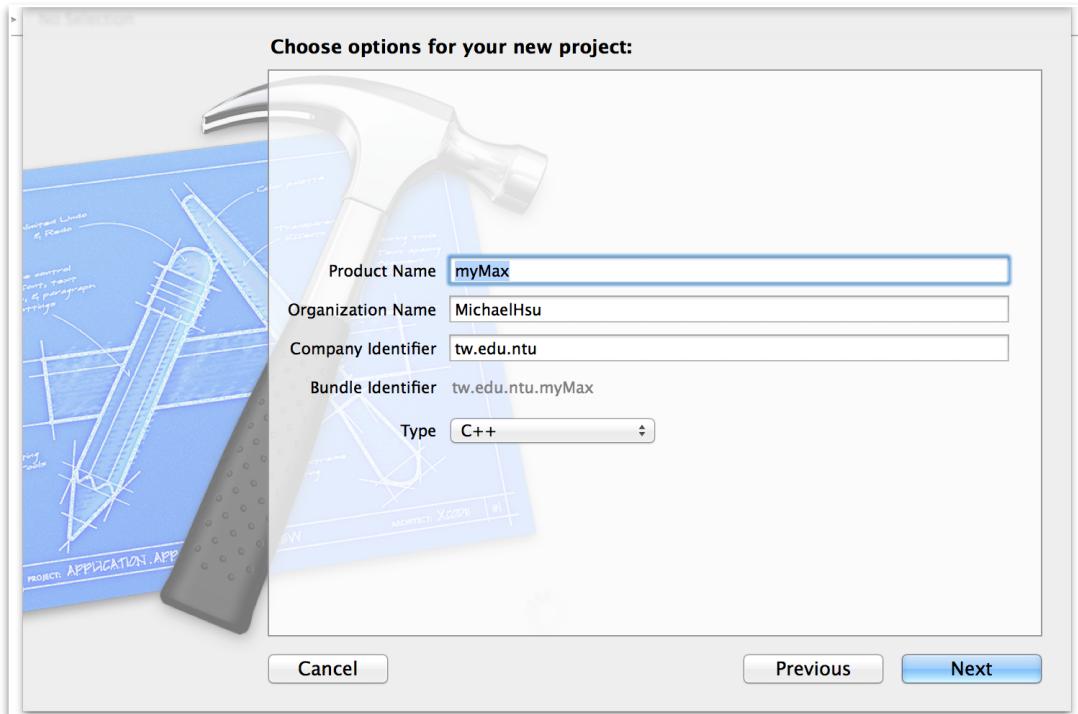
▲ Figure 4: Create a new Xcode project.



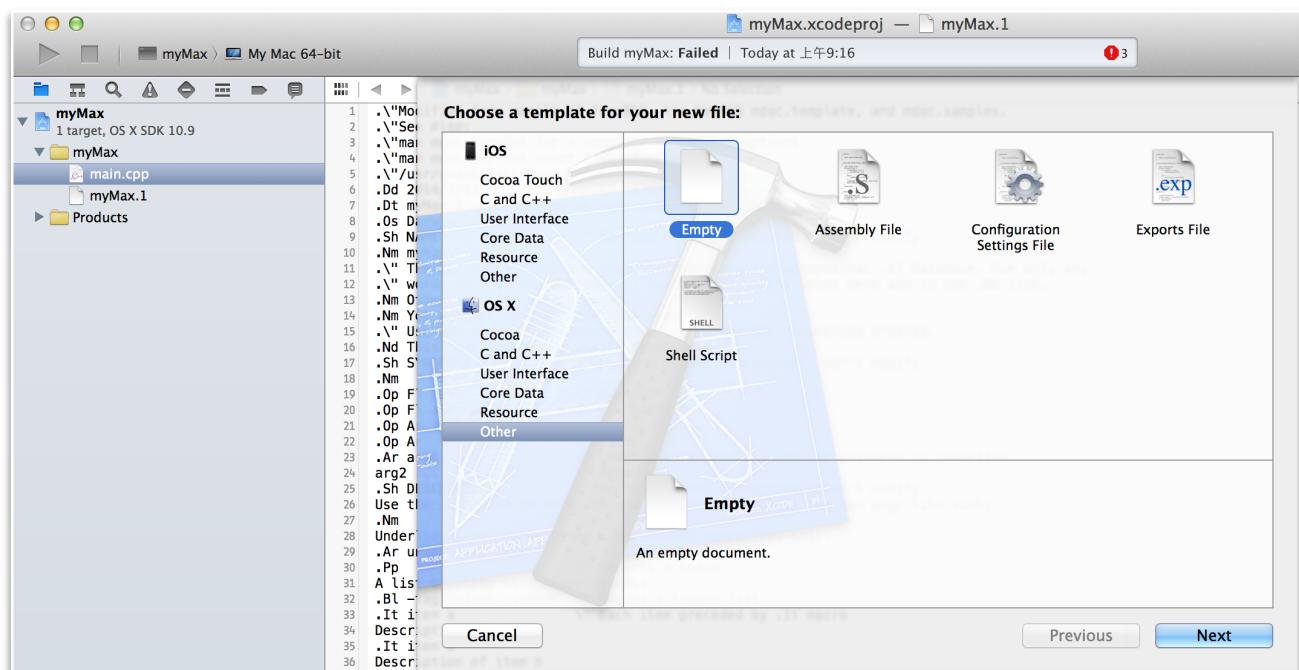
▲ Figure 5: **OSX > Application > Command Line Tool**

C++ Tutorial for the Mac

In Figure 6, you need to name the product first, and keep the product type as C++ (of course). Then, we put all of the downloaded source codes in the project, but we need some tips to do that. Now, create two empty files manually (**File > new > File**), and those will be your *cpp* file and *header* file (Figure 7 and Figure 8).

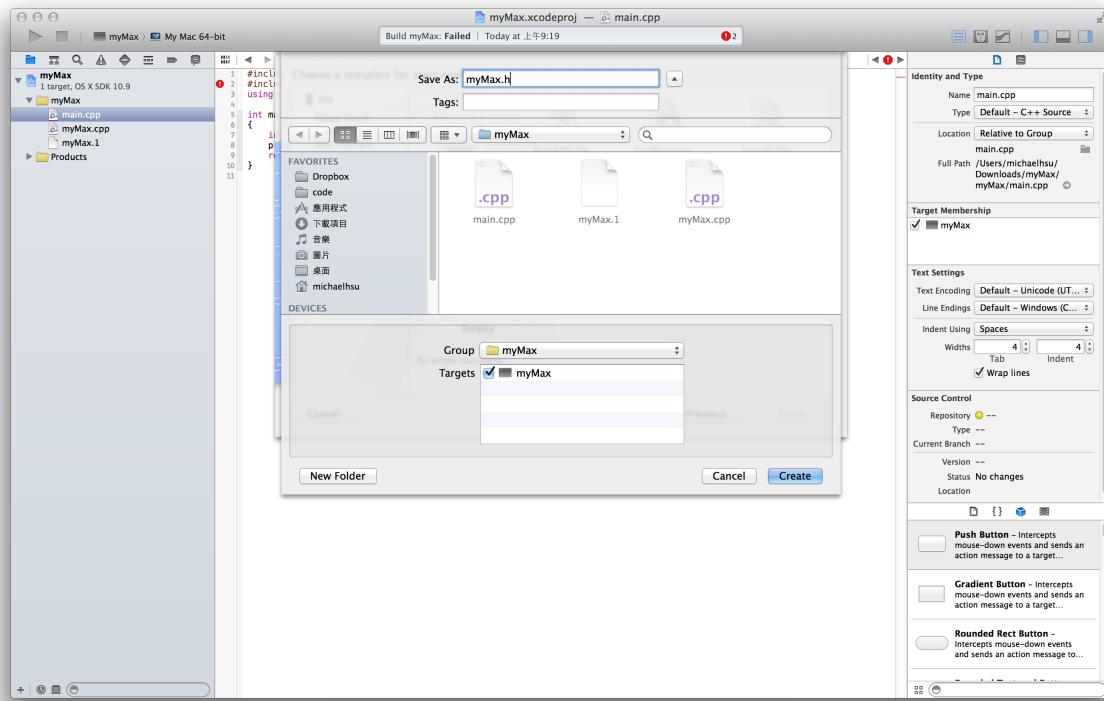


▲ Figure 6: Name the Product.



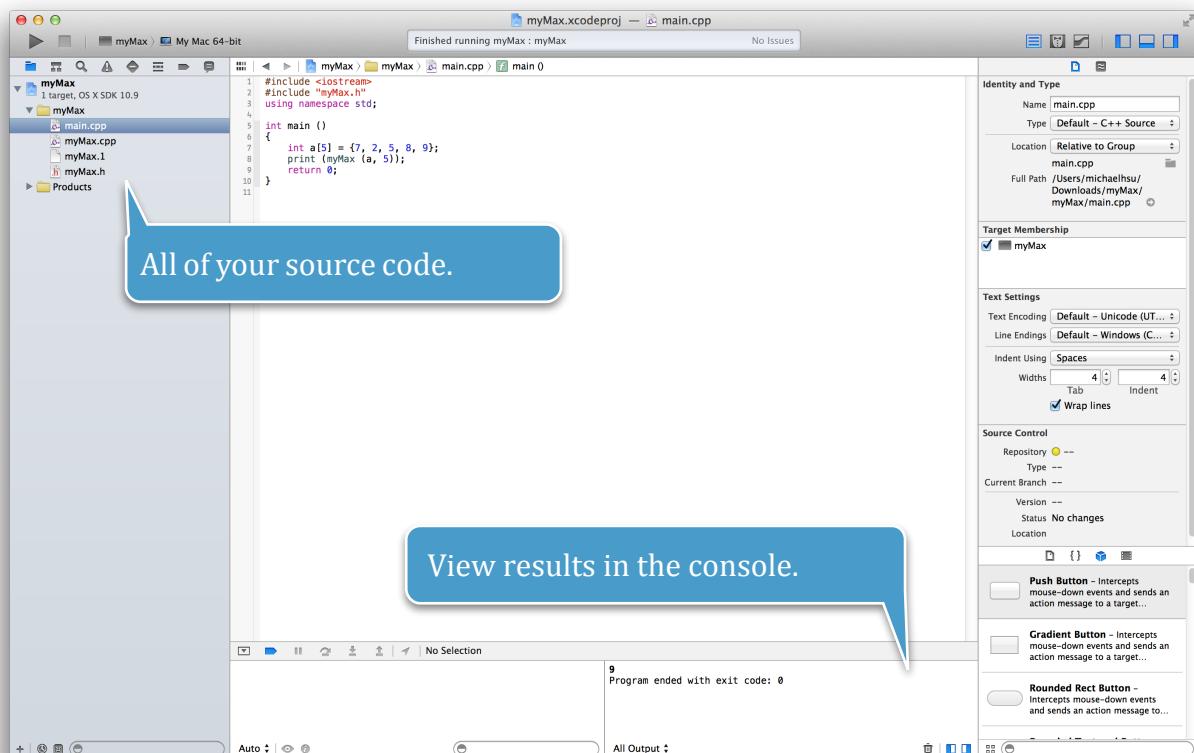
▲ Figure 7: Create two empty files manually.

C++ Tutorial for the Mac



▲ Figure 8: Those are your *cpp* file and *header* file.

Finally, we can execute your project, and we will get the results in the console window in the bottom of Xcode (Figure 9).



▲ Figure 9: Execute Project.