October 1, 2015

Tools and Techniques of Computational Science - Fall 2015
Assignment 0

Architecture and Unix

This homework is due by midnight on Monday, October 12. This exercise can be completed in any plain text format or LaTeX. Future homework will require the use of LaTeX, so if you are not familiar with it and have the time now, it might benefit you to go ahead and bite the bullet.

Later on in the class, I will provide a quick primer on LaTeX, but for now, you can check out latex-project.org.

Please submit homework to the canvas site and upload using either a plain text file (ascii file) or a PDF. Do not upload a word document.

Exercises

Exercise 0.0:

Use the top500.org website to generate interesting statistics. Present these results in your favorite graphing program. At this time, you have the right to choose what is your favorite. Later on this class, you will be introduced to both `gnuplot` and `matplotlib` and future plotting will be required using one of these packages.

Additionally, tell a story about any interesting or surprising trends. Bonus points will be given for creative (and correct) narratives.

Exercise 0.1:

Why is Stampede (2.7 GHz processors) "faster" than lonestar (3.33 GHz processors) even though the clock speed is slower?

Exercise 0.2:

Construct a "Theoretical Performance" breakdown for Stampede similar to the one presented in Lecture #1 for the Intel Woodcrest.

Filters and Pipelines

A *filter* is a small and (usually) specialized program in Unix-like operating systems that transforms data in some way.

As is typically the case with command line programs in Unix, filters read data from standard input and write to standard output. Standard input is the data source for the program; by default, this is the input from the keyboard. However, programs (and thus filters) can receive input from a file or from the output of another program. This is referred to as *redirection*.

Standard output is the display by default. Thus, if standard output for a program is not redirected to a file or a device (or another program), it will be displayed directly to your terminal. If, however, you want to use output from one program as input for another program (a filter, for instance), you can connect the data streams with *pipes*.

By using filters, redirection, and pipes, it is possible to construct a sequence of commands to accomplish a highly specialized task.

Example: How many users on Stampede have a name that contains the string "Chris"?

```
[19:21:31][csim@login2:~]$ getent passwd | grep Chris | wc -l
100
```

Note that we often refer to command strings like the example above as *one-liners* in that they accomplish a desired function with a single line of Unix commands. We start this particular one-liner by generating output with the command "getent". Here, *getent passwd* has queried the TACC authentication server and returned a colon (:) separated list that includes user names. By using a pipe (|), we redirect this output to be filtered as input by the "grep" command. Grep only returns those usernames that contain the string "Chris". Once again, we redirect standard output from grep with a pipe to the "wc" program. This filter then counts the number of lines in the input stream (wc -l) and returns the result to standard output. Since there is no additional redirection of this output, it is written to the terminal.

Example of filters:

- awk
- cat
- cut
- expand
- fold
- grep
- head
- more (less)
- sed
- sort
- strings
- tail
- tac
- tee
- uniq
- wc

Exercise 0.3:

Explain what the following command line invocation is doing. Include in your description how the flow of standard input and standard output is used to obtain the desired result.

```
[19:23:23][csim@login2:~]$ getent passwd  | cut -f 7 -d: | sort | uniq

/bin_bash
/bin/bash
/bin_csh
/bin/csh
/bin/sync
/bin/tcsh
```

```
/bin/zsh
/sbin/halt
/sbin/nologin
/sbin/shutdown
[19:24:00][csim@login2:~]$
```

Exercise 0.4:

How many users on Stampede use bash? How many use tcsh? What is the most popular shell that isn't bash or tcsh?

Exercise 0.5:

Who has the longest TACC username? How many characters is it? How many TACC users have usernames that are exactly 8 characters long?

Exercise 0.6:

Generate your own question involving parsing "getent passwd" output and then generate the one-liner necessary to answer your question.

Note: don't forget about your friendly neighborhood man pages for detailed information on various Unix commands.

Exercise 0.7:

Most UNIX distributions come with a dictionary file: /usr/share/dict/words. Examine this file and describe how the entries are arranged and formatted. Grab a copy of this file to keep in your own directory to complete this exercise.

Construct one liners using regular expressions to answer the following questions:

1. count the number of words beginning with each letter of the alphabet and prints the result in the following format:

   ```
   a:31788
   b:25192
   ...
   ```

2. How many words end with the letter "s"? Of these how many are possessive?

3. Write a one-liner that counts and prints out the number of words that contain the substring *foo* but not the substrings *food, fool* or *foot*. (Hint: this can be done most logically with 1, 2, or 4 calls to grep. See if you can do it in 1, but full credit will be given for any answer that works.)

4. Generate your own question involving parsing the dictionary file and then generate the one-liner necessary to answer your question.

Exercise 0.8:

List and describe the filesystems found on the login nodes of Stampede. Indicate whether they are are backed up and have a quota, and describe their intended usage. Describe the hardware of the servers that provide each type of storage (if applicable) including the type of interconnect.

Exercise 0.9:

1. What kernel version are the Stampede login nodes running and how did you determine this?
2. Download the kernel.org version of this kernel using a one-liner. How did you do this? (note that you will not find the exact same version because Stampede is using a Red Hat distriubtion; look for the closest revision).
3. How large is this kernel directory? How many files does it contain and how many of those files are .c files?
4. How many directories does the kernel source contain? Answer this question at least three different ways with one example using *find*, one with *ls* and finally one with *tree*
5. What is the largest file and how many lines does it have?
6. How many total lines of code (including comments) are in the linux kernel? (Hint: source files for the kernel are those that end with .c or .h)
7. Generate your own question involving the linux kernel source and then generate the one-liner necessary to answer your question.