1.1

```
1. $ date
   Thu Jan-23  08:19:43  IST 2025
2. $ date +%A
   Thursday
3. $ date +%B
   January
4. $date +%m
   01
5. $date +%h
   Jan
6 $date +%d
   23
7. $ date +%y
   25
8. $date +%H
   08
9. $date +% M
   33
10. $date +% S
   07
11. $ echo "Welcome to OS"
   Welcome to OS
12. $ cal Oct 2024
          October 2024
   Su Mo Tu We Th Fr Sa
          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 31
13. $ bc
   3+5
   8
```

Ex No: 1b)

Date: 31|1|25

BASIC LINUX COMMANDS

1.1 GENERAL PURPOSE COMMANDS

1. The 'date' command:

The date command displays the current date with day of week, month, day, time (24 hours clock) and the year.

SYNTAX: $ date

The date command can also be used with following format.

Format	Purpose	Example
+ %m	To display only month	$ date + %m
+ %h	To display month name	$ date + %h
+ %d	To display day of month	$ date + %d
+ %y	To display last two digits of the year	$ date + %y
+ %H	To display Hours	$ date + %H
+ %M	To display Minutes	$ date + %M
+ %S	To display Seconds	$ date + %S

2. The echo'command:

The echo command is used to print the message on the screen.

SYNTAX: $ echo

EXAMPLE: $ echo "God is Great"

3. The 'cal' command:

The cal command displays the specified month or year calendar.

SYNTAX: $ cal [month] [year]

EXAMPLE: $ cal Jan 2012

4. The 'bc' command:

14. $who

```
root      pts/0    2025-01-23   08:14 (:0)
cse368    pts/1    2025-01-23   08:16 (172.16.9.18)
cse387    pts/5    2025-01-23   08:17 (172.16.9.11)
cse369    pts/12   2025-01-23   08:18 (172.16.9.17)
  :
```

15. $who am i

```
cse369    pts/12   2025-01-23   08:18 (172.16.9.17)
```

16. $id

```
uid=1370(cse369) gid=1370(cse369) groups=1370(cse369)
context=unconfined_u:unconfined_r:unconfined_t:s0-s0:
c0.c1023
```

17. $tty

```
/dev/pts/12
```

18. $man cat

```
NAME
    cat - concatenate files and print on the standard output
SYNOPSIS
       cat [OPTION]...   [FILE]...
DESCRIPTION
       Concatenate FILE(S) to standard output
       With no FILE, or when FILE is -, read standard input.

       -A,  --show-all
            equivalent to -vET
       -b,  --number-nonblank
        :   number nonempty output lines, overrides -n
```

(11) $PS

```
PID   TTY          TIME      CMD
1730  pts/12       00:00:00  bash
5025  pts/12       00:00:00  ps
```

$PS -e

```
  PID  TTY          TIME      CMD
   1    ?           00:00:02  systemd
   2    ?           00:00:00  kthreadd
   :    :              :        :
```

Unix offers an online calculator and can be invoked by the command bc.

SYNTAX: $ bc

EXAMPLE: bc –l

16/4

5/2

5. The 'who' command

The who command is used to display the data about all the users who are currently logged into the system.

SYNTAX: $ who

6. The 'who am i' command

The who am i command displays data about login details of the user.

SYNTAX: $ who am i

7. The 'id' command

The id command displays the numerical value corresponding to your login.

SYNTAX: $ id

8. The 'tty' command

The tty (teletype) command is used to know the terminal name that we are using.

SYNTAX: $ tty

9. The 'clear' command

The clear command is used to clear the screen of your terminal.

SYNTAX: $ clear

10. The 'man' command

The man command gives you complete access to the Unix commands.

SYNTAX: $ man [command]

11. The 'ps' command

The ps command is used to the process currently alive in the machine with the 'ps' (process status) command, which displays information about process that are alive when you run the command. 'ps;' produces a snapshot of machine activity.

SYNTAX: $ ps

EXAMPLE: $ ps

$ ps –e

$ps -aux

(12) $ uname -n

```
localhost.localdomain
$ uname -s
Linux
$ uname -v
#1 SMP Thu Jun 29 20:38:21 UTC 2017
```

(1.2)

1. $ pwd

```
/home/cse369
```

2. $mkdir 123

```
$ cd 123
[cse369@localhost 123] $ cd
[[cse369@localhost ~] $ vi add.c
[[cse369@localhost ~] $ cc add.c
[[cse 369@localhost ~] $ .a/a. ./a.out
[[cse 369@localhost

$ ls
add.c
```

3. $ cd:

```
[cse 369@localhost 123] $ cd
[cse 369@localhost ~] $
```

(4) $ls :

```
add.c
```

12. The 'uname' command

The uname command is used to display relevant details about the operating system on the standard output.

-m -> Displays the machine id (i.e., name of the system hardware)

-n -> Displays the name of the network node. (host name)

-r -> Displays the release number of the operating system.

-s -> Displays the name of the operating system (i.e.. system name)

-v -> Displays the version of the operating system.

-a -> Displays the details of all the above five options.

SYNTAX: $ uname [option]

EXAMPLE: $ uname -a

1.2 DIRECTORY COMMANDS

1. The 'pwd' command:

The pwd (print working directory) command displays the current working directory.

SYNTAX: $ pwd

2. The 'mkdir' command:

The mkdir is used to create an empty directory in a disk.

SYNTAX: $ mkdir dirname

EXAMPLE: $ mkdir receee

3. The 'rmdir' command:

The rmdir is used to remove a directory from the disk. Before removing a directory, the directory must be empty (no files and directories).

SYNTAX: $ rmdir dirname

EXAMPLE: $ rmdir receee

4. The 'cd' command:

The cd command is used to move from one directory to another.

SYNTAX: $ cd dirname

EXAMPLE: $ cd receee

5. The 'ls' command:

1.3.

```
1. $cat >filemane
   $ cat >sub.~~py~~c

2. $ cat filename

   $cat sub.~~py~~c
   #include <stdio.h>
   int main ()
   {
      int x,y;
      x=5;
      y=3;
      z=x-y
      printf ("%d",z);
   }

3. $cp
   $cp sub.c add.c

4. $ rm ad
   $rm add.c

5. $mv
   $mv sub.c add.c

6. $ file
   $ file add.c
   add.c: C source, ASCII text
```

The ls command displays the list of files in the current working directory.

SYNTAX: $ ls

EXAMPLE: $ ls

$ ls –l

$ ls –a

1.3 FILE HANDLING COMMANDS

1. The 'cat' command:

The cat command is used to create a file.

SYNTAX: $ cat > filename

EXAMPLE: $ cat > rec

2. The 'Display contents of a file' command:

The cat command is also used to view the contents of a specified file.

SYNTAX: $ cat filename

3. The 'cp' command:

The cp command is used to copy the contents of one file to another and copies the file from one place to another.

SYNTAX: $ cp oldfile newfile

EXAMPLE: $ cp cse ece

4. The 'rm' command:

The rm command is used to remove or erase an existing file

SYNTAX: $ rm filename

EXAMPLE: $ rm rec

$ rm –f rec

Use option –fr to delete recursively the contents of the directory and its subdirectories.

5. The 'mv' command:

The mv command is used to move a file from one place to another. It removes a specified file from its original location and places it in specified location.

SYNTAX: $ mv oldfile newfile

EXAMPLE: $ mv cse eee

6. The 'file' command:

The file command is used to determine the type of file.

SYNTAX: $ file filename

EXAMPLE: $ file receee

1.4

1. Semicolon

```
$ who ; date
student pts/0    2025-01-25    13:30 (:0)
student pts/1    2025-01-25    13.42 (:0)
Sat Jan 25  14:14:52  IST 2025
```

2. &&

```
$ who && date
 student  pts/0    2025-01-25    13:30 (:0)
 student  pts/1    2025-01-25    13.42 (:0)
 Sat Jan 25  14:14:52  IST  2025
```

EXAMPLE:

$ chmod u –wx college

Removes write & execute permission for users for 'college' file.

$ chmod u +rw, g+rw college

Assigns read & write permission for users and groups for 'college' file.

$ chmod g=wx college

Assigns absolute permission for groups of all read, write and execute permissions for 'college' file.

14. The 'Octal Notations' command:

The file permissions can be changed using octal notations also. The octal notations for file permission are

Read permission	4
Write permission	2

EXAMPLE:

$ chmod 761 college

Execute permission	1

Assigns all permission to the owner, read and write permissions to the group and only executable permission to the others for 'college' file.

1.4 GROUPING COMMANDS

1. The 'semicolon' command:

The semicolon(;) command is used to separate multiple commands at the command line.

SYNTAX: $ command1;command2;command3…………….;commandn

EXAMPLE: $ who;date

2. The '&&' operator:

The '&&' operator signifies the logical AND operation in between two or more valid Unix commands.It means that only if the first command is successfully executed, then the next command will executed.

SYNTAX: $ command1 && command && command3…………….&&commandn

EXAMPLE: $ who && date

3. $'||' operator

```
$ who || date
student  pts/0       2025-01-25    13:30 (:0)
student  pts/1       2025-01-25    13:42 (:0)
```

4. $ head

```
$ head sub.c
#include <stdio.h>
int main ()
{
  int x,yz;
  x=5;
  y=3;
  z=x-y;
  printf ("%d", z);
}
```

5. $tail

```
$ tail sub.c
#include <stdio.h>
int main ()
{
  int x,y,z;
  x=5;
  y=3;
  z=x-y;
  printf("%d",z);
}
```

6. $more

```
$ls -l | more
-rw-rw-r--, 1 student student  0 Jan 23 10:33 282
-rw-rw-r--, 2 student student 11 Jan 23 10:37 Rec
```

3. The '||' operator:

The '||' operator signifies the logical OR operation in between two or more valid Unix commands.It means, that only if the first command will happen to be un successfully,it will continue to execute next commands.

SYNTAX: $ command1 || command || command3……………..||commandn

EXAMPLE: $ who || date

1.5 FILTERS

1. The head filter

It displays the first ten lines of a file.

SYNTAX: $ head filename

EXAMPLE: $ head college Display the top ten lines.

$ head -5 college Display the top five lines.

2. The tail filter

It displays ten lines of a file from the end of the file.

SYNTAX: $ tail filename

EXAMPLE: $ tail college Display the last ten lines.

$tail -5 college Display the last five lines.

3. The more filter:

The pg command shows the file page by page.

SYNTAX: $ ls –l | more

4. The 'grep' command:

This command is used to search for a particular pattern from a file or from the standard input and display those lines on the standard output. "Grep" stands for "global search for regular expression."

SYNTAX: $ grep [pattern] [file_name]

EXAMPLE: $ cat> student

Arun cse

Ram ece

Kani cse

$ grep "cse" student

Arun cse

Kani cse

5. The 'sort' command:

The sort command is used to sort the contents of a file. The sort command reports only to the

```
4. cat abn.txt
   123633
   33258
   69733
```

5. $ grep

~~$ grep abn.txt~~

```
$ grep "33" abn.txt
  1236-33
  33258
  69733
```

```
6. $ sort
   $ sort abn.txt
   123633
   33258
   69733
```

```
7. $ sort -n
  $ sort -n abn.txt
  69733
  33258
  69733
```

```
8. nl abn.txt
   1. 123633
   2. 33258
   3. 69733
```

```
9.$ cut -c 3 abn.txt
   3
   2
   7
```

screen, the actual file remains unchanged.

SYNTAX: $ sort filename

EXAMPLE: $ sort college

OPTIONS:

Command	Purpose
Sort –r college	Sorts and displays the file contents in reverse order
Sort –c college	Check if the file is sorted
Sort –n college	Sorts numerically
Sort –m college	Sorts numerically in reverse order
Sort –u college	Remove duplicate records
Sort –l college	Skip the column with +1 (one) option.Sorts according to second column

6. The 'nl' command:

The nl filter adds lines numbers to a file and it displays the file and not provides access to edit but simply displays the contents on the screen.

SYNTAX: $ nl filename

EXAMPLE: $ nl college

7. The 'cut' command:

We can select specified fields from a line of text using cut command.

SYNTAX: $ cut -c filename

EXAMPLE: $ cut -c college

OPTION:

-c – Option cut on the specified character position from each line.

1.5

① $free

```
        total     used     free     shared  buff     available
Mem     1993876   507092   457020   55300   1029764  1840432
Swap    2125820   0        2125820
```

② $top

```
top -15:00:33 up 1.01 min, 2 users, load average; 0.01,0.03
Tasks : 159 total, 2 running, 157 sleeping, 0stopped
%.Cpu(S): 0.8 US, 0.3 Sy, 0.0ni, 98.9id, 0.0wa, 0.0hi
 KiB Mem: 4062328 total, 2702276 free 487024
```

③ PS

```
PID  user    PR  NI   VIRT   RES  SHRS  %.CPU  %.MEM  TIME+
                                                       COMMAN
                                  0 S   0.0    0.0    0.0034 top
10   77root  20  0    0      0
                                  0 S   0.3    0:01.28 systemd
1    root    20  0    32264  0
2    root    0   -20  0      0    0S    0.0    0.0    0:00.0 kworker
```

1.5 OTHER ESSENTIAL COMMANDS

1. free

Display amount of free and used physical and swapped memory system.

synopsis- free [options]

<u>example</u>

```
[root@localhost ~]# free -t

total used free shared buff/cache available  Mem: 4044380 605464 2045080

148820 1393836 3226708  Swap: 2621436 0 2621436

Total: 6665816 605464 4666516
```

2. top

It provides a dynamic real-time view of processes in the system.

synopsis- top [options]

<u>example</u>

```
[root@localhost ~]# top

top - 08:07:28 up 24 min, 2 users, load average: 0.01, 0.06, 0.23

Tasks: 211 total, 1 running, 210 sleeping, 0 stopped, 0 zombie

%Cpu(s): 0.8 us, 0.3 sy, 0.0 ni, 98.9 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

KiB Mem : 4044380 total, 2052960 free, 600452 used, 1390968 buff/cache  KiB Swap:

2621436 total, 2621436 free, 0 used. 3234820 avail Mem   PID USER PR NI VIRT RES

SHR S %CPU %MEM TIME+ COMMAND

        1105 root 20 0 175008 75700 51264 S 1.7 1.9 0:20.46 Xorg  2529 root 20 0 80444

32640 24796 S 1.0 0.8 0:02.47 gnome-term
```

3. ps

It reports the snapshot of current processes

synopsis- ps [options]

<u>example</u>

```
[root@localhost ~]# ps -e
```

5. PS -e

```
PID    TTY    TIME CMD
1      ?      00:00:01   systemd
2      ?      00:00:00   kthreadd
3      ?      00:00:00   kworkal/0:00
```

4. vmstat

```
procs            memory            swap        io   system  CPO
r b      swpd    free         buff cache       SI   so  bi  incsos
0 0      0       2702788      65940807036      0    0   103  202 264
```

5. df

```
File system     1t-blocks   used   Available   use
dev tmpFs       2020176     0      2020176     0%    /dev
tmpFs           2031164     0      2031164     0%    /devS
```

6. pin 172.16.41

```
PING 172.16.41 (172.16.4.1) 56(8H) bytes of data
64 bytes from 172.16.4.1 : icmp-seq=1t+l=64tin=0.0
64 bytes from 172.16.41 : icmp-seq=2++ly=64 tin=0.0
64 bytes
```

7. ifconfig

```
enp350: flags=4163<up, BROAD CAST, RUNNING,
  MUTICAST > mtu, 500
  inet 172.16.9.6  net mask 255.255.252.0
  brodcast 172.16.11.255
```

```
PID TTY TIME CMD
1 ? 00:00:03 systemd
2 ? 00:00:00 kthreadd
3 ? 00:00:00 ksoftirqd/0
```

4. vmstat

It reports virtual memory statistics

synopsis- vmstat [options]

example

```
[root@localhost ~]# vmstat
procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu---
-- r b swpd free buff cache si so bi bo in cs us sy id wa st  0 0 0 1879368
1604 1487116 0 0 64 7 72 140 1 0 97 1 0
```

5. df

It displays the amount of disk space available in file-system.

Synopsis- df [options]

example

```
[root@localhost ~]# df
Filesystem 1K-blocks Used Available Use% Mounted on
devtmpfs 2010800 0 2010800 0% /dev  tmpfs 2022188 148 2022040 1% /dev/shm
tmpfs 2022188 1404 2020784 1% /run  /dev/sda6 487652 168276 289680 37% /boot
```

6. ping

It is used verify that a device can communicate with another on network. PING stands for Packet Internet Groper.

synopsis- ping [options]

```
[root@localhost ~]# ping 172.16.4.1
PING 172.16.4.1 (172.16.4.1) 56(84) bytes of data.
64 bytes from 172.16.4.1: icmp_seq=1 ttl=64 time=0.328 ms
64 bytes from 172.16.4.1: icmp_seq=2 ttl=64 time=0.228 ms
```

8. traceroute wwin.rajalakshmi.org
traceroute to www.rajalakshmi.org (14.99.10.232)
30 hops max, 60 byte packets

1. rajalakshmi.org (14.99.10.232) 31.620 ms **

```
64 bytes from 172.16.4.1: icmp_seq=3 ttl=64 time=0.264 ms
64 bytes from 172.16.4.1: icmp_seq=4 ttl=64 time=0.312 ms
^C
--- 172.16.4.1 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3000ms
rtt min/avg/max/mdev = 0.228/0.283/0.328/0.039 ms
```

7. ifconfig

It is used configure network interface.

synopsis- ifconfig [options]

example

```
[root@localhost ~]# ifconfig

enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu
1500  inet 172.16.6.102 netmask 255.255.252.0 broadcast 172.16.7.255  inet6
fe80::4a0f:cfff:fe6d:6057 prefixlen 64 scopeid 0x20<link>
ether 48:0f:cf:6d:60:57 txqueuelen 1000 (Ethernet)


 RX packets 23216 bytes 2483338 (2.3 MiB)
RX errors 0 dropped 5 overruns 0 frame 0
TX packets 1077 bytes 107740 (105.2 KiB)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0 8.
```

traceroute

It tracks the route the packet takes to reach the destination.

synopsis- traceroute [options]

example

```
[root@localhost ~]# traceroute www.rajalakshmi.org
traceroute to www.rajalakshmi.org (220.227.30.51), 30 hops max, 60 byte
packets  1 gateway (172.16.4.1) 0.299 ms 0.297 ms 0.327 ms
2 220.225.219.38 (220.225.219.38) 6.185 ms 6.203 ms 6.189 ms
```

Result:

Thus the general purpose commands, the directory commands, file handling commands, grouping commands and other essential commands have been successfully observed and executed.