**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

September 1, 2015

prepared by:

Robert Irwin  
Email: robert.irwin@temple.edu

# Problem

In this assignment we are analyzing a data set using the command line. More specifically, we will be counting the total number of directories and files in the data set, searching for files that contain specific words, and creating histograms of all the words in these files. This will be done using a shell script.

# Approach

The first step of the assignment was to place an executable in the bin directory, and change our default path so we can execute the script in any directory. This was done by editing the .bash\_profile file in the home directory of our local machine.

Then we began to analyze the data set. Our first task was, “Count the total number of directories in the database. Count the total number of files. The code is shown below.

# Beginning searching and counting for

# all files and directories

find "../data" -type f > files.txt

FILECOUNT="$(wc -l < files.txt)"

find "../data" -type d > output.txt

DIRCOUNT="$(wc -l < output.txt)"

echo "File count: " $FILECOUNT

echo "Directory count: " $DIRCOUNT

Here we use find with the option [-type f] to search the path for all files located under that path. The output of find is written to files.txt. The output is written with each file separated by a newline character. Therefore, we can use the wc command, with the –l option to count the lines in the file “files.txt”. The output of this is saved to FILECOUNT. The process is repeated for the directories.

Then we were to, “Count the number of sessions that have a first name that begins with “R” and a last name that begins with “S” and occurred between 2010 and 2013.” The code is shown below.

# Search for dates and first/last/ names

# This is done in steps and each of the search results

# of each search are written to a txt file to

# verify the output

grep "/R" output.txt > out1.txt

grep "\_S" out1.txt > out2.txt

grep "2010\|2011\|2012\|2013" out2.txt > final.txt

FILECOUNT="$(wc -l < final.txt)"

echo "Files that match search criteria: " $FILECOUNT

To complete this task, grep was used. Grep searches for a user defined pattern in a directory. Our search pattern was defined based on the format of the files in files.txt. These commands could have been pipelined, but for debugging purposes, we filtered out files based on one search criteria at a time.

Then we had to, “Count the number of files in the database that have the word “spike” occurring at least once. Call this subset “A”. Repeat this for files containing the word “seizure,” which we will refer to as subset B. Produce a histogram of the distribution of these words in subsets A and B, listing the most frequently occurring first and least frequently occurring last. Show both the absolute counts and the percentage (often referred to as the cumulative distribution).” The code for this is shown below.

grep -iRlw "spike" /Users/robertirwin/test/test1\* > suba.txt

SPIKE="$(wc -l < suba.txt)"

echo "Files with <spike>: " $SPIKE

grep -iRlw "seizure" /Users/robertirwin/test/test1\* > subb.txt

SEIZURE="$(wc -l < subb.txt)"

echo "Files with <seizure>: " $SEIZURE

echo "Creating Histogram of subset A..."

xargs cat < suba.txt | tr -sc '[A-Z][a-z]' '[\012\*]' > suba.words

sort -f suba.words | uniq -ci | sort -nr > suba.hist

#pdf

sed 's/^ \*//g' < suba.hist > suba1.hist

cut -f 1 -d ' ' suba1.hist > num.list

#sum up all histogram entries

sum=$(awk '{SUM += $1} END { print SUM}' num.list)

file='num.list'

lines=`cat $file`

#calculate perentages

for num in $lines;

do

percent=`awk 'BEGIN{printf("%0.2f", '$num' / '$sum' \*100)}'`

echo "$percent%" >> per.list

done

#now we simply paste the lists

paste suba.hist per.list > hista.hist

rm per.list num.list

echo "Done... saved in hista.hist"

echo "Creating Histogram of subset B..."

xargs cat < subb.txt | tr -sc '[A-Z][a-z]' '[\012\*]' | sort -f | uniq -i -c | sort -nr > subb.his\

t

#pdf

sed 's/^ \*//g' < subb.hist > subb1.hist

cut -f 1 -d ' ' subb1.hist > numb.list

#sum up all histogram entries

sum=$(awk '{SUM += $1} END { print SUM}' numb.list)

file='numb.list'

lines=`cat $file`

for num in $lines;

do

percent=`awk 'BEGIN{printf("%0.2f", '$num' / '$sum' \*100)}'`

echo "$percent%" >> per.list

done

#now we simply paste the lists

paste subb.hist per.list > histb.hist

rm per.list num.list

echo "Done... saved in histb.hist"

In order to compete this task, we had to use the cat command. The cat command concatenates text files together. The output of this is passed to [tr] with the options, -s and –c. Tr is a command that translates or deletes characters. The –c option returns the first compliment of SET1, which is the ‘[A-Z][a-z]’ seen in the command. The –s replaces each occurrence of a repeated character listed in SET1 with a single character. This output is then piped to sort which is piped to “uniq” which removes duplicate lines from a file, while –c option prefixes the line with the number of occurrences of the each line. This output is then passed to sort again with the –n and –r options. The –n options sorts according to numeric value, and –r orders this list in reverse, so that is in descending order. This is then written to a text file. The xargs command is critical to the code. Before xargs was called, the list of files was too long for the cat command to handle. Therefore, xargs is called. Xargs is a function used to reads items from the standard input, and executes the command one or more times with any initial-arguments followed by items read from standard input. Xargs enables the cat command to handle a larger amount of files by breaking the standard input into smaller pieces.

To produce the pdf, the command sed and cut were used. Sed was utilized to remove all whitespace in the beginning of the line. This was essential to the cut command, which cut the first field out of the file and stored it to a new file. A field is every character up to a delimiter, which we specified to be a space. These numbers were then summed up using the awk command, and a for loop was run to divide each number by sum and multiply the result by 100. Then paste was utilized to append the percentages behind the words of the histogram.

Our last assignment was to, “for subset A, produce a histogram of all two-word sequences that occur in this subset of the database”. The code is shown below.

# 4) For Subset A, produce a histogram of all two-word sequences that

# occur in this subset of the database

echo "Creating Histogram of all two word sequences in subset A..."

tail -n +2 suba.words > suba.next

#Merge the two lists

paste suba.words suba.next | sort -f | uniq -ci | sort -nr > bi.hist

#pdf

sed 's/^ \*//g' < bi.hist > bi1.hist

cut -f 1 -d ' ' bi1.hist > num.list

#sum up all histogram entries

sum=$(awk '{SUM += $1} END { print SUM}' num.list)

file='num.list'

lines=`cat $file`

#calculate perentages

for num in $lines;

do

percent=`awk 'BEGIN{printf("%0.2f", '$num' / '$sum' \*100)}'`

echo "$percent%" >> per.list

done

#now we simply paste the lists

paste bi.hist per.list > histbi.hist

echo "Done... saved in histbi.hist"

#Clean up files

rm output.txt out1.txt out2.txt final.txt files.txt suba.txt subb.txt suba.words suba.next per.li\

st bi.hist num.list suba1.hist subb1.hist suba.hist subb.hist bi1.hist

The first command in this code block is tail. By default, the tail command prints the last 10 lines of a file. The option [–n +k] prints out the rest of the file starting with the kth line. In this line of code we are essentially creating a text file to push next to our text file with all the words. Because the .next file has all the same words, except the first one, when these files are pushed together the result will be a file with all the words that appear next to each other. Paste accomplishes this task for us, and then we pass this to our normal sorting pipe.

# Results

Below shows the changes made to the default path in the .bash\_profile file that allows us to execute our script in any directory.

export PATH="/Users/robertirwin/anaconda3/bin:/Users/robertirwin/bin:$PATH"

To illustrate that we can in fact execute our hello world script in any directory, we show a sequence of command line inputs below.

Roberts-MacBook-Pro:hw1 robertirwin$ cd

**Roberts-MacBook-Pro:~ robertirwin$ cd bin**

**Roberts-MacBook-Pro:bin robertirwin$ ls -l**

**total 8**

**-rw-r--r-- 1 robertirwin staff 28 Aug 24 14:27 run.sh**

**Roberts-MacBook-Pro:bin robertirwin$ sh run.sh**

**Hello World**

Roberts-MacBook-Pro:bin robertirwin$ cd

Roberts-MacBook-Pro:~ robertirwin$ cd software\_tools/hw1

Roberts-MacBook-Pro:hw1 robertirwin$ ls -la

total 221080

-rw-r--r-- 1 robertirwin staff 29445110 Aug 31 09:13 -c

drwxr-xr-x 17 robertirwin staff 578 Aug 31 10:12 .

drwxr-xr-x 7 robertirwin staff 238 Aug 30 12:36 ..

-rw-r--r-- 1 robertirwin staff 343729 Aug 31 09:45 bi.hist

-rw-r--r-- 1 robertirwin staff 3513 Aug 31 10:12 count.sh

-rw-r--r-- 1 robertirwin staff 3514 Aug 31 09:26 count.sh~

-rw-r--r-- 1 robertirwin staff 11755976 Aug 31 09:30 files.txt

-rw-r--r-- 1 robertirwin staff 11993 Aug 31 09:30 final.txt

-rw-r--r-- 1 robertirwin staff 233604 Aug 31 09:30 out1.txt

-rw-r--r-- 1 robertirwin staff 19836 Aug 31 09:30 out2.txt

-rw-r--r-- 1 robertirwin staff 5218355 Aug 31 09:30 output.txt

-rw-r--r-- 1 robertirwin staff 40174 Aug 31 09:38 suba.hist

-rw-r--r-- 1 robertirwin staff 29458134 Aug 31 09:44 suba.next

-rw-r--r-- 1 robertirwin staff 1432408 Aug 31 09:33 suba.txt

-rw-r--r-- 1 robertirwin staff 29458143 Aug 31 09:37 suba.words

-rw-r--r-- 1 robertirwin staff 59934 Aug 31 09:44 subb.hist

-rw-r--r-- 1 robertirwin staff 5687600 Aug 31 09:36 subb.txt

**Roberts-MacBook-Pro:hw1 robertirwin$ sh run.sh**

**Hello World**

Notice that run.sh (hello world script) is in the bin directory. We then switch to a different directory, namely software\_tools/hw1. After listing all the files in that directory, it is evident that run.sh is not in that directory. Because we changed our default path to look in the bin directory, we see that when we attempt to execute run.sh in the software\_tools/hw1 directory, the script can be executed.

Now we will look at counting all the files and directories in the data set. Using the code described in the approach section, the following output was obtained.

File count: 200000

Directory count: 110022

The next step of this assignment required us to find all files containing the word “spike”. Count.sh produced the following result.

Files with <spike>: 15955

Files with <seizure>: 63349

Now we will look at the results of the histogram producing code. Keep in mind that this is only a small portion of the histogram of all words in subset A. Because the histogram for subset B required the same code, subset B’s histogram will not be shown.

225289 THE 4.9048%

146612 and 3.1919%

146610 OF 3.1919%

99756 With 2.1718%

99628 A 2.1690%

92897 In 2.0225%

92620 is 2.0165%

90381 TO 1.9677%

67242 EEG 1.4639%

51127 PATIENT 1.1131%

49346 WAs 1.0743%

46695 There 1.0166%

45909 At 0.9995%

42951 SEIZURES 0.9351%

41414 This 0.9016%

38426 activity 0.8366%

37960 were 0.8264%

37168 CLINICAL 0.8092%

36848 Spike 0.8022%

34673 Left 0.7549%

33092 performed 0.7205%

32419 As 0.7058%

31601 Sleep 0.6880%

31128 wave 0.6777%

29820 RECORD 0.6492%

28665 Hz 0.6241%

28461 are 0.6196%

26843 No 0.5844%

26555 FOR 0.5781%

25200 HISTORY 0.5486%

25081 Right 0.5460%

24295 Temporal 0.5289%

23815 which 0.5185%

23755 seen 0.5172%

21759 waves 0.4737%

21560 ABNORMAL 0.4694%

21465 be 0.4673%

21077 Electrode 0.4589%

20710 Stimulation 0.4509%

20656 An 0.4497%

20643 Hyperventilation 0.4494%

20230 SEIZURE 0.4404%

20189 but 0.4395%

20079 Not 0.4371%

19450 Bursts 0.4235%

19360 On 0.4215%

We notice that it is in descending order, which is what we want.

It is important to note that when creating the histogram, our code ignores case. This was accomplished using the –f option on sort, along with the –i option on uniq.

Now we will show the histogram of all two word sequences of subset A.

42202 The patient 0.9188%

38408 OF THE 0.8362%

28229 There is 0.6146%

27396 In the 0.5964%

26316 THE RECORD 0.5729%

26273 Spike and 0.5720%

23156 and wave 0.5041%

19861 with a 0.4324%

19225 The left 0.4186%

18914 WAs performed 0.4118%

17938 Photic stimulation 0.3905%

16576 DESCRIPTION OF 0.3609%

16115 CLINICAL CORRELATION 0.3508%

16090 CLINICAL HISTORY 0.3503%

15844 wave activity 0.3449%

15145 year old 0.3297%

15091 electrode placement 0.3286%

14717 is a 0.3204%

14713 EEG WAs 0.3203%

13128 The EEG 0.2858%

12691 Due to 0.2763%

12512 Sharp waves 0.2724%

12504 using standard 0.2722%

12475 Digital video 0.2716%

12305 The right 0.2679%

12234 video EEG 0.2664%

12007 anterior temporal 0.2614%

11799 INTRODUCTION Digital 0.2569%

11161 This EEG 0.2430%

11107 Patient was 0.2418%

10932 THE RECORDING 0.2380%

10911 In wakefulness 0.2375%

10674 On the 0.2324%

10671 Abnormal EEG 0.2323%

10345 standard system 0.2252%

10186 EEG due 0.2218%

10067 This is 0.2192%

10053 HISTORY year 0.2189%

9935 Bursts of 0.2163%

9875 as well 0.2150%

9844 to be 0.2143%

9841 Activating procedures 0.2143%

9681 to The 0.2108%

9605 well as 0.2091%

9572 RECORD The 0.2084%

9570 which is 0.2084%

# Analysis

To verify part 2 of the assignment, we will make a test directory with sub-directories and dummy files. We will then run the code on this main directory and ensure that the code produces the correct result. This process is shown below.

Roberts-MacBook-Pro:~ robertirwin$ cd test

Roberts-MacBook-Pro:test robertirwin$ ls -l

total 16

drwxr-xr-x 2 robertirwin staff 68 Aug 26 22:39 ad

-rw-r--r-- 1 robertirwin staff 21 Aug 26 22:40 another.txt

-rw-r--r-- 1 robertirwin staff 519 Aug 26 22:38 count.sh

drwxr-xr-x 3 robertirwin staff 102 Aug 26 22:38 test1

Roberts-MacBook-Pro:test robertirwin$ cd ad

Roberts-MacBook-Pro:ad robertirwin$ ls -l

Roberts-MacBook-Pro:ad robertirwin$ cd ..

Roberts-MacBook-Pro:test robertirwin$ cd test1

Roberts-MacBook-Pro:test1 robertirwin$ ls -l

total 0

drwxr-xr-x 3 robertirwin staff 102 Aug 26 22:38 test2

Roberts-MacBook-Pro:test1 robertirwin$ cd test2

Roberts-MacBook-Pro:test2 robertirwin$ ls -l

total 8

-rw-r--r-- 1 robertirwin staff 9 Aug 26 22:38 test.txt

Roberts-MacBook-Pro:test2 robertirwin$

From the command line sequence above, we know that our code should find four directories (test, ad, test1, test2) and three files (another.txt, count.sh, test.txt). Count.sh produces the following result:

File count: 3

Directory count: 4

Therefore, we are confident that our code is in fact producing the correct result.

Now we will analyze the first/last name search sequence of the code. We will use our same test directory to test this portion of the code. As seen in the contents of our test directory, there are no files matching this search criterion. When we run the code, count.sh, we expect an output of 0. The result is shown below.

Roberts-MacBook-Pro:test robertirwin$ sh count.sh

File count: 3

Directory count: 4

**Files that match search criteria: 0**

We have shown that our counter will not count any files that does not match the search criterion. To prove that our code fully works, a portion of the list of the 248 files that it counted is shown below.

../data/book\_10/00002503\_20101227/Rauhe\_Samantha

../data/book\_10/00003245\_20101107/Ravenscroft\_Sau

../data/book\_10/00004673\_20101114/Ravo\_Scottie

../data/book\_10/00005640\_20101213/Raudales\_Salena

../data/book\_10/00007131\_20101227/Rauh\_Sam

../data/book\_10/00007364\_20101121/Ratz\_Sabine

../data/book\_10/00007481\_20101121/Ratulowski\_Sabina

../data/book\_10/00009468\_20101213/Rauda\_Sal

../data/book\_10/00009567\_20101107/Ravenscraft\_Saturnina

../data/book\_11/00000011\_20130316/Reckley\_Sierra

../data/book\_11/00000377\_20130308/Reash\_Shella

../data/book\_11/00000462\_20130426/Raybourn\_Shakira

../data/book\_11/00000477\_20130430/Rayo\_Shanta

../data/book\_11/00000489\_20130308/Reavis\_Sherie

../data/book\_11/00000527\_20130312/Rayson\_Shantelle

../data/book\_11/00000555\_20101213/Reauish\_Shera

../data/book\_11/00000602\_20130223/Rawi\_Seema

../data/book\_11/00000661\_20130310/Reddinger\_Son

../data/book\_11/00000676\_20130428/Rayno\_Shannan

../data/book\_11/00000698\_20130320/Redell\_Sophie

../data/book\_11/00000818\_20130315/Reckleben\_Sidney

../data/book\_11/00000866\_20130228/Redmond\_Sterling

../data/book\_11/00000888\_20130423/Raysor\_Shanti

../data/book\_11/00000905\_20130311/Reagan\_Shaun

../data/book\_11/00000946\_20130229/Reader\_Sharonda

../data/book\_11/00000955\_20130227/Recar\_Shirlee

../data/book\_11/00000984\_20130309/Reddicks\_Solomon

As we can see, all the files match the search criteria.

To test the file count, as well as the histogram, the code was edited to search a particular directory containing two files. The contents of the files, are shown below.

Roberts-MacBook-Pro:test1 robertirwin$ more test.txt

spike

seizure

I

am

a

text

file

he

he

He

she

She

a

he

A

the

will

AM

am

he

he

he

he

he

he

he

he

he

he

he

he

he

he

Roberts-MacBook-Pro:test1 robertirwin$ more dummy.txt

dsjhfksjdhfka

In the files above, we see that only 1 file contains the words seizure and spike. The file count that matches the search, as well as the histogram is shown below.

Files with <spike>: 1

Files with <seizure>: 1

Creating Histogram of subset A...

Done... saved in hista.hist

Roberts-MacBook-Pro:test robertirwin$ more hista.hist

18 He 54.55%

3 AM 9.09%

3 A 9.09%

2 She 6.06%

1 will 3.03%

1 the 3.03%

1 text 3.03%

1 spike 3.03%

1 seizure 3.03%

1 file 3.03%

1 I 3.03%

Finally, we will show the histogram of all words that are next to each other in the test.txt file.

15 he He 45.45%

1 will AM 3.03%

1 the will 3.03%

1 text file 3.03%

1 spike seizure 3.03%

1 she She 3.03%

1 seizure I 3.03%

1 he A 3.03%

1 he 3.03%

1 file he 3.03%

1 am he 3.03%

1 am a 3.03%

1 a text 3.03%

1 a he 3.03%

1 She a 3.03%

1 I am 3.03%

1 He she 3.03%

1 AM am 3.03%

1 A the 3.03%