**CSC 209 UNIX Tools**

|  |  |  |
| --- | --- | --- |
|  | **Assignment 8** |  |

**Objectives: Objectives: UNIX Commands (grep and sed) and Regular Expression**

**Note**: **Read** the entire assignment carefully and carry out the following tasks one by one. For some steps, I may provide the required UNIX command. For others, identify the UNIX commands you need. **UNIX is case sensitive**. Create folders and files exactly as indicated below, paying special attention to case. Complete the report as indicated below and submit the same. I do not want to see copy-and-paste of your computer output in your report. By requiring you to write I am hoping that you will remember the UNIX commands used and the results observed.

1. Login to Courses server using your Webmail Net-ID and password.

Take a note of how the command prompt placed by the shell appears and report below:

Shell prompt: [51] [etay1@courses2016:~]$

Type an appropriate command to print the working directory.

Command to print the working directory: pwd

Your login directory: /home/etay1

Use an appropriate command to switch to the **csc209** folder which you created in Assignment # 1.

Command to switch from login directory to **csc209**: cd csc209

1. Complete the following steps:

Determine the current working directory.

Command to print the working directory: pwd

Your current working directory: /home/etay1/csc209

You should currently be located in **csc209**. Create a new folder named **asgn08** under **csc209**. Set the access permissions for **asgn08** as **rwx --- ---** by typing a command:

**chmod 700 asgn08**

Command to create the **asgn08** folder: mkdir asgn08

Command to set permissions, even if not needed: chmod 700 asgn08

Move to **asgn08** and determine the current working directory.

Command to switch from **csc209** directory to **asgn08**: cd asgn08

Command to print the working directory: pwd

Your current working directory: /home/etay1/csc209/asgn08

Shell prompt: [61] [etay1@courses2016:~/csc209/asgn08]$

**From now on, unless explicitly permitted, you should not move away from this asgn08 directory.**

**If you happen to log out in the middle of the exercise, be sure to use appropriate command(s)**

**and move to asgn08 before you continue.**

2. With a single command, without changing directory, copy the file **test01** in my directory **~nyu/csc209/asgn08** to your directory **asgn08**.

Command to copy **test01**: cp ~nyu/csc209/asgn08/test01 .

Verify the success of copying with **ls -l** command for this file. Write down the output.

Command to verify success: ls -l

Output: -rw------- 1 etay1 domain users 273 Nov 25 17:15 test01

3. Display the file with line number and with a **$** character printed after the last character in each line.

Command to display file with line number and **$** at end: grep -n $ test01

Number of lines in the file displayed: 20

The file contains several lines. Each line contains a sequence of characters with a newline (**\n**)character as the last one to mark the end of the line. In the display of the file using the **cat** command above, a **$** sign appears to show the end of the line so trailing blanks if any can be easily recognized. Do not count the **$** sign as a character in the line.

Below I ask you to visually identify and count lines with certain characteristics so you can check the results of your **egrep** commands on pages 4 to 6.

Visually count the number of lines containing no characters, not even blank characters.

(Step 4, p. 4) Count of the number of lines with no characters: 9

Lines with no characters (empty lines) (list number): 4, 6, 7, 8, 13, 15, 16, 19 ,20

(Step 5) Count of the number of lines with one or more characters: 10

Lines with one or more characters (non-empty lines) (list number): 1, 2, 3, 5, 9, 11, 12, 14, 17, 18

(Step 6) Count of the number of lines with only blank character(s): 1

Line with only blank character(s) (list number): 10

(Step 7, p. 5) Count of the number of lines with no blank character: 4

Line with no blank character (list number): 1, 9, 11, 14

Visually count the number of lines with leading blank(s), and the number of lines with trailing blank(s). A line has leading blank(s) if the first character is blank. A line has trailing blank(s) if the last character is blank. A line with one blank character has a leading and trailing blank.

(Step 8) Count of the number of lines with leading blank(s): 6

Lines with leading blank(s) (list number): 2, 3, 5, 12, 17, 18

(Step 9) Count of the number of lines with trailing blank(s): 11

Lines with trailing blank(s) (list number): 1, 2, 3, 5, 9, 10, 11, 12, 14, 17, 18

Visually count lines with both leading and trailing blanks.

(Step 10, p. 6) Count of the number of lines with both leading and trailing blank(s): 6

Lines with both leading and trailing blank(s) (list number): 2, 3, 5, 12, 17, 18

While writing the regular expressions below, **indicate a space (blank) character as 🗆**

Also, note that while writing the regular expression, there is no need to quote it. When using the regular expression in **grep** or **egrep** command, do quote them with single quotes.

I recommend you use **egrep** even if I say **grep**.

4. Give one **grep** command to display, with line number, only empty lines (lines with no characters, not even blanks) in the file. Write a command with pipe to count the number of such empty lines.

Regular expression to be used to display empty line(s): egrep ‘^$’ test01

Command to display empty lines, with line number: egrep -n ‘^$’ test01

Line number for the lines listed: 4, 6, 7, 13, 15, 16, 19

Command to count empty lines:

egrep -n ‘^$’ test01

|

egrep ‘^$’ test01

Count obtained: 7

(Check line numbers and count against your visual count on page 3)

5. Give one **grep** command to display, with line number, only non-empty lines (lines with one or more characters) in the file. Write a command with pipe to count the number of such non-empty lines.

Regular expression to be used to display non-empty line(s): egrep -v ‘^$’ test01

Command to display non-empty lines, with line number: egrep -nv ‘^$’ test01

Line number for the lines listed: 1, 2, 3, 5, 8, 9, 10, 11, 12, 14, 17, 18, 20

Command to count non-empty lines: egrep -nv '^$' test01 | egrep -cv '^$' test01

Count obtained: 13

(Check line numbers and count against your visual count)

6. Give one **grep** command to display, with line number, only non-empty lines in the file which contain only blank(s). Write a command with pipe to count the number of such non-empty lines with only blank(s) in them.

Regular expression to display non-empty line(s) with only blank(s): egrep '^[[:blank:]]\*$' test01

Command to display blank non-empty lines, with line number: egrep -n ‘^[[:blank:]]\*$ ' test01

Line number for the lines listed: 4, 6, 7, 8, 10, 13, 15, 16, 19, 20

Command to count blank non-empty lines: egrep -n '^[[:blank:]]\*$ ' test01 | egrep '^[[:blank:]]\*$' test01

Count obtained: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(Check line numbers and count against your visual count)

7. Give one **grep** command to display, with line number, only lines in the file which contain **no** blanks. Write a command with pipe to count the number of such lines which contain no blanks.

Regular expression to be used to display line(s) with **a** blank: egrep -n ‘ ‘ test01

Command to display lines with **no** blanks: egrep -nv ‘ ‘ test01

Line number for the lines listed: 4, 6, 7, 9, 13, 15, 16, 19

Command to count lines with no blanks: egrep -vc ' ' test01

Count obtained: 8

(Check line numbers and count against your visual count)

8. Give one **grep** command to display, with line number, only lines in the file which contain leading blank(s). Write a command with pipe to count the number of such lines which contain leading blank(s).

Regular expression to be used to display line(s) with leading blank(s): grep '^ ' test01

Command to display lines with leading blank(s): grep -n '^ ' test01

Line number for the lines listed: 2, 3, 5, 8, 10, 12, 17, 18, 20

Command to count lines with leading blank(s): grep -c '^ ' test01

Count obtained: 9

(Check line numbers and count against your visual count)

9. Give one **grep** command to display, with line number, only lines in the file which contain trailing blank(s). Write a command with pipe to count the number of such lines which contain trailing blank(s).

Regular expression to display line(s) with trailing blank(s): grep '\ ' test01

Command to display lines with trailing blank(s): grep -n '\ ' test01

Line number for the lines listed: 1, 2, 3, 5, 8, 10, 11, 12, 14, 17, 18, 20

Command to count lines with trailing blank(s): grep -c '\ ' test01

Count obtained: 12

(Check line numbers and count against your visual count)

10. Give one command with pipe to display, with line number, only lines in the file which contain both leading and trailing blank(s). Write a command with pipe to count the number of such lines which contain both leading and trailing blank(s).

Command to display lines with both leading and trailing blank(s):

grep -n '\ ' test01 | egrep -n '^ ' test01

Line number for the lines listed: : 2, 3, 5, 8, 10, 12, 17, 18, 20

Command to count lines with both leading and trailing blank(s):

grep -c '\ ' test01 | egrep -c '^ ' test01

Count obtained: 9

(Check line numbers and count against your visual count)

11. With one command, create five directories named **.dir01, .dir02, dir03, dir04** and .**dir05** under **asgn08**. (Note that three of them are hidden directories.)

Command to create the five directories: mkdir .dir01 .dir02 dir03 dir04 .dir05

Also create an empty file named **sedprog** at this stage.

Command to create **sedprog**: touch sedprog

12. The **ls** command when used with option -**F** will flag each executable file with an **\*** at the end of the name. Without changing directory, and using a relative path, execute the **ls –aF** for the directory **asgn01**.

Command **ls –aF** on **asgn01**: ls -aF ../asgn01

Output:

./ ../ hello.bash\*

Now pipe the output of the previous **ls** command to a **grep** command to select only executables. Remember that executables are flagged with an **\*** after the name.

Command with pipe to select executables: ls -aF ../asgn01 | egrep '\\*'

Output of the previous command: hello.bash\*

Now construct one command to list the names of all executable files in all eight folders **asgn01**-**asgn08**. Use shell wildcard for the directory names.

Command to locate all executables under **asgn01**-**asgn08**:

ls -aF ../asgn\* | egrep '\\*'

Output of the previous command:

hello.bash\*

hello\*

hello.py\*

a.out\*

GCD\*

a.out\*

helloWithCount\*

helloWithCount.c\*

**Note the use of \* (rather than using ?) for two different purposes in the previous command (one as a shell wildcard symbol and the other as a character in the regular expression.)**

13. The **ls** command when used with option -**F** flags each directory file with a **/** at the end of the name. Write a command with two pipes that counts the number of non-hidden directories under the current directory **asgn08**.

Command to display directory names with **/** at the end: ls -aF | egrep '\/'

Command with one pipe to display only non-hidden directories: ls -F | egrep '\/'

Command with two pipes to count the number of non-hidden directories:

ls -F | egrep ‘\/’ | egrep -c .

Count obtained: 2

14. Write alternative command with two pipes that counts the number of non-hidden directories under the current working directory **asgn08**. Proceed as follows:

* Perform an **ls-l** command and see the output.
* Consider sending this output through a pipe to a **grep** command. Construct the **grep** command that selects only directory entries (with **d** in the beginning of the line) in the input it receives. Construct the complete command with the pipe and observe the output.
* We need to count the number of lines. Use a pipe again and the **wc –l** command

Command to count the number of non-hidden directories:

ls -l | egrep -n '^d' | wc -l

Output (number of non-hidden directories): 2

15. Give one **sed** command to display the file **test01** with all occurrences of the pattern **line** substituted by **sentence** in lines 1-6 alone.

**sed** command: sed '1,6 s/line/sentence/g' test01

Which lines were modified? 2, 5

Display the contents of **test01** and check if it has changed in any way.

Command to display **test01**: cat test01

Has **test01** changed in any way? Lines returned to normal

16. Give one **sed** command to display the file **test01** on screen with **>** (greater than) character added at the end of each line.

**sed** to add **>** at the end of each line: sed 's/$/>/' test01

What did you observe? Adds > to the end of each line

17. Write one **sed** command to display the file **test01** with double quotes character at the beginning and the end of each line. (You may use a pipe.)

**sed** double quotes around lines: sed 's/^/"/' test01 | sed 's/$/"/'

What did you observe? Puts “ at the beginning and end of every line

18. Write a command with three pipes that counts the number of hidden directories as well as non-hidden directories under the current working directory. Follow the same logic used in Step 14. Obviously you must start with **ls –al**. Assume that no directory name ends with a period. Use a suitable **sed** command, in between **grep** and **wc**, to filter out the two directory entries **.** and **..**

Command to show hidden directories as well as non-hidden directories with the entries **.** and **. .** :

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Command to show hidden directories as well as non-hidden directories by filtering out entries **.** and **. .** :

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Command to count the number of hidden directories as well as non-hidden directories without **.** and **. .** :

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Output (number of hidden and non-hidden directories without **.** and **. .**):

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

19. Write a small **sed** program to remove all unnecessary blanks between words in a line (only one blank between words), and all leading and trailing blanks in a line. The **sed** program should also delete all lines with no characters in it, i.e., empty lines. First construct the **sed** editing instructions needed.

Contents of **sedprog** file (When **sed** editing instructions are placed in a file do not quote them).

Recognize that you are only writing the editing instruction here (what you normally put within quotes in a **sed** command)

**sed** editing instruction to remove leading blanks: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**sed** editing instruction to remove trailing blanks: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**sed** editing instruction to squeeze out unnecessary blanks: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**sed** editing instruction to delete empty lines: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Edit the **sed** program file **sedprog** and enter all the editing instructions above. Again, when **sed** editing instructions are placed in a file do not quote them.

Command to edit the file **sedprog**: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Execute the **sed** command with this program file on **test01** and create **test02** by output redirection as follows:

**sed –f sedprog test01 > test02**

Command to use the **sedprog** file on **test01** and create **test02**:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Display the file **test02** with line numbers and with a **$** sign marking the end of the line.

Command to display: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Visually inspect to see that no line has leading or trailing blanks, there is one and only one blank between all words and that all empty lines have been removed. If that is not the case, go back and fix your problems.

**See the #20, which is essential to show your work in #19 above.**

20. The UNIX command **script** is used to record a terminal session in a file and print the file later.

Assignment 2 (on Blackboard, but not used this semester) tells how this is done in #9, pages 6 and 7. It is convenient to name the file with your last name so it can be easily identified. Then type a series of commands for which the output must be captured.

It is very important not to use any control character as every character entered is captured in the file as is, and control characters mess up the display. **So, DO NOT backspace or use arrow keys or open an editor in the script**. Type your commands very carefully. **If there are mistakes, hit return and retype.** Finally, to close the script process, use the **exit** command.

For example, for a user with the last name **smith**, the session appears as follows:

**script smith # start recording**

**. . . series of commands**

**exit # stop recording**

As proof of the correct working of your previous step, create a **script** command output file by carrying out the following steps:

* Display your name by executing an **echo** command, such as echo YourName.
* Display your **sed** program file **sedprog** with **cat** command
* Display the original data file **test01** with **cat -nE** command
* Display the new file **test02** with **cat -nE** command
* Display the original data file **test01** with **od –c** command
* Display the new file **test02** with **od –c** command
* Print line, word and byte count of **test01** with **wc** command
* Print line, word, and byte count of **test02** with **wc** command

Print the **script** command output file and submit. Be sure your name appears on the output.

**Attach the printed script file to the back of your Assignment 8 (copy & paste).**

The easiest ways to print the script file are probably the following:

1. From the SSH client provided in Drake Library, don’t even bother to highlight the script file; just open a notepad file and enter cntrl-V to copy the text into notepad, making sure that just the script file lines are saved in notepad. Print out the script file using the printer in Drake.
2. From the SSH client provided in Drake Library, display the script file just created, open a notepad file and enter cntrl-V to copy the text into notepad, etc.
3. In a PuTTY client, hold down your mouse button and scan over the text to highlight it, etc.
4. Other apps, such as WinSCP, are available that will help you copy the text for printing.