CSE 391, Spring 2019
Assignment 2: More Unix Shell
Due Tuesday, April 16, 2019, 11:59 PM

This assignment continues to practice using the `bash` shell and basics of combining commands using redirection and pipes. For Task 0 there is nothing to submit. For Tasks 1, 2, and 3, you will submit your responses to the Google Form linked on the course website and calendar.

Some parts of this assignment depend on compiling and running Java programs from the command line. Many distributions of Linux do not include Sun's Java Development Kit (JDK). You may need to install JDK or use a different Linux machine that already has JDK installed, such as the CSE Virtual Machine (VM), CSE basement lab machines or the shared `attu` server. See the course web site for directions about how to install JDK on your own Linux machine.

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 1.

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 `hw2` directory for your files for this assignment.

```
wget http://courses.cs.washington.edu/courses/cse391/19sp/homework/2/hw2.zip
unzip hw2.zip
```

Task 1: Learn more about the system

The following are questions about your Linux system for you to investigate and discover the answers. Each question can be answered by running one or more Linux commands from the shell. For this task, you should submit the **answer to the question**, not the command(s) you used.

Each Linux system is different; you will receive credit if your answer is plausible for a typical Linux system.

1. What is the total number of files and directories stored directly within the `/bin` folder on this machine's file system? Do not include the contents of any subdirectories of `/bin` . (*Hint: There are a lot of files; it would take too long to count them all by hand. Use the `wc` command to count words or lines in a given input or file.)*

2. *(Self-Discovery)* The file `/etc/passwd` stores a list of all users' names and user account names on the system, along with a bit of other information such as what shell program they use. (The default shell for most users, and the one we have been learning about in this course is the Bash shell, stored in the file `/bin/bash` .)

   How many users exist on this Linux system that use the Bash shell by default? *(Hint: To figure this out, you will need to search for lines in the `passwd` file that mention bash.)*

   You may assume that no line of `/etc/passwd` contains the phrase "/bash" other than to specify the Bash shell.

3. *(Self-Discovery)* What is the full path of the `wc` program on this machine?

4. In class we talked about links, which allow one file to refer to another file. There are two kinds of links: "hard" and "soft" links. Create a ***soft*** link the file `lyrics.txt` and call it `link_to_lyrics.txt`. Do an `ls -l` (long format) command and paste the line referring to `link_to_lyrics.txt` into the Google Form.

5. Edit `link_to_lyrics.txt` to add another line to the file. What happens to `lyrics.txt`? Did it change?

(continued on next page)

Task 2: Java program w/ command-line arguments

Write and turn in a Java program called `Backwards` in a file named `Backwards.java`. Your program should print all of its *command-line arguments* with their characters reversed. Submit your program to the Google Form. **Be sure to name your file `Backwards.java`, and submit only the .java file,** not the .class file.

For example, if the user runs the program from a terminal using:

```
java Backwards hello how-are you? "I'm fine"
```

The program should produce the output below.

```
olleh
era-woh
?uoy
enif m'I
```

Task 3: 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 input/output redirection operators such as `>`, `<`, and `|`.

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 `hw2.zip`. Use `man` pages or the *Linux Pocket Guide*, or post on the course message board, if you need help.

1. The file `animals2.txt` contains an alphabetized list of animal names. It includes many duplicates. Output the first 16 distinct animals from the file, one per line. (The last one should be `adlie penguin` .)
2. Compile the Java program stored in the file `Crunch.java`.
3. Run the Java Crunch program stored in `Crunch.class`, suppressing (hiding) its console output. See the lecture slides for how to hide the console output of a program. (The program creates a file called `crunch.txt` and also prints output to the console. If your command is correct, there will be no console output, but the output file `crunch.txt` will still be created.)
4. Combine the contents of files `song1.txt`, `song2.txt`, and `song3.txt` into a new file `big_song.txt`.
5. Run the Java `Pow` program stored in the file `Pow.class`, redirecting its input to come from the file `numbers.txt` instead of from the console. (`Pow` accepts integers from standard input and computes exponents. If you ran it normally, it would sit there waiting for input. You'd have to press Ctrl-D to end the input.)
6. Display all lines from `animals.txt` that contain the text "`husky`" ignoring case, in reverse-alphabetical order and *with no duplicates*. Give the command that will output the lines themselves only.
7. Output the names of all files/folders in the current directory whose names do NOT contain the phrase "`txt`", one file name per line. *(Hint: Use the same command that you'd use to find lines that *do* contain a pattern.)*
8. Run the Java program Fresh (stored in the file Fresh.class), displaying no output on the console, and instead capturing the first 4 lines of output produced by the program in a file named Four.txt
9. *(Self-Discovery)* 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. Using `curl`, output the number of lines in the text file at the following URL into the file `hamlet_result.txt`:

   https://courses.cs.washington.edu/courses/cse391/19wi/homework/2/hamlet.txt

   Count these lines without using any graphical program (such as a web browser) to download the file to your computer. The line count should be the only thing that appears in `hamlet_result.txt`; don't show the number of words/characters, file name, etc. *Note:* You may want to (but may not need to) find the appropriate command-line argument(s) to suppress some of `curl`'s normal output and run it in "silent mode". Recall that you can search `man` pages for a phrase using the `/` key. Note that you can also redirect the stderr stream using `2>`.