Xcode Notes for EECS 381

Revised 9/6/2019

David Kieras, University of Michigan

Current Version

At the time of this revision, Xcode 10.x on Mac OS X 10.14.x appears to work correctly, and is the recommended version. However, you should not update your programming tools in the middle of a project unless it is absolutely essential! And before you update, prepare a fall-back so if something is wrong, you can easily get back to where you were easily. For example, a complete backup of your system before updating is a good idea. At least, make a backup of the tool you are using before the update, or find out whether you can download the previous version and install it in place of the new version.

Introduction

Mac OS X is based on Unix, and if you download and install the free developer tools package, you get a complete Unix GNU toolset for programming - open a Terminal window, and start using makefiles! You can also use the Xcode IDE, with its excellent project management support, great editing environment, and valuable graphical debugger.

However, like MS's Visual Studio, Xcode is intended to support complex application development, making it at least as confusing as Visual Studio for the relatively simple projects in this course. Like all such tools, there is a learning curve to scramble up, with many details to learn.

These notes focus only on things which are specific to setting up Xcode for use in this course; it is not a tutorial on using Xcode as a whole.

Which Version to Use

For C projects, pretty much any version of Xcode will work fine, especially since we will be using a restricted version of the C99 Standard, which is quite old.

For C++ projects, it is essential to be able to use C++11/14/17 features, so use the latest recommended version of Xcode and Mac OS. This recipe assumes the use of the LLVM Clang compiler and LLVM C++ Standard Library; both are excellent implementations.

Setting up an Xcode Standard C or C++ Project

1. Start Xcode and set Preferences.
   - First, I recommend going into Preferences, Locations, pick the Locations tab, and set Derived Data to Relative. This makes the folder "DerivedData" appear in your project or workspace folder. This makes it easier to find your project executable — it will be in your project directory as DerivedData/ProjectName/Build/Products/Debug or Release.

- Files that need to be read by the executable can be placed in the same directory as the executable, and files written by the executable will by default appear in that directory as well. This is true whether you run the program in Xcode or from a Terminal app Unix shell window.
- Directly running the executable in a Unix shell can be convenient for testing purposes. You can drag it out to a convenient directory for testing via the Terminal app, or in Terminal, drag the Debug or Release directory into a cd command to change to that directory.
- Note that you can manually clean up the project or reduce its disk footprint by deleting the DerivedData folder for a project — it will get recreated when you open the project and build it again. Be sure to keep a copy of any input files you have placed in the Debug or Release directory.

2. Create a new project.
- Select File/New/Project
- Select the project template for macOS/Application/Command Line Tool.
- Give it a good name, e.g. "project1" — this will be the name of the project folder, the source code file within that folder, and the executable default name.
- Set the Language to be C or C++.
- Navigate to desired location, select New Folder if needed, but the project will already get its own folder.
- Note that here you can choose to create a Git repository local to your Mac.

3. Fix the project Build Settings.
The default settings unfortunately do not work well for doing the relatively simple but Standard-based programming that we need for this course. In the project window, in the sidebar, click on click on project icon, then select Build Settings and then with "All" and "Combined" view. Then select or confirm the following settings. If a setting is not mentioned, its defaults should be good. Any settings related to asm, Codewarrior, Pascal, Swift, or Objective-C can be set to No or Off - this is optional; no ill effects should appear if you leave these at their defaults.

Under Architectures:
- Select Standard Architectures or Native Architecture of Build Machine (use the selecting list under the tiny buttons on the right).
- Base SDK: macOS
- Build Active Architecture Only: yes

Under Apple Clang - Code Generation:
- Optimization Level: I recommend None for Debug, Fastest, Smallest for Release.

Under Apple Clang - Language:
- C Language Dialect: C99

Under Apple Clang Language - C++
- C++ Language Dialect - C++17 [-std=c++17]
- C++ Standard Library - libc++ (LLVM C++ standard library with C++11 support) - works for C++17 as well.

Apple Clang - Language - Modules:

- If this is a C project, set all of these to No to avoid some confusing name conflicts. Here, "module" is mac-ese for this automatic mechanism, and is related to, but different from, how "module" is used in this course. (There has been a defect in the C project template, probably a spill-over from the Objective-C project template) that causes some macOS application headers to be automatically included, and libraries automatically added to the build, which can cause name conflicts.) The problem does nor arise if the C++ Language was selected.

Apple Clang Warning Policies:

- Pedantic warnings - yes.

Apple Clang Warnings:

- For C projects especially, Missing Function Prototypes - yes
- The defaults are pretty good, but you can turn on other warnings as you wish; I suggest the following: Check switch statements, mismatched return type, missing braces, missing newline at end of file, sign comparison, unused variables.

5. The "create a new project" will normally produce a "hello world" project main function file. You can do a build and a run to check whether everything seems to work. Then select and delete the hello world files.

6. Then start adding your own files to the project. To create a new file, choose File/New/File … which will open a template chooser in which you can select a header file, a C file, or a C++ file. If you select a C or C++ file, it will give you the option of also creating a header file with the same name. The new files will be added to the project, ready for you to start editing.

However, be sure to correct Apple's dubious choice of contents for the header file (the project file templates apparently weren't prepared by mainstream C/C++ programmers!). Also change the extension of a C++ header file from .hpp (preferred by some people) to .h (preferred by many C++ wizards, and used in this course throughout). Correct the include guard symbol — it violates the coding standards for the course. Also delete their misguided #include of <stdio.h>, or <iostream> which should almost never be in a header file (See C++ Header File Guidelines handout). Including these in a new header by default is misguided.

If you already have the files, drag them into the project folder into the folder where the source files go (confusingly, with the same name as the project), and then select File/Add files to project and select the files to be added to the project.

To remove a file from the project, select it in the sidebar window and hit delete; you can choose whether to keep the file but remove the reference to the file from the project, or to trash it. Normally you only want to remove the reference because this is an easy way to change the top-level module of a project for testing purposes.

I find it useful to keep the file structure on disk parallel to the structure shown in the project sidebar window. Keeping all the sources in a single directory by themselves makes it easier to upload to my CAEN account later.

7. IDEs are very complex, and sometimes get confused. So, if during your work, the situation appears confused, try doing a Product/Clean to discard the compiler outputs, and then do a rebuild to recompile everything. If things still seem to be confused, quit Xcode, delete the Derived Data directory, and reopen the project. If things seem to hopelessly confused, quit Xcode, set aside your source code files, trash the entire project folder, and start over with creating a new project with the settings that you want.