CSE 390, Spring 2010
Assignment 2: More Unix Shell
Due Tuesday, October 19, 2010, 12:30 PM

This assignment continues to practice using the `bash` shell and basics of combining commands using redirection and pipes. Electronically turn in a file `homework2.txt` that contains your answers to several questions below.

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 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 / Prepare your home 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://www.cs.washington.edu/education/courses/cse390a/10sp/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. You should create a text file named `homework2.txt` and save it somewhere in your home directory, such as in your 390 folder. In this file, write your answers to the questions below, one per line. You don't need to explain how you found your answers.

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

1. What is the full path of the `diff` program on this machine?

2. What is the total number of files and directories stored directly within the `/bin` folder on this machine's file system? (Not including the contents of any subdirectories within `/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.)*

3. *(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.)

4. *(Self-Discovery)* 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. See the lecture slides to recall the difference between these two types of links.

   What happens when file `a` represents a "hard link" to another file `b`, but then `b` is deleted? (Does `a` also disappear from the file system? Does `a` remain but become a "broken link" that fails to function? Etc.) What about if `a` is a "soft link" to `b`? Test this for yourself by creating a link from one file to another, and deleting the linked-to file and investigating what happens. See if a is still there and/or if it can still be used after deleting `b`.

5. What is a reason it might be useful to have a link from one file to another? Why not just make a copy of the file?

(continued on next page)

Task 2: Java program w/ command-line arguments

Write and turn in a Java class called `Backwards` in a file named `Backwards.java`. Your program should print all of its command-line arguments reversed. For example, if the user runs the program from a terminal using the command:

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

The program should produce the output below. Submit `Backwards.java` along with your `homework2.txt` file.

```
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 `|`. Write these commands into your `homework2.txt` file, one per line. In your file, **write the command** that will perform the task described for each numbered item; don't write the actual output that such a command would produce.

To test your commands, you should have unzipped `hw2.zip`. Use `man` pages or the *Linux Pocket Guide* to help you.

1. Compile the Java program stored in the file `Crunch.java`.
2. 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` .)
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 produces both console and file output. If your command is correct, there will be no console output, but the output file `crunch.txt` will still be created.)
4. 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.)
5. Combine the contents of files `verse1.txt`, `verse2.txt`, and `verse3.txt` into a new file `lyrics.txt`.
6. 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.)*
7. *(Self-Discovery)* The `du` command shows the disk space used by all files in a given directory and its contents.

   Using `du`, output the total number of bytes (not kilobytes, megabytes, etc.) used by the `/etc/X11` folder and its subdirectories. Output a single line containing the total bytes (it's okay if the line also shows the folder name). (`/etc/X11` contains configuration files for the X window system, Unix/Linux's graphical user interface.)
8. *(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 words in the text file at the following URL:

   http://www.cs.washington.edu/education/courses/cse390a/10au/homework/2/hamlet.txt

   Count these words without using any graphical program (such as a web browser) to download the file to your computer. The word count should be the only output; don't show the number of characters/lines, file name, etc.

   Note: You will have 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.
9. Display all lines from `animals.txt` that contain the word "`growl`" ignoring case, in reverse-ABC-sorted order and with no duplicates. Output the lines themselves only.
10. Run the Java program stored in `Fresh.class`. Do not display any of the program's output on the console; instead, capture the first 4 lines of output produced by the program into a file named `willsmith.txt` .