Lab 2

Each lab will begin with a recap of last lab and a brief demonstration by the TAs for the core concepts examined in this lab. As such, this document will not serve to tell you everything the TAs will in the demo. It is highly encouraged that you ask questions and take notes.

In order to get credit for the lab, you need to be checked off by the end of lab. For non-zero labs, you can earn a maximum of 3 points for lab work completed outside of lab time, but you must finish the lab before the next lab. For extenuating circumstance, contact your lab TAs and the course instructor.

Special Exception: You may receive full points on Lab 1 if you show your work to a TA in your assigned section during week 2. This is a special exception since many students enrolled late and weren't able to show their work during the normal week 1 lab period.

Pair Programming: In part of this lab, you can choose to work in pairs. If you choose this option, you must be checked off together for the programming work**. You only need one computer for pair programming.** One person will be the driver, who controls the computer and codes, while the other is the navigator, who observes and advises. After 20 minutes, you will switch driver and navigator, continuing this pattern until the task is complete. Please read more about pair programming and the benefits: CS161 Pair Programming Handout

The quiz and Linux portion of this lab should be completed independently, but the remainder of the lab can be done using pair programming.

(2 pts) Lab Quiz from Lab #1

1. The following are some of the most commonly used Linux commands. Provide a definition for each.

   ls, cd, vim, mkdir, pwd

2. Using the above commands, outline the following tasks:
   - Create a new file, assign1.cpp, in a new directory, ~/assign/assign1. You can assume the assign directory already exists in your home (~) directory.

3. The following are some of the most common commands in vim. Provide an explanation for each.

   i, Esc, :wq, :q

(2 pts) More Linux

1. Now, open your secure shell (SSH) client and connect to:
   access.engr.oregonstate.edu

2. In Linux, every command has a manual page, which provides you information on what the command means, how to use the command, and a description of the available options for a command. You can use the **man** command to read the manual page for a command. (e.g. man Linux commands do not have spaces in them and they are lower case. This is important because **Linux is case sensitive**!!! Following a Linux command is a space followed by arguments. Some arguments may be required and some are optional such as options, which are preceded by a dash.

   linux_command –option required

   If an argument is optional in Linux, then it is enclosed in brackets, [], and required arguments do not have brackets. For example, **man ls**, and notice that everything supplied to ls is optional. You can also use the command and --help to get a brief manual page for the command, i.e. **ls --help**

3. In order to get more familiar with the Linux/UNIX environment, you will do a few Linux-type exercises at the beginning of each lab. Today, we will learn the copy, move, and remove commands, i.e. cp, mv, and rm. First, look at the manual page for each of these commands. **Remember to use the space bar, b, and q for moving around in the manual pages.
   man cp
   man mv
   man rm
4. First, let's use these new commands before moving forward. Copy your hello.cpp program from the labs/lab1 directory to your home directory.
   cp labs/lab1/hello.cpp ~
   This says, copy the hello.cpp file located in the labs/lab1 directory into my home directory. Use **ls** to list the directory contents and make sure you have a hello.cpp file in your home directory.

5. Now, rename the file to hello2.cpp by using the move command.
   mv hello.cpp hello2.cpp.
   Use **ls** to list the directory contents and make sure you no longer have a hello.cpp file and you now have a hello2.cpp file.

6. Create a test directory, and then change into that directory.
   mkdir test
   cd test

7. Copy the hello2.cpp file from your home directory to the test directory you are currently in. **Remember that **..** is up/back a directory. You could also say cp ~/hello2.cpp . because you know that hello2.cpp is in your home directory.
   cp ../hello2.cpp .
   You could also say **cp ~/hello2.cpp .** because you know that hello2.cpp is in your home directory.

8. Now, go back to your home directory or up/back a directory, and remove the file hello2.cpp file in your home directory and remove the test directory and its contents, which contains the file hello2.cpp. Use ls to make sure you see the hello2.cpp file and the test directory in your home directory.
   cd ..
   ls
   rm hello2.cpp (if prompted press n so you don't remove it)
   rm –f hello2.cpp (notice no prompt, -f forcefully removes without asking)
   rm test (notice it won't remove a directory, even with -f)
   rm –r test (notice the prompt, -r recursively descends into a directory to remove it and its contents, note you can use –rf together to avoid all the prompts)
   ls (you shouldn't see hello2.cpp or test)

9. Change into your labs directory, create a lab2 directory, and then change into that directory. **DO NOT use spaces in directory or file names in Linux.
   cd labs
   mkdir lab2
   cd lab2

10. There are a few shortcuts in Linux that you want to be familiar with using. One is the use of up/down arrows to move through your **history of commands**. At the shell prompt, press the up arrow and note what happens, and then press the down arrow and note what happens.

11. Another useful shortcut is **tab completion**. Go up two directories with **cd ../..**, and then let's change back into the labs directory. This time, start typing cd and l (a lowercase "L"), then press the tab key. This will complete your labs word because it is the only option in your home directory that starts with an l. Now, try changing into the lab2 directory again using tab completion, but this time you'll be presented with two options that start with an l.

A convenient Linux cheat sheet is available online at:
https://files.fosswire.com/2007/08/fwunixref.pdf
It might be useful to keep a copy of this nearby while you are learning to use Linux.

You can find more Linux and vim resources and tutorials on the **Useful Links** page of our Canvas website.

Helpful vim hint: When you are in vim escape mode, then you can use a command to show the line numbers on the left, which can be helpful for debugging. Show the line numbers in vim by typing **:set number.** **Please make sure you refer to lab 1 for a list of other helpful vim commands.

(6 pts) Prepare for Assignment #2 (this can be done in pairs)
In your assignment, you have variables, the need to use rand(), and conditionals.

Create a program called **rand_numbers.cpp**, and type the following program into the file.

```
#include <iostream>
#include <ctime>     /*included to allow time() to be used*/
#include <cstdlib>   /*include to allow rand() and srand() to be used*/

using namespace std;

int main() {
   int x;    /* variable to hold our random integer*/
   srand(time(NULL)); /*seeds random number generator. Do this just once*/

   x = rand();
   cout << "x = " << x << endl;

   x = rand();
   cout << "x = " << x << endl;

}
```

Compile and run the above program 3 times.

- **What is the result of each rand() for the different executions?**

Comment out the srand() function call. Compile and run the above program 3 times.

- **What is the result of each rand() for the different executions?**

Now add an additional srand() function call in between the two rand/cout lines of code, so there are two srand() function calls. Compile and run the above program 3 times.

- **What is the result of each rand() for the different executions?**

Edit your **rand_numbers.cpp** program so that it **chooses a random int that is in the range 0-5**. Print the random number to the screen.
Hint: I think the mod (short for modulo) operator % would be very useful here. Recall that the mod operator returns the remainder of a division operation.
What is anything mod 5 or 6? How can this help you?

Now, use this number, 0-5, to select which message to print:

- If the number is 0, print “Bummer, I’m zero!!!”
- Else if the number is odd, 1, 3 or 5, print “I’m an odd number!”

- Else it’s an even number, print “I’m an even number!”

Show your completed work to the TAs for credit. If you are working in pairs, make sure that the TA knows who the two group members are. You must separately show your individual work to the TAs (lab quiz and Linux work).