CSCI 2000U: Assignment 1

due Oct. 3, 2015, 17:00 EDT

The goals of this assignment are

- to gain familiarity with more Unix commands (`wc`, `sort`, `grep`, etc.) and use them to work with files;
- to use pipes and redirection operators to manipulate input and output;
- to learn how to write and execute shell scripts to automate sequences of shell commands; and
- to use the Unix `tar` command to compress a directory tree into a single file for submission.

To begin, create a directory `Assignment-1` that will contain all the files created for the assignment. Depending on how you have structured your home directory, the full path to your `Assignment-1` could be, say, `/home/mobile/Desktop/csci-2000/Assignments/Assignment-1`.

1. Open a terminal window running a `bash` shell and execute each of the following commands at the command prompt. At the same time, open a text editor in a different window and explain briefly what each command does in a text file called `sample_commands.txt`. You may want to use the Unix command `man` to learn more about each command, but paraphrase the `man` description in more common language.
   (a) `echo hello world`
   (b) `echo {con,pre}{sent,fer}{s,ed}`
   (c) `date`
   (d) `whoami`
   (e) `cal 2000`
   (f) `cal 9 1752` (do you notice anything unusual?)
   (g) `bc -l` (type quit or press Ctrl-d to quit)
   (h) `echo 5+4 | bc -l`

   In question 4, you will submit the file `sample_commands.txt` for grading.

2. Use any text editor to create a text file that contains the following lines (save the file as `simple.sh`):

```
#!/bin/bash
# This is a comment
echo "The number of arguments is $#"
echo "The arguments are $*"
echo "The first argument is $1"
echo "My process number is $$"
echo "Enter a number from the keyboard: "
read number
echo "The number you entered was $number"
```

The term "script" refers to a file containing a collection of commands to be executed in sequence; we could just as easily use the term "program." Text files are sometimes given the suffix `.sh` to indicate that the contents are a shell script—this is not mandatory.

Notice the first two characters of this file are the special combination "`#!`"—sometimes pronounced *sha-bang*—followed by the path `/bin/bash` Usually, the character "`#`" denotes the start of a comment. However, when the first line of a script file contains "`#!`" followed by the path to a program, the command shell is instructed to use that program (in this case `/bin/bash`) to interpret the lines that follow. When the script file is made executable (see below), the script can then be invoked by typing `./simple.sh` at the command prompt. It is alternatively possible to execute the shell script `simple.sh` by invoking `bash simple.sh` at the command prompt.

To understand the rest of the script `simple.sh`, we observe that any command-line arguments passed to the script can be accessed through "`$1`," "`$2`," "`$3`," etc. The special combination of characters "`$*`" stands for all the command-line input arguments and the characters "`$#`" represent the number of input arguments. Similarly, the process number of the shell executing the script is given by "`$$`." Finally, the command "`read number`" assigns keyboard input to a variable.

- To make the script `simple.sh` executable, change the permissions of the file `simple.sh`:
  `ls -l simple.sh`
  `chmod a+x simple.sh`
  `ls -l simple.sh`
  You should notice that the permission bits of the file `simple.sh` (as displayed by "`ls -l`") changed as a result of running `chmod`.
- Verify that `simple.sh` is in fact executable by running the following command:
  `./simple.sh hello world`
  When prompted to enter a number, enter "`5`" and hit RETURN. Notice that you have to preface the command `simple.sh` by "`./`" to execute an executable file in the current working directory.
- Finally, enter the following command:
  `echo "five" | ./simple.sh hello world > simple.out`
  This will generate a file called `simple.out`.

In question 4, you will submit the files `simple.sh` and `simple.out` for grading.

3. This exercise involves manipulating data obtained from experiments involving *cochlear implants*. A cochlear implant is a small electronic device that is surgically implanted in the inner ear to give deaf people a sense of hearing. More than a quarter of a million people have them, but there is no widely-accepted benchmark to measure their effectiveness. To establish a baseline for such a benchmark, a clinical trial was set up using teenagers with cochlear implants. The subjects were asked to listen to audio files on their computer and report:

   - the quietest sound they could hear;
   - the lowest and highest tones they could hear; and
   - the narrowest range of frequencies they could discriminate.

   In each experiment, a lab technician played an audio sample for a subject, and recorded the subject's observations—when the subject first heard the sound or first heard a difference in the sound. Each set of test results was written out into a text file, one set per file. Each participant had a unique subject ID, and a made-up subject name. Each experiment had a unique experiment ID. The experiment has collected 351 files so far.

   Unfortunately, the data is a bit of a mess! There are inconsistent file names, there are extraneous `NOTES` files to get rid of, and the data is spread across several directories. We are going to use shell commands to get this data into shape. By the end, we want the data cleaned up in a single directory.

   - If you have not already done so, create a directory `Assignment-1` that contains all the files created for the assignment. Depending on how you have structured your home directory, the full path to your `Assignment-1` could be, say, `/home/mobile/Desktop/csci-2000/assigments/Assignment-1`.
   - Download the archive file `cochlear-implant-trials-data.tar.gz` into the directory `Assignment-1`. The file is available on BlackBoard and can be downloaded using a web browser.
   - Extract the archive file `cochlear-implant-trials-data.tar.gz` within `Assignment-1`. To do this, enter `tar xzvf cochlear-implant-trials-data.tar.gz` at the command prompt. This will create a subdirectory `Assignment-1 /data` within `Assignment-1`.
   - Clean up the data files from the directory `data` as follows:
     – Use a `find` command to delete the extraneous `NOTES` files in all the subdirectories of `data`.
     – Create a directory `cleaned_data` directly beneath `Assignment-1`, i.e., create `Assignment-1 cleaned_data`.
     – Move all the files from the subdirectories of `data` into `cleaned_data`.
     – Rename all the files in `cleaned_data` to end with the suffix `.txt` (it is permissible for a file name to with the suffix `.txt.txt`).

     Use the `history` command to recall which precise commands you used to clean up the data. Create an executable script file `cleaning.sh` to store those commands so that, when executed from within the directory `Assignment-1` with the subdirectory `Assignment-1 /data` in its original state, the script `cleaning.sh` should execute the four cleaning steps listed above.

     Hint: If you make a mistake and need to start over just do the following:
     – Navigate to the `Assignment-1` directory.
     – Delete the `data` directory using `rm -rf data`.
     – Re-extract the data from the tar file, i.e., execute `tar xzvf cochlear-implant-trials-data.tar.gz` at the command prompt. You should see that the directory `data` has reappeared in its original state.

     Once you have successfully created the directory `Assignment-1 /cleaned_data` with appropriate files within, delete the old directory `data` and the archive file `cochlear-implant-trials-data.tar.gz` from the directory `Assignment-1`.

     In question 4, you will submit the file `cleaning.sh` and the directory `cleaned_data` for grading.

4. If you have not already done so, create a directory `Assignment-1` that contains all the files created for the assignment. Depending on how you have structured your home directory, the full path to your `Assignment-1` could be, say, `/home/mobile/Desktop/csci-2000/assigments/Assignment-1`.

   After completing questions 1, 2, and 3, make sure the required files and directories are stored in `Assignment-1`. That is, the directory `Assignment-1` should contain *precisely* the following things:

   - the file `sample_commands.txt` (from question 1);
   - the file `simple.sh` (from question 2);
   - the file `simple.out` (from question 2);
   - the file `cleaning.sh` (from question 3); and
   - the directory `cleaned_data` (from question 3).

   Make sure that your files are all named *exactly as specified* (i.e., matching upper- and lower-case letters, no spaces in the file names, underscores, same suffixes, etc.). Remember to delete the old directory `data` and archive file `cochlear-implant-trials-data.tar.gz` from the directory `Assignment-1` containing the "uncleaned" data from question 3.

   To submit your work, you will construct an archive in the form of a *gzipped tar* file (sometimes called a "tar-ball"). The command `tar` stands for "tape archive" (referring to the fact that computer file systems were backed up on magnetic tapes). The command `gzip` is a file compression program (e.g., like `zip` from DOS or Windows).

   - Navigate to the *parent* directory of `Assignment-1` (i.e., the directory directly above `Assignment-1`).
   - Create a shell variable `STUDENT_NUM` that stores your student number. For instance, if your student number is 123456789, you would enter
     `STUDENT_NUM=123456789`
     at the command prompt (make sure there are no spaces around the equals sign).
   - Enter the following command at the command prompt to create the `tar` file:
     `tar czvf ${STUDENT_NUM}-Assignment-1.tar.gz Assignment-1`
     Make sure the file name is *exactly* as specified here:
     - The first 9 characters of the file name should be the digits of your student number.
     - The word "`Assignment`" should be capitalized.
     - There should be hyphens (not spaces or underscores) separating your student number, the word "`Assignment`," and the digit "`1`."
     - The suffix `tar.gz` should use all lower case letters.
   - You can verify that the gzipped tar file was created by entering `ls -l` at the command prompt.
   - You can verify the contents of the tar-ball by entering
     `tar tzvf ${STUDENT_NUM}-Assignment-1.tar.gz`
     at the command prompt.
   - **Upload the appropriately-named `tar.gz` file to BlackBoard.**