Linux

Regex & Text Manipulation

$pwd → Present working directory.

eg: /home/ubuntu

$ echo "Hello World" ⇒ Displays "Hello World" as output like print statement.

→ $ echo "Hello World" > peanut.txt.

↓ Redirect/output ↳ Filename

This command will redirect the output to the given file. (If the file doesnot exist, it will create file with the given name)

→ $ cat peanut.txt (↗ filename.) ⇒ Displays the contents of file

→ $ echo "Hello Earth" > peanut.txt ⇒ This command will overwrite the contents of file.

Instead of overwriting, if u want to add the line to existing content:

→ $ echo "Hello Jupiter" >> peanut.txt

↓ Append command.

Output:

Hello Earth

Hello Jupiter

Taking Inputs:

$ cat < peanut.txt > sample.txt

↓ std in ↓ stdout

This means take the contents of peanut.txt file as input and redirect the content to sample.txt.

stdin - 0 ; stdout - 1 ; stdError - 2

(<) (>) (or) (1>) (2>)

StdError : Redirects error message to a file.

eg: $ ls /fake/dir > file1.txt

(Here no such directory is present)

output: ls cannot access '/fake/dir': No such file or directory

↓

To redirect this error message to file as content stdError (2>) is used

eg: $ ls /fake/dir 2> file1.txt ⇒ The error message for this command will be moved to file1.txt

$ cat file1.txt

output: ls cannot access '/fake/dir': No such file or directory.

/dev/null ⇒ This is like a nothing file. Nothing can be stored.

If user decides not to view the output and also not to store the output, /dev/null can be used.

eg: $ ls /fake/dir 2> /dev/null

$ cat peanut.txt 1> /dev/null

It is like redirecting to nothingness.

Pipe command (|):

Taking output of one command and passing it as input to another command.

eg: ls -la /home | less

↳ opens the content in separate editor window

↓

List out all files/folders from /home directory

↳ Output from this command will be opened in separate window.

tee command:

This command will display the output and also saves the output in a file

→ $ ls 1>% out.txt → This command will only save the output in a file.

→ $ ls | tee newfile.txt → Redirects the output to newfile.txt and also displays the output (like a cat command)

ls lists out the file/folder

| passes the output of ls command as input to next command.

Environment variables

→ $ printenv

Prints all the environment variables (like inbuilt variables)

Now to add ur own variable with value

→ $ export name="Varshini"

→ $ printenv ⇒ You should see name="Varshini"

Now if you just use ur variable in command:

→ $ name

output: Varshini

→ $ echo "Hello $name"

output: Hello Varshini

To delete the variable:

→ $ unset name

→ $ printenv ⇒ You should not see name variable now.

To between User roles:

→ $ sudo su - root

su → switch user

root → username

- Root is superior user

Regular Expression (Regex)

This is mainly used when u r trying to search for particular pattern of characters from system generated logs like ip address, url etc...

Also on websites, you can see email related fields where it might ask u to enter valid email ID. If u give invalid ID, the system recognizes and throws u error. Here system is performing a pattern checking

Ref: regex101.com

[ ] → set of character eg: [a, b, c], [0-9]

\A → Checks whether the doc starts with specific word beginning of doc.. /Z → End of doc

\A →

\B → Between words, eg: sand /B and /B ⇒ Between eg: sand, hand

\b → Whole words eg: /b and /b ⇒ and not sand, hand

\d → Digits ⇒ [0-9]

\D → Any character other than digits [0-9]

\s → Space character (

\S → other than whitespace

\w - [a-z] [A-Z] [0-9], _ (underscore)

\W - Not [a-z] [A-Z] [0-9], _

. → Any character except newline (Enter)

^ → Start of every line eg: ^The → Identifies the line starts with 'The'

$ → End of every line eg: apple$ → Identifies the line ends with apple

* - Zero or more occurances

eg: abc*

↳ ab, abc, abcc, abccc..... abcabccc

ab*c

↳ ac, abbc, abc, abbbc, abbb....c

+ - one or more occurances

eg: abc+

abc, abcc, abccc..., abcabbabc

? - zero or one occurances

eg: abc?

ab, abc, abc ab abb

{} - specifying no. of occurances

eg: abc{2}

ab, abc, abcc, abcc

Write a regex pattern to find valid mail ID

eg: varsh123v@yahoo.com

~~[a-z~~A-Z0-9]+[@][a-z~~A-Z~~]+[.]~~[a-z A-Z]~~+

^[a-zA-Z0-9]+[@][a-zA-Z]+[.][a-zA-Z]+$

^ start of every line

[@] one or more occurances

[.] one or more occurances

$ at end

varsh123v @ yahoo . com

[a-zA-Z0-9]+ [@] [a-zA-Z] [.] [a-zA-Z]+$

grep command:

→ $ grep "pattern" filename

eg: $ grep "apple" abc.txt.

→ $ grep -i "apple" abc.txt
↳ case insensitivity

$ grep -l "apple" /home/ubuntu ?

→ $ grep -l "apple" sample.txt file2.txt file3.txt
↳ searches for the word in all the give file and displays the filename if the word is present.

→ $ grep -w "apple" abc.txt
↳ Searches whole word

→ $ grep -iw "apple" abc.txt.

→ $ grep -n "apple" abc.txt.
↳ searches the line, displays the line with the given word.

Regex in grep:

→ $ grep "[A-Z]*" abc.txt
↳ Displays all the word that has upper case
eg: CSE, Apple, He, Eva, BTech ....

sed command ⇒ Find & Replace

→ $ sed 's/is/IS/g' abc.txt

- s → substitute
- is → search word
- IS → Replace word
- g → global occurances (All places)
- abc.txt → filename.

awk command :

eg: employee.txt

ajay Manager 45000
varun accountant 35000

If we want retrieve particular data, we can use awk

→ $ awk '[*print]' employee.txt => Prints the data.

→ $ awk '[print $1]' employee.txt => Prints 1st column.

→ $ awk '[print $1 $3]' employee.txt => Prints 1st & 3rd col.

→ $awk '/accountant [print $1 $3]' employee.txt

↳ Prints 1st & 3rd column that has only accountant
like filtering out accountants.

head & tail commands :

→ $tail filename (default)
↳ Displays last 10 line from the file.

→ $ tail -n 3 file.txt
↳ Displays last 3 line.

→ $ head filename (Default.)
↳ Displays first 10 lines from file

→ $ head -n 5 file.txt
↳ Displays first 5 lines from file.

tail -f => give live updates of line.

→ If u want to view 30 - 45 line

→ $ head -n 45 file.txt | tail -n 15 file.txt

↳ Displays 30-45th line

head 45
tail 15

vim - text editor for linux

$ vi file.txt

↳ Opens the file in vim text editor.

press 'i' to edit data

press 'Esc' to come out of edit mode & goes to command mode.

:wq ⇒ save & quit editor mode.
(w → save, q → quit)

Go to a particular line, press D twice, the line will be deleted

:%d ⇒ deletes all line.

:q! ⇒ Force quit goes to linux terminal.

sort command :

→ $ sort filename

sorts the content in ascending order

→ $ sort -r filename

sorts the content in descending order

→ $ ls -l | sort

↳ Sorts the list of file in ascending order (by size - default)

unique command: Always compare adjacent lines.

→ $ uniq file.txt.

↓

Displays unique words from the content.

→ $ uniq -c file.txt

↓

Displays unique words & no. of their occurances

→ $ uniq -d file.txt

↳ Displays duplicate values

Word count (wc) command

$ wc file.txt → No. of lines, No. of words, bytes

14 14 71

$ wc -l new.txt → lines only

$ wc -c " → word count only