ECE 282 Lab06

Lab Due: 11 pm, February 28th, 2019

1. Command practice

Tutorial: http://eecs.mines.edu/Courses/csci274/Content/09_output.html
Scripting help: https://www.tutorialspoint.com/unix/unix-vi-editor.htm

What do the following commands do?

a) mount / umount (**Hint: http://www.tutorialspoint.com/unix/unix-file-system.htm**)

b) tty

c) stty

d) seq

e) head

f) tail

g) echo

h) sed

Write down the commands that can do the following tasks:

a) Create a file s1.txt that has 100 lines, each line is a number from 1 to 100

b) Create a file s2.txt that has 50 lines with the pattern 2 4 6… 100
Hint: man seq

c) Output the first 5 lines of s1.txt to the **terminal**

d) Output the last 5 lines of s2.txt to a file **temp.txt**

e) Output the lines 5 to 9 **from the file s1.txt** to the terminal
Hint: head pipeline to tail

f) Filter the input file (acmart.html) to find the lines including link to .png files
Hing: try grep png

2. Some sed (from http://www.grymoire.com/Unix/Sed.html#uh-1)

Sed is the ultimate stream editor. If that sounds strange, picture a stream flowing through a pipe. Okay, you can't see a stream if it's inside a pipe. That's what I get for attempting a flowing analogy. You want literature, read James Joyce.

Anyhow, sed is a marvelous utility. Unfortunately, most people never learn its real power. The language is very simple, but the documentation is terrible. The Solaris on-line manual pages for sed are five pages long, and two of those pages describe the 34 different errors you can get. A program that spends as much space documenting the errors as it does documenting the language has a serious learning curve.

Do not fret! It is not your fault you don't understand sed. I will cover sedcompletely. But I will describe the features in the order that I learned them. I didn't learn everything at once. You don't need to either.

Sed has several commands, but most people only learn the substitute command: s. The substitute command changes all occurrences of the regular expression into a new value. A simple example is changing "day" in the "old" file to "night" in the "new" file:

```
sed s/day/night/ <old >new
```

Or another way (for UNIX beginners),

```
sed s/day/night/ old >new
```

and for those who want to test this:

```
echo day | sed s/day/night/
```

This will output "night".

If you are replacing simple texts, you can avoid using quotes. Otherwise, your string need to be enclosed with quotes:

```
sed 's/day/night/' <old >new
```

I must emphasize that the sed editor changes exactly what you tell it to. So if you executed

```
echo Sunday | sed 's/day/night/'
```

This would output the word "Sunnight" because sed found the string "day" in the input.

Another important concept is that sed is line oriented. Suppose you have the input file:

```
one two three, one two three
four three two one
one hundred
```

and you used the command

```
sed 's/one/ONE/' <file
```

The output would be

```
ONE two three, one two three
four three two ONE
ONE hundred
```

And some more:

Backslash as the escape character. Just like strings in C.

& as matched string. Search a string and at some characters

```
sed 's/abc/(abc)/' <old >new
sed 's/abc/(&)/' <old >new
```

* for multiple occurrence of a pattern

```
sed 's/[a-z]*/(&)/' <old >new
```

. for any character

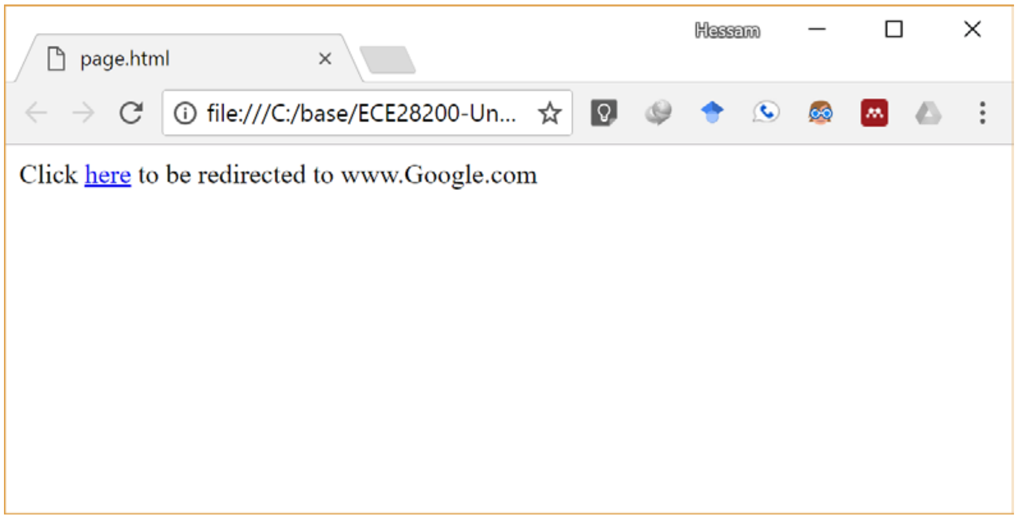
```
sed 's/a.*c/(&)/' <old >new # find and replace any string
starting with ‘a’ and ending with ‘c’
```

3. Lab assignment

In HTML, links are indicated by the `<a href="PATH">TEXT</a>`. PATH is a relative or absolute path to a file or resource and TEXT is the text that is going to be displayed in for that link. Ex:

```
Click <a href="www.google.com">here</a> to be redirected to www.Google.com
```

To see the result, just copy the above line in a text file, save it with html extension (file.html) and open it with a browser. In the following is the same webpage, opened with Chrome,



Click here to be redirected to www.Google.com

Get the acmart.html from Canvas. This file is downloaded from the following link:

`http://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/acmart/`

Extract all web links from this file, convert them to absolute paths (instead of relative), and store them in a file named links.

The links in this file are like the following:

```
<a href="ACM-Reference-Format.dbx">ACM-Reference-Format.dbx</a>
```

These paths are relative to the current directory on the server. However, for our case, you need to convert the links to absolute paths, preceding with the link of the webpage where the file was downloaded from. The above html link must be converted to the following.

```
http://ctan.math.utah.edu/ctan/tex-archive/macros/latex/contrib/acmart/ACM-
Reference-Format.dbx
```

For this, you need to first filter all the line that include the string ‘`href=`’. Use grep for this.

Then use sed and proper string substitution multiple times to rip away any string before ‘`href="`’ and after `</a>`.

Finally redirect the generated lines to the file `links`. The generated output is already on Canvas, named `links-lab05`. You could compare you output against contents of this file.

Include the script(s) in a bash file, so that after running the bash file, the input file will be processed and the corresponding output be generated.

To make a bash script file:

1) Make a text file (ex. Lab06.sh)
2) First line must be #!<path to bash program>
   Usually bash is located at /bin/bash, so the first line of this line will be #!/bin/bash

3) Set the executable permission of the bash file
   chmod +x lab5.sh (this command will set x permission of the file for all fields)
4) Include the Linux commands in the bash script file
5) Run the file
   ./lab06.sh

You could just see the `sample-script.sh` bash script on canvas and modify it. Just remove the Unix commands and insert you own command instead. Or you could do every other thing that you want. Explore and be creative.

Submit all your files to Canvas as a single ZIP file.