CSE 391, Autumn 2019
Assignment 5: Even More Unix Shell!
Due Tuesday, November 5, 2019, 1:00 PM

This assignment continues to practice using the `bash` shell and combining commands using redirection and pipes. For Task 0 there is nothing to submit. For Task 1, you will submit your responses to Gradescope.

Task 0: Log in and Prepare a directory

First, **log in to a machine running Linux** and launch a **Terminal** window as described in previous homeworks.

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 `hw5` directory for your files for this assignment.

```
wget http://courses.cs.washington.edu/courses/cse391/19au/homework/5/hw5.zip
unzip hw5.zip
```

Task 1: Getting Comfortable With a Text Editor (Copy + Cut + Paste)

The following are exercises and questions are meant to help you become more comfortable with a text editor that is built into the command line. You can choose either vim or emacs. It does not matter which editor you choose, but we recommend that you pick one and stick to learning it for the remainder of the quarter. While the answers to the questions themselves are relatively easy to find by simply looking them up, **the real learning will come from you actually practicing these commands yourself**. While we won't be able to know whether you've really been practicing, this is not for our benefit, it's for yours. We also recommend getting even more practice by writing the answers to your task1.sh and task1.sh files using this editor ☺

1. What is the command to cut or delete 5 lines of text, starting from your current cursor position?

2. What is the command to paste the 5 lines of text you just cut in the previous question?

3. What is the command to copy 8 lines of text, starting from your current cursor position?

Task 2: 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 all commands, do not create any files except those indicated.

To test your commands, you should have unzipped `hw5.zip` into the current directory. You can assume you are in the `hw5` directory when doing these problems. Use `man` pages or the *Linux Pocket Guide*, or post on the course message board, if you need help.

In response to each question, you will provide **the command that will perform the task described**, not the output that the command produces. Write your commands in on the indicated lines in the `task2.sh` file in the `hw5` folder.

4. Write a single line (this could be several commands but all typed on one line) that 1) creates a directory called `HW5output` 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 `HW5output` 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.)

(continued on next page)

5. 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.

6. (a) 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/hw5`, it would create an empty file named `billybob` with a full path of: `~/391/hw5/billybob` .

   (b) Repeat the above, but create the file with a .txt extension, as in `billybob.txt`.

7. *(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.

8. Write a command that compiles all of the java files in your current directory and all its subdirectories (and sub-sub-directories, etc. recursively)

9. The `curl` command fetches the contents of a document at a given URL. While `wget` downloads and saves the file to your local disk, `curl` instead outputs it to the terminal. In lecture #4 (from October 22), we wrote a command to process the html from the CS faculty website and extract the faculty emails. This command read from a file that Zorah had downloaded ahead of time.

   Modify the command from lecture so that in a single command you fetch the html from https://www.cs.washington.edu/people/faculty, extract the faculty emails from the html, and both output the emails to the terminal as well as write them to a file called `faculty-emails.txt`.

   Note: You should also find the appropriate command-line argument(s) to suppress some of `curl`'s normal output and run it in "silent mode".