CSE 390, Spring 2011
Assignment 3: Multi-user Unix Environments
Due Tuesday, April 19, 2011, 1:30 PM

This assignment continues to practice using the `bash` shell and basics of combining commands using redirection and pipes. Electronically turn in a file `homework3.txt` that contains your answers to several questions below.

Task 0: Log in to attu

First, **log in to the shared `attu` server** using a terminal as shown in lecture and in the Working at Home section of the course web site. Use a terminal and an `ssh` program, or stand-alone SSH software such as Tectia on Windows. Log in to `attu` using your usual CSE user name and password. Contact the instructor/TA or CSE support if you cannot log in.

There is no ZIP archive of support files for this assignment. But there are some files we have placed on attu that you will need to be able to access in order to answer the questions in this assignment.

Task 1: Learn more about attu

The following are questions about the shared `attu` Linux system for you to investigate and discover the answers. Create a text file named `homework3.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. If you like, you can also write the shell command(s) you used to find each answer, though this is not required. (In Task 2 later, you *will* need to write the commands used for each item.)

The state of `attu` changes over time; you will receive credit if your answer is plausible for `attu` in general.

1. Run the program `~stepp/cowsay` with any string of your choice as a command-line parameter, and show us the output that is produced. Paste the output into the top of your `homework3.txt` file.
2. The `attu` server is actually "mirrored" as several physical host computers to spread the load of all the students using the server. Which of these hosts are you logged into now? *(What is its full host name?)*
3. What is the full path of your home directory on the `attu` server?
4. What version of the Linux kernel does `attu` use? *(It begins with 2.xxx.)*
5. *(Self-Discovery)* What distribution of Linux does `attu` use *(e.g. Ubuntu, etc.)*, and what version of it? There isn't a foolproof single command to discover this information, but many Linux systems have a file named `/etc/`**`distributionname`**`-release` whose text contains the distribution name and number.
6. How much total memory (RAM) does the `attu` system have, in kilobytes (k)? *(Hint: The program that lists which of `attu`'s processes consume the most CPU% also gives stats about `attu`'s total and available memory.)*
7. What is the process ID number (PID) of the `bash` process you are currently running for your session on `attu`?
8. How many total processes are currently running on `attu` by all users, as reported by the `ps` command?
9. 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.)*
10. What are the usernames and names of 3 other people logged into `attu` right now?
11. A small change has been made between the two files below on `attu`. What is the difference between the two files? (Show the literal change between the files, and explain the change in your own words.)
    `~stepp/390/babynames1.txt` and `~stepp/390/babynames2.txt`

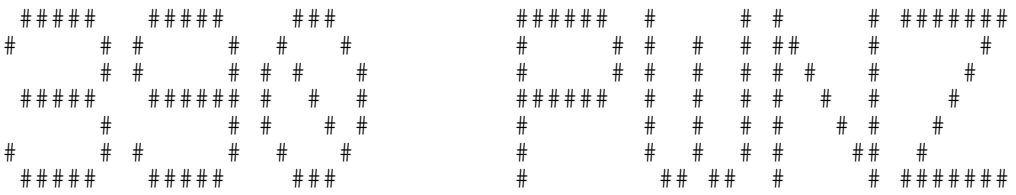
(continued on next page)

Task 2: Bash shell commands on attu

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 `homework3.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 some commands, you should be running on `attu`. Use `man` pages or the *Linux Pocket Guide* to help you.

1. Create an alias `p` for the `pico` editor. For example, if the user types `p foo.txt`, it will run `pico foo.txt` .
2. Create an alias `ll` (two lowercase Ls) that will run `ls` with appropriate parameters to list all files (including hidden file entries whose names begin with `.`) in long detailed format.
3. 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.
4. Display the differences between the files `message 1.txt` and `message 2.txt` (note the spaces in the file names) in directory `~stepp/390/`, in side-by-side format, ignoring differences in spacing. *(You may want to resize your terminal window wider.)*
5. Create a new empty file whose name is the same as your user name. In other words, if user `billybob` ran this command from directory `~/390/hw2`, it would create an empty file named `~/390/hw2/billybob` . (For an extra challenge, can you make it create the file with a .txt extension, such as `billybob.txt`?)
6. Send a terminal message with the following large text to the user named `stepp` (or another user of your choosing). Notice that the text appears all on one line, not two lines.



7. Display a list of which processes are using the most of `attu`'s CPU or memory; make it update once per second.
8. 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.
9. *(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. *(Test your command using the file `~stepp/390/Questions.class`; you'll want to copy this file to a directory under your control and run it there, it won't work if you try to run it from the ~stepp/390 directory. If you do it correctly, the y or n text won't appear on the console.)*
10. Write a single command that copies the source file `Fresh.java` from directory `~stepp/390/` to the current directory, then compiles the program in the current directory, and if it compiles successfully, runs the program.

    (This is similar to an exercise on the previous assignment, except we want it to copy and compile as well as run. But if the copying or compilation produces an error, don't run the program. Part of the difficulty is in achieving all of this with a single-line command: copying, 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.)
11. Write a single command that runs the Java program stored `Box.class` (assumed to be in the current directory), which reads text from standard input and surrounds it with a text box. It accepts a command-line parameter of the character to use for the box edges; run the program in such a way that it uses the tilde character, `~` , for this parameter. Also redirect the program's standard input to come from the file `boxinput.txt` , and make it write its output to the file `box output.txt` (note the space in the file name) in the current directory. If such a file already exists, append the output to the end of this file rather than overwriting.

    (*Hints:* Find the files `Box.class` and `boxinput.txt` in the `~stepp/390/` directory and copy them to your own working directory for testing. Try using an `x` for the box border first before using the trickier `~` tilde.)