Section 1 Lesson 0

Section 1 Lesson 0 – MinGW setup

I How to get started

This lesson shows how we download and set up our tools, describe the course, and discuss design philosophy. Please read the three essays included with this lesson before you continue. They will describe what we will be covering and why I chose the tools to use for them. Once you are done reading those essays this lesson will show you how to get ready for all the subsequent lessons of this course.

Essays in the S1L0_MinGW folder:

1) Introduction

2) Beginnings

3) Why command-line tools

II How to install MinGW

After you have installed MinGW you can use the command prompt to do a lot of work for you. MinGW doesn't block any normal Windows behavior. Rather it is an extension which lets you use gcc and a few handy tools for file manipulation and navigation. You could load many more Linux features but the basics are enough for now.

Go to http://www.mingw.org/

Now scroll down until you see Downloads: on the left. From there you can download the installer at https://osdn.net/projects/mingw/releases/ and start the download. I use the 32 bit file but one day will move on to the 64 bit version.

Once you have MinGW loaded you will have to modify your PATH variable so the Windows operating system (OS) can find the new tools. PATH is an environment variable which tells the OS where to look when you type a command at a prompt. I test a lot of code but I don't always remove it when I am done. This makes my path variable very long. To handle this problem I copy the entire path string and paste it into an editor, add what I need, then paste the longer string back again.

You will need to look at your path environment variable now and then to figure out why something is not linking or compiling. You probably don't have the include files or the libraries in a location which

the path variable includes. Under the Windows OS, at a command prompt, type: **path > path.txt** and you will have a current record of which directories can be found along your search path.

To locate your PATH variable hit the start button (I am using Win7 so your directions may be a little different). Type **path** in the search window to the left of the Shut down button. Click on the second choice, Edit the system environment variables, and a dialog window will open. Click on the Advanced tab and then on Environment Variables… button in the lower right. A new dialog box will open with your user variables in the top window and system variables in the lower one. In the topmost window scroll down until you see Path. Click on it and then on Edit… Another dialog box opens. Copy everything in the Variable value: window and paste it into your text editor.

I like to look at what the command-line path says first so I keep it in my editor window. Usually you only need to add what you want at the end. Make sure of the semicolons though. Here is what my PATH variable holds :

```
PATH=C:\ProgramData\Oracle\Java\javapath;C:\Windows\system32;C:\Windows;C:\
Windows\System32\Wbem;C:\Windows\System32\WindowsPowerShell\v1.0\;C:\MinGW\
bin;C:\MinGW\mingw32\bin;C:\MinGW\msys\1.0\bin;
```

Work through the string cutting it at each semicolon. Like this:

PATH =

```
C:\ProgramData\Oracle\Java\javapath;
C:\Windows\system32;
C:\Windows;
C:\Windows\System32\Wbem;
C:\Windows\System32\WindowsPowerShell\v1.0\;
```

and what I have added for MinGW

```
C:\MinGW\bin;
C:\MinGW\mingw32\bin;
C:\MinGW\msys\1.0\bin;
```

Your OS searches these folders for the new tools. You will need to add those last three choices to your path. If you placed MinGW somewhere other than at C:\MinGW you will have to modify your PATH to reflect this change. Otherwise, your OS will not be able to find the MinGW tools. You will have to open a new command prompt window after you have edited your path variable. Any command prompt windows which were open previously won't ‘see’ the changes you have just made. Something you need to remember.

Now that you have MinGW up and running you can try a few new tricks.

type: **dir > files.txt** at a command prompt

then: **cat files.txt**

I know this is a roundabout way to display the directory but you just created a text file called files.txt. That lets you manipulate it however you want. Searches, sorts, report creation, etc. The greater than symbol means open a file (or create it if necessary) and then stream the output directly into the text file. You have now gained the power of the Unix OS. You can chain commands together with pipes (the | symbol denotes this). By adding the MinGW tools you now have a much more powerful command-line interface. You have 'merged' Windows and Unix commands for your convenience. It really helps things work better and faster.

I am more used to typing **ls** than I am **dir.** **ls** is the list command from Unix. **ls -l** is the longer form of the command. There are dozens of options denoted by the flags (-l for instance) but **ls** and **ls -l** will do for 99% of life. **ls** is also easier to type than **dir** :)

III Choose your editor

Text editors are not the same as word processing applications, such as LibreOffice Write or Microsoft Word. They will not wrap text at the end of a line. They use monofonts which require the same space for every character in the alphabet. In this course of study you will have C and C++ source and header files, as well as makefiles and resource files. You can choose to have your editor highlight key words for you. I have set mine to format correctly with little assistance from me.

Text editors come in handy when you are unsure of the format of a file. With the editor you can simply look at it. If it is a binary file you will see a lot of strange characters and some glitchy bits of blinking and jumping about. But there are many files which are not labeled as text which can be read by a text editor. Especially data files. By examining those you can find the patterns so you can write a program to read the files and manipulate the data. I did this with some digital elevation model files from the U. S. Geological Survey. I converted those numbers into a gray scale with topographic lines to map the local terrain.

However, there is no need for a fancy editor if you are working on a computer foreign to you. Under Windows you can use MS Notepad. WordPad will not work the same because it adds all sorts of formatting instructions as a header. You truly want the simplest editor you can get, a text editor. You see every character you type, as well as the whitespace characters if you desire. In makefiles the tab character is expected so being able to see it is helpful.

I like my editor to be available quickly so I pick simple tools which load rapidly. However, you really don’t need any editor or even a computer to write code. I have written many lines of code on a pad of paper while sitting by a stream. I find I rewrite my code more often that way, as if I was editing prose. Factoring my code into small, reusable functions happens naturally because I don’t want to rewrite the same thing over and over again.

Often I just put arrows on the paper pointing at a function. That way I can thread the thoughts together into a program. A habit I picked up from writing code in Forth. Remember to carry paper and pencil for when there are no computers around. Writing down the pseudo code floating in your head helps you expand those ideas. Once you start thinking in modules you can work at a higher level, merging those modules into a variety of applications. When you have your ideas roughed out on paper you can easily transfer them to an electronic form.

A white board is also a good tool. Start from pseudo code of an algorithm you found and progress to a detailed outline for a program. The medium is easy to erase and rewrite if you can stand the fumes :) A black board served the same purpose for years. Dad had one in his workshop for keeping track of his designs. We bought it at an auction from one of the local one-room school houses. Real slate is a joy to use. Once again the medium is easy to erase and rewrite only it has no smell, just dusty.

My list of choices while running under Windows:

PSPad, Eclipse, Notepad+, MS Notepad

While running under Linux:

Geany, gedit, Eclipse, Sublime Text 2, nano, vim, etc.

When you don't have access to a computer:

Blackboard, whiteboard, paper and pencil

IV How to install PSPad

Go to the download page, https://www.pspad.com/en/download.htm , and grab the Portable (zip) file. Unzip the file and install the code. Follow the directions and put it where you normally add applications to your computer. This is normally the C: drive.

PSPad setup

Settings menu | Program Settings... | Editor (part 1) or Editor (part 2)

Editor (part 1) :

I have scroll past EOL, EOF, and Trim Trailing Spaces all checked.

Check Auto Indent Mode, Extended HOME Key, and Extended END key

Also check Highlight Matching Brackets, Completion of chars ({[<"' , and Smart chars completion.

Editor (part 2)

Check Real Tabs

Set Tab Width to 3

and leave the rest of the settings as they are.

These settings will allow you to write code in the Whitesmiths style pretty easily.

There are many features in PSPad I do not use. But I do use the Projects features. The rightmost entry on the toolbar is a cube. If you click on that cube you will toggle the project window on the left of the display area. If you click on the Projects menu instead you will get a list of commands you can use to create, edit, or manipulate your projects. I have included project files for each lesson (.ppr files). When you open your lesson folder you will find the .ppr file. Double click on it and PSPad will open with all of the files in the project ready to edit.

If you want to create your own project you can use the following recipe. Click on the cube to open the project window if it is not already visible. Open all of the files you want in your project in the edit window. I just noticed I had not created the S1L1 project so we can start there as an example. I opened lesson1.c and the makefile in the editor. The project window now reads New project with a folder icon beneath it. Right click on the folder icon and choose Add all open files. Now the files are in your unnamed project. Let's fix that by right clicking on New project. Choose Rename and type **lesson1**. Now go to the File menu and choose Save As… Navigate to S1L1_new and save your new project as lesson1.ppr (you really only need to type **lesson1** the .ppr part gets added automatically). You now have lesson1.ppr ready for your next editing session. If you click on the cube icon you can hide the project window. That way you can use the entire space for editing your code, makefile, and resources.

How to install Eclipse

Go to https://www.eclipse.org/ and click on the Orange Download button in the upper right of the window. The download site will offer you the best fit for your computer and operating system. Currently I am running under Ubuntu Linux so the site is offering me the 64 bit Linux version of Eclipse. When you are running under Windows the site will offer you the proper version for your computer, be it a 32 or a 64 bit machine. Download the compressed file and extract it to a convenient location. Then follow the directions as you install your copy of Eclipse.

Eclipse Setup

While in Eclipse go to the Window menu and choose Preferences which opens an extensive dialog box of many possibilities. However, I am only interested in the editor settings for now. Click on Text Editors (Under General) and set the tab width to 3. Now left click the triangle next to C/C++. Then scan down the choices to Code Style and left click on its triangle. Left click Formatter and you are presented with a dialog which allows you to set your formatting preferences to Whitesmiths.

I open Whitesmiths [built-in] and edit it. Open the Indentation tab and change the indentation size and the tab size both to 3.

Check the following boxes under Indent and clear the others.

'public', 'protected', 'private' within body class

Declarations relative to 'public', 'protected', 'private'

Statements within 'switch' body

Statements within 'case' body

'break' statements.

Open the other tabs if you like so you can see how they are set.

Save your new profile as Whitesmiths3 (for instance). I want to make sure I keep the reference to the original but give it a designator showing it has been changed by me.

As with PSPad there are many other features which I do not use. Eclipse requires you to use a workspace. However, I have been able to use the editor on any folder I like by following this script. Click on the File menu and choose Open Projects from File System… which opens a dialog you can use to navigate to your folder and import it. Your directory needs to have all the files you need to build your app just like the .ppr file does under PSPad. Once you navigate to that folder click on the Finish button in the lower right of the dialog box. Once you have imported the folder you can use Eclipse to modify it. You haven't really moved anything but now you can use the Eclipse editor to modify things. I find this is quite valuable when I download some terribly formatted code. Just import the ugliness into Eclipse, type ctl-shift-f and it instantly is formatted using Whitesmiths3 style. This feature alone pays for all of the time it took to find the preference file and fix it. Wow! The really nice thing is you can give someone else the same code in their favorite formatting style.

One thing Eclipse makes difficult is creating a new file. Once you have any sort of file Eclipse can work on it fine, but creating one where you want it is hard. Eclipse wants you to use that workspace to

encompass everything. So I avoid creating anything with Eclipse. I will use Notepad or nano to create an unformatted file. Once you have yourfile.c you can open it with Eclipse, format it, and edit it as you please. You were able to create the new file, in the folder of your choice, next to all the other files in your project. It is best you create all your new files in your folder of choice before you import it into the Eclipse environment. While you can edit and format them using drag and drop they won't be in a project. When you are done creating new files use the File | Open Projects from the File System… option. But if you are in a hurry just drag and drop your new file onto the Eclipse window and start editing. However, Eclipse whines much less if you use the Open Project option.

cmake or make?

Some Windows operating systems do not have make installed. They have cmake instead. It seems to be a superset of make. Before I could use a makefile I had to modify my Windows box. My solution was simple: I made a copy of cmake and saved it as make in the same directory. Now when I type **make** in my command prompt window I run my makefile. Not very complicated but it does what I want, when I want it. Control and freedom.

V Assignments

1) Read all the essays in the S1L0 folder.
2) Install MinGW and get it running.
3) Create a few folders with **mkdir** and navigate them using **cd**.
4) Choose an editor, install it, and then write some files with it.
5) Once you have created a few files list them with **ls** or **ls -l**
6) When you are ready, open S1L1_new and proceed with the next lesson.

VI Links

http://mingw.org/

https://osdn.net/projects/mingw/downloads/68260/mingw-get-setup.exe/

http://www.mingw.org/wiki/InstallationHOWTOforMinGW

http://www.mingw.org/wiki/Getting_Started

http://www.cs.colby.edu/maxwell/courses/tutorials/maketutor/

https://www.pspad.com/en/download.htm

http://www.transmissionzero.co.uk/computing/win32-apps-with-mingw/

https://en.wikipedia.org/wiki/Indentation_style#Whitesmiths_style

http://www.terminally-incoherent.com/blog/2009/04/10/the-only-correct-indent-style/

http://www.learn-c.org

https://www.learnpython.org/