Lab 7b - More Linux Text Parsing Practice

Reading

The Linux Command Line, chapter 20, Text Processing, pp. 233 - 266 in the printed book or pp.273 - 314 in the pdf.

Lab

A good Linux admin can quickly parse huge log files from the command line (and jump tall buildings in a single bound.) This lab takes another look at file parsing.

The file, LinuxLab2LogFile, contains some SMTP (mail server) logs. The format of the file is:

Date Time SenderIPAddress SenderName ServiceName MailServerName MailServerIP Commands

We want to answer these questions:

1. how many separate senders sent mail?
2. how many messages did each sender send?
3. who sent the most messages?

Open a browser in your CentOS VM and go to Canvas. Download LinuxLab2LogFile. Open a terminal and change directory to ~/Downloads, which is where your browser puts things.

View the file from the command line:

```
less LinuxLab2LogFile.
```

First, some cleanup. On a few lines, the server tried to send binary data. Some of the bytes in the binary data looked like carriage returns, so there are extra lines of junk in the file. You can see some if you use `less` and page through the data (line 708 for example. While in `less`, type **708**) Note that all of the valid lines start with the date, so the following command will remove any lines that don't start with 2014-. (The "^" means start at the beginning of the line. You could use a fancy regular expression in grep to match any line in the correct format no matter what the year, but we'll start simply.)

```
grep '^2014-' LinuxLab2LogFile | less
```

Now, on to answering the questions. To do this, you'll have to find out how to use the command, cut. Use --help, man, or Google to do this.

Question 1: How many separate senders (servers) sent mail?

Step 1: Cut the SenderIPAddress column from the results of your grep command, above. Replace the less at the end of the command with your cut command. Check to be sure you got the right column; you should see IP addresses.

```
grep '^2014-' LinuxLab2LogFile | [your cut command] | less
```

Step 2: Now pipe the output into sort. (Note: this is to help the uniq command work, and not to sort them into a human-nice format.) Look at the data to make sure it is what you expect.

```
grep '^2014-' LinuxLab2LogFile | [your cut command] | sort | less
```

Step 3: Use the uniq command to extract the unique IP addresses. Note, that this only works if you sorted the list first. Check to see that the data is correct.

```
grep '^2014-' LinuxLab2LogFile | [your cut command] | sort | uniq |
less
```

Step 4: Use the wc command to count the number of lines in temp3. Note: I got 19.

```
grep '^2014-' LinuxLab2LogFile | [your cut command] | sort | uniq |
[your wc command]
```

Note: when you are making long strings of commands like this it may be helpful to take small steps. I will often pipe the output of the first command into less to see if I like the output. Then I pipe the first command to the second and then to less and check again. I keep going this way until I finally have the entire command working correctly.

Question 2: How many messages did each sender send?

Each message causes several lines in the log file, as the file records the EHLO, MAIL FROM, RCPT TO, DATA and QUIT commands that are part of the SMTP protocol. We will assume that one DATA command means one message. Use the grep command to extract the lines containing "DATA". Then use cut and sort as before. The uniq command has an option to count the number of duplicates (messages in this case) it finds. The output will show the number of messages from each IP address.

Hint: your command will look something like this

```
grep '^2014-' LinuxLab2LogFile | [grep for DATA] | [same cut command
as before] | [same sort] | [uniq with option to count]
```

Question 3: who sent the most messages?

Since there are only 19 senders, you can just look at the output from the last question. If there were many senders, you could pipe the output into
`sort -rn | head` (or you could use `less`)
The -r option causes sort to sort in reverse order so the big numbers are at the top, and the -n option causes sort to put data in numeric rather than alphabetic order. This would also work, where you have the small numbers at the top of the file and look at the end to see which one had the most.
`sort -n | tail`.

BTW If you'd like to get an idea of what's possible from the command line, look at the blog "Command Line Kung Fu".

Hand in

Hand in the answers to questions 1-3.