

## ECE 1574 Lab Assignment 2 Introduction to Cygwin, the GNU compiler, and C++ programming

CRNs 11720-11727

### Introduction

In this lab, you will create and edit programs, organize files and directories, and examine the storage and representation of data types on the computer.

### Creating and editing programs

Please follow the tutorial that gives directions on how to setup your cygwin environment to allow command line invocation of Textpad. In cygwin, execute the command  
textpad main.cpp & <enter>

This should cause textpad to execute and it will ask you if it is okay to create main.cpp. Tell it yes. At this point, you have a cygwin window open as well as a textpad window open. You can now enter a program in textpad. Start with the following simple program:

```
#include <stdio.h>
int main()
{
    printf("Hello\n");
    return(0);
}
```

Save the program and quit textpad. To edit an already created file, you can invoke textpad in exactly the same way

textpad main.cpp & <enter>  
and that file will be ready for editing in textpad.

*Note: Why is there an "&" at the end of the line? This tells cygwin to execute a command "in the background," but still be ready to execute other commands while that command is running.*

*Note: When you followed the steps in the tutorial, you were changing "environment variables." Environment variables are used in Windows, cygwin, and linux to customize the user environment; most users are not aware of them. The environment variable we changed, PATH, tells cygwin what directories to search for a command that you issue. This allows the computer to not have to search every darn directory every time you issue a command.*

### Organize Files and Directories

Please refer to the online useful commands in cygwin (unix commands). We'll use some of those commands to organize your directory structure. Directories (termed folders in Windows) are simple containers for files and other directories. In lab 1, you created a directory called lab1. It is generally a good idea to logically organize directories to make finding things more manageable. Create a directory called "labs" by issuing the command

`mkdir labs.`

Now move into that directory by issuing the command

`cd labs`

Next, move the lab1 directory into labs

`mv ../lab1 .`

The use of “..” refers to “the directory above the one I’m currently in” and the use of “.” refers to “the directory I am current in.”

Now create a directory called lab2

`mkdir lab2`

and then move main.cpp into it

`mv ../main.cpp lab2`

Now move into the lab2 directory

`cd lab2`

Now copy the makefile from lab1 into this directory

`cp ../lab1/makefile .`

*Note: What the heck is this cygwin thing anyway? It is a windows program that provides an interface to windows to make it behave like Linux/Unix. This allows the use of many of the tools that have been developed in the Linux/Unix world without requiring us to install and run a second operating system.*

*Note: You can run multiple cygwin windows if you find it makes life easier.*

### Time to Program

First, we need to edit the makefile so that the “source” file is main.cpp instead of lab1.cpp. To do that, start textpad

`textpad makefile &`

Find and change anything that says “lab1” to “main” and then save & quit.

Now when you run make, it should create an executable “main” that will print out “hello” when you run it.

Now, start to edit main.cpp using textpad. You can leave textpad running and still compile programs using make on the command line. You just need to make sure that you save the program before running make. Your program should perform the steps below in the order given below (delete the original “hello” printf and leave the “return” statement at the end of the program):

- (1) Declare variables of the following data types: character, unsigned integer, and floating point.
- (2) Assign a value of 0 to each of these variables (even the character).
- (3) Print each of these variables using the following printf statements:  
`printf(“Character is %c\n”,your_character_variable_name_goes_here);`  
`printf(“Unsigned Integer is %d\n”,your_unsigned_int_name_goes_here);`  
`printf(“Floating point is %f\n”,your_floating_point_name_goes_here);`
- (4) Assign a value of ‘k’ to the character variable.
- (5) Assign the value in the character variable to the unsigned integer.
- (6) Assign the value in the integer to floating point value.

- (7) Print all of the variables again.
- (8) Now print out the hexadecimal representation of the unsigned integer using  
`printf("Hexadecimal value is %x\n",your_unsigned_int_name_goes_here);`

---

Please show the GTA your work so that he/she can validate that you have created the appropriate directories and that your program executes properly. Explain to the TA why the print statements print the values that they print. This lab (and all further labs) must be validated during your lab section.

---

Your Name

---

Date

---

GTA name / Signature