CSE 303, Spring 2009
Homework 1: The Bourne Identity (using `bash`)
50 points

Due Thursday, April 9, 2009, 11:30 PM

This assignment focuses on using the `bash` shell to execute common Unix commands. There are two parts to this assignment: A set of Unix commands you must write, and a set of questions to which you must discover the answers.

Turn in two files, one named `answers.txt` and one named `commands`, from the Homework section of the course web site. You will also want the support file `hw1.zip` from the Homework section of the course web site; place it into a folder on your Linux/Unix machine, and extract its contents for testing your commands. (If you're on a Unix/Linux machine, you can unzip this file by typing `unzip hw1.zip` .)

In terms of grading, most of the points will come from the correctness of your answers: Does the command you gave perform the action specified? Some points will also come from the elegance of your commands. If you use an unnecessarily clunky solution to solve a problem that can be solved with a simpler command, you may lose partial points.

Note: The answers to all questions in this assignment can be found entirely using commands shown in the lecture notes from the first week. You may use other commands if you like, but you should constrain yourself to those from lecture or from the *Linux Pocket Guide* textbook. Ask the instructor if you are unsure whether a particular command is allowed.

Part 1 of 2. `attu` Server Questions/Tasks (`answers.txt`):

The following are questions for you to investigate and discover the answers. You should create a text file named `answers.txt` and write in it your answers to the questions below, one per line. Your file should have 7 lines total.

First, connect to the `attu.cs.washington.edu` server using `ssh` . The first time you connect, you must change your shell to `bash` by running the `chsh` command. When `chsh` asks you what shell you want to use, type `/bin/bash` . You will then need to reconnect to `attu`. (It will remember your preference for `bash` from then on.)

1. Try running each of the following commands on `attu`, one at a time. (You don't need to turn in the output of the commands.) What is the last command doing? Describe what it does in one sentence.
```
ps
ps -ef
ps -ef | more
ps -ef | grep bash
```
2. What is the full path of your home directory on the `attu` server? Run an appropriate command to find out.
3. How many total users have home directories within the same folder as your home directory? (The immediate parent of your home directory)
4. What version of Linux does `attu` use? We mean the version of the actual kernel (the core) of the Linux operating system, not the version of the distribution such as Red Hat / Fedora / Ubuntu. (Hint: It begins with 2.6.)
5. How much total memory (RAM) does the `attu` system have, in kilobytes (k)?
6. How many total processes are currently running on `attu` by all users, as reported by the `ps` command?
7. What is the process ID number (PID) of the `bash` process you are currently running for your session on `attu`?
8. Run the program named `banner` stored in Marty Stepp (user name `stepp`) 's home directory. This command accepts a word or phrase as arguments and outputs a large ASCII text version of that word/phrase. Try running this program with a short favorite word/phrase of yours (10 or fewer letters).

   Set up a `.plan` file for yourself that contains this banner output, so that when other users use `finger` on you, they will see it. (Test whether it works by running `finger` on yourself.) You don't need to put any text in your `answers.txt` for this problem, but we will check whether you have a `.plan` file by running `finger` on you.

(continued on next page)

Part 2 of 2. Bash Commands (`commands`):

For each of the numbered items below, determine a single `bash` shell statement that will perform the operation(s) requested. Note that for some items you must redirect input/output or combine commands using `<`, `>`, and `|` . You can work on your solutions to these problems either on an actual computer running Linux/Unix (such as in the basement labs) or on the `attu` server. We recommend using a machine other than `attu` to get more of a Linux experience.

For full credit, each of your solutions must be a single one-line shell statement. In other words, the `commands` file you submit should be 14 lines long, where line *N* contains the command to perform the operation from item *N* below.

Unless otherwise specified, a statement may not include the `;` operator (which executes two separate statements on one line) or the `&&` or `||` operators (which join two commands together with a boolean and/or condition).

Each Linux/Unix box can be slightly different; for full credit, your commands must be able to work properly either on `attu`, or on the basement lab computers, or on a fresh Ubuntu installation (with Java installed). Each command should run without changing the shell's working directory (in other words, don't use `cd` as part your solutions.) If you like, your `commands` file can optionally be a runnable script with a `#!/bin/bash` header, which we'll discuss next week in lecture.

1. List all files in the `/var` directory, in the standard format (no special parameters required).
2. Copy all the web pages (files whose names end with `.html` or `.css`) from the current folder's `website` subdirectory to the current directory.
3. List all files (including hidden files) in the `songs` subdirectory of the current directory, along with all subdirectories of `songs`, all in reverse alphabetical order, one file shown per line.
4. Figure out which Java programs in the current folder's `programs` subdirectory use the Java `DrawingPanel` class (see which files construct a `new DrawingPanel` object). List the names of these files, one per line. (Do not output the actual lines of the files that construct `DrawingPanel` objects; merely output the files' names.)
5. Set the file `HW4.java` to have a last-modified date of March 15, 4:56pm.
6. Create an alias so that when the user types `q` , it will exit the shell.
7. Create an alias so that when the user types `cheat` followed by a file name, it will change that file's last-modified date to be March 15 of this year at 4:56pm.
8. Combine the files `verse1.txt`, `verse2.txt`, and `verse3.txt` into a new file `lyrics.txt` . Your command should not modify the contents of any of the three original part files.
9. 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. The program's output should display on the console as normal. (`Pow` accepts integers from standard input and computes exponents. If you run it normally, it will just sit there waiting for input. You'll have to press Ctrl-D to end your input and get out.)
10. 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.
11. Create a new file named `otherfiles.txt` that contains the names of all files/folders in the current directory whose names do NOT contain the phrase "`txt`", one file name per line.
12. Output the number of words in the text file at the following URL:
    http://www.cs.washington.edu/education/courses/cse303/09sp/homework/1/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.
13. Output the number of processes that the user `root` is running whose names contain the phrase "`sh`". Do not output the names of these processes, only the count of how many there are.
14. Compile the Java program stored in `Fresh.java` and run it. 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` . Part of the difficulty of this problem is in achieving all of this with a single-line command: compiling, running, and putting the first 4 lines into the output file. (You may use a semicolon or `&&` operation in this command.)