1) **cal** :- Displays a calendar

   Syntax:- cal [options] [month] [year]

   Description :-
   - ➢ cal displays a simple calendar. If arguments are not specified, the current month is displayed.
   - ➢ In addition to cal, the ncal command ("new cal") is installed on some Linux systems. It provides the same functions of cal, but it can display the calendar vertically (with weeks in columns), and offers some additional options.

Option	Use
-1	Display single (current) 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
-w	Print the number of the week under each week column

   Example:-

2) **clear** :- It clears the terminal screen.

   Syntax :- clear

   Description :-
   - ➢ clear command clears your screen if this is possible, including its scroll back buffer.
   - ➢ It ignores any command-line parameters that may be present.

   Example:-

3) **man** :- man command which is short for manual, provides in depth information about the requested command (or) allows users to search for commands related to a particular keyword.

   Syntax:- man command name [options]

Example:-

```
[test1990@server-1 ~]$man ls

LS(1)                          User Commands                          LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information  about  the  FILEs (the current directory by
       default).  Sort entries alphabetically if  none  of  -cftuvSUX
       nor --sort is specified.

       Mandatory  arguments  to  long options are mandatory for short
       options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters
```

4) **pwd :-** Displays path from root to current directory

    Syntax :- pwd [options]

    Example:

```
[root@localhost Lab_1]# pwd
/root/OS/Lab_1
```

5) **cd** :- It is used to change 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 directory name or ..

Option	Use
cd ..	Change Current directory to parent directory
cd Lab_1	Change from current working directory to lab_1

Example:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ cd lab_1
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/lab_1 $ cd ..
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ cd ..
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC /home $ []
```

6)   ls :- Lists the contents of a directory

Syntax :- ls [options] [file|dir]

Description :-

Option	Use
-a	Shows you all files, even files that are hidden (these files begin with a dot.)
-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
-t	Sort by time & date
-S	Sort by file size
-R	Includes the contents of subdirectories

Example:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ ls -l
total 123816
drwxr-xr-x  2 dietstaff dietstaff      4096 Dec 15  2015 abc
drwx------ 12 dietstaff dietstaff      4096 Jul  7 08:57 ADA_Lab
-rwxr-xr-x  1 dietstaff dietstaff      9030 Aug 25 10:50 a.out
-rw-r--r--  1 dietstaff dietstaff       843 Aug 29  2016 c1.c
-rw-r--r--  1 dietstaff dietstaff       336 Feb 19  2016 calc1.sh
```

Field Explanation:

➢ If first character is – then it is normal file

➢ If it is d then it is 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, rwxr-xr-x indicates read-write-execute permission for user, read-execute permission for group, and read-execute 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 "dietstaff".

➢ **Field 4 – Group**: Fourth field specifies the group of the file. In this example, this file belongs to "dietstaff" group.

➢ **Field 5 – Size**: Fifth field specifies the size of file. In this example, '4096' 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, 'Jul 7 08:57' specifies the last modification time of the file.

➢ **Field 7 – File or directory name**: The last field is the name of the file or directory. In this example, the file name is c1.c.

7) exit :- It is used to terminate a program, shell or log you out of a network normally.
Syntax :- exit

8) echo :- It prints the given input string to standard output.
Syntax :- echo string

Description :-

Option	Use
-n	Do not output a trailing newline
-e	Enable interpretation of backslash escape sequences

Example:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ echo "Hello Linux"
Hello Linux
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ echo -n "hello"
hellodietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ echo -e "hello"
hello
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $
```

9) who :- who command can list the names of users currently logged in, their terminal, the time they have been logged in, and the name of the host from which they have logged in.
Syntax :- who [options] [file]

Description:-

Option	Use
-b	Prints time of last system boot
-H	Print column headings above the output
-a	Display all details of current logged in user
-q	Prints only the usernames and the user count/total no of users logged in

Example :-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ who
dietstaff tty8          2018-12-31 11:55 (:0)
dietstaff pts/2         2018-12-31 11:57 (:0)
dietstaff pts/1         2018-12-31 11:57 (:0)
dietstaff pts/6         2018-12-31 11:57 (:0)
```

10) whoami:- Print effective userid
Syntax :- whoami

Description: - Print the user name associated with the current effective user id.

Example :-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ whoami
dietstaff
```

11) **mkdir** :- This command is used to create a new directory
Syntax :- mkdir [options] directory

Description :-

Option	Use
-m	Set permission mode
-p	No error if existing, make parent directories as needed
-v	Print a message for each created directory

Example:-

```
[root@localhost ~]# mkdir OS
[root@localhost ~]# ls
OS        dos        hello.c
[root@localhost ~]# mkdir OS
mkdir: can't create directory 'OS': File exists
[root@localhost ~]# mkdir -p OS
[root@localhost ~]# mkdir -v Lab_2
created directory: 'Lab_2'
[root@localhost ~]# mkdir -p /root/D1/D2
[root@localhost ~]# ls
D1        Lab_2        OS        dos        hello.c
```

12) **rmdir** :- It is used to delete/remove a directory and its subdirectories.
Syntax :- rmdir [options..] Directory

Description :-
  ➢ It removes only empty directory.

Option	Use
-p	Remove directory and its ancestors

Example:-

```
[root@localhost Lab_2]# ls
dir1
[root@localhost Lab_2]# rmdir dir1
[root@localhost Lab_2]# ls
[root@localhost Lab 2]#
[root@localhost ~]# pwd
/root
[root@localhost ~]# mkdir -p /root/a/b/c
[root@localhost ~]# ls
D1       Lab_2    OS        a        abc       dir1      dos        hello.c
[root@localhost ~]# rmdir -p a/b/c
[root@localhost ~]# ls
D1       Lab_2    OS        abc      dir1      dos       hello.c
```

13) bc :- bc command is used for command line calculator. It is similar to basic calculator.

By using which we can do basic mathematical calculations.

Syntax :- bc [options]

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.

Option	Use
-q	To avoid bc welcome message
-l	To include math library functionalities

Example:-

```
[root@localhost ~]# cat calc.txt
12+4
[root@localhost ~]# bc -l calc.txt
bc 1.06.95
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
16
```

14) uname :- It is used to print system information.

Syntax :- uname [options]

Description :-

> ➢ Print certain system information.

Option	Use
-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:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ uname
Linux
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ uname -r
3.13.0-24-generic
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ uname -s
Linux
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ uname -v
#46-Ubuntu SMP Thu Apr 10 19:11:08 UTC 2014
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ uname -n
dietstaff-HP-Elite-7100-Microtower-PC
```

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 :-

```
[test1990@server-1 ~]$tty
/dev/pts/27
```

16) stty :- 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 :-

```
[test1990@server-1 ~]$stty
speed 38400 baud; line = 0;
ixany
tab3
-echok
```

17) cat :- It is used to create, display and concatenate file contents.

Syntax : - cat [options] [FILE]…

Description :-

Option	Use
-b	Omits line numbers for blank space in the output
-E	Displays a $ (dollar sign) at the end of each line
-n	Line numbers for all the output lines
-s	If the output has multiple empty lines it replaces it with one empty line
-T	Displays the tab characters as ^I in the output

- ➢ Basically three uses of the cat command.
1) Create new files.
2) Display the contents of an existing file.
3) Concatenate the content of multiple files and display.

Example :-

```
[root@localhost ~]# cat > demo.txt
hello
linux
^C
[root@localhost ~]# cat demo.txt
hello
linux
[root@localhost ~]#
```

18) cp :- cp command copy files from one location to another. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

Syntax :- cp [option] source destination/directory

Description:-

- ➢ It will copy the content of source file to destination file.
- ➢ If the destination file doesn't exist, 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.

Option	Use
-i	interactive - ask before overwrite
-f	force copy by removing the destination file if needed
-v	print informative messages
-l	link files instead of copy
-s	follow symbolic links
-n	no file overwrite
-u	update - copy when source is newer than destination
-R	copy directories recursively

Example:-

```
[root@localhost ~]# cat > f1.txt
hi
good morning
^C
[root@localhost ~]# ls
D1        Lab_2    OS        abc        dir1      dos       f1.txt    file1.c   hello.c
[root@localhost ~]# cp f1.txt new.txt
[root@localhost ~]# ls
D1        OS        dir1      f1.txt     hello.c
Lab_2     abc       dos       file1.c    new.txt
[root@localhost ~]# cat new.txt
hi
good morning
```

19) **rm** :- It is used to remove/delete the file from the directory.

Syntax :- rm [options..] [file|directory]

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.

  ➢ 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.

Option	Use
-d	Delete an empty directory
-r	Remove directories and their contents recursively
-f	Ignore non-existent files, and never prompt before removing
-i	Prompt before every removal

Example :-

```
[root@localhost ~]# ls
dos        hello.c
[root@localhost ~]# rm hello.c
[root@localhost ~]# ls
dos
```

```
[root@localhost ~]# ls
dos        f1.txt   f2.txt   f3.txt
[root@localhost ~]# rm -i *.txt
rm: remove 'f1.txt'? y
rm: remove 'f2.txt'? y
rm: remove 'f3.txt'? n
[root@localhost ~]# ls
dos        f3.txt
```

20) **mv** :- It is used to move/rename file from one directory to another.

Syntax :- mv [options] oldname newname

Description :-

➢ mv command which is short for move.
➢ mv command is different from cp command as it completely removes the file from the source and moves to the directory specified, where cp command just copies the content from one file to another.
➢ 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.

Option	Use
-i	Prompts before overwriting another file
-f	Force move by overwriting destination file without prompt
-n	Never overwrite any existing file
-u	Update – move when source is newer than destination
-v	Print informative messages

Example:-

```
[root@localhost ~]# ls
D1          OS          data.txt    dos         f2.txt      hello.c     new.txt
Lab_2       abc         dir1        f1.txt      file1.c     myData.txt  new1.txt
[root@localhost ~]# mv f1.txt dir1
[root@localhost ~]# ls
D1          OS          data.txt    dos         file1.c     myData.txt  new1.txt
Lab_2       abc         dir1        f2.txt      hello.c     new.txt
[root@localhost ~]# cd dir1
[root@localhost dir1]# ls
f1.txt
```

21) **nl** :- nl numbers the lines in a file.

Syntax: - nl [OPTION] [FILE]

Description :-

Option	Use
-i	line number increment at each line
-s	add STRING after (possible) line number
-w	use NUMBER columns for line numbers

Example :-

22) **cut** :- cut command is used to cut out selected fields of each line of a file. The cut command uses delimiters to determine where to split fields.

Syntax :- cut [options] filename

Description :-

Option	Use
-c	The list following -c specifies character positions
-d	The character following -d is the field delimiter
-f	Select only these fields on each line
-b	Select only the bytes from each line as specified in LIST

Example :-

23) paste:- paste command is used to paste the content from one file to another file. It is also used to set column format for each line.

Syntax :- paste [option] 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.

Option	Use
-d	Specify of a list of delimiters
-s	Paste one file at a time instead of in parallel

Example:-

```
[test1990@server-1 ~]$cat empID.txt
1
2
3
4
[test1990@server-1 ~]$cat empName.txt
abc
pqr
xyz
demo
[test1990@server-1 ~]$paste empID.txt empName.txt
1       abc
2       pqr
3       xyz
4       demo
[test1990@server-1 ~]$paste -s empID.txt empName.txt
1       2       3       4
abc     pqr     xyz     demo
```

24) more:- Displays text one screen at a time.

Syntax :- more [options] filename

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.

Option	Use
-c	Clear screen before displaying
-n	Specify how many lines are printed in the screen for a given file
+n	Starts up the file from the given number
-s	Doesn't display extra blank lines

Example:-

25) cmp :- It compares two files and tells you which line numbers are different.

Syntax : - cmp [options..] file1 file2

Description :-

➢ If a difference is found, it reports the byte and line number where the first difference is found.

➢ If no differences are found, by default, cmp returns no output.

Option	Use
-b	Print differing bytes
-i	Skip a particular number of initial bytes from both the files
-n	Compare at most LIMIT bytes
-l	Print byte position and byte value for all differing bytes

Example:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ cat f1.txt
hi linux good morning
how r u
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ cat f2.txt
hello ab good morning
how r u
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ cmp -l f1.txt f2.txt
 2 151 145
 3  40 154
 5 151 157
 6 156  40
 7 165 141
 8 170 142
```

26) **comm** :- compare two sorted files line by line

Syntax:- comm [option]... FILE1 FILE2

Description :-

➢ Compare sorted files FILE1 and FILE2 line by line.

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

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

Option	Use
-1	suppress lines unique to FILE1
-2	suppress lines unique to FILE2
-3	suppress lines that appear in both files
--check-order	check that the input is correctly sorted, even if all input lines are pairable
--nocheck-order	do not check that the input is correctly sorted

Example :-

```
[test1990@server-1 ~]$cat f1.txt
abc
def
ghi
[test1990@server-1 ~]$cat f2.txt
abc
ghi
klm
[test1990@server-1 ~]$comm f1.txt f2.txt
                abc
def
                ghi
        klm
```

➢ 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 a prefix.

27) diff :- It is used to find differences between two files.

Syntax :- diff [options..] fileone filetwo

Description :-

➢ Diff is the third command that can be used to display file differences. Unlike its fellow members, cmp and comm, it tells us which lines in one file have is to be changed to make the two files identical.

Option	Use
-b	Ignore any changes which only change the amount of whitespace (such as spaces or tabs)
-B	Ignore blank lines when calculating differences
-i	Ignore changes in case. consider upper- and lower-case letters equivalent

Example:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ cat f3.txt
hello
good morning
all
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ cat f4.txt
hello
good morning
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ diff f3.txt f4.txt
3d2
< all
```

- **d – a line was deleted**
- **c – a line was changed**
- **a – a line was added**

28) chmod :- chmod command allows you to alter / Change access rights to files and directories.

Syntax:- chmod [options] [MODE] FileName

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

#	File Permission
0	none
1	execute only
2	write only
3	write and execute
4	read only
5	read and execute
6	read and write
7	set all permissions

Example :

```
[root@localhost ~]# chmod 764 f2.txt
[root@localhost ~]# chmod u=rwx,g=rw,o=r f3.txt
[root@localhost ~]# ls -l
total 20
drwxrwxrwx     3 root      root          163 Aug 21  2011 dos
-rwxrw----    1 root      root           12 Jan 13 15:35 f1.txt
-rwxrw-r--    1 root      root           13 Jan 13 15:36 f2.txt
-rwxrw-r--    1 root      root           16 Jan 13 15:54 f3.txt
-rw-r--r--    1 root      root          242 Jul 15  2017 hello.c
```

29) chown :- Command for system V that changes the owner of a file.

Syntax :- chown [options] newowner filename/directoryname

Example :-

```
$chown rimmer myfile
$ls -l myfile
-rw-r--r--  1 rimmer  scifi  112640 Jan 04 10:49 myfile
```

30) chgrp :- chgrp command is used to change the group of the file or directory. This is an admin command. Root user only can change the group of the file or directory.

Syntax:- chgrp [options] newgroup filename/directoryname

Example :-

```
$ ls -l
-rw-r--r-- 1 john john 210 2018-01-04 16:01
/home/john/sample.txt
$ chgrp user /home/john/sample.txt
$ ls -l
-rw-r--r-- 1 john user 210 2018-01-04 16:01 /home/john/sample.txt
```

31) **file** :- file command tells you if the object you are looking at is a file or a directory.

Syntax:- file [options] directoryname/filename

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.

Option	Use
-i	To view the mime type of a file rather than the human readable format

Example :-

```
[test1990@server-1 ~]$ls
d1   f1   f1.c   f1.txt   f2   f2.txt   f3.txt
[test1990@server-1 ~]$file f1.txt
f1.txt: ASCII text
```

32) **finger** :- finger command displays the user's login name, real name, terminal name and write status (as a ''*'' after the terminal name if write permission is denied), idle time, login time, office location and office phone number.

Syntax:- finger [username]

Description :-

Option	Use
-i	Force long output format
-m	Match arguments only on user name (not first or last name)

Example :-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ finger dietstaff
Login: dietstaff                        Name: ced
Directory: /home/dietstaff              Shell: /bin/bash
On since Thu Jan 10 12:26 (IST) on tty8 from :0
    22 minutes 18 seconds idle
On since Thu Jan 10 12:27 (IST) on pts/4 from :0
    4 seconds idle
No mail.
No Plan.
```

33) **sleep** :- Delay for a specified amount of time

Syntax :- sleep NUMBER[SUFFIX]

Description:-

➢ The sleep command pauses for an amount of time defined by NUMBER.

➢ SUFFIX may be "s" for seconds (the default), "m" for minutes, "h" for hours, or "d" for days.

Example :-

```
[root@localhost ~]# sleep 5
```

34) **ps** :- It is used to report the process status. ps is the short name for Process Status.
Syntax:- ps [options]

Description :-

Option	Use
-e	Display every active process on a Linux system in generic (Unix/Linux) format.
-x	View all processes owned by you
-u	Filter processes by its user
-F	To provide more information on processes.

Example :-

```
[test1990@server-1 ~]$ps
  PID TTY          TIME CMD
26299 pts/329   00:00:00 sh
26375 pts/329   00:00:00 sleep
26377 pts/329   00:00:00 ps
```

35) **kill** :- kill command is used to kill the background process.
Syntax:- kill [options] 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.

Option	Use
-s	send the specified signal to the process
-l	list all the available signals.
-9	Force to kill a process.

Example :-

```
[test1990@server-1 ~]$ps
  PID TTY             TIME CMD
26299 pts/329  00:00:00 sh
26712 pts/329  00:00:00 less
27875 pts/329  00:00:00 ps
[test1990@server-1 ~]$kill -9 26299
Session closed.
```

36) **ln** :- ln command is used to create link to a file (or) directory. It helps to provide soft link for desired files.

Syntax:- ln [options] existingfile(or directory)name newfile(or directory)name
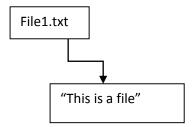
Description:-

What Is A Link?

➢ A link is an entry in your file system which connects a filename to the actual bytes of data on the disk. More than one filename can "link" to the same data. Here's an example. Let's create a file named file1.txt:
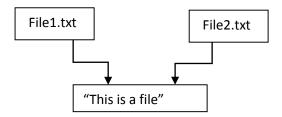
$ echo "This is a file." > file1.txt

➢ This command echoes the string "This is a file". Normally this would simply echo to our terminal, but the > operator redirects the string's text to a file, in this case file1.txt

➢ When this file was created, the operating system wrote the bytes to a location on the disk and also linked that data to a filename, file1.txt so that we can refer to the file in commands and arguments.

➢ If you rename the file, the contents of the file are not altered; only the information that points to it.

➢ The filename and the file's data are two separate entities.

```
┌─────────────┐
│  File1.txt  │
└──────┬──────┘
       │
       ▼
┌─────────────┐
│"This is a file"│
└─────────────┘
```

➢ What the link command does is allow us to manually create a link to file data that already exists.

➢ So, let's use link to create our own link to the file data we just created. In essence, we'll create another file name for the data that already exists.

$ link file1.txt file2.txt

➢ The important thing to realize is that we did not make a copy of this data. Both filenames point to the same bytes of data on the disk. Here's an illustration to help you visualize it:

- If we change the contents of the data pointed to by either one of these files, the other file's contents are changed as well. Let's append a line to one of them using the >>operator:
  $ echo "Hello Linux" >> file1.txt
- Now let's look at the contents of file1.txt:
  $ cat file1.txt
  This is a file
  Hello Linux
- Now let's look at the second file, the one we created with the link command.
  $ cat file2.txt
  This is a file
  Hello Linux
- ln, by default, creates a hard link just like link does. So this ln command:
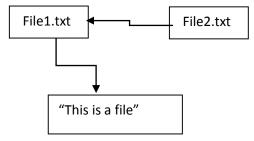 **$ ln file1.txt file2.txt**
- It is the same as the following link command. Because, both commands create a hard link named file2.txt which links to the data offile1.txt.
  $ link file1.txt file2.txt
- However, we can also use ln to create symbolic links with the -s option. So the command:
  $ ln -s file1.txt file2.txt
- It will create a symbolic link to file1.txt named file2.txt. In contrast to our hard link example, here's an illustration to help you visualize our symbolic link:

- You should also be aware that, unlike hard links, removing the file (or directory) that a symlink(symbolic linkl) points to will break the link. So if we create file1.txt:
  $ echo "This is a file." > file1.txt
- Now, create a symbolic link to it:
  $ ln -s file1.txt file2.txt
- we can cat either one of these to see the contents:
  $ cat file1.txt
  This is a file.
  $ cat file2.txt
  This is a file.

> But, if we remove file1.txt:
> **$ rm file1.txt**
> we can no longer access the data it contained with our symlink:
> **$ cat file2.txt**

cat: file2.txt: No such file or directory

Option	Use
-s	Makes it so that it creates a symbolic link
-f	If the destination file or files already exist, overwrite them
-i	Prompt the user before overwriting destination files

37) head :- head command is used to display the first ten lines of a file, and also specifies how many lines to display.

Syntax:- head [options] filename

Description:-

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

Option	Use
-n	To specify how many lines you want to display
-n number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in lines
-c number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in bytes

Example :-

```
[root@localhost ~]# cat data.txt
1 ab
2 hello
3 linux
4 how
5 are
6 you
7 good
8 noon
9 to
10 all
11 demo
12 here
13 data
```

```
[root@localhost ~]# head data.txt
1 ab
2 hello
3 linux
4 how
5 are
6 you
7 good
8 noon
9 to
10 all
[root@localhost ~]#
```

38) tail :- tail command is used to display the last or bottom part of the file. By default it displays last 10 lines of a file.

Syntax :- tail [options] filename

Description:-

Option	Use
-c number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in bytes

-n number	The number option-argument must be a decimal integer whose sign affects the location in the file, measured in lines

Example : -

39) **sort** :- It is used to sort the lines in a text file.

Syntax:- sort [options] filename

Description:-

➢ By default, the sort command sorts file assuming the contents are ASCII. Using options in sort command, it can also be used to sort numerically.

Option	Use
-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
-n	Compare according to string numerical value
-nr	To sort a file with numeric data in reverse order
-k	Sorting a table on the basis of any column

Example:-

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

Description :-

  ➢ Find is one of the powerful utility of Unix (or Linux) used for searching the files in a directory hierarchy

Option	Use
-name filename	Search for files that are specified by 'filename'
-newer filename	Search for files that were modified/created after 'filename'
-user filename	Search for files owned by user name or ID 'name'
-size +N/-N	Search for files of 'N' blocks; 'N' followed by 'c' can be used to measure size in characters
-empty	Search for empty files and directories
-perm octal	Search for the file if permission is 'octal'

Example:-

41) **uniq** :- Report or filter out repeated lines in a file.
Syntax:- uniq [option] filename

Description : -

  ➢ It can remove duplicates, show a count of occurrences, show only repeated lines, ignore certain characters and compare on specific fields.

Option	Use
-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
-D	Print all duplicate lines
-f	Avoid comparing first N fields
-i	Ignore case when comparing
-s	Avoid comparing first N characters
-u	Prints only unique lines

Example:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ cat hello.txt
hello
hello
good morning
linux
linux
linux
how r u
all
all
linux
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ uniq hello.txt
hello
good morning
linux
how r u
all
linux
```

42) tr :- Translate characters.

Syntax:- tr [options] set1 [set2]

Description:-

- ➢ It supports a range of transformations including uppercase to lowercase, squeezing repeating characters, deleting specific characters and basic find and replace.
- ➢ It can be used with UNIX pipes to support more complex translation.
- ➢ tr stands for translate.
- ➢ POSIX Character set supported by tr command :
  - • [:digit:] Only the digits 0 to 9.
  - • [:alnum:] Any alphanumeric character.
  - • [:alpha:] Any alpha character A to Z or a to z.
  - • [:blank:] Space and TAB characters only.
  - • [:xdigit:] Hexadecimal notation 0-9, A-F, a-f.
  - • [:upper:] Any alpha character A to Z.
  - • [:lower:] Any alpha character a to z.

Option	Use
-c	Use the complement of SET1
-d	Delete characters in SET1, do not translate
-s	Replace each input sequence of a repeated character that is listed in SET1 with a single occurrence of that character

Example:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ cat data.txt
hello
good morning
linux
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ cat data.txt | tr "[a-z]" "[A-Z]"
HELLO
GOOD MORNING
LINUX
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ cat data.txt | tr "[:lower:]" "[:upper:]"
HELLO
GOOD MORNING
LINUX
```

43) history :- history command is used to list out the recently executed commands in the number line order.
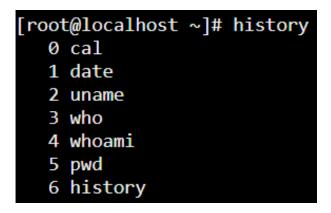
Syntax:- history [options]

Description:-

➢ The history command performs one of several operations related to recently-executed commands recorded in a history list.

Option	Use
-c	Clear the history list by deleting all of the entries

Example :-

```
[root@localhost ~]# history
    0 cal
    1 date
    2 uname
    3 who
    4 whoami
    5 pwd
    6 history
```

44) write :- Send a message to another user.

Syntax:- write person [ttyname]

Description:-

➢ The write utility allows you to communicate with other users, by copying lines from your terminal to theirs.

➢ When you run the write command, the user you are writing to gets a message of the format:

Message from yourname@yourhost on yourtty at hh:mm ...

➢ Any further lines you enter will be copied to the specified user's terminal. If the other user wants to reply, they must run write as well.

➢ When you are done, type an end-of-file or interrupt character. The other user will see the message 'EOF' indicating that the conversation is over.

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'

Example :-

```
                              Terminal                        _
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ write dietstaff pts/6
hello
linux
```

```
                              Terminal
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $
Message from dietstaff@dietstaff-HP-Elite-7100-Microtower-PC on pts/2 at 12:17 ...
hello
linux
```

45) grep :- It selects and prints the lines from a file which matches a given string or pattern.

Syntax:- grep [options] pattern [file]

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 match characters, or regular expressions.
 ➢ grep stands for globally search for regular expression and print out.

Option	Use
-i	Ignore case distinctions
-v	Invert the sense of matching, to select non-matching lines.
-w	Select only those lines containing matches that form whole words
-x	Select only matches that exactly match the whole line.
-c	Print a count of matching lines for each input file.
-n	Display the matched lines and their line numbers
-o	Print only the matched parts of a matching line

Example :-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ cat data.txt
hello
good morning hello
linux
linux
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ grep hello data.txt
hello
good morning hello
```

46) **pwd** :-Displaying your current directory name (Print working directory).
Syntax:-pwd [options]

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:-
```
[root@localhost Lab_1]# pwd
/root/OS/Lab_1
```

47) **wc** :- Word Count (wc) command counts and displays the number of lines, words, character and number of bytes enclosed in a file.
Syntax: - wc [options] [filename]

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.

Option	Use
-l	Print the newline counts
-w	Print the word counts
-c	Print the byte counts
-L	Print the length of the longest line

Example :-
```
[root@localhost ~]# wc -L hello.c
38 hello.c
```

49) **| (Pipeline command)** :- The Pipe is a command in Linux that lets you use two or more commands such that output of one command serves as input to the next.
Syntax: - command_1 | command_2 | command_3 | …. | command_N…

Description:-
  ➢ The symbol '|' denotes a pipe.
  ➢ Pipes help you mash-up two or more commands at the same time and run them consecutively.
  ➢ In short, the output of each process directly as input to the next one like a pipeline.

Example:-
```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ cat data.txt
1 abc 45,000 Rajkot
5 abc 45,000 Surat
4 xyz 42,00 Rajkot
3 emp 52,000 Surat
2 pqr 33,000 Ahmedabad
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~/OsLab $ cat data.txt | head -5 | tail -2
3 emp 52,000 Surat
3 emp 52,000 Surat
```

Basic system administration commands of unix

1) **date** : - Prints or sets the date and time.

   Syntax :- date[options] [+format] [date]

   Description :-
   - ➢ Display the current date with current time, time zone.
   - ➢ 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.

Format	Use
%a	Abbreviated weekday(Tue)
%A	Full weekday(Tuesday)
%b	Abbreviated month name(Jan)
%B	Full month name(January)
%c	Country-specific date and time format
%D	Date in the format %m/%d/%y
%j	Julian day of year (001-366)
%p	String to indicate a.m. or p.m.
%T	Time in the format %H:%M:%S
%t	Tab space
%V	Week number in year (01-52); start week on Monday

Example:-

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date
Fri Dec 21 10:38:44 IST 2018
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%a
Fri
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%A
Friday
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%b
Dec
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%B
December
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%D
12/21/18
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%j
355
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%p
AM
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%T
10:39:26
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ date +%V
51
```

2) 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.
> ➢ wall sends a message to everybody logged in with their mesg permission set to yes.

Example : -

```
                              Terminal                       – + ×
dietstaff-HP-Elite-7100-Microtower-PC dietstaff # cat msg.txt
hello
how r u all
dietstaff-HP-Elite-7100-Microtower-PC dietstaff # wall msg.txt

Broadcast Message from dietstaff@dietstaf
        (/dev/pts/2) at 12:28 ...

hello
how r u all

dietstaff-HP-Elite-7100-Microtower-PC dietstaff #
```

3) passwd :- It is used 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.
> ➢ First user has to insert current password. Then new password will be asked followed by confirm new password field.
> ➢ passwd command can also be used to change the home directory where the path stands for the home directory.

Example : -

```
dietstaff@dietstaff-HP-Elite-7100-Microtower-PC ~ $ passwd
Changing password for dietstaff.
(current) UNIX password: ▯
```