

Lab 2

Get checked off for up to 3 points of incomplete work from the previous lab within the first 10 minutes of lab.

Next, TAs will distribute a 10-minute quiz for 3 pts followed by a 30-40 minute worksheet for 2 pts. The worksheet will contain core concepts covered in this week's lab, lecture, and assignment, and the TA will go over the worksheet with the class, after you are given some time to work on the sheet in groups. After the TA goes over the worksheet, you will have a 50-60 minute hands-on coding activity for 5 pts that you can complete on your own or in pairs.

Pair Programming: In this lab, you can choose a partner for pair programming on the coding portion of the lab. **You must be checked off together, and 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: [Pair Programming Student Handout](#)

In order to get credit for the lab, you need to submit your quiz to a TA and get the worksheet and code 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 an extenuating circumstance, contact your lab TAs and Jennifer Parham-Mocello.

(1 pt) 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. 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 (when 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, after typing **cd** and **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**.

Here is a Linux and vim cheat sheet to help you reference some of these commands quickly. <http://classes.engr.oregonstate.edu/eecs/winter2019/cs161-001/labs/CheatSheet.pdf>

You can find more Linux and vim cheat sheets and tutorials on the links page of our class website: <http://classes.engr.oregonstate.edu/eecs/winter2019/cs161-001/links>

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.**

Prepare for Assignment #2

(1 pts) 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?**

(3 pts) Edit your `rand_numbers.cpp` program so that it **chooses a random int that is in the range 0-5** based on the value returned from `rand()`. I think the mod operator `%` would be very useful here. What is anything mod 5 or 6? How can this help you? **Print the random 0-5 number to the screen.**

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!"

Now, determine which odd number or even number you have by using nested if blocks.

```

if (num==0) {
    cout << "Bummer, I'm zero!!!" << endl;
}
else if ((number%2)==1) {
    cout << "I'm an odd number!" << endl;
    if (num==1) {
        cout << "I'm the number 1!" << endl;
    }
    ...
}
...

```

Show your completed work to the TAs for credit. You will not get points if you do not get checked off!