Lab 2: The Shell and more C Review

Objectives

- Learn to use the Linux command line and common commands
- Practice File IO and passing parameters from the command line

You may complete this lab individually or with one colleague from this class using pair programming rules (see the course canvas page).

1 Linux prompt and commands

Although in Linux, you may use a window manager for a GUI interface similar to Windows/MacOS, a lot of functionality is best available through the shell with a command line where one can issue direction commands to control the machine.

- A **shell** is a program that communicates with the operating system. It is an interface between the users of the operating system and the computer resources the operating system manages.
- A **terminal** is a window with a prompt at which a user can issue commands to the operating system in text. Commands types into this window are processed by the shell to be carried out by the operating system.
- A **command line** is a text prompt where you can type shell commands.

You will need to use shell commands for this course and this skill can be immensely powerful in your future computing experience. If you have learned how to operate on the shell elsewhere, then you probably know this material already. You still need to do this part of the lab and turn in the answers. You may also want to lend your expertise and help others who are new to the command line environment.

1.1 Basic Shell Commands

Here is a list of basic terms and commands you will need.

- A path is a list of folders and subfolders where something resides. For example, your Linux home directory ("directory" and "folder" are synonymous) may be located at the path `/home/hacker/`. The Windows operating system also uses paths, e.g.,

  ```
  C:\Windows\ApplicationData\Mozilla\Firefox
  ```

- A prompt is a symbol or set of characters where you type commands. By default, your prompt displays your username, the computer you are on, and your current folder so it may look similar to `hacker@system:~/cmps2300`
- The `ls` command (listing) displays the contents of the current folder.

Lab 2: The Shell and more C Review

- The `pwd` command (print working directory) displays the name and path of the current folder.
- The `cd` command (change directory) changes the current directory from one folder to another.
- The `mkdir` command (make directory) creates new folders.
- The `cp` command copies files and directories.
- The `mv` command moves and/or renames files and directories.
- The `rm` command deletes (removes) filed and directories.
- The `&` command runs another command in the background so that you can continue to use the current shell.

1.2 Practice

Begin editing the hand-in file for this lab using `vi` by entering the command vi `lab1.txt` at your prompt and hitting the Enter key.

When you start the vi program, you begin in **command mode**. To enter **insert mode**, press the "i" key. Now type your name and lab number (Lab 2) at the top of the file. Leave insert mode by pressing ESC. Save the file by pulling up a prompt in the bottom left of the screen by pressing ":". Type "w" (for write) and press enter. Likewise, you can quit by entering the "q" command in the same manner.

Number your answers in the following format:

Question 1: Here does my answer to question 1...

1 Question 1: What is your prompt? (Your actual prompt, not the definition of the term "prompt".

2 Displaying your path. When you opened the terminal, you started out in your home directory. This is the base directory for your account. Windows uses the name concept.

You can tell that you are in your home directory because your prompt shows a tilde (~) for your current folder name. The tilde is just a shortcut for "one's home directory". The actual path to your home directory is longer. To see the full path to your current directory, enter the `pwd` command at your prompt.

```
$ pwd
```

Question 2: What is the full path to your home directory?

Lab 2: The Shell and more C Review

3 Displaying the contents of a directory.

To see the contents of the current directory, enter the `ls` command at your prompt (that's a lowercase letter l in ls).

```
$ ls
```

If coloring is enabled, files and folders will be shown in different colors.

Question 3: What files and folders are inside your home directory?

4 Creating and changing directories. Let's create a folder for this course. Call it `cmps2300`.

To create the folder, issue the command

```
$ mkdir cmps2300
```

Now let's change directories into this folder.

```
$ cd cmps2300
```

Your prompt should now show that you are in the cmps2300 directory.

Question 4: What is the full path to your cmps2300 directory?

To go back up a folder, issue the command

```
$ cd ..
```

(the `cd` command followed by two dots.) You can also navigate in a folder from any other folder by specifying its path (full or shortcut). The shortcut path to your cmps2300 folder is `~/cmps2300`. To get to this folder using its path, issue the command

```
$ cd ~/cmps2300
```

You can always return to your home directory by issuing the command `cd ~` (just plain `cd` will also work.)

5 Copying, moving, and renaming files.

Files and folders can be copied, moved, and renamed as needed. Let's make a copy of your `lab1.txt` file. Navigate into your home directory. Issue the command

```
$ cp lab1.txt copy.txt.
```

Lab 2: The Shell and more C Review

Use `ls` to see that `copy.txt` now exists.

You can move files from one directory to another using the `mv` command. Let's move `copy.txt` into your `cmps2300` directory.

```
$ mv copy.txt cmps2300/
```

Verify that copy.txt has been moved by using the `ls` and `cd` commands.

You can also rename files using `mv`. In you are now there already, navigate into `~/cmps2300`. Rename `copy.txt` to `extra.txt`.

```
$ mv copy.txt extra.txt
```

Verify that the file has been renamed using `ls`.

These commands can be used locally or with paths to move a file from one directory into another. Navigate back into your home directory then issue the following command.

```
$ mv ~/cmps2300/extra.txt copy.txt
```

Notice that the file have been moved back into your home directory and renamed back to `copy.txt` (use `ls` to verify!)

Question 5: What is the command to move your `lab1.txt` into your `cmps2300` directory?

Quit editing `lab1.txt` (within vi, issue the :w and :q commands – they can be combined into a single command :wq). Then move the file, edit it again, and enter the command you used.

6 Deleting files.

The copy.txt file should currently be found in your home directory. Navigate home. Let's delete this file:

```
$ rm copy.txt
```

You can also use paths (as above) with the `rm` command.

To delete a folder, you have to recursively delete all files and subfolders within it. To test this out, first use `mkdir` to create a folder named `extra`.

```
$ mkdir extra
```

To delete it:

```
$ rm -r extra
```

1.3 Beyond the Basics

We are only scratching the surface in this lab. We haven't even introduced one of the shell's most powerful features – the pipe (look it up!). This is only one of the many powerful features that you can explore. In addition, the shell has loops, conditionals, and variables. That is, you can write computer programs directly in the shell. These shell programs are known as *shell scripts*, and with them you can perform powerful feats such as automating, maintaining, and managing whole fleets of computers.

If you ever need more information about any shell command, you can directly access it on the shell by examining its manual page. You can see the man page for `ls`:

```
$ man ls
```

Hit "q" to quit the man program.

Likewise, `$ man intro` provides an introduction to shell commands.

`$ man bash` provides all the information you could ever want about bash.

2 C programming

2.1 Hello, world!

Write a small program that calls the following two functions. Use the program to understand the differences in their behavior

```
void swap(int x, int y)
{
  int temp;
  temp = x;
  x = y;
  y = temp;
  return;
}

void swap(int *x, int *y)
{
  int temp;
  temp = *x;
```

```
  *x = *y;
  *y = temp;
  return;
}
```

Question 6: In your own words, explain how the * and & operators work.

2.2 File IO

Recall the performance statistics program from the previous week:

Design a program that lets the user analyze performance of a computing system. The user will enter the total time the system has been idle on each day of the week, in minutes. The program will store this information in an array.

The program will then display the total idle time, the performance rate of the system during this week (i.e. percentage of idle time with respect to total length of the week), the average daily idle time, and the days with the lowest and highest load.

Augment your solution to that program so that now the time in minutes is entered from file whose name is supplied as the first command-line argument to the program and the output is written to a file that is supplied as a second command-line argument to the program. Write this augmented solution in `fileio.c`.

The first line in the input file will contain the number of days stored as an integer number (so you will no longer be working with a week time period). The subsequent lines will contain measurements of idle time in minutes, stored as integer numbers, one number per line.

Submission

Upload your `lab1.txt` (with its 6 answers) and `fileio.c` to Canvas before the deadline.