CSE 391, Spring 2019
Assignment 3: Even More Unix Shell!
Due Tuesday, April 23, 2019, 11:59 PM

This assignment continues to practice using the `bash` shell and combining commands using redirection and pipes.

Task 0: Log in and Prepare a directory

First, **log in to a machine running Linux** and launch a **Terminal** window as described previously in Homework 2.

We have set up a ZIP archive full of support files that you must download to your Linux machine. Download/unzip it to a directory on your system. We suggest creating a `hw3` directory for your files for this assignment.

```
wget http://courses.cs.washington.edu/courses/cse391/19sp/homework/3/hw3.zip
unzip hw3.zip
```

Task 1: Bash shell commands

For each item below, **determine a single `bash` shell statement that will perform the operation(s) requested**. Each solution must be a one-line shell statement, but you may use operators such as `>`, `>>`, `<`, `|`, `&&`, and `;`. For commands that make use of files from `hw3.zip`, you may assume you are in a directory that contains those files.

Submit your answers to the Google Form. Note that you can return to the Google Form and modify and re-submit your answers multiple times before the deadline. In response to each question, **write the command that will perform the task described**, *not the output that the command produces*.

To test your commands, you should have unzipped `hw3.zip`. Use `man` pages or the *Linux Pocket Guide*, or post on the course message board, if you need help.

1. Write a single line (this could be several commands but all typed on one line) that 1) creates a directory called `HW3output` in your current directory, 2) compiles `Fresh.java`, and 3) if it compiles successfully, runs the program, sending its output into the file `output.txt` in the `HW3output` directory. Hint: remember we have ways to run more than one command on a single line.

   (Each step of the process should only occur if the previous step(s) succeeded. Part of the difficulty is in achieving all of this with a single-line command: creating a directory, compiling, and running. Once your command works for a valid `Fresh.java`, test it for a bad program by making a copy of `Fresh.java` and editing it to insert a syntax error.)

2. The Java program stored in `Box.class` reads text from standard input and surrounds it with a text box. It also requires a **command-line parameter** of the character to use for the box edges. Write a single line command that runs the program in such a way that it uses the character A, for this parameter. This single line command should also redirect the program's standard input to come from the input file `box input.txt` (note the space in the file name), and make it write its output to the file `boxoutput.txt` in the current directory. If such a file already exists, append the output to the end of this file rather than overwriting.

3. Create a new empty file whose name is the same as whatever user is running the command. In other words, if user `billybob` ran this command from directory `~/391/hw3`, it would create an empty file named `billybob` with a full path of: `~/391/hw3/billybob` . For an extra challenge, can you make it create the file with a .txt extension, such as `billybob.txt`?

4. *(Self-Discovery)* The `yes` command repeatedly outputs the string `y` ; it is useful only when combined with other commands that expect the user to confirm many actions by typing `y` or `n`. (a) Write a command that runs the Java program `Questions.class` and automatically answers `y` to all its questions. (b) Write a second version that answers `n` to all the questions. (If you do it correctly, the y or n text won't appear on the console.) Looking at the man page for the `yes` command will be useful here.

5. *(Self-Discovery)* Display the differences between the files `message 1.txt` and `message 2.txt` (note the spaces in the file names), in **side-by-side format**, **ignoring differences in spacing**. (You may want to resize your terminal window wider.)

(continued on next page)

6. Output the number of processes that the user `root` is running. Do not output the names of these processes, only the count of how many there are.
7. Display a list of which processes are using the most of this Linux machine's CPU or memory so that the list updates continuously every two seconds.
8. Create an alias `c` for the command `cat`. For example, if the user types: `c foo.txt`, it will run `cat foo.txt` .
9. Create an alias `lt` that will run `ls` with appropriate parameters to list files in long format and sorted by modification time, newest first.
10. Create an alias `lss` that will run `ls` with appropriate parameters to list files in long format sorted by file size (largest first).
11. Create an alias so that when the user types `trick` followed by a file name, it will change that file's last-modified date to be April 1, 2018 at 4:56pm. Note that you can check that the timestamp is correct using the command: `ls --full-time`