CSCE 251 - Sec 700 Assignment 2 Spring 2020

Instructions Follow instructions *carefully*, failure to do so may result in points being deducted.

- Completely answer all of the questions. You must submit your homework via CSE Handin (`http://cse.unl.edu/handin`) by Friday, March 13th, at 11:59pm.
- Solutions to the written portion of this homework should be submitted as a text file. The scripts should be submitted using the names specified for each of them.
- Verify your scripts using the webgrader for this course (`cse.unl.edu/~cse251/grade`). The webgrader is a simple tool that runs all of your scripts on the CSE server and redirects their output to a single web page. **Scripts that do not run on the CSE server can earn at most half credit.**
- There are 60 points available on this assignment, but it will be graded out of 50 points.
  Question 6 is bonus.
- As always, the CSE academic integrity policy is in effect (see `http://cse.unl.edu/academic-integrity-policy`).

Question	Points	Score
1	10	
2	8	
3	5	
4	12	
5	15	
6	10	
Total:	60	

1 Written

1. [10 points] Answer each of the following assuming that the permissions setting string is `-rwxr-xr--`.

   (a) Is this a file or a directory?

   (b) Who has the most control in this case: the user, group, or others?

   (c) Who all has both read and execute permission, but doesn't have write permission?

   (d) Who all has read permission?

   (e) Who all has write permission?

2. [8 points] Answer the following questions about the **chmod** command. Give the entire command in your answer.

   (a) Using **symbolic** notation, how do you add read and write permission for the group?

   (b) Using **symbolic** notation, how would you set the permission of the user to be read and write, but no execute?

   (c) Using **numeric** notation, how would you assign the permission string `-rwxr-x--x` ?

   (d) Using **numeric** notation, how would you assign the permission string `-rw-------` ?

2 Scripting

It may be helpful to look ahead to module 21, Shell Scripting - Part 1. Remember that all of your scripts should start with the line `#!/bin/bash`, read hash-bang-bin-bash. I point this out both because it's fun to say, and because this line is an *interpreter directive*, which tells the machine to use the `bash` interpreter when running your script. Inclusion of this line is an exercise in good, defensive programming. Failure to include it risks unexpected behaviour by your scripts on systems on which `bash` is not the default scripting shell (e.g. the CSE server).

In the spirit of encouraging good habits, one point will be deducted on each script which does not begin with the appropriate interpreter directive.

For Windows users: If you find yourself with an error like

```
1  /bin/bash^M: bad interpreter: No such file or directory
```

then your file was saved using Windows-style endlines. Windows uses a pair of characters, `\r\n`, to denote newlines, while Unix systems use only `\n`. The extraneous `\r` causes headaches. In the example above, the machine was looking for an interpreter literally named `bash^M`, but couldn't find one. To avoid this problem, most text editors and IDEs meant for Windows have an option that allows for saving files with Unix-style endlines.

3. [5 points] We'll start with some practice. There is an example shell script named `helloWorld.sh` provided with this assignment on Canvas. Rename it to `3.sh`, submit it on handin, and test it using the webgrader.

4. [12 points] Listing Files

   Submit a separate shell script for each of the following three items. The scripts should be named `4a.sh`, `4b.sh`, and `4c.sh`. Your outputs should match the expected outputs shown on the webgrader.

   (a) Write a script that displays the contents of the current directory, using long listing format and human-readable file sizes.

(b) Write a script that prints out the names of the ten smallest files in the current directory.

(c) Write a script that lists all of the `.png` files inside of the parent of the current directory as a comma-separated list.

5. [15 points] Viewing Files

Provided with this assignment is a file called `roster.txt`, which is the roster for a Philosophy 101 course.

Submit a separate shell script to do each of the following five items. The scripts should be named `5a.sh`, `5b.sh`, etc. Your outputs should match the expected outputs shown on the webgrader.

(a) Display the entire contents of `roster.txt`.

(b) Using `head`, output the file description at the top of `roster.txt`.

(c) Using `tail` output the names of the TAs in `roster.txt`.

(d) Output only the names of the students listed in `roster.txt`.

(e) Output all of the names in `roster.txt`, but none of the other lines.

6. [10 points] **Bonus**

This question, like the previous, uses the Philosophy 101 roster. Submit your solution for this question as `6.sh`.

1. There's a group project coming up for Philosophy 101, and the students need to be split into groups. Write a script that reads the student names from `roster.txt` and **randomly** sorts them into groups of three.
2. Your script may either assume that it will be run in the same directory as a file named `roster.txt`, or it may take the name of a text file as input (using positional arguments, **not read**). In either case, you can safely assume that the roster your script runs on will be formatted the same way as `roster.txt`, though the number of students may be variable.

```
$ ./6.sh roster.txt
```

should output something like:

```
Group 1
Hume
Epicuris
Schopenhauer

Group 2
Wittgenstein
Kierkegaard
Locke

etc...
```