# Computer lab exercises and homework for Oct. 3

## Preliminaries

The goal of this assignment is to do some exploratory data analysis in a Unix/Linux environment using some of the simple text processing utilities, and to learn a "little language" called Awk. To get started, look at Section 1 in the PDF writeup, which covers regular expressions and the grep utility. The following commands are useful in your day-to-day work on Agave:

- file manipulation: ls mv cp rm chmod
- directory handling: cd pwd
- text processing: head tail sort grep more wc

Each of these commands has a manual page; man grep, for instance, prints the manual page for the grep command. Here's a thumbnail description of the text processing commands:

- head -n *count* file prints the first count lines of file. The default count is 10 lines, and head reads from the standard input if no file is specified.
- tail -n *count* file Like head, but prints the last *count* lines of the input.
- sort sorts text. See the manual page for the options -n, -k, and -r. For instance, sort -n -k3 sorts in numerical order using the third field of each line as the key.
- grep *pattern* file prints all the lines in *file* that contain *pattern*. In the simplest case, *pattern* is a sequence of letters, digits, and spaces. (If the pattern contains spaces, enclose it in quotation marks.)
- more *file* displays *file* one screenful at a time; the standard input is used if no file is specified. Hit the space bar to advance by one screenful, the return key to advance by one line, and q to quit.
- wc ("word count") displays a count of lines, words, and characters in a file or list of files.

Awk is a little language that can accomplish many useful tasks in a single line of code; Section 2.1 of the Awk writeup gives a few examples of "one liners" that are handy for simple text processing jobs. Awk has inspired many successors; for example, Perl, Python, and Ruby are full-featured scripting languages that extend Awk's capabilities in many ways and are suitable for large programming projects. They are well worth learning, but the effort takes considerable time. You can become proficient in Awk in an hour.

## Use Agave for these exercises

1. When you are ready, log into Agave with your ASUrite ID and password:

   ```
   ssh yourname@agave.asu.edu
   ```

   You will need to start the Cisco VPN on your computer to log into Agave from outside the ASU campus network.

2. Once logged in, start an interactive session on one of the compute nodes by typing

   ```
   interactive
   ```

3. If you type `pwd` now, you will see your home directory. Create a new folder for today's exercise by typing (for example)

   ```
   mkdir oct02
   ```

   You may use any name that you like, of course, but it is convenient on Linux systems to use names that do *not* have embedded spaces or nonalphanumeric characters.

4. Type

   ```
   cd oct02
   ```

   to change your working directory to `oct02`. You may return to your home directory at any time by typing `cd` with no arguments.

5. Download the data files from Blackboard and save them in a convenient place on your computer. Upload the files using PuTTY on Windows. On a Mac or Linux machine, you may type the following command in a terminal window:

   ```
   scp cardinals.txt stats yourname@agave.asu.edu:oct02
   ```

   The latter command copies the files directly from the current folder on your laptop to the directory `oct02` that you just created.

6. When you are finished, be sure to log out of the compute node by typing `exit` and from the login node by typing `exit` once more.

## Exercises

**Due date:** Friday, Oct. 12 at midnight. You may simply copy the commands that you type (and the results) as a Word or PDF file for grading.

**Exercise 1.** Since football season is starting, we'll use the Arizona Cardinals active roster as of Oct. 2 for the input. It is stored as `cardinals.txt`. Use your browser to save this file to your home directory or other convenient location.

Using any combination of the text processing tools mentioned above, devise a command to perform each of the following tasks (one command per task):

(a) count the number of players on the roster

(b) display the roster in order by jersey number

(c) display the heaviest five players

(d) display all players who went to college in Oklahoma

(e) display all the quarterbacks (QB)

(f) count the number of wide receivers (WR).

Strive to do each task with a one-line command. If you need more steps, that's OK too. Do *not* write a C or Java program to do these!

## The stats program

The stats program itself is an Awk script that accepts input lines consisting of at most one number per line. It computes and displays the mean, the smallest and largest values, and the total number of values in the list. Save this file as stats and give it execute permission with

```
chmod +x stats
```

(This command turns the Awk script into its own executable program. Experiment with it by creating your own little data file with one number per line.) I invite you to inspect the Awk code to get a feel for how Awk works.

**Exercise 2.** The objective here is to write a simple Awk command to extract selected fields from the Cardinals roster, and, if appropriate, pipe them to stats or other appropriate program to answer each question. In other words, for each of the following tasks, you will write a command line of the form

```
awk command cardinals.txt
```

or

```
awk command cardinals.txt | other program
```

where the other program can be stats or one of the other standard Unix utilities. Please read Sections 2.1–2.4 of the Awk handout first.

In the simplest cases, an Awk script has the form

```
pattern {action}
```

If *pattern* is omitted, then every line is matched, and if *action* is omitted, then Awk simply prints each matched line. For example,

```
awk '{print $1}' file
```

prints the first field of every line in file (the pattern is omitted, so every line is matched). So, here we go (one command per task):

(a) What is the average weight of a Cardinals player?

(b) How much do the lightest and heaviest players weigh?

(c) How many rookies are on the team? (A rookie is a player with no previous professional experience.)

(d) What are the names and positions of the players with more than 8 years of professional experience?

(e) Who are the cornerbacks (CB)?

**Exercise 3.** See Example 14 in the handout on awk. You will need to place your Awk scripts for parts (a) and (b) in a file (say height.awk and bmi.awk, respectively) and invoke Awk as

```
awk -f height.awk cardinals.txt
```

This is because quotation marks are significant to the shell, and it's just too difficult to write a single shell command line that passes all the required quotation marks to Awk in the right place.

(a) Write an Awk script that displays each player's height in inches. (There are 12 inches in a foot.)

(b) Write an Awk script that displays each player's name, position, and body mass index. The body mass index is defined as

$$\text{BMI} = w/h^2$$

where $w$ is weight (well, mass) in kilograms and $h$ is height in meters. One inch is exactly 0.0254 meters, and you may assume that one kilogram equals 2.2 pounds.

(c) Identify the players whose BMI is greater than 30. Don't modify your script in (b) to answer this question; instead, pipe the output that you get in (b) to another appropate Awk command.

(d) What is the average BMI of the offensive line (OL)? Use your script in (b) without modification and pipe its output to another appropriate command(s).

(e) What is the average BMI of the quarterbacks? Wide receivers? (Same comments as (d).)

## Semester homework: Learn a text editor

To use a Linux-based system effectively, you need to learn how to use a text editor from a terminal. The following three editors are widely available:

- pico, which is very simple
- vi (or vim), which has intermediate complexity
- emacs, which is the most sophisticated Linux-based editor but also has the longest learning curve.

I recommend that you invest the time and effort to learn one of the latter two editors; it is a useful professional skill. We will not talk about these editors in class, but there are many online resources to help you get started. You may find one of the following links helpful:

- Emacs tutorial
- Vi/Vim tutorial