

EX.NO:1B

DATE:

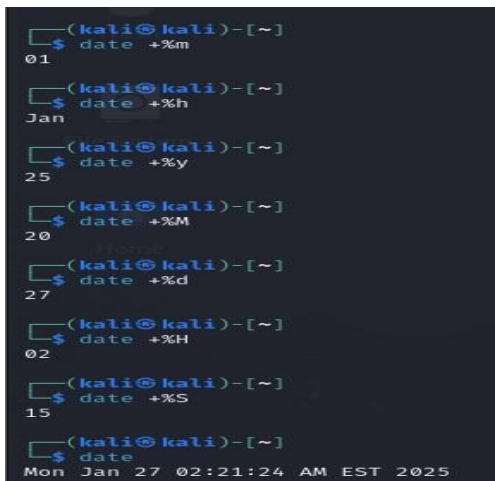
BASIC LINUX COMMANDS

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



```

(kali㉿kali)-[~]
└─$ date +%m
01

(kali㉿kali)-[~]
└─$ date +%h
Jan

(kali㉿kali)-[~]
└─$ date +%y
25

(kali㉿kali)-[~]
└─$ date +%M
20

(kali㉿kali)-[~]
└─$ date +%d
27

(kali㉿kali)-[~]
└─$ date +%H
02

(kali㉿kali)-[~]
└─$ date +%S
15

(kali㉿kali)-[~]
└─$ date
Mon Jan 27 02:21:24 AM EST 2025
  
```

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"

```
(kali@kali)-[~]  
$ echo "God id great"  
God id great
```

3. The 'cal' command: The cal command displays the specified month or year of the calendar

SYNTAX: \$ cal [month]

[year]

EXAMPLE:

\$ cal Jan 2012

```
(kali@kali)-[~]  
$ cal  
January 2025  
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
```

```
(kali@kali)-[~]  
$ cal june 2024  
June 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
```

4. The 'bc' command: Unix offers an online calculator and can be invoked by the command bc.

SYNTAX: \$ bc

EXAMPLE: bc -l

16/4

5/2

```
bc 1.07.1
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Softw
are Foundation, Inc.
This is free software with ABSOLUTELY NO WARRANTY.
For details type `warranty'.
16/4
4
5/2
2
2*3
6
4+7
11
6-3
3
```

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

```
(kali@kali)-[~]
$ who
kali      tty7      2025-01-27 02:10 (:0)
```

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

SYNTAX: \$ who am i

```
(kali@kali)-[~]
$ whoami
kali
```

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

SYNTAX: \$ id

```
(kali@kali)-[~]
$ id
uid=1000(kali) gid=1000(kali) groups=1000(kali),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugin),100(users),101(netdev),106(blueetooth),113(scanner),136(wireshark),137(kaboxer)
```

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

SYNTAX: \$ tty

```
(kali@kali)-[~]
$ tty
/dev/pts/0
```

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]

```
File Actions Edit View Help
who(1)
NAME
  who - show who is logged on

SYNOPSIS
  who [OPTION]... [ FILE | ARG1 ARG2 ]

DESCRIPTION
  Print information about users who are currently logged in.

  -a, --all
        same as -b -d --login -p -r -t -T -u

  -b, --boot
        time of last system boot

  -d, --dead
        print dead processes

  -H, --heading
        print line of column headings

  -l, --login
        print system login processes

  --lookup
        attempt to canonicalize hostnames via DNS

  -m
        only hostname and user associated with stdin

  -p, --process
        print active processes spawned by init

  -q, --count
        all login names and number of users logged on

  -r, --runlevel
        print current runlevel

  -s, --short
        print only name, line, and time (default)

  -t, --time
        print last system clock change

  -T, -w, --msg
        log file:

```

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

```
(kali@kali)-[~]
$ ps
  PID TTY          TIME CMD
 2746 pts/0        00:00:06 zsh
14712 pts/0        00:00:00 ps

(kali@kali)-[~]

```

\$ ps -e

```

kali@kali:~$ ps -e
  PID TTY          TIME CMD
    1 ?        00:00:00 system
    2 ?        00:00:00 kthreadd
    3 ?        00:00:00 pool_workqueue_release
    4 ?        00:00:00 [worker/R-rcu_
    5 ?        00:00:00 [worker/R-rcu_
    6 ?        00:00:00 [worker/R-slab
    7 ?        00:00:00 [worker/R-netns
   12 ?        00:00:00 [worker/R-netns
   13 ?        00:00:00 rcu_tasks_kthread
   14 ?        00:00:00 rcu_tasks_cdb_kthread
   15 ?        00:00:00 rcu_tasks_trace_kthread
   16 ?        00:00:00 ksoftirqd/0
   17 ?        00:00:00 rcu_govert
   18 ?        00:00:00 migration/0
   19 ?        00:00:00 [kfsd/0]
   20 ?        00:00:00 [kfsd/1]
   21 ?        00:00:00 [kfsd/2]
   22 ?        00:00:00 [kfsd/3]
   23 ?        00:00:00 [kfsd/4]
   24 ?        00:00:00 [kfsd/5]
   25 ?        00:00:00 [kfsd/6]
   26 ?        00:00:00 [kfsd/7]
   27 ?        00:00:00 [kfsd/8]
   28 ?        00:00:00 [kfsd/9]
   29 ?        00:00:00 [kfsd/10]
   30 ?        00:00:00 [kfsd/11]
   31 ?        00:00:00 [kfsd/12]
   32 ?        00:00:00 [kfsd/13]
   33 ?        00:00:00 [kfsd/14]
   34 ?        00:00:00 [kfsd/15]
   35 ?        00:00:00 [kfsd/16]
   36 ?        00:00:00 [kfsd/17]
   37 ?        00:00:00 [kfsd/18]
   38 ?        00:00:00 [kfsd/19]
   39 ?        00:00:00 [kfsd/20]
   40 ?        00:00:00 [kfsd/21]
   41 ?        00:00:00 [kfsd/22]
   42 ?        00:00:00 [kfsd/23]
   43 ?        00:00:00 [kfsd/24]
   44 ?        00:00:00 [kfsd/25]
   45 ?        00:00:00 [kfsd/26]
   46 ?        00:00:00 [kfsd/27]
   47 ?        00:00:00 [kfsd/28]
   48 ?        00:00:00 [kfsd/29]
   49 ?        00:00:00 [kfsd/30]
   50 ?        00:00:00 [kfsd/31]
   51 ?        00:00:00 [kfsd/32]
   52 ?        00:00:00 [kfsd/33]
   53 ?        00:00:00 [kfsd/34]
   54 ?        00:00:00 [kfsd/35]
   55 ?        00:00:00 [kfsd/36]
   56 ?        00:00:00 [kfsd/37]
   57 ?        00:00:00 [kfsd/38]
   58 ?        00:00:00 [kfsd/39]

```

\$ ps -aux

```

kali@kali:~$ ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1   0.0  0.0  22228  9224 ?        Ss   02:00  0:02 /sbin/init splash
root         2   0.0  0.0   0      0 ?        S    02:00  0:00 [kthreadd]
root         3   0.0  0.0   0      0 ?        S    02:00  0:00 [pool_workqueue_release]
root         4   0.0  0.0   0      0 ?        S    02:00  0:00 [worker/R-rcu_
root         5   0.0  0.0   0      0 ?        S    02:00  0:00 [worker/R-rcu_
root         6   0.0  0.0   0      0 ?        S    02:00  0:00 [worker/R-slab
root         7   0.0  0.0   0      0 ?        S    02:00  0:00 [worker/R-netns
root        12   0.0  0.0   0      0 ?        S    02:00  0:00 [worker/R-netns
root        13   0.0  0.0   0      0 ?        S    02:00  0:00 rcu_tasks_kthread
root        14   0.0  0.0   0      0 ?        S    02:00  0:00 rcu_tasks_cdb_kthread
root        15   0.0  0.0   0      0 ?        S    02:00  0:00 rcu_tasks_trace_kthread
root        16   0.0  0.0   0      0 ?        S    02:00  0:00 ksoftirqd/0
root        17   0.2  0.0   0      0 ?        S    02:00  0:02 [rcu_govert]
root        18   0.0  0.0   0      0 ?        S    02:00  0:00 migration/0
root        19   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/0]
root        20   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/1]
root        21   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/2]
root        22   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/3]
root        23   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/4]
root        24   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/5]
root        25   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/6]
root        26   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/7]
root        27   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/8]
root        28   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/9]
root        29   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/10]
root        30   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/11]
root        31   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/12]
root        32   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/13]
root        33   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/14]
root        34   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/15]
root        35   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/16]
root        36   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/17]
root        37   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/18]
root        38   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/19]
root        39   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/20]
root        40   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/21]
root        41   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/22]
root        42   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/23]
root        43   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/24]
root        44   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/25]
root        45   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/26]
root        46   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/27]
root        47   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/28]
root        48   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/29]
root        49   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/30]
root        50   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/31]
root        51   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/32]
root        52   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/33]
root        53   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/34]
root        54   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/35]
root        55   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/36]
root        56   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/37]
root        57   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/38]
root        58   0.0  0.0   0      0 ?        S    02:00  0:00 [kfsd/39]

```

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

```
(kali@kali)-[~]
└─$ uname
Linux
(kali@kali)-[~]
└─$ uname -m
x86_64
(kali@kali)-[~]
└─$ uname -r
6.8.11-amd64
(kali@kali)-[~]
└─$ uname -n
kali
(kali@kali)-[~]
└─$ uname -s
Linux
(kali@kali)-[~]
└─$ uname -v
#1 SMP PREEMPT_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30)
(kali@kali)-[~]
└─$ uname -a
Linux kali 6.8.11-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.8.11-1kali2 (2024-05-30) x86_64 GNU/Linux
```

1.2 DIRECTORY COMMANDS

1. The 'pwd' command:

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

SYNTAX: \$ pwd

```
(kali@kali)-[~]
└─$ pwd
/home/kali
```

2. The 'mkdir' command: The mkdir is used to create an empty directory in a disk.

SYNTAX: \$ mkdir dirname

EXAMPLE: \$ mkdir receee

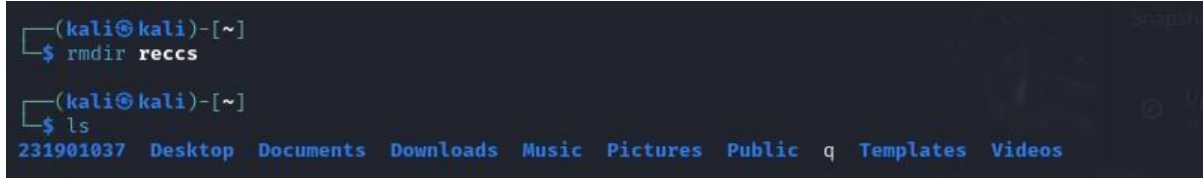
```
(kali@kali)-[~]
└─$ mkdir 231901037
(kali@kali)-[~]
└─$ mkdir reccs
(kali@kali)-[~]
└─$ ls
231901037 Desktop Documents Downloads Music Pictures Public q reccs Templates Videos
```

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 reeeee



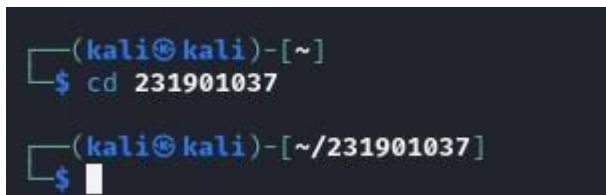
```
(kali@kali)-[~]
$ rmdir reecs

(kali@kali)-[~]
$ ls
231901037 Desktop Documents Downloads Music Pictures Public q Templates Videos
```

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

SYNTAX: \$ cd dirname

EXAMPLE: \$ cd reeeee



```
(kali@kali)-[~]
$ cd 231901037

(kali@kali)-[~/231901037]
$
```

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

SYNTAX: \$ ls

EXAMPLE: \$ ls

\$ ls -l

\$ ls -a



```
(kali@kali)-[~]
$ ls
231901037 Desktop Documents Downloads Music Pictures Public q reecs Templates Videos

(kali@kali)-[~]
$ ls -a
. 231901037 .bashrc .cache Desktop Documents .face .gnupg .java .mozilla Pictures Public reecs Templates .Xauthority .zshrc
.. .bash_logout .bashrc.original .config .dirc Downloads .face.icon .ICEauthority .local Music .profile q .sudo_as_admin_successful Videos .xsession-errors
```

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

```
(kali㉿kali)-[~/231901037]
$ cat > rec
hihi
hello
^C

(kali㉿kali)-[~/231901037]
$ cat rec
hihi
hello
```

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

```
(kali㉿kali)-[~/231901037]
$ cp rec recce

(kali㉿kali)-[~/231901037]
$ cat recce
hihi
hello
```

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.

```
(kali㉿kali)-[~/231901037]
$ rm rec

(kali㉿kali)-[~/231901037]
$ ls
recce
```

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

```
(kali㉿kali)-[~]
$ mv cse rec

(kali㉿kali)-[~]
$ ls
231901037 Desktop Documents Downloads Music Pictures Public q rec Templates Videos

(kali㉿kali)-[~]
$ cat rec
hihi
hello
```

6. The 'file' command: The file command is used to determine the type of file.

SYNTAX: \$ file filename

```
(kali㉿kali)-[~]
$ file rec
rec: ASCII text
```


7. “wc command” : The wc command is used to count the number of words, lines and characters in a file.

SYNTAX: \$ wc filename

EXAMPLE: \$ wc receee

```
(kali@kali)-[~]  
$ wc rec  
2  2 12 rec
```

8. The ‘Directing output to a file’ command:

The ls command lists the files on the terminal (screen). Using the redirection operator ‘>’ we can send the output to file instead of showing it on the screen.

SYNTAX: \$ ls > filename

EXAMPLE: \$ ls > cseeee

```
(kali@kali)-[~]  
$ ls > list  
  
(kali@kali)-[~]  
$ cat list  
231901037  
Desktop  
Documents  
Downloads  
list  
Music  
Pictures  
Public  
q  
rec  
Templates  
Videos
```

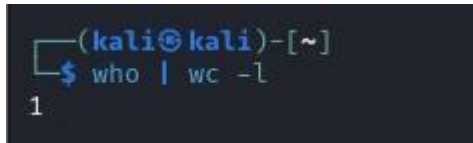
9. The ‘pipes’ command:

The Unix allows us to connect two commands together using these pipes. A pipe (|) is a mechanism by which the output of one command can be channeled into the input of another command. SYNTAX: \$ command1 | command2 EXAMPLE: \$ who | wc -l

10. The ‘tee’ command:

While using pipes, we have not seen any output from a command that gets piped into another command. To save the output, which is produced in the middle of a pipe, the tee command is very useful. SYNTAX: \$ command | tee filename

EXAMPLE: \$ who | tee sample | wc -l



11. The 'Metacharacters of unix' command:

Metacharacters are special characters that are at higher and abstract level compared to most of other characters in Unix. The shell understands and interprets these metacharacters in a special way.

* - Specifies number of characters

?- Specifies a single character

[]- used to match a whole set of file names at a command line.

! – Used to Specify Not EXAMPLE:

\$ ls r** - Displays all the files whose name begins with 'r'

\$ ls ?kkk - Displays the files which are having 'kkk', from the second characters irrespective of the first character.

\$ ls [a-m] – Lists the files whose names begins alphabets from 'a' to 'm'

\$ ls [!a-m] – Lists all files other than files whose names begins alphabets from 'a' to 'm'

12. The 'File permissions' command:

File permission is the way of controlling the accessibility of file for each of three users namely Users, Groups and Others.

There are three types of file permissions are available, they are

r-read

w-

write

x-

execute

The permissions for each file can be divided into three parts of three bits each.

First three bits	Owner of the file
Next three bits	Group to which owner of the file belongs
Last three bits	Others

EXAMPLE: \$ ls college

```
-rwxr-xr-- 1 Lak std 1525 jan10 12:10 college
```

Where,

-rwx The file is readable, writable and executable by the owner of the file.

Lak Specifies Owner of the file. r-x Indicates the absence of the write permission by the Group owner of the file. Std Is the Group Owner of the file. r-- Indicates read permissions for others.

```
(kali㉿kali)-[~]
$ ls -l rec
-rw-rw-r-- 1 kali kali 12 Jan 27 02:42 rec
```

13. The 'chmod' command:

The chmod command is used to set the read, write and execute permissions for all categories of users for file.

SYNTAX: \$ chmod category operation permission file

Category	Operation	permission
u-users	+ assign	r-read
g-group	-Remove	w-write
o-others	= assign absolutely	x-execute
a-all		

EXAMPLE:

```
$ chmod u -wx college
```

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

```
(kali㉿kali)-[~]
$ chmod u-wx rec

(kali㉿kali)-[~]
$ ls -l rec
-r--r--r-- 1 kali kali 12 Jan 27 02:42 rec
```

```
$ chmod u +rw, g+rw college
```

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

```
(kali㉿kali)-[~]
$ chmod u+rw,g+rw rec

(kali㉿kali)-[~]
$ ls -l rec
-rwxrw-r-- 1 kali kali 12 Jan 27 02:42 rec
```

\$ chmod g=wx college

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

```
(kali㉿kali)-[~]
$ chmod g=wx rec

(kali㉿kali)-[~]
$ ls -l rec
-rwx-wxr-- 1 kali kali 12 Jan 27 02:42 rec
```

14. The 'Octal Notations' command:

The file permissions can be changed using octal notations also. The octal notations for file permission are
\$ chmod 761 college

Read permission	4
Write permission	2

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.

```
(kali㉿kali)-[~]
$ chmod 761 rec

(kali㉿kali)-[~]
$ ls -l rec
-rwxrw--x 1 kali kali 12 Jan 27 02:42 rec
```

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

```
(kali㉿kali)-[~]
$ who;date
kali      tty7          2025-01-27 02:10 (:0)
Mon Jan 27 02:51:30 AM EST 2025
```

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 be executed.

SYNTAX: \$ command1 && command2 && command3 &&commandn

EXAMPLE: \$ who && date

```
(kali㉿kali)-[~]
$ who&&date
kali      tty7          2025-01-27 02:10 (:0)
Mon Jan 27 02:51:48 AM EST 2025
```

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 unsuccessful, it will continue to execute next commands.

SYNTAX: \$ command1 || command2 ||

command3... ||commandn

EXAMPLE: \$ who || date

```
(kali㉿kali)-[~]
$ who || date
kali      tty7          2025-01-27 02:10 (:0)
```

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.

```
(kali㉿kali)-[~]
$ head -3 college
lerumipsum
lorumsipsum
loremipsum
```

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.

```
(kali㉿kali)-[~]
$ tail -3 college
loremipsum
loremipsum
loremipsum
```

3. The more filter:

The pg command shows the file page by page.

SYNTAX: \$ ls -l | more

```
(kali㉿kali)-[~]
$ ls -l | more
total 56
drwxrwxr-x 2 kali kali 4096 Jan 27 02:38 231901037
-rw-rw-r-- 1 kali kali 8 Jan 27 02:46 akkw
-rw-rw-r-- 1 kali kali 155 Jan 27 02:53 college
-rw-rw-r-- 1 kali kali 0 Jan 27 02:46 csecs
drwxr-xr-x 2 kali kali 4096 Jan 27 02:10 Desktop
drwxr-xr-x 2 kali kali 4096 Jan 27 02:10 Documents
drwxr-xr-x 2 kali kali 4096 Jan 27 02:14 Downloads
-rw-rw-r-- 1 kali kali 88 Jan 27 02:44 list
drwxr-xr-x 2 kali kali 4096 Jan 27 02:10 Music
drwxr-xr-x 2 kali kali 4096 Jan 27 02:22 Pictures
drwxr-xr-x 2 kali kali 4096 Jan 27 02:10 Public
-rw-r--r-- 1 kali kali 3052 Jan 27 02:31 q
-rwxrw-x 1 kali kali 12 Jan 27 02:42 rec
drwxr-xr-x 2 kali kali 4096 Jan 27 02:10 Templates
drwxr-xr-x 2 kali kali 4096 Jan 27 02:10 Videos
```

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

```
(kali@kali)-[~]
$ grep i rec
hiii
```

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

```
(kali@kali)-[~]
$ sort rec
helloo
hiii

(kali@kali)-[~]
$ sort -r rec
hiii
helloo

(kali@kali)-[~]
$ sort -n rec
helloo
hiii

(kali@kali)-[~]
$ sort -m rec
hiii
helloo

(kali@kali)-[~]
$ sort -u rec
helloo
hiii
```

6. The 'nl' command:

The nl filter adds line 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

```
(kali@kali)-[~]
$ nl rec
 1 hiii
 2 helloo
```

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.

```
(kali@kali)-[~]
$ cut -c 1 rec
h
h
```

1.5 OTHER ESSENTIAL COMMANDS

1. free

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

synopsis- free

[options]

example

[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

```
(kali@kali)-[~]
$ free -t
```

	total	used	free	shared	buff/cache	available
Mem:	2012808	1485464	174980	53892	557528	527344
Swap:	1048572	589452	459120			
Total:	3061380	2074916	634100			

2. top

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

system. synopsis- top [options]

example

```
[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

```
(kali@kali)-[~]
$ top
top - 03:00:06 up 50 min, 2 users, load average: 0.65, 0.39, 0.35
Tasks: 224 total, 2 running, 222 sleeping, 0 stopped, 0 zombie
%Cpu(s): 0.8 us, 1.9 sy, 0.0 ni, 95.8 id, 0.8 wa, 0.0 hi, 0.7 si, 0.0 st
MiB Mem : 1965.6 total, 270.9 free, 1349.4 used, 551.1 buff/cache
MiB Swap: 1024.0 total, 453.4 free, 570.6 used, 616.2 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM    TIME+    COMMAND
 3273 kali      20   0   11.5g   356368 128932 S   3.6   17.7   5:29.19 firefox-esr
 1113 root       20   0   493592  114776  51468 S   2.3    5.7   2:11.18 Xorg
 3731 kali      20   0   3120380 393100 106516 S   2.3   19.5   1:55.43 Isolated Web Co
 210 root      0 -20      0      0      0 I   0.7    0.0   0:00.63 kworker/1:1H-kblockd
2284 kali      20   0   388036  26660  18904 S   0.7    1.3   0:28.18 xfwm4
2505 kali      20   0   375076  23720  16632 S   0.7    1.2   0:15.80 panel-13-cpugra
 7494 kali      20   0   2855304 229996  99180 S   0.7   11.4   1:28.28 Isolated Web Co
29681 kali      20   0   9340    5120   3072 R   0.7    0.3   0:00.08 top
 534 root      20   0   317412  6656   6144 S   0.3    0.3   0:07.33 vmtoolsd
2507 kali      20   0   337688  16864  12208 S   0.3    0.8   0:11.20 panel-15-genmon
 6051 root      0 -20      0      0      0 I   0.3    0.0   0:01.17 kworker/2:2H-kblockd
16659 root      0 -20      0      0      0 I   0.3    0.0   0:00.23 kworker/0:2H-kblockd
20652 kali      20   0   458124 100856  87296 S   0.3    5.0   0:05.74 qterminal
25849 root      20   0      0      0      0 I   0.3    0.0   0:00.04 kworker/u65:0-writeback
25930 root      0 -20      0      0      0 I   0.3    0.0   0:00.10 kworker/3:2H-kblockd
   1 root      20   0   22216   9124   7020 S   0.0    0.5   0:02.48 systemd
   2 root      20   0      0      0      0 S   0.0    0.0   0:00.06 kthreadd
   3 root      20   0      0      0      0 S   0.0    0.0   0:00.00 pool_workqueue_release
   4 root      0 -20      0      0      0 I   0.0    0.0   0:00.00 kworker/R-rcu_g
   5 root      0 -20      0      0      0 I   0.0    0.0   0:00.00 kworker/R-rcu_p
   6 root      0 -20      0      0      0 I   0.0    0.0   0:00.00 kworker/R-slub_
   7 root      0 -20      0      0      0 I   0.0    0.0   0:00.00 kworker/R-netns
  12 root      0 -20      0      0      0 I   0.0    0.0   0:00.00 kworker/R-mm_pg
  13 root      20   0      0      0      0 I   0.0    0.0   0:00.00 rcu_tasks_kthread
  14 root      20   0      0      0      0 I   0.0    0.0   0:00.00 rcu_tasks_rude_kthread
  15 root      20   0      0      0      0 I   0.0    0.0   0:00.00 rcu_tasks_trace_kthread
  16 root      20   0      0      0      0 S   0.0    0.0   0:00.21 ksoftirqd/0
  17 root      20   0      0      0      0 I   0.0    0.0   0:04.32 rcu_preempt
  18 root      rt   0      0      0      0 S   0.0    0.0   0:00.09 migration/0
  19 root     -51   0      0      0      0 S   0.0    0.0   0:00.00 idle_inject/0
  20 root      20   0      0      0      0 S   0.0    0.0   0:00.00 cpuhp/0
  21 root      20   0      0      0      0 S   0.0    0.0   0:00.00 cpuhp/1
  22 root     -51   0      0      0      0 S   0.0    0.0   0:00.00 idle_inject/1
  23 root      rt   0      0      0      0 S   0.0    0.0   0:00.17 migration/1
  24 root      20   0      0      0      0 S   0.0    0.0   0:00.16 ksoftirqd/1
```

3. ps

It reports the snapshot of current processes synopsis- ps [options]

example : [root@localhost ~]# ps -e

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

```
(kali㉿kali)-[~]
$ vmstat
procs  -----memory-----  --swap--  ----io-----  -system--  -----cpu-----
r  b   swpd   free   buff  cache   si   so    bi    bo    in   cs  us  sy  id  wa  st  gu
1  0  541068 290796 23700 533252   64  234   490   540  869   3   4   5  92   0   0   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

```

kali@kali:~$ df
Filesystem      1k-blocks    Used Available Use% Mounted on
udev            963592      0   963592  0% /dev
tmpfs           201284     1340   199944  1% /run
/dev/sda1      82083148 1508894 62778656 28% /
tmpfs           1086484      0   1086484  0% /dev/shm
tmpfs           5120        0    5120    0% /run/lock
tmpfs           1024        0    1024    0% /run/credentials/systemd-journald.service
tmpfs           1024        0    1024    0% /run/credentials/systemd-udev-load-credentials.service
tmpfs           1024        0    1024    0% /run/credentials/systemd-timesetup-dev-early.service
tmpfs           1024        0    1024    0% /run/credentials/systemd-timesetup.service
tmpfs           1086484     592  1085812  1% /tmp
tmpfs           1024        0    1024    0% /run/credentials/systemd-sysusers.service
tmpfs           1024        0    1024    0% /run/credentials/systemd-timesetup-dev.service
tmpfs           1024        0    1024    0% /run/credentials/systemd-timesetup.service
tmpfs           1024        0    1024    0% /run/credentials/getty@tty1.service
tmpfs          201280     136   201144  1% /run/user/1000

```

It is used to verify that a device can communicate with another on the network.
PING stands

[illegible]

CSE CYBER SECURITY

```
[root@localhost ~]# ifconfig
```

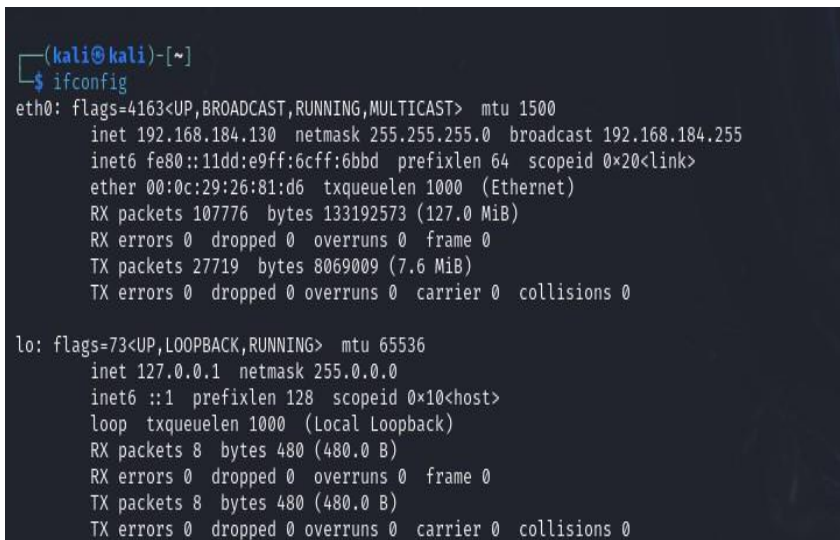
```
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu
1500 inet 172.16.6.102 netmask 255.255.252.0 broadcast
172.16.7.255 inet6 fe80::4a0f:cfff:fe6d:6057 prefixlen
64 scopeid 0x20<link> ether
48:0f:cf:6d:60:57 txqueuelen 1000 (Ethernet)
```

```
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
```



```
(kali@kali)-[~]
$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.184.130 netmask 255.255.255.0 broadcast 192.168.184.255
    inet6 fe80::11dd:e9ff:6cff:6bbd prefixlen 64 scopeid 0x20<link>
    ether 00:0c:29:26:81:d6 txqueuelen 1000 (Ethernet)
    RX packets 107776 bytes 133192573 (127.0 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 27719 bytes 8069009 (7.6 MiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 8 bytes 480 (480.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 8 bytes 480 (480.0 B)
    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
```

```
(kali@kali)-[~]
$ traceroute www.rajalakshmi.org
traceroute to www.rajalakshmi.org (14.99.10.232), 30 hops max, 60 byte packets
 1  192.168.184.2 (192.168.184.2)  1.489 ms  1.062 ms  0.870 ms
 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  * * *
```

RESULT:

Hence, basic linux commands are executed successfully