

Ex No: 1 b

Date:

BASIC LINUX COMMANDS

1.1 GENERAL PURPOSE COMMANDS

1. The 'date' command:

The date command display 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

OUTPUT

```
(shanthosh@kali)-[~]
$ date +%m
06
(shanthosh@kali)-[~]
$ date +%h
Jun
(shanthosh@kali)-[~]
$ date +%d
01
(shanthosh@kali)-[~]
$ date +%y
24
(shanthosh@kali)-[~]
$ date +%H
07
(shanthosh@kali)-[~]
$ date +%M
09
(shanthosh@kali)-[~]
$ date +%S
51
```

2. The echo'command:

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

SYNTAX: \$ echo

EXAMPLE: \$ echo "God is Great"

OUTPUT

```
(shanthosh@kali)-[~]
$ echo "hello"
hello
```

3. The 'cal' command:

The cal command displays the specified month or year calendar.

SYNTAX: \$ cal [month] [year] **EXAMPLE:**

\$ cal Jan 2012

OUTPUT

```
(shanthosh@kali)-[~]
$ cal Jan 2022
January 2022
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
```

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

OUTPUT

```
(shanthosh@kali)-[~]  
$ bc  
bc 1.07.1  
Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006, 2008, 2012-2017 Free Software Foundation, Inc.  
This is free software with ABSOLUTELY NO WARRANTY.  
For details type `warranty'.  
1+2  
3  
3*5  
15  
|
```

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

OUTPUT

```
(shanthosh@kali)-[~]  
$ who  
shanthosh tty7      2024-06-01 06:47 (:0)
```

6. The 'who am i' command

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

SYNTAX: \$ who am i

OUTPUT

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

7. The 'id' command

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

SYNTAX: \$ id

OUTPUT

```
(shanthosh@kali)-[~]  
$ id  
uid=1000(shanthosh) gid=1000(shanthosh) groups=1000(shanthosh),4(adm),20(dialout),24(cdrom),25(floppy),27(sudo),29(audio),30(dip),44(video),46(plugdev),100(users),106(netdev),118(wireshark),121(bluetooth),134(scanner),141(vboxsf),142(k  
abover)
```

8. The 'tty' command

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

SYNTAX: \$ tty

OUTPUT

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

9. The 'clear' command

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

SYNTAX: \$ clear

OUTPUT

**** Clears entire screen or terminal****

10. The 'man' command

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

SYNTAX: \$ man [command]

OUTPUT

```
TTY(1) User Commands
NAME
    tty - print the file name of the terminal connected to standard input

SYNOPSIS
    tty [OPTION] ...

DESCRIPTION
    Print the file name of the terminal connected to standard input.

    -s, --silent, --quiet
        print nothing, only return an exit status

    --help
        display this help and exit

    --version
        output version information and exit

AUTHOR
    Written by David MacKenzie.

REPORTING BUGS
    GNU coreutils online help: <https://www.gnu.org/software/coreutils/tty>
    Report any translation bugs to <https://translationproject.org/team/>

COPYRIGHT
    Copyright © 2023 Free Software Foundation, Inc. License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.
    This is free software: you are free to change and redistribute it. There is NO WARRANTY, to the extent permitted by law.

SEE ALSO
    Full documentation <https://www.gnu.org/software/coreutils/tty>
    or available locally via: info '(coreutils) tty invocation'

GNU coreutils 9.4
Manual page tty(1) line 1/35 (END) (press h for help or q to quit)
January 2024
```

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

OUTPUT

```
(shanthosh@kali)-[~]
$ ps
  PID TTY          TIME CMD
 13554 pts/0    00:00:01 zsh
 17957 pts/0    00:00:00 ps
```

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

OUTPUT

```
(shanthosh@kali)~$ uname
Linux

(shanthosh@kali)~$ uname -m
x86_64

(shanthosh@kali)~$ uname -r
6.6.15-amd64

(shanthosh@kali)~$ uname -n
kali

(shanthosh@kali)~$ uname -s
Linux

(shanthosh@kali)~$ uname -a
Linux kali 6.6.15-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.6.15-2kali1 (2024-04-09) 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

OUTPUT

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

2. The 'mkdir' command:

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

SYNTAX: \$ mkdir dirname

EXAMPLE: \$ mkdir receee

OUTPUT

```
(shanthosh@kali)-[~]  
$ mkdir test  
mkdir: cannot create directory 'test': File exists
```

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

OUTPUT

```
(shanthosh@kali)-[~]  
$ rmdir test
```

The

4. 'cd' command:

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

SYNTAX: \$ cd dirname

EXAMPLE: \$ cd receee

OUTPUT

```
(shanthosh@kali)-[~]  
$ cd ..  
  
(shanthosh@kali)-[/home]  
$ cd ~
```

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

OUTPUT

```
(shanthosh@kali)-[~]  
$ ls test
```

1.3 FILE HANDLING COMMANDS

1. The 'cat' command:

The cat command is used to create a file.

SYNTAX: \$ cat > filename

EXAMPLE: \$ cat > rec

OUTPUT

```
(shanthosh@kali)-[~]  
$ cat > test1.c  
^C
```

2. 'Display contents of a file' command:

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

The

SYNTAX: \$ cat filename

OUTPUT

Show Contents of File

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

OUTPUT

```
(shanthosh@kali)-[~]  
$ cp receiver.c test1.c
```

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.

OUTPUT

```
(shanthosh@kali)-[~]  
$ rm test.c
```

File Removed from ~ directory

5. '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

OUTPUT

```
(shanthosh@kali)-[~]  
$ mv test1.c test.c
```

The

6. The 'file' command:

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

SYNTAX: \$ file filename

EXAMPLE: \$ file receee

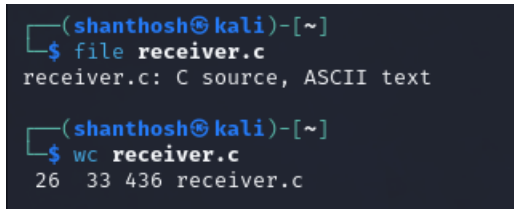
7. The 'wc' command:

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

SYNTAX: \$ wc filename

EXAMPLE: \$ wc receee

OUTPUT



```
(shanthosh@kali)-[~]  
$ file receiver.c  
receiver.c: C source, ASCII text  
  
(shanthosh@kali)-[~]  
$ wc receiver.c  
26 33 436 receiver.c
```

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

OUTPUT

9. 'pipes' command:

The Unix allows us to connect two commands together using these pipes. A pipe (|) is an 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:

The

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

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

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'

11. 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.

12. 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.

\$ 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.

13. 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
Execute permission	1

EXAMPLE:

\$ chmod 761 college

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. 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 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 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

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 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

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

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	

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

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

Hence the basic linux commands has been successfully executed in the terminal and an output has been received.