Lab Assignment 2: Simple I/O and Calculations

Due: Friday, September 14, 11:59 PM

Description:

The purpose of this assignment is to write code that will prompt the user to enter two numbers, make a calculation, and display the results. You will also create a Makefile that you will use to compile your program. The program for this lab will calculate the Wind Chill Index (T_{wc}), given an air temperature (T_a) in degrees Fahrenheit, and a wind speed (V) in miles per hour.

Preparation:

- Become familiar with the CS 262 Programming Style Guide (found under Course Content on Blackboard). Note the example on how you should read input values into your program.
- Read the man pages for fgets(), sscanf() and printf() noting how format strings are used in each function. Pay careful attention to how floating point values are printed with printf() and how to create a reference to a variable when using sscanf() (use the & before the variable that is to contain the input).
- Read the man page for the pow() function. Note that you will need to add an additional #include file (*math.h*), besides the normal *stdio.h* and *stdlib.h* header files. You will also need to add a link option (-lm) to your

gcc compile command.

- You may also wish to read about how the wind chill index is calculated. Some information about this can be found on Wikipedia: https://en.wikipedia.org/wiki/Wind_chill

Instructions:

Make a lab2 directory in your CS262 directory on zeus. Don't forget to use the proper naming conventions. Refer to the instructions from Lab 1 if necessary. The source file for this assignment will be named Lab2_<username>_<labsection>.c and will include the required header information at the top:

```
// <Your Name and G Number>
// CS 262, Lab Section <Lab Section Number>
// Lab 2
```

Your program will prompt the user to enter two values. The first value will be a temperature in degrees Fahrenheit (T_a), and will be of type int. The second value will be the wind speed in miles per hour (V), and will be of type double. You will use the following formula to calculate the Wind Chill Index (T_{wc}) which will be of type double:

$$T_{wc} = 35.74 + 0.6215T_a - 35.75V^{+0.16} + 0.4275T_aV^{+0.16}$$

Once you have calculated the Wind Chill Index, you will then display the temperature, wind speed and wind chill using descriptive statements such as:

Temperature: 38 degrees Fahrenheit
Wind Speed: 17.4 mph
Wind Chill Index: 28.6 degrees Fahrenheit

Floating point numbers should be output to one decimal place. Also, values of a magnitude less than 1 should display a leading 0 (ex. 0.7). Even though the temperature will be read as type int and wind speed will be read as a type double, the user may or may not enter values with a decimal point. In other words, expect the user to enter values such as 25, 25.0 and 25.35 for both temperature and wind speed.

Note that wind chill calculations only work correctly for temperatures at or below 50 degrees Fahrenheit and wind speeds at or above 3 mph. For this lab, you must check inputs to ensure that the values are within a proper range. If an improper value is entered, an error message should be printed and the program will exit (do not re-prompt the user for a correct value).

For input, you must use the fgets()/sscanf() combination of Standard I/O functions. **Do not use the scanf() or fscanf() functions.** Although using scanf() for retrieving numerical data is convenient, it is problematic when used in conjunction with character string input. Therefore, it is best to get into the habit of using the fgets()/sscanf() combination exclusively.

Note from the sscanf() manpage that when reading a floating point value, if the data type of the variable that will receive the value is of type *double*, you must use "%lf" (long float) instead of "%f" (float) for your format string.

You will compile your program using gcc following the compile steps introduced in Lab 1 and also create a Makefile (described below).

Your executable file will be named
Lab2_<username>_<labsection>

Testing your completed program:

After compiling, run your program to ensure that it calculates the proper values and displays the information in a logical and readable format. Check that negative temperature values also work correctly. To check your results, you may use an online Wind Chill Index calculator such as the one found at: https://www.weather.gov/epz/wxcalc_windchill

Editing a Makefile:

For this lab, you will modify the simple Makefile from Lab 1. To do this, copy the Makefile from your Lab 1 directory to your Lab 2 directory. Then, edit it in the following manner:

1. Change all instances of
   lab1_<username>_<labsection> to
   lab2_<username>_<labsection>
2. Add the following line at the top of the file (after the initial comments):

```
CFLAGS=-g -Wall
```

1. Replace the "`-g -Wall`" in the compile line (after the gcc) with $(CFLAGS)
2. Add "-lm" at the end of your compile line.

These changes basically make CFLAGS a variable where you can modify the compile line options, and it will change it wherever the variable is found in the Makefile. For this file, there is only one instance where you need the CFLAGS, but in future Makefiles, you may have several instances of the gcc command. Making the compile options a variable will make it possible to change all instances of gcc commands by changing only a single line. Also, because you are using a function from the Math library (pow()), you need to let the linker know that it will need to pull in a function from that library. The way to do this is with the "-lm" option. You will learn more about linking to libraries later in the semester.

Check your Makefile by compiling your program with the *make* command to ensure that it works properly. Also ensure that *make clean* works as well.

Submission:

You will submit a tarfile containing your source file, Makefile and a typescript file that shows a listing of your code and that it compiles and runs without errors. Follow this procedure to produce your tarfile:

1) Create a typescript file named Lab2_typescript_<username>_<labsection>. To do this, type the following command:

```
script Lab2_typescript_<username>_<labsection>
```

2) Show that you are logged onto Zeus. Type: `uname -a`

3) Show a listing of the current directory to ensure the source file and Makefile is present. (use the *ls* command)

4) Show a listing of your code. (*cat* command)

5) Show a listing of your Makefile.

5) Remove any versions of Lab2_<username>_<labsection> that may appear. (*make clean* command)

6) Compile the code using make.

7) Show that the executable file was created from the make command. (*ls* command again)

8) Run the executable using ***at a minimum*** the following pairs of input samples:

Temperature	Wind Speed	
32	10.0	
32.0	10	
32.5	10.0	(Note - the temperature should be treated as the integer 32)
-4	16.4	
55	16	(Note - program should exit with an error statement before the wind speed can be entered)
0.0	0.0	(Note - program should exit with an error statement immediately after the wind speed is entered)
0.0	15.9	
0	15.9	
-40	10	
8	4	(Result: 0.4. Tests the leading 0 for values between -1.0 and 1.0)

You may add additional input samples if you wish.

9) Remove the executable with the *make clean* command.

10) Show that the executable file is no longer present. (*ls* command again)

11) End the typescript by typing Control-d

12) Verify your typescript file is correct (use the *cat* command)

13) Create a tarfile of your Lab 2 directory. Reference the instructions from Lab 1 if necessary.

14) Copy this tarfile to your local machine.

15) Submit the tarfile to Blackboard as Lab 2.

Points to Review:

- Learn and remember the Unix commands that you need to use often in performing your work.
- Get used to using vim or vi on zeus to complete your assignments. It is generally much faster to edit your code directly on zeus than it is to edit code locally and then copy it over to zeus for compiling.
- Remember the correct syntax for compiling executables using gcc
- Also remember that the *man* command is always there to help you with particular commands or functions.

Congratulations! You have completed Lab 2.