DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Homework Assignment No. 01:

COMMAND LINE PROGRAMMING

Submitted to:

Professor Joseph Picone
ECE 3822: Software Tools for Engineers
Temple University
College of Engineering
1947 North 12th Street
Philadelphia, Pennsylvania 19122

8/30/2015

Prepared by:

Devin Trejo
Email: devin.trejo@temple.edu

1. PROBLEM

In this first assignment of the semester we simply want to familiarize ourselves with command line tools. We start by learning how to personalize our Linux configuration by editing the .bash_profile and .bashrc files in ~/. Aliases and manipulating the environment path allow us to run commands from any directory on our machines.

1. Edit the environment path so a 'hello world' command can be ran from any directory. Then create an alias by modifying the .bash_profile.

We then switch gears and learn some commonly used commands. Commands like 'grep', 'find', 'wc', 'echo', etc. are powerful commands that we should know. Specifically we will work with a large data set of clinical EEGs and query for three cases:

2. Patient Names whose first names start with R and last names start with S who had an EEG in the date range 2010-13
3. EEG reports that contain the word 'spike'. EEG reports that contain the word 'seizure'. We then produce a histogram of the words in these reports.
4. For EEG reports that contain the word 'spike' produce a histogram of bi-grams.

2. APPROACH

In this introduction to basic Linux commands we start by looking up which commands will prove useful to accomplish our tasks.

In the case of the first problem we need to edit our environment variable and add aliases to our login. In order to understand which files to edit, we did a quick search on the .bashrc and .bash_profile files found in a Linux user's home directory. On stackoverflow.com we learn that in essence both these files are the same. The .bashrc file is ran last. In order to run a command from any location on our computer we need to add the path to our 'hello world' script the environment. We also add an alias so that we can run a command such as "ece_3822_d" that will run "ls –la".

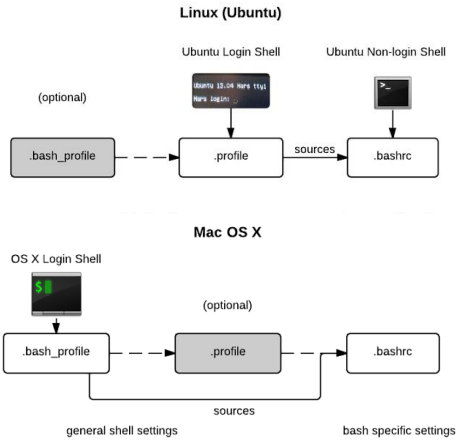

Figure 1: .bashrc, .bash_profile, .profile execution order in MAC OS X and Ubuntu.

<Source: http://dghubble.com/>

The next three problems revolve around the same premise of processing a large data set. We use a combination of 'find' and 'grep' to locate the files that match the search criteria. We also reference the man (man {find, grep}) to learn about the different arguments we can pass to these commands. For example we run

```
find /path/to/data/ -type d | wc –l
```

to obtain a list of directories. The word count command allows to count the total number of new line characters which correspond to the number of directories.

The third part asks us to create a histogram which we reference "Unix for Poets" to learn how to make. To get the contents of the files in our data directory we use the 'grep' command. Then we translate the new all the letters so that they are on separate lines using the 'tr' command. By running the 'uniq –c' command we can create our histogram.

3. RESULTS

All source code and outputs can be found on my GitHub page linked below:

https://github.com/dtrejod/myece3822/tree/master/hw1

3.1. Part 1: Environment Path and Aliases

Editing the .bash_profile and .bashrc is straight forward. We add the path to our /bin folder to the environment variable. This allows us to run our “hello world” script from any location.

```
# .bash_profile

# Get the aliases and functions
if [ -f ~/.bashrc ]; then
        . ~/.bashrc
fi

# User specific environment and startup programs

PATH=$PATH:$HOME/bin:$HOME/projects/github/dtrejod/myece3822/hw1/bin


export PATH
```

```
# .bashrc

# Source global definitions
if [ -f /etc/bashrc ]; then
        . /etc/bashrc
fi

# User specific aliases and functions
alias ece_3822_d="ls -la"
```

Now we demonstrate our script runs from any directory and our alias works:

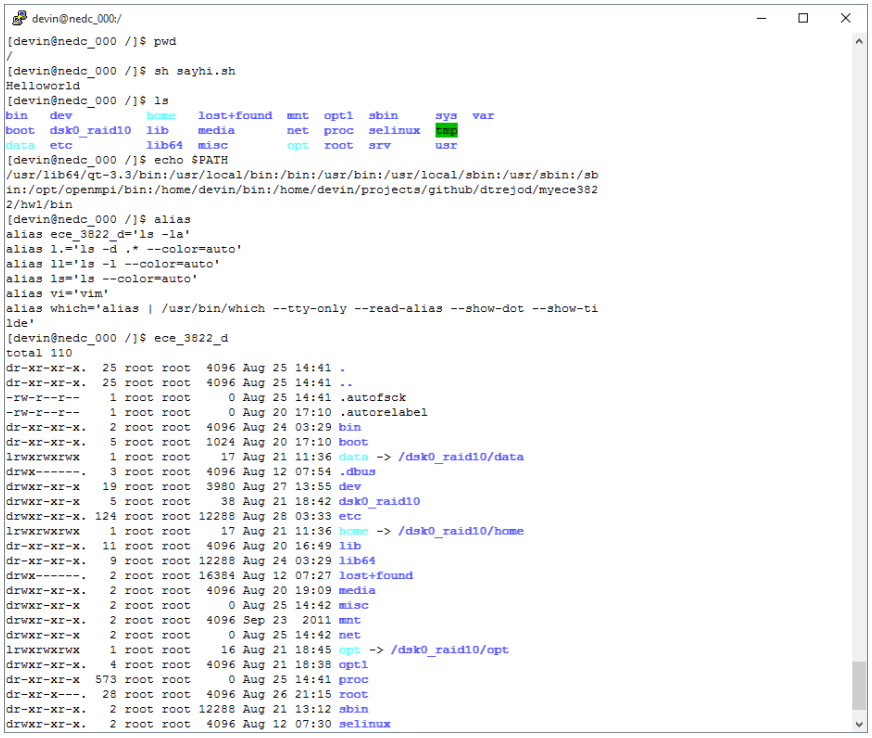

Figure 2: Demonstration of Path and alias.

3.2. Part 2: Patient Names whose first names start with R and last names start with S who had an EEG in the date range 2010-13

We begin by counting the number of directories and files in our /data/ directory using the –type {d,f} arguments respectively. We then move on to find file patients who meet the search criteria by analyzing the file name. If there is an arrangement of characters such as "/R" and "_S" we say that patient first name starts with R and last name starts with S. The year of the EEG sessions is also printed in the in the path.

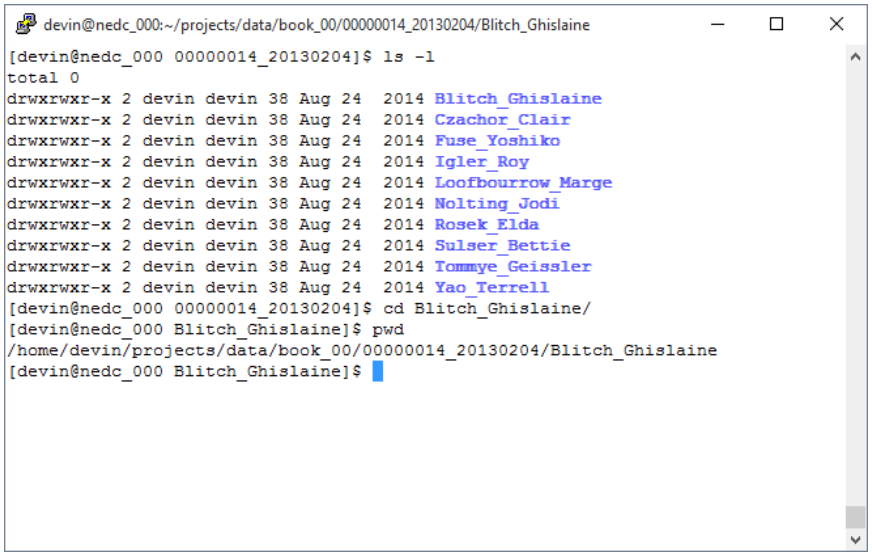

Figure 3: Screengrab showing the structure of the file names within our data folder.

```
DATA_ECE_3822="/home/devin/projects/data/";

# Count the total number of directories
echo "Total number of Dirs:"
find $DATA_ECE_3822 -type d | wc -l

echo ""
# Count the number of
echo "Total number of Files:"
find $DATA_ECE_3822 -type f | wc -l

echo ""
# Count number of  that begin in "R" and "S" last name and 2010-13
echo "Number of Patients Names that begin with 'R' first name and 'S' last name:"
find $DATA_ECE_3822 -type d -path '*/R*' | grep '_S' | grep '_2010\|_2011\|_2012\|_2013' | wc -l
```

3.3. Part 3: EEG reports that contain the word ‘spike’. EEG reports that contain the word ‘seizure’. We then produce a histogram of the words in these reports.

Next we create our histograms. We generate a list of files that produce a file listing of reports that match the desired criteria. We print the counts to the stdout.

From the generated subset lists we now want to produce our histograms. One problem we ran into was an “Augment list too long” for when we pass the file to the ‘cat’ command. After some research we found the ‘xargs’ command which will build commands for you by parsing your input into smaller pieces. The output histogram is saved as a *.hist file.

```
# If ran before clean up previous output
rm -f subseta.hist subsetb.hist subsetc.hist subseta_bi.hist

# grep
#    -i: ignore case
#    -R: recursive (search in sub-directories)
#    -w: match whole word
#    -l: stop searching the file once a match is found (avoid duplicates)
# <Source: http://www.cyberciti.biz/faq/howto-search-find-file-for-text-string/>

echo "Data Reference:"
echo "    Subset A are files with the word 'spike'"
echo "    Subset B are files with the word 'seizure'"
echo "    Subset C are files with the word 'spike' and 'seizure'"
echo ""

echo "Number of files that match Subset A:"
grep -iwlR 'spike' $DATA_ECE_3822 > subseta.list
wc -l subseta.list
echo "Number of files that match Subset B:"
grep -iwlR 'seizure' $DATA_ECE_3822 > subsetb.list
wc -l subsetb.list
echo "Number of files that match Subset C:"
grep -ilRE 'spike.*seizure' $DATA_ECE_3822 > subsetc.list
wc -l subsetc.list

echo ""
# Produce Histogram
echo "Producing histogram of words in Subset A."
xargs cat < subseta.list | tr -sc '[A-Z][a-z]' '[\012*]' > subseta.words
sort subseta.words | uniq -c | sort -nr > subseta.hist
echo "    Histogram saved to subseta.hist"

echo "Producing histogram of words in Subset B."
xargs cat < subsetb.list | tr -sc '[A-Z][a-z]' '[\012*]' | sort | uniq -c | sort -nr >> subsetb.hist
echo "    Histogram saved to subsetb.hist"

echo "Producing histogram of words in Subset C."
xargs cat < subsetc.list | tr -sc '[A-Z][a-z]' '[\012*]' | sort | uniq -c | sort -nr >> subsetc.hist
echo "    Histogram saved to subsetc.hist"
```

3.4. Part 4: For EEG reports that contain the word ‘spike’ produce a histogram of bi-grams.

For the last part we want to create a histogram but with bigrams (every two adjacent words). To accomplish this we take our subseta.words (a listing of all words in our specified subset A) and shift them down by one word. The shift is accomplished by running the “tail –n +2” command. We then combine the two lists by using the paste command and run the same histogram script we used previously.

```
# Assignment hw1 (part4):
# For subset A, produce a histogram of all two-word sequences that occur in this
#    subset of the database
echo "Producing histogram of bigrams in Subset A."

## Create a list of subseta.words+1
tail -n +2 subseta.words > subseta.nextwords
## Merge the two words lists together and create histogram
paste subseta.words subseta.nextwords | sort | uniq -c | sort -nr > subseta_bi.hist
echo "    Histogram saved to subseta_bi.hist"

# File cleanup
rm -f subseta.list subsetb.list subsetc.list subseta.words subseta.nextwords
```

3.5. Output

Standard output after running hw1.sh.

```
Total number of Dirs:
110022
Total number of Files:
200000
Number of Patients Names that begin with 'R' first name and 'S' last name:
248
Data Reference:
   Subset A are files with the word 'spike'
   Subset B are files with the word 'seizure'
   Subset C are files with the word 'spike' and 'seizure'
Number of files that match Subset A:
15955 subseta.list
Number of files that match Subset B:
63349 subsetb.list
Number of files that match Subset C:
4506 subsetc.list

Producing histogram of words in Subset A.
   Histogram saved to subseta.hist
Producing histogram of words in Subset B.
   Histogram saved to subsetb.hist
Producing histogram of words in Subset C.
   Histogram saved to subsetc.hist
Producing histogram of bigrams in Subset A.
   Histogram saved to subseta_bi.hist
```

First 10 lines of the histograms (full list on my GitHub: https://github.com/dtrejod/myece3822/tree/master/hw1):

Subseta.hist	**Subsestb.hist**	**Subsetc.hist**	**Subseta_bi.hist**
225289 the	673541 the	80669 the	42202 the patient
146612 and	467403 of	52060 and	38408 of the
146610 of	396454 and	43221 of	28229 there is
99756 with	318213 a	35643 to	27396 in the
99628 a	316648 is	30540 with	26316 the record
92897 in	313817 with	30330 a	26273 spike and
92620 is	304079 in	28196 is	23156 and wave
90381 to	238269 to	23393 in	19861 with a
67242 EEG	230377 EEG	22524 EEG	19225 the left
51127 patient	171348 was	22236 at	18914 was performed

The last histogram is a bigram of subset A.

4. ANALYSIS

To show our scripts make sense we concentrate the script to only run on a subsection of the data. The specific folder path is listed below. The contents of the folder can be seen in Figure 3:

"data/book_00/00000014_20130204/"

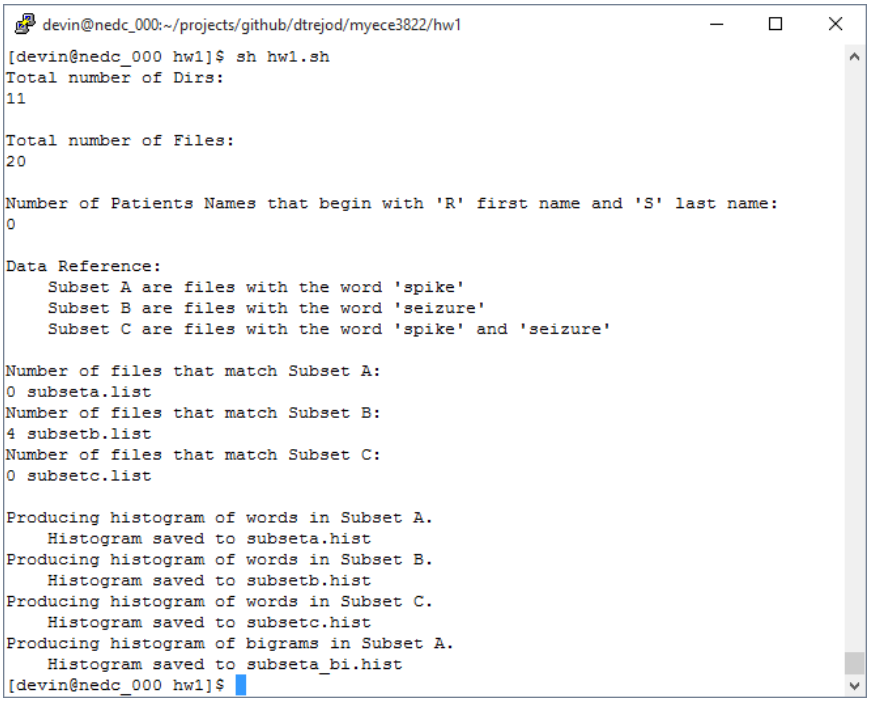

Figure 4: Script ran on smaller dataset higher /data/

From Figure 3 we see we have a hierarchy directory *"00000014_20130204"* with 10 sub directories. Our output thus shows 11 directories. There are 20 files as well inside this specific directory.

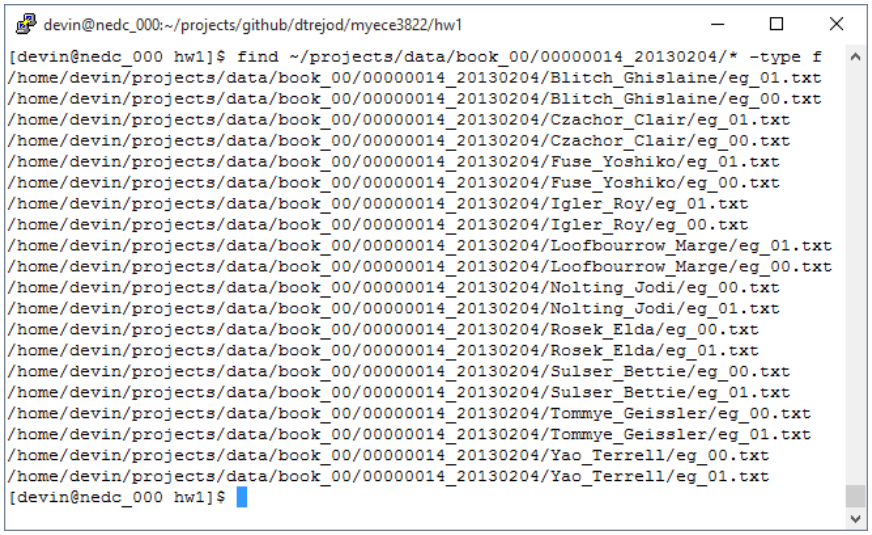

Figure 5: 20 Files inside our smaller data set.

We can see now that now of these directories contain a person whose first name begins with ‘R’ and last name begins with ‘S’ thus our count returns zero. We can also see that none of these files contain whole word ‘spike’ but four reports do contain the word ‘seizure’.

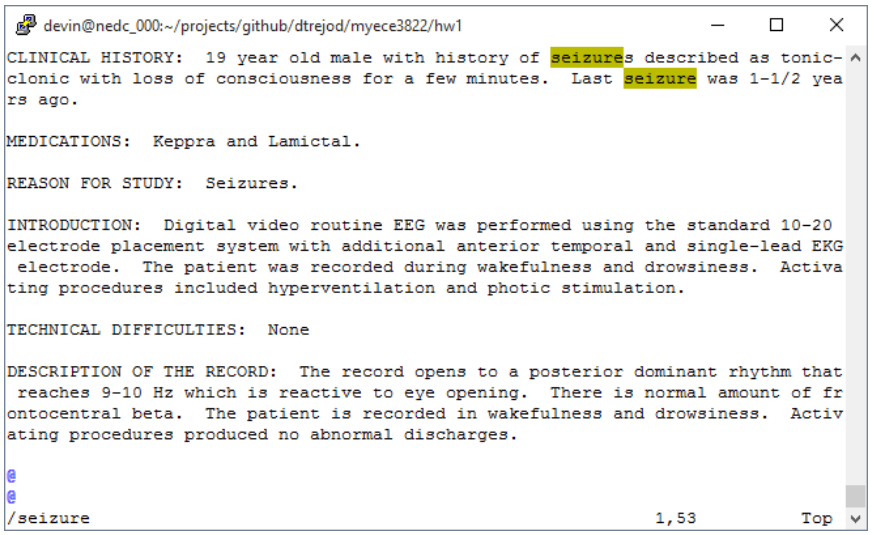

Figure 6: Found file that contains the word seizure.

In conclusion we have shown how powerful commands like ‘find’, and ‘grep’ can be. They work fast even when analyzing large datasets such as ours (size of 823MB).