1) **Cal**:- Displays a calendar
Syntax:- cal [[month] year]

Description :-

- Cal displays a simple calendar. If arguments are not specified, the current month is displayed. The options are as follows:

-1	Display single month output. (This is the default.)
-3	Display prev/current/next month output
-s	Display Sunday as the first day of the week (This is the default.)
-m	Display Monday as the first day of the week
-j	Display Julian dates (days one-based, numbered from January 1)
-y	Display a calendar for the current year

Example:-
$cal
Or
$cal 01 2014

Jan 2014

Sun	Mon	Tue	Wen	Thur	Fri	Sat
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	30	

2) **Date** : - Print or set the system date and time
Syntax :- date
Description :-

- Display the current time in the given FORMAT.
- The command can also be used with suitable format specifies as arguments. Each format is preceded by a + symbol, followed by the % operator, and a single character describing the format.

Example:-
$date

```
Mon Jan 13 02:55:15 IST 2014
$ date +%m
01
$ date +%h
Jan
```

3) **Clear** :- Clear command clears the screen

Syntax :- clear

Description :-

- Clear clears your screen if this is possible. It looks in the environment for the terminal type and then in the terminfo database to figure out how to clear the screen.

4) **Cd** :- Changes the directory.
Syntax :- cd [directory]

Description:-

- Used to go back one directory on the majority of all UNIX shells. It is important that the space be between the cd and the..

Example:-

```
$ pwd
  /home/kumar
$ cd progs
$ pwd
 /home/kumar/progs
$ cd ..
  /home/kumar
```

5) **Ls** :- Lists the contents of a directory
Syntax :- ls [-a] [-A] [-d] [-l] [-p] [-r] [-R][pathnames]

Description :-

-a	Shows you all files, even files that are hidden (these files begin with a dot.)
-A	List all files including the hidden files. However, does not display the working directory (.) or the parent directory (..).
-d	If an argument is a directory it only lists its name not its contents
-l	Shows you huge amounts of information (permissions, owners, size, and when last modified.)
-p	Displays a slash (/) in front of all directories
-r	Reverses the order of how the files are displayed
-R	Includes the contents of subdirectories

Example:-
$ ls – l

```
-rw-r----- 1 ramesh team-dev 9275204 Jun 13 15:27 mthesaur.txt.g
```

Field Explanation

- - normal file
- d directory

- **Field 1 – File Permissions:** Next 9 character specifies the files permission. Each 3 characters refers to the read, write, execute permissions for user, group and world In this example, -rw-r—– indicates read-write permission for user, read permission for group, and no permission for others.
- **Field 2 – Number of links:** Second field specifies the number of links for that file. In this example, 1 indicates only one link to this file.
- **Field 3 – Owner:** Third field specifies owner of the file. In this example, this file is owned by username 'ramesh'.
- **Field 4 – Group:** Fourth field specifies the group of the file. In this example, this file belongs to "team-dev' group.
- **Field 5 – Size:** Fifth field specifies the size of file. In this example, '9275204′ indicates the file size.
- **Field 6 – Last modified date & time:** Sixth field specifies the date and time of the last modification of the file. In this example, 'Jun 13 15:27′ specifies the last modification time of the file.
- **Field 7 – File name:** The last field is the name of the file. In this example, the file name is mthesaur.txt.gz.

6) **Exit**:- Allows you to exit from a program, shell or log you out of a Unix network.
Syntax :- exit

7) **Echo** :- display a line of text
Syntax :- echo
Example:-

```
$ echo "hi"
hi
```

8) **Who** :- show who is logged on
Syntax :- who [OPTION]

Description :-

-b	time of last system boot
-d	print dead processes
-m	only hostname and user associated with stdin
-q	all login names and number of users logged on
-t	print last system clock change
-u	list users logged in

Example :-
$who

```
raj pts/1 2014-01-17 22:42 (:0.0)
swatipts/2 2014-01-18 09:30 (:0.0)
jay pts/3 2013-12-25 08:52 (:0.0)
```

Here first column shows user name, second shows name of the terminal the user is working on. Third& fourth column shows date and time of logging, last column shows machine name.

9) **Who am i**:- Print effective userid
Syntax :- who am i
Description: - Print the user name associated with the current effective user id.
Example :-
$ who am i

```
jay pts/3 2013-12-25 08:52 (:0.0)
```

Here first column shows user name, second shows name of the terminal the user is working on. Third & fourth column shows date and time of logging, last column shows machine name.

10) **Mkdir**:- This command is used to create a new directory
Syntax :- mkdir [option] directory

Description :-

-m mode	Set permission mode (as in chmod)
-p	No error if existing, make parent directories as needed
-v	Print a message for each created directory
directory	The name of the directory that you wish to create

Example:- $mkdiraaa

- The above command will create directory named aaa under the current directory. We can also create number of subdirectories with one mkdir command.

11) **Rmdir**:- Deletes a directory.
Syntax :-rmdir [OPTION]... DIRECTORY

Description :-

-p, --parents	Remove DIRECTORY and its ancestors. E.g., `rmdir -p a/b/c' is similar to `rmdir a/b/c a/b a'
-v, --verbose	output a diagnostic for every directory processed

12) **Bc**:- Calculator
Syntax :- bc

Description :-

- Bc is a language that supports arbitrary precision numbers with interactive execution of statements.
- Bc starts by processing code from all the files listed on the command line in the order listed. After all files have been processed, bc reads from the standard input. All code is executed as it is read.

Example:-

```
$ bc

bc 1.04
Copyright (C) 1991, 1992, 1993, 1994, 1997 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
12+10
22
```

Use [ctrl-d] to quit bc.

- Bc can also be used in converting numbers from one base to another. For example , setting ibase=2 , we can give the input in binary form and we will get output in decimal. And reverse is possible by setting obase=2.

$ bc

```
bc 1.04 Copyright (C) 1991, 1992, 1993, 1994, 1997 Free Software Foundation, Inc. This is free
software with ABSOLUTELY NO WARRANTY. For details type `warranty'. ibase=2 11001010 202
```

13) **Uname**:- print system information.
Syntax :-uname [OPTION]...

Description :-

- Print certain system information. With no OPTION, same as -s.

-s	print the kernel name
-n	print the network node hostname
-r	print the kernel release
-v	print the kernel version
-m	print the machine hardware name
-o	print the operating system

Example:-

```
$uname
Linux
```

14) **Passwd**:- Allows you to change your password.

Syntax:- passwd

Description :-

- Passwd changes the password or shell associated with the user given by name or the current user if name is omitted. By default, the current password is prompted for, and a new one must be entered and confirmed.passwd command can also be used to change the home directory where the path stands for the home directory.

15) **Tty**:- Print the file name of the terminal connected to standard input.
Syntax :- tty

Description :-

- Tty writes the name of the terminal that is connected to standard input onto standard output. Command is very simple and needs no arguments.

Example :-

```
  $ tty
/dev/tty10
```

16) **Stty**:- Sets options for your terminal. OR change and print terminal line settings.
Syntax :- sty

Description :-

- Stty sets certain terminal I/O modes for the device that is the current standard input; without arguments, it writes the settings of certain modes to standard output.

Example :-

```
 $ stty
speed 19200 baud, 25 rows, 79 columns
kill = ^U
```

17) **Cat** :- Creating and displaying files.
Syntax : - cat > Filename
Description :-

- If we want to create a new file. Just type the command.
  $cat > test
- Now press the enter key and you would find the cursor positioned in the next line you want to store in the file 'test' Type in two lines of text.
- After press the keys Ctrl d
- The Keys ctrl d indicate the EOF or end of file character.
- You would once again get back the $ prompt on the screen.
- Command prompt displays the contents of file recipe on the screen.
- To see the contents of the file test that we created above we should say

$cat test

- Two uses of the cat command.
- One is to create new files and another to display the contents of an existing file.
- It can concatenate the contents of two files and store then in the third file.
- The contents of sample1 and sample2 should get appended to it then you should use the append output redirection operator >> as shown below

$cat sample1 sample2 >>newsample

Examples :- $ cat > file1
process management
memory management
file mgmt

[ctrl-d]

- The above command creates file named file1. You can write down information in the file and when you have done you need to use ctrl+d to have prompt back on your screen. Cat can also be used to read the file contents.

$ cat file1

memory management
process management
file mgmt

18) **Cp** :- Copies files from one location to another.
Syntax :- cp [OPTION]... SOURCE DEST cp [OPTION]... SOURCE... DIRECTORY

Examlple :- $ cp file1 file2

Description:-

- Here after cp command contents of both file1 and file2 files are the same. If the destination file doesn't exists , it will be created. If it exists then it will be overwritten without any warning. If there is only one file to be copied then destination can be the ordinary file or the directory file.

19) **Rm**:- Remove files.
Syntax :-rm [OPTION]... FILE...

Description :-

- Files can be deleted with rm. It can delete more than one file with a single invocation. For deleting a single file we have to use rm command with filename to be deleted.

filename	The name of the file you want to remove directory

Examlple :- $ rmos

- Deleted file can't be recovered. rm can't delete the directories. If we want to remove all the files from the particular directory we can use the * symbol.
- Following command deletes all the files from the sub directory.

$ rm *

20) **Mv**:- Renames a file or moves it from one directory to another directory.
Syntax :- mv oldnamenewname.

Description :-

- mv has two functions: it renames a file and it moves a group of files to a different directory. Mv doesn't create a copy of the file , it merely renames it. No additional space is consumed on disk during renaming. For example if we rename a file os to os1 and then if we try to read file os we will get error message as it is renamed to os1 there is no existence of file named os.

```
$ cat file1
 Memory
 Process
Files
$ mv file1 file2
$ cateos -ksh: cate: not found [No such file or directory]
$ file1 file2
Memory
 process
files
```

- If the destination file doesn't exist it will be created. mv can also be used to rename a directory. A group of files can also be moved to a directory. mv doesn't prompt for overwriting destination file if it exists.

-f	mv will move the file(s) without prompting even if it is writing over an existing target. Note that this is the default if the standard input is not a terminal
-i	Prompts before overwriting another file
oldname	The oldname of the file renaming
newname	The newname of the file renaming
filename	The name of the file you want to move directory - The directory of were you want the file to go

21) **Cut** :- Cut out selected fields of each line of a file.
Description :-

- We can extract both columns and fields from the file with the cut command. Columns are specified with the –c option and fields with –f.

```
$ cat > shortlist

2233|a.k.shukla|g.m.|sales|12/12/52|6000 9876|jai
```

```
sharma|director|production|12/03/50|7000 5678|sumit
chakrobarty|d.g.m|marketing|19/04/43|6000 2365|barun
sengupta|director|personnel|11/05/47|7800
```

i) Cutting columns: to extract specific columns , we need to follow the –c option with a list of column numbers, delimited by a comma. Ranges can also be used with hyphen. For example following command extracts the name from the shortlist file.

```
$ cut -c 6-15 shortlist
a.k.shukla
 jai Sharma
sumitchak
barunseng
n.k.gupta
```

- Cut can also be used with multiple ranges as follows which gives columns from 6 to 15 and 22 to 26. And here 30- shows column number 30 to the end of the line.

```
$ cut -c 6-15,22-26,30- shortlist

a.k.shuklasales/12/52|6000
 jai sharmator|puction|12/03/50|7000
sumit chaky|d.gmarketing|19/04/43|6000
barun sengirectpersonnel|11/05/47|7800
n.k.gupta|an|ad|30/08/56|5400
```

ii) Cutting fields: -c option is useful for fixed length lines. But suppose now we want to cut 2 and 3 fields from our file then we can use-f option. For example following command cuts 2 and 3 fields from our file. We have to use –d option to specify the delimiter. In our file fields delimiter is | symbol that we have specified in double quotes. And we have to use –f option to specify field list.

```
$ cut -d "|" -f 2,3 shortlist
a.k.shukla|g.m.
 jai sharma|director
sumitchakrobarty|d.g.
m barunsengupta|director
n.k.gupta|chairman
```

file	A path name of an input file. If no file operands are specified, or if a file operand is -, the standard input will be used
-c	The list following -c specifies character positions
-d	The character following -d is the field delimiter

22) **Paste** :- Merge corresponding or subsequent lines of files.
Syntax :- paste [file]
Description :-

- Paste prints lines consisting of sequentially corresponding lines of each specified file. In the output the original lines are separated by TABs. The output line is terminated with a newline.

-d	Specify of a list of delimiters.
-s	Paste one file at a time instead of in parallel.
File	A path name of an input file. If - is specified for one or more of the file s, the standard input will be used.

Example:-

```
$ cat > a
Hiii
$ cat > b
Hello
$ paste a b
Hiii Hello
```

23) **More** :- Displays text one screen at a time.
Syntax :- more [+ linenumber] [file ...]

Description :-

- More command displays its output a page at a time. For example we are having a big file with thousands of records and we want to read that file then we should use more command. Consider a file named employees then we will use following command to read its contents a page at a time.

$ more employees

+line number	Start up line number
filename	The name of the file

24) **Cmp**:- Compares two files and tells you what line numbers are different.
Syntax : - cmpfirstfilesecondfile

Description :-

- Let's create a file named os2. And use cmp command to compare os and os1files.

Example:-

```
$ cat file1
memory
process
files
$ cat > file2
memory
process
```

```
files mgmt
$ cmp file1 file2
File1 file2 differ: char 21, line 3
```

- The two files are compared byte by byte and the location of the first mismatch is echoed to the screen. Cmp doesn't bother about possible subsequent mismatches .

Firstfile	First file that you wish to compare
secondfile	Second file that you wish to compare to

25) **Comm**:- Select or reject lines common to two files.
Syntax:- comm [-1] [-2] [-3] file1 file2

Description :-

- Requires two sorted files and lists differing enteries in different columns. produces three text columns as output:

1 Lines only in file1.
2 Lines only in file2.
3 Lines in both files.

- For example let's create two files named file1 and file2 with following data.

$ cat > file1
```
c.k.shukla
chanchalsanghvi
s.n.dasgupta
sumit [1] + Stopped
```

$ cat > file2
```
anilaggarwal
barunsengupta
c.k.shukla
lalit
```

- Now let's use comm. Command with these two files.

```
$ comm file1 file2
anilaggarwal
barunsengupta
c.k.shukla
chanchalsanghvi
lalit
s.n.dasgupta
sumit
```

- In the above output we can see that first column contains two lines unique to the first file and second column contains three lines unique to the second file and the third column contains two lines common to both the files. Comm. Can produce the single column output using 3 options -1,-2 or -3. To drop a particular column , simply use its column number as an prefix.

-1	Suppress the output column of lines unique to file1
-2	Suppress the output column of lines unique to file2
-3	Suppress the output column of lines duplicated in file1 and file2
file1	Name of the first file to compare
file2	Name of the second file to compare

26) **Diff** :- Displays two files and prints the lines that are different.
Syntax :- diff [-b] [-i] [-w] [fileonefiletwo] [directoryonedirectorytwo].

Description :-

- Diff is the third command that can be used to display file differences. Unlike its fellow members ,cmp and comm. , it also tells us which lins in one file have to be changed to make the two files identical. Let's use the above created same files with diff commands.

```
diff file1 file2
0a1,2
>anilaggarwal
>barunsengupta
2c4
>chanchalsanghvi
>lalit
4d5
>sumit
```

- Diff uses some special symbols and instructions to indicate changes that are required to make two files identical. Each instruction uses an address combined with an action that is applied to the first file. For example in above output 0a1,2 means appending 2 lines after line 0 , which becomes lines 1 and 2 in the second file. 2c4 changes/replaces line 2 with line 4 in the second file. And 4d5 means delete line 4.

27) **Chmod**:- Changes the permission of a file.
Syntax:- chmod [*options*] mode file ...

Description :-

- Chmod command is used to set the permissions of one or more files for all three categories of users (user,group and others). It can be run only by the user and super user. Command can be used in two ways. Let's first take a look at the abbreviations used by chmod command.

Category		Operation		Permission	
u	User	+	Assigns permission	r	Read permission
g	Group	-	Removes permission	w	Write permission
o	Others	=	Assigns absolute	x	Execute permission

- **Relative permissions** : when changing permissions in a relative manner , chmod only changes the permissions specified in the command line and leaves the other permissions unchanged. In this mode it uses the following syntax:

Example : let's first check the permission of file shortlist. It shows read/write permission. Let's assign execute permission to this file for owner.

```
$ ls –l
 -rw-r--r—1 Hiral None 241 Dec 21 04:02
 $ chmodu+x shortlist
 $ ls -l shortlist
-rwxr--r-- 1 Hiral None 241 Dec 21 04:02
```

- Here chmod uses expression u+x means – u shows user category that is user, + shows we need to assign the permission and x shows execute permission. Sameway we can assign other permissions.

- **Absolute permissions:** sometimes you don't need to know what a file's current permissions are , but want to set all nine permission bits explicitly. The expression used by chmod here is a string of three octal numbers. Each type of permission is assigned a number as shown:

  Read permission-4
  Write permission-2
  Execute permission -1

- For each category we add numbers that represent the assigned permission. For instance , 6 represents read and write permission , and 7 represents all permissions.
- **For Example :**chmod 666 shortlist , shows read and wrie (4 +2) permissions for all three types of users.

28) **Chown**:- Command for system V that changes the owner of a file.
Syntax :- chown [-R] newowner filenames

Description :-

-R	Change the permission on files that are in the subdirectories of the directory that you are currently in
Newowner	The alias/username of the new owner of the file
Filenames	The file that you are changing the rights to

Example :-
```
$ls -l demo.txt
-rw-r--r-- 1 root root 0 Aug 31 05:48 demo.txt
```

```
$ chown vivek demo.txt
$  ls -l demo.txt
-rw-r--r-- 1 vivek root 0 Aug 31 05:48 demo.txt
```

29) **Chgrp**:- For system V, changes the group that has access to a file or directory.
Syntax:- chgrp newgroup filenames [-f]

Description :-

newgroup	Specifies the new group that you wish to allow access to
Filenames	Specifies the files that the newgroup has access to
-f	Force. Do not report errors

Example :-

```
$ ls -l demo.txt
-rw-r--r-- 1 root root 0 Aug 31 05:48 demo.txt
 $ chgrpvivek demo.txt
-rw-r--r-- 1 root vivek 0 Aug 31 05:48 demo.txt
```

30) **File**:- Tells you if the object you are looking at is a file or if it is a directory.
Syntax:- file name
Description:-

- File command is used to determine the type of file , especially of an ordinary file. We can use it with one or more filenames as arguments. For example we can use file command to check the type of the os1 file that we have created.

  $ file os1
  ```
   os1: short english text
  ```
- File correctly identifies the basic file types (regular , directory and device). We can use * to signify all files.

$ file *

```
cn: short text
os1: short english text
sub: directory, searchable
```

- This command identifies the file type by examining the magic number that is embedded in the first few bytes of the file. Every file type has a unique magic number. File recognizes text files , and can distinguish between shell programs , C source and object code. It can also identify DOS executables, compressed files, PDF documents.

31) **Finger**:- Lists information about the user.
Syntax:- finger [-l] [-q] [-s] [username]
Description :-

-l	Force long output format
-s	Force short output format

Example :-

```
$ finger rahul
Login: rahul                    Name: (null)
Directory: /home/sathiya              Shell: /bin/bash
On since Mon Nov  1 18:45 (IST) on :0 (messages off)
On since Mon Nov  1 18:46 (IST) on pts/0 from :0.0
New mail received Fri May  7 10:33 2010 (IST)
Unread since Sat Jun  7 12:59 2008 (IST)
No Plan.
```

32) **Sleep**:- Waits a *x* amount of seconds.
Syntax :- sleep time

Description:-

Time	The amount of seconds to wait

33) **Kill**:- terminate a process.
Syntax:- Kill pid

Description :-

- The command kill sends the specified signal to the specified process or process group.
- If no signal is specified, the TERM signal is sent. The TERM signal will kill processes which do not catch this signal.
- For other processes, it may be necessary to use the KILL (9) signal, since this signal cannot be caught.

pid..	Specify the list of processes that kill should signal

34) **Users**: - print the user names of users currently logged in to the current host.
Syntax:- users

Description:-
- Output who is currently logged in.

35) **Ps**:- Reports the process status.

Syntax:- ps [-a] [-A] [-f] [-j] [-l]
Description :-

-a	List information about all processes most frequently requested: all those except process

	group leaders and processes not associated with a terminal
-A	List information for all processes. Identical to -e, below
-f	Generate a full listing
-j	Print session ID and process group ID
-l	Generate a long listing

Example :-

$ps

```
PID  TTY      TIME  CMD
291 console 0:00   bash
```

36) **Ln**:- Creates a link to a file.
Syntax:- ln [-s] existingfilenewname

Description:-

-s	Makes it so that it does not create a symbolic link
Existingfile	Specifies file(s) that you want to create a link to
Newname	The new name of the file

Example :-

$ ln index manual

This links index to the new name, manual/index.

37) **Head**:- Displays the first ten lines of a file, unless otherwise stated.
Syntax:- head -number filename

Description:-

- Head command displays the top of the file. When used without an option , it diplays the first ten lines of the file.

  $head emp.lst

  ```
  2233|a.k.shukla|g.m.|sales|12/12/52|6000 9876|jai
  sharma|director|production|12/03/50|7000 5678|sumit

  chakrobarty|d.g.m|marketing|19/04/43|6000 2365|barun
  sengupta|director|personnel|11/05/47|7800
  5423|n.k.gupta|chairman|admin|30/08/56|5400 1006|chanchal
  sanghvi|g.m.|accounts|05/06/63|6300 6213|karuna
  ganguly|director|sales|03/09/38|6700
  1265|s.n.dasgupta|manager|sales|12/09/63|5600 4290|jayant
  choudhry|executive|production|07/09/50|6000 2476|anil
  ```

```
aggarwal|manager|sales|01/05/59|5000
```

- We can use –n option to specify a line count and display say , first three lines of the file.

  $ head - 3 emp.lst

  ```
  2233|a.k.shukla|g.m.|sales|12/12/52|6000 9876|jai
  sharma|director|production|12/03/50|7000 5678|sumit
  chakrobarty|d.g.m|marketing|19/04/43|6000
  ```

-number	The number of the you want to display
Filename	The file that you want to display the x amount of lines

38) **Tail**:- Delivers the last part of the file.
Syntax :- tail filename

- Tail displays the end of the file. When used without option it displays the last ten lines from the file. The last three lines are display using following command.

  $ tail -3 emp.lst

  ```
  3564|sudhir agarwal|executive|personnel|06/07/47|7500
  2345|j.b.saxena|g.m.|marketing|12/03/45|8000
  0110|v.k.agarwl|g.m.|marketing|31/12/40|9000
  ```

- We can also address lines from the beginning of the file instead of the end. The +count option allows to do that , where count represents the line number from where the selection should begin. For example following command selects 11th line onwards from emp.lst file.

  $ tail +11 emp.lst

  ```
   6521|lalit chowdury|director|marketing|26/09/45|8200 3212|shyam
  saksena|d.g.m|accounts|12/12/55|6000 3564|sudhir
  agarwal|executive|personnel|06/07/47|7500
  2345|j.b.saxena|g.m.|marketing|12/03/45|8000

  0110|v.k.agarwl|g.m.|marketing|31/12/40|9000
  ```

-number	The number of the you want to display
Filename	The file that you want to display the x amount of lines o

39) **Sort**:- Sorts the lines in a text file.
Syntax:- sort [options]... [file]

Description:-

-b	Ignores spaces at beginning of the line
-c	Check whether input is sorted; do not sort
-r	Sorts in reverse order
-u	If line is duplicated only display once
Filename	The name of the file that needs to be sorted

Example:-

- The following example will create the a file first then it will sort the a file

```
cat > a      sort a
B             A
H             B
A             E
V             H
E          S
S          V
H
```

40) **Find**:- Finds one or more files assuming that you know their approximate path.
Syntax :- find path

Description :-

Path	A path name of a starting point in the directory hierarchy

41) **Unique**:- Report or filter out repeated lines in a file.
Syntax:- uniq [-c | -d | -u] input_file

Description :-

-c	Precede each output line with a count of the number of times the line occurred in the input
-d	Suppress the writing of lines that are not repeated in the input
-u	Suppress the writing of lines that are repeated in the input
input_file	A path name of the input file. If input_file is not specified, or if the input_file is -, the standard input will be used

Example:-
$ cat >abc
d.g.m
admin
accounts
sales
d.g.m

```
marketing
advisor
```

- Now we want to sort this file and want to remove the duplicate lines. For that we can use the following command which sorts the designation file and uniq removes the duplicate lines and puts the data in uoutput file.

```
$ sort abc | uniqdef
$ cat def
 Accounts
 Admin
 Advisor
d.g.m
marketing
 sales
```

42) **Tr**:- Translate characters.
Syntax:- tr [-c] [-d] [-s] [string1] [string2]

Description:-

- The tr filter manipulates individual characters in a line. It translates characters using one or more compact expressions. tr takes input only from standard input , it doesn't take filename as argument. By default tr translates each character in expression1 to its mapped counterpart in expression2. The first character in expression1 is replaced by the first character in second expression and so on. For example consider the following command where w e want to replace | with ~ and / with @ symbol in the shortlist file

Example:-

```
$ tr '|/' '~@' <shortlist
2233~a.k.shukla~g.m.~sales~12@12@52~6000
9876~jai sharma~director~production~12@03@50~7000
5678~sumit chakrobarty~d.g.m~marketing~19@04@43~6000
2365~barun sengupta~director~personnel~11@05@47~7800
5423~n.k.gupta~chairman~admin~30 @08@56~5400
```

- We can also specify the ranges in the expression. Consider the following command which takes first 3 lines of the shortlist file and changes the case from lower to upper.

```
$ head -n 3 shortlist | tr '[a-z]' '[A-Z]'
2233|A.K.SHUKLA|G.M.|SALES|12/12/52|6000
9876|JAI SHARMA|DIRECTOR|PRODUCTION|12/03/50|7000
5678|SUMIT CHAKROBARTY|D.G.M|MARKETING|19/04/43|6000
```

-c	Complement the set of characters specified by string1
-d	Delete all occurrences of input characters that are specified by string1
string1	First string or character to be changed
string2	Second string or character to change the string1

43) History:- Manipulate the history list.
Syntax:- history

Description:-

- The history command performs one of several operations related to recently-executed commands recorded in a history list. Each of these recorded commands is referred to as an ``event''.

44) Write:- Send a message to another user.
Syntax:- write person [ttyname]

Description:-

Person	If you wish to talk to someone on your own machine, then person is just the person's login name. If you wish to talk to a user on another host, then person is of the form 'user@host'.
Ttyname	If you wish to talk to a user who is logged in more than once, the ttyname argument may be used to indicate the appropriate terminal name, where ttyname is of the form 'ttyXX' or 'pts/X'.

45) Wall: - send a message to everybody's terminal.
Syntax :- wall [message]

- Wall sends a message to everybody logged in with their mesg(1) permission set to yes. The message can be given as an argument to *wall*, or it can be sent to *wall*'s standard input. When using the standard input from a terminal, the message should be terminated with the EOF key (usually Control-D).
- The length of the message is limited to 20 lines.

46) Grep:-
Syntax:- grep options pattern filename

Description:-

- This command searches the specified input fully for a match with the supplied pattern and displays it.
- While forming the patterns to be searched we can use shell metacharacters,or regular expressions.
- Let us begin with the simplest example of usage of grep.

$grep picture newsfile

- This would search the word 'picture' in the file newsfile and if found, the line s containing it would be displayed on the screen.
- We can use grep to search a pattern in several file .

Example :-

```
$grep picture newsfilestoryfile
```

- Here the word picture would be searched in both the files, newsfile and storyfile and if found the lines containing it would be displayed along with the name of the file where it occurred.

47) Pwd:-Displaying your current directory name(Print working directory).
Syntax:-pwd

Description:-

- At the time of logging in user is placed in the specific directory of the file system. You can move around from one directory to another , but any point of time , you are located in only one directory. This directory is known as your current directory. pwd command tells your current directory.

Example:-

```
$ pwd
/C/Documents and Settings/Bhavesh
```

48) wc:- counting lines , words and characters
Syntax: - wc [*options*] [file ...]

Description:-

- This command counts lines , words and characters depending on the options used. It takes one or more filenames as its arguments and displays four-columnar output. For example let's read our os1 file. And we use wc command with that filename.

Example :-

```
$ cat file1
 Memory
 Process
 Files
$ wc file1
3  3  21
```

- Wc counts 3 lines, 3 words and 21 characters. The file name is also shown in the fourth column. Here line means any group of characters not containing a new line , a word means a group of characters not containing a space, tab or newline and character means the smallest unit of information and includes a space, tab and new line.

l, --lines	Writes the line counts
w --words	Writes the word counts
c – chars	Writes the Char counts

49) **Man**:- The man command is short for manual and provides in depth information about the requested command or allows users to search for commands related to a particular keyword.

Syntax:- man Command name
Example:-
```
$man finger
```

50) **| (Pipeline command)** :- Used to combine more than one command. Takes output of 1st command as input of 2nd command.

Syntax:- commmand1| command2|……
Example:-
```
$ls -l | grep "Aug"
-rw-rw-rw-   1 john  doc     11008 Aug  6 14:10 ch02
-rw-rw-rw-   1 john  doc      8515 Aug  6 15:30 ch07
-rw-rw-r--   1 john  doc      2488 Aug 15 10:51 intro
-rw-rw-r--   1 carol doc      1605 Aug 23 07:35 macros
```

51) **Explain the basic system administration commands of unix**

The basic system administration commands are :

1) su command : Acquiring the super user status
Description:-
- Any user can acquire superuser status with su command if she knows the root password.

Example:-
```
$ su
Password : ******
# pwd
/home/juliet
```
- Here user juilet becomes superuser.
- Here current directory doesn't change but # prompt indicates Juliet now has powers of a super user.

2) date : setting the system date

Description:-
- Normal user uses date command to display the system date.
- The Administrator uses same command with numeric argument to set the system date.
- This argument is usually an eight character string of the form MMDDhhmm, optionally followed by two or four digit year string.

Example:-
```
# date 01092124
Thu Jan 9 21:24:00 IST 2003
```

3) wall : communicating with users.

Description:-

- Wall command addresses all users simultaneously.

Example:-

```
# wall
The machine will be shut down today at 14:30 hrs.
[ctrl +d]
```

- All users currently logged in will receive this message on their terminal.

4) ulimit : Setting limits on file size

Description:-

- The ulimit command imposes a restriction on the maximum size of the file that a user is permitted to create , so that there is less wastage of disk space.

Example:-

Ulimit 20971510 *Measured in 512 – byte blocks*

5) passwd : Changing any password

Description:-

- Passwd prompts for the exixting password when the command is used by a nonprivileged user.
- However , when super user uses this command , it behaves differently.

Example:-

```
# passwd
 Changing password for root
Enter new password (mininmum of 5 , maximum of 8 characters)
Please use a combination of upper and lower case letters and numbers.
New password : *********
Re – enter password : *********
Password changed.
```

- Administrator has privilege of changing any users password.

6) SUID : set user id.

Description:-

- Sets special mode to the file .
- This mode enables a process to have the privilege of owner of the file during the instance of the program.

Example :-

```
# chmod u+s a.out ; ls –l a.out
-rwsr-xr-rw 1 root staff 2113 Mar 24 11:18 a.out
```

7) SGID : set group id.

Description:-

- SGID is similar to SUID except that a program with SGID set allows the user to have same power as the group which owns the program.

Example :-

```
# chmod g+s a.out ; ls –l a.out
-rw-sxr-rw 1 root staff 2113 Mar 24 11:18 a.out
```

8) du : disk usage.

Description:-

- du command list the usage of entire file system.

Example :-

```
# du /home/sales/tml
11554  /home/sales/tml/forms
12820  /home/sales/tml/data
638     /home/sales/tml/database
```

-a	Write counts of all files , not just directories.
-b	Print size in bytes
-c	Displays at last grand total of all files sizes.
-h	Human readable format

```
# du -ah
0          ./redhat/rh7
4.0K       ./redhat
4.0K       ./testfile.txt
0          ./linuxKernel
0          ./ubuntu/ub10
4.0K       ./ubuntu
```

1) Write a shell script to check whether given number is positive or negative.

Program :-

```
echo "enter num"
read a
if [ $a -eq 0 ]
then
echo "$a : You must give/supply one integers"
exit 1
fi
if [ $a -gt 0 ]
then
echo "$1 number is positive"
else
echo "$1 number is negative"
fi
```

Output :-

```
[student@localhost ~]$ sh positive.sh
enter num
5
 number is positive
[student@localhost ~]$ sh positive.sh
enter num
-5
 number is negative
```

2) Write a shell script to check whether given number is even or odd.

Program :-

```
echo "Even And Odd Numbers in a given Range:" echo "Enter range:"
read n
k=1
m=`expr $k % 2`
echo "Odd numbers are"
while [ $k -le $n ]
do
if [ $m -eq 0 ] then
k=`expr $k + 2`
fi
echo "$k"
k=`expr $k + 2`
```

```
done
k=2
echo "Even numbers are"
while [ $k -le $n ]
do
echo "$k" k=`expr $k + 2`
 done
```

Output :-

```
[student@localhost ~]$ sh Even.sh
And Odd Numbers in a given Range:
Enter range: 10
Odd numbers are 1 3 5 7 9
Even numbers are 2 4 6 8 10
```

3) To find the largest number from given three numbers.

Program :-

```
echo "To find Greatest among three numbers"
echo "Enter a value:"

read a

echo "Enter b value:"
 read b

echo "Enter c value:"
read c

if [ $a -ge $b -a $a -ge $c ]
then

echo "a=$a is Greater among $a,$b,$c"
elif [ $b -gt $a -a $b -ge $c ]

then

echo "b=$b is Greater among $a,$b,$c"
else

echo "c=$c is Greater among $a,$b,$c"
```

```
 fi
```

Output :-

```
[student@localhost ~]$ sh largest.sh
find Greatest among three numbers Enter a value:

45

Enter b value: 67

Enter c value: 99

c=99 is Greater among 45, 67, 99
```

4) To check whether given number is digit or alphanumeric or special character.

Program :-

```
echo "To know what type of character is given input"
echo "Enter your input:"
read ch
case $ch in
[a-zA-Z]* )echo "$ch is an ALPHABET";;
[0-9]* ) echo "$ch is a DIGIT";;
 ?) "$ch is a SPECIAL CHARACTER";;
*)
echo "You have entered more than one Character"
Esac
```

Output :-

```
[student@localhost ~]$ sh digit.sh
To know what type of character is
given input Enter your input:
a

a is an ALPHABET
To know what type of character is
given input Enter your input:

7
7 is a DIGIT
To know what type of character is
given input Enter your input:
```

```
*
* is a SPECIAL CHARACTER
```

5) To check given number is Prime or not.

Program :-

```
echo “Enter Any Number”
read n
i=1
c=1
while [ $i -le $n ]
do
i=$(expr $i + 1)
r=$(expr $n % $i)
if [ $r -eq 0 ]
then
c=$(expr $c + 1)
fi
done
if [ $c -eq 2 ]
then
echo “Prime”
else
echo “Not Prime”
fi
```

Output :-

```
[student@localhost ~]$ sh primeornot.sh
“Enter Any Number”
15
“Not Prime”
[student@localhost ~]$ sh primeornot.sh
“Enter Any Number”
5
“Prime”
```

6) To find factorial of a given number.

Program :-

```
echo "Total no of factorial wants"
read fact
ans=1
```

```
counter=0
while [ $fact -ne $counter ]
do
   counter=`expr $counter + 1`
   ans=`expr $ans \* $counter`
done
echo "Total of factorial is $ans"
```

Output :-

```
[student@localhost ~]$ sh fact.sh
Total no of factorial wants
5
Total of factorial is 120
```

7) To print Fibonacci series of given number.

Program :-

```
echo "Fibonacci Series"

a=0
b=1
echo "Enter no.of Terms"
read n
while [ $n -ge 1 ]
do
c=`expr $a + $b` a=$b
b=$c
n=`expr $n - 1` echo "$c"
done
```

Output :-

```
[student@localhost ~]$
sh fib.sh
Fibonacci Series
Enter no.of Terms 5
1 2 3 5 8
```

8) To find sum, average and product of given four numbers.

Program :-

```
echo Enter four integers with space between
read a
read b
read c
read d
sum=`expr $a + $b + $c + $d`
avg=`expr $sum / 4`
dec=`expr $sum % 4`
dec=`expr \( $dec \* 1000 \) / 4`
product=`expr $a \* $b \* $c \* $d`
echo Sum=$sum
echo Average=$avg.$dec
echo Product=$product
```

Output :-

```
[student@localhost ~]$ sh avg.sh
Enter four integers with space between
5
4
9
4
Sum=22
Average=5.500
Product=720
```

9) To check whether the given number is Armstrong or not

Program :-

```
echo "To know the given number is Armstrong or not" echo "Enter any number:"
read n p=$n
while [ $n -gt 0 ]
 do
r=`expr $n % 10`
sum=`expr $sum + $r \* $r \* $r`
 n=`expr $n / 10`
done
if [ $p = $sum ] then

echo "The given number $p is Armstrong"
else
```

```
echo "The given number $p is Not Armstrong"
fi
```

Output :-

```
[student@localhost ~]$ sh amstrong.sh
To know the given number is Armstrong or not Enter any number:

153

The given number 153 is Armstrong
```

10) To check whether the given year is leap year or not.

Program :-

```
echo "To know if the given year is leap year ornot"
 echo "Enter the year:"
read year
if(($year % 4==0 && $year %100!=0))
then
echo "The given year $year is a LEAP YEAR" else
echo "The given year $year is NOT A LEAP YEAR"
fi
```

Output :-

```
[student@localhost ~]$ sh leap.sh
To know if the given year is leap
year or not Enter the year:

2004

The given year 2004 is a LEAP YEAR
```

11) To print reverse of given number.

Program :-

```
echo -n "Input no : "
read no
length=`expr length $no`
while [ $length -ne 0 ]
```

```
do
b=$b`expr substr $no $length 1`
length=`expr $length - 1`
done
echo "Reverse no is " $b
```

Output :-

```
[student@localhost ~]$ sh reverse.sh
Input no : 45698
Reverse no is  89654
```

12) To check whether the given number perfect or not.

Program :-

```
echo "To know if the given number is Perfect or not" echo "Enter any number"
read n
for((i=1;i<n;i++))
 do
if(($n % i==0))
then
sum=`expr $sum + $i`
 fi
done
if(($n==$sum))
then
echo "The given number $n is PERFECT"
 else
echo "The given number $n is NOT PERFECT"
 fi
```

Output :-

```
[student@localhost ~]$ sh perfect.sh
To know if the given number is Perfect or not Enter any number

6
The given number 6 is PERFECT
```

13) To print combination of 123.

Program :-

```
echo "The combination of 1 2 3"
  for i in 1 2 3
  do
  for j in 1 2 3
do
  for k in 1 2 3
  do
  if [ $i -ne $j -a $j -ne $k -a $k -ne $i ]
  then
  echo "$i $j $k"
  fi
 done
 done
 done
```

Output :-

```
[student@localhost ~]$ sh combination.sh
The combination of 1 2 3
12 3
1 3 2
2 1 3
2 3 1
3 1 2
3 2 1
```

14) To find the value of one number raised to power of another.

Program :-

```
echo "To find the power value of two numbers"
echo "Enter a number:"
read n
echo "Enter another number"
 read m
p=1
for((i=1;i<=m;i++))
do
p=`expr $p \* $n`
 done
echo "The value of $n to the power of $m is $p"
```

Output :-

```
[student@localhost ~]$ sh power.sh
find the power value of two numbers Enter a number:
2
Enter another number 4
The value of 2 to the power of 4 is 16
```

15) To print mark sheet from given marks.

Program :-

```
echo "Enter the five subject marks for the student"
read m1
read m2
read m3
read m4
read m5
sum1=`expr $m1 + $m2 + $m3 + $m4 + $m5`
echo "Sum of 5 subjects are: " $sum1
per=`expr $sum1 / 5`
echo "Percentage: " $per
if [ $per -ge 60 ]
then
echo "You get Distinction"
elif [ $per -ge 50 ]
then
echo "You get First class"
elif [ $per -ge  40 ]
then
echo "You get Second class"
else
    echo "You get Fail"
fi
```

Output :-

```
student@localhost ~]$ sh marks.sh
Enter the five subject marks for the student
45
50
60
70
```

```
85
Sum of 5 subjects are:  310
Percentage:  62
You get Distinction
```

16) To display the following details in a pay list Pay slip details, House rent allowance, Dearness allowance, Provident fund. HRA is to be calculated at the rate of 20% of basic, DA at the rate of 40% of basic and PF at the rate of 10% of basic.

Program :-

```
echo "Please enter your Basic:" read basic
echo "PAY SLIP DETAILS"
echo "1. HOUSE RENT ALLOWANCE"
 echo "2. DEARNESS ALLOWANCE"
echo "3. PROVIDENT FUND"
echo "your choice:"
read ch
case $ch in
1) hra=`expr $basic \* 20 / 100`
echo Your HOUSE RENT ALLOWANCE is Rs. $hra;;
2) da=`expr $basic \* 40 / 100`
echo Your DEARNESS ALLOWANCE is Rs. $da;;
3) pf=`expr $basic \* 10 / 100`
echo Your PPOVIDENT FUND is Rs. $pf;;
*) echo "Not a valid choice";;
Esac
```

Output :-

```
student@localhost ~]$ sh basic.sh
Please enter your Basic:
5890
PAY SLIP DETAILS
HOUSE RENT ALLOWANCE
DEARNESS ALLOWANCE
PROVIDENT FUND
your choice: 1
Your HOUSE RENT ALLOWANCE is Rs. 1178
```

17) To display a list of all files in the current directory to which you have read, write and execute permissions.

Program :-

```
for File in *
do
if [ -r $File -a -w $File -a -x $File ]
then
echo $File
fi
done
```

Output:-

```
[student@localhost ~]$ sh file1.sh
Desktop
Documents
Downloads
Music
Pictures
Public
Templates
Videos
```

18) To perform addition, subtraction, multiplication and division of two numbers using case.

Program :-

```
clear
sum=0
i="y"
echo " Enter one no."
read n1
echo "Enter second no."
read n2
while [ $i = "y" ]
do
echo "1.Addition"
echo "2.Subtraction"
echo "3.Multiplication"
echo "4.Division"
echo "Enter your choice"
read ch
case $ch in
```

```
    1)sum=`expr $n1 + $n2`
     echo "Sum ="$sum;;
    2)sum=`expr $n1 - $n2`
     echo "Sub = "$sum;;
   3)sum=`expr $n1 \* $n2`
     echo "Mul = "$sum;;
   4)sum=`expr $n1 / $n2`
     echo "Div = "$sum;;
   *)echo "Invalid choice";;
esac
echo "Do u want to continue ?"
read i
if [ $i != "y" ]
then
   exit
fi
done
```

Output:-

```
[student@localhost ~]$ sh case.sh
Enter one no.
1
Enter second no.
2
1.Addition
2.Subtraction
3.Multiplication
4.Division
Enter your choice
1
Sum =3
```

19) To find the GCD of given number.

Program :-

```
echo Enter two numbers with space in between
read a b
m=$a
if [ $b -lt $m ]
then
m=$b
fi
```

```
while [ $m -ne 0 ]
do
x=`expr $a % $m`
y=`expr $b % $m`
if [ $x -eq 0 -a $y -eq 0 ]
then
echo gcd of $a and $b is $m
break
fi
m=`expr $m - 1`
done
```

Output:-

```
[student@localhost ~]$ sh gcd.sh
Enter two numbers with space in between
2 4
gcd of 2 and 4 is 2
```

20) To print the pyramid structure for the given number.

Program:-

```
echo "enter the number"
read n
for((i=1;i<=n;i++))
do
for ((j=1;j<=i;j++))
do
echo -n "$i "
done
echo " "
done
```

Output:-

```
[student@localhost ~]$ sh pyramid.sh
enter the number
3
1
2 2
3 3 3
```