

Accessing file contents

- **cat** file1
- **cat -n** file1 //gives the numbering

head and tail

- **head** file1
- **head -n** file1 //gives first n lines
- **tail** file1
- **tail -n** file1 //gives last n lines E.g **head -n20** file1

more and less

- **more** file1 //allows one page at a time
- **ls -al |more** //allows listing of directories one page at a time.
- **less** file1 //allows page Up and page Down

Press Enter key to scroll down line by line(or)

Use d to go to next page

Use b to go to previous page

Use / to search for a word in the file

Use v to go vi mode where you can edit the file and once you save it you will get back to less/more command

Find

Options

- name : For searching a file with its name
- inum : For searching a file with particular inode number
- type : For searching a particular type of file
- user : For files whose owner is a particular user
- group : For files belonging to particular group

find / -name test1.txt → find file named test1.txt in root dir and subdirectories
find . -name test1.txt → find all files whose name is test1.txt in current directory

E.g

```
find / -name dpkg.log
```

```
cd /etc
```

```
find . -name 50-cloud-init.yaml
```

```
find /etc/ -group shadow -ls
```

```
find / -type l -ls → find files with symbolic link
```

```
find / -type d -name Pictures
```

locate pattern

The locate command works reading contents from the database.

```
apt-get install mlocate
```

The db must be updated

```
sudo updatedb
```

E.g

```
#locate dpkg.log
```

```
#locate -i readme.md
```

Grep and Regular expression

Tool to search string in a file. Basically are a character or group of characters that represents certain things in a string.

```
$ cat word.txt
```

```
Surya is Studentx1000
```

```
surya is studentx1000
```

```
studentx1000 is surya
```

studentx1000 is Surya

surya

surya is surya

Timmy

Tommy

TImmy

TOmmy

Linux is a great OS

I like Linux M 10

This is a random Number 9018234103874

Sometimes I mess up case like this sUrYa

\$ grep surya word.txt --> to find every line that has word surya but it does not give lines with different cases.

\$ grep -n surya word.txt --> show line number

\$ grep -i surya word.txt --> ignores the case and displays every line that has word surya.

\$ grep [sS]urya word.txt --> displays the only lines that start with a capital or lower case but not random capital letters.

Ths characters placed inside the square bracket are the replacement values.

\$ grep -i tommy word.txt

\$ grep -i timmy word.txt

Using regular expressions.

\$ grep T[oO]mmy word.txt

\$ grep T[iI]mmy word.txt

\$ grep T[iOIO]mmy word.txt

\$ grep ^Surya word.txt --> prints lines that begin with capital Surya

\$ grep ^Surya word.txt --> caret ^ symbol represents lines that begin with.

\$ grep ^[Ss]urya word.txt --> gives all lines that have capital and lower-case s.

\$ grep [Ss]urya\$ word.txt --> gives all the lines that end with capital or lowercase s.

\$ grep [s]urya\$ word.txt --> gives lines ending with lower surya.

\$ grep [S]urya\$ word.txt --> gives a line ending with capital Surya.

\$ grep ^surya\$ word.txt --> gives a line that contains surya alone.

\$ grep [0-9] word.txt --> searching numeric value. gives all the lines that contain numeric values.

\$ grep [0-9]\$ word.txt --> gives lines that end with numeric value.

\$ grep [0-9][0-9][0-9]\$ word.txt --> gives lines that ends with the last three characters numeric.

\$ grep ^\$ word.txt --> to print blank lines.

\$ grep -v ^\$ word.txt --> inverts the search and print line that have contents only, not blank lines.

sed

sed stands for stream editor, which is used to search a word in the file and replace it with the word required to be in the output

```
$sed -e 's/find this/replace with this/g' /etc/passwd
```

- Here -e indicates giving some expression.
- s indicates find.
- Find these contents to be replaced.
- Replace with this, contents that replace the existing one.
- g indicates global search and replace.
 1. sed -e 's/ansible/devops/g' /etc/passwd
 2. sed -n '2p' /etc/passwd
 - Here -n flag (silent/negation) ignores all the non important lines. p is for printing and 2 is for the 2nd line.
 3. sed -n '\$p' /etc/passwd
 - Print last line
 4. sed -n '3,6!p' /etc/passwd
 - Print all lines except 3rd to 6th line
 5. sed -i 's/Suryaraj/raj/g' abc.txt
 - -i option allows to write on the same file

Cut

Extract texts

```
cut -c1 /etc/passwd
```

→ To print the 1st character of every row of the 1st column.

```
cut -c1-5 /etc/passwd
```

→ To print the 1st to 5th character.

```
cut -d ':' -f1 /etc/passwd
```

Here -d is the delimiter(breaks) and f1 represents field1 column 1.

```
cut -d ':' -f2-3 /etc/passwd
```

Here f2-3 represents the fields 2 to field 3.

To delimit spaces and print the field

```
cut -d " " -f1 filename
```

cut -d -f filename (where d stands for delimiter eg. :, " " etc and f stands for field)

AWK

AWK is a family of tools that are primarily used for processing text files. The most basic use of AWK is parsing the files and generating reports.

Syntax: `awk '/search pattern/ {Actions}' file`

- Search pattern in a regular expression.
- Actions -> Statements to be performed.
- Several patterns & actions are possible in AWK.
- File -> Input file
- Single quotes around the program is to avoid shell not to interpret any of its special characters.

‘Print only specific field’

AWK has a number of built-in variables.

For each record i.e line, it splits the record delimited by whitespace character by default & it stores in the \$n variables.

E.g If the line has 4 words, it will be stored in \$1, \$2, \$3, \$4.

\$0 represents the whole line.

```
awk -F ':' '{print $1}' /etc/passwd
```