



RAJALAKSHMI
ENGINEERING COLLEGE
An AUTONOMOUS Institution
Affiliated to ANNA UNIVERSITY, Chennai

DEPARTMENT OF ARTIFICIAL INTELLIGENCE AND DATA

SCIENCE LAB MANUAL

CS23431 – OPERATING SYSTEMS

(REGULATION 2023)

RAJALAKSHMI ENGINEERING COLLEGE
Thandalam, Chennai-602015

Name: Ramya A

Register No: 231801135

Year / Branch / Section: 2nd / AI&DS / FB

Semester: IV

Academic Year: 2024 - 2025

INDEX

Exp.No	Title	Page No
1a	Installation and Configuration of Linux.	4
1b	Basic Linux Commands.	9
2a	Shell script - Arithmetic Operation -using expr command Basic Calculator.	27
2b	Shell script - Check leap year using if-else.	31
3a	Shell script - Reverse the number using while loop.	32
3b	Shell script - Fibonacci series using for loop.	33
4a	Text processing using Awk script - Employee average pay.	34
4b	Text processing using Awk script - Results of an examination.	36
5	System calls –fork(), exec(), getpid(),opendir(), readdir().	38
6a	FCFS	41
6b	SJF	43
6c	Priority	46
6d	Round Robin	48
7	Inter-process Communication using Shared Memory.	51
8	Producer Consumer using Semaphores.	55
9	Bankers Deadlock Avoidance algorithms.	58
10a	Best Fit	61
10b	First Fit	63
11a	FIFO	65

11b	LRU	67
11c	Optimal	70
12	File Organization Technique- single and Two level directory.	73

Ex No: 1a
Date: 21/1/25

INSTALLATION AND CONFIGURATION OF LINUX

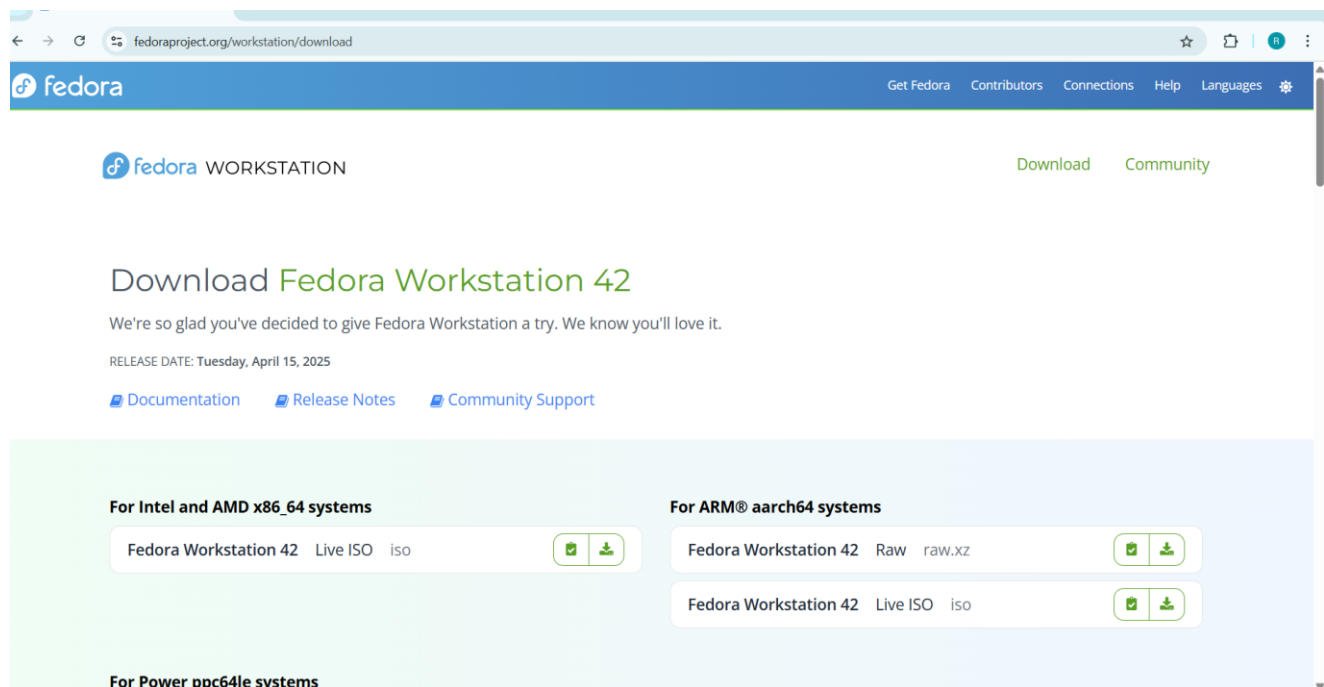
AIM:

To install and configure Linux operating system in a Virtual Machine.

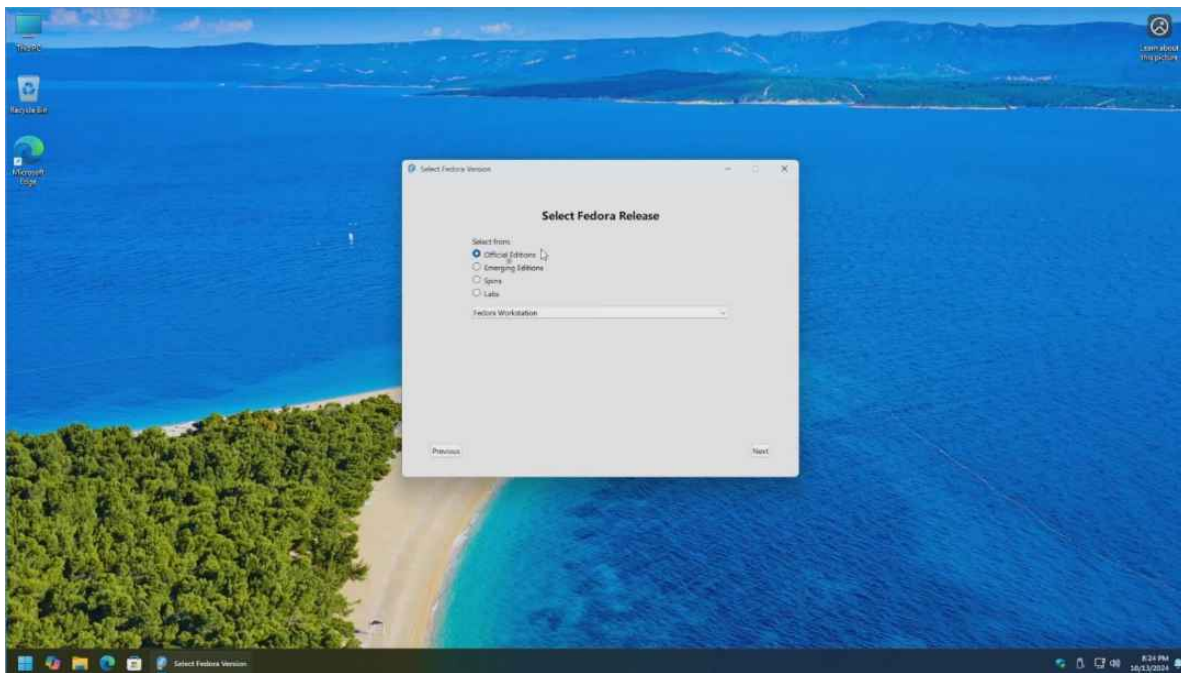
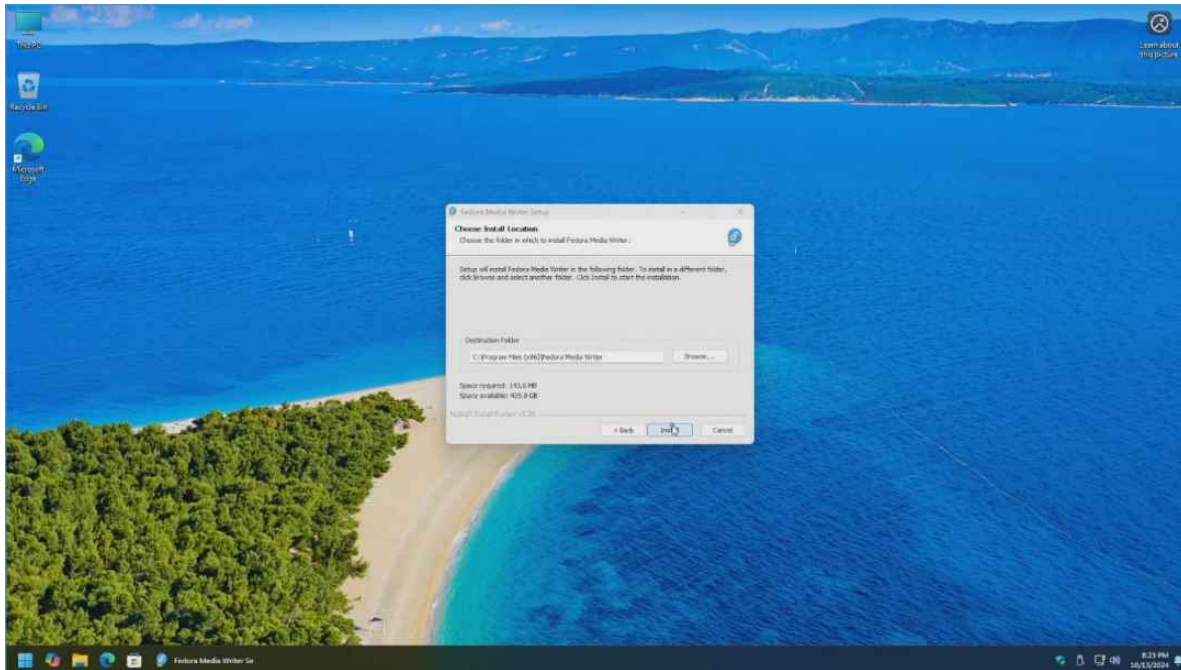
INSTALLATION/CONFIGURATION STEPS:

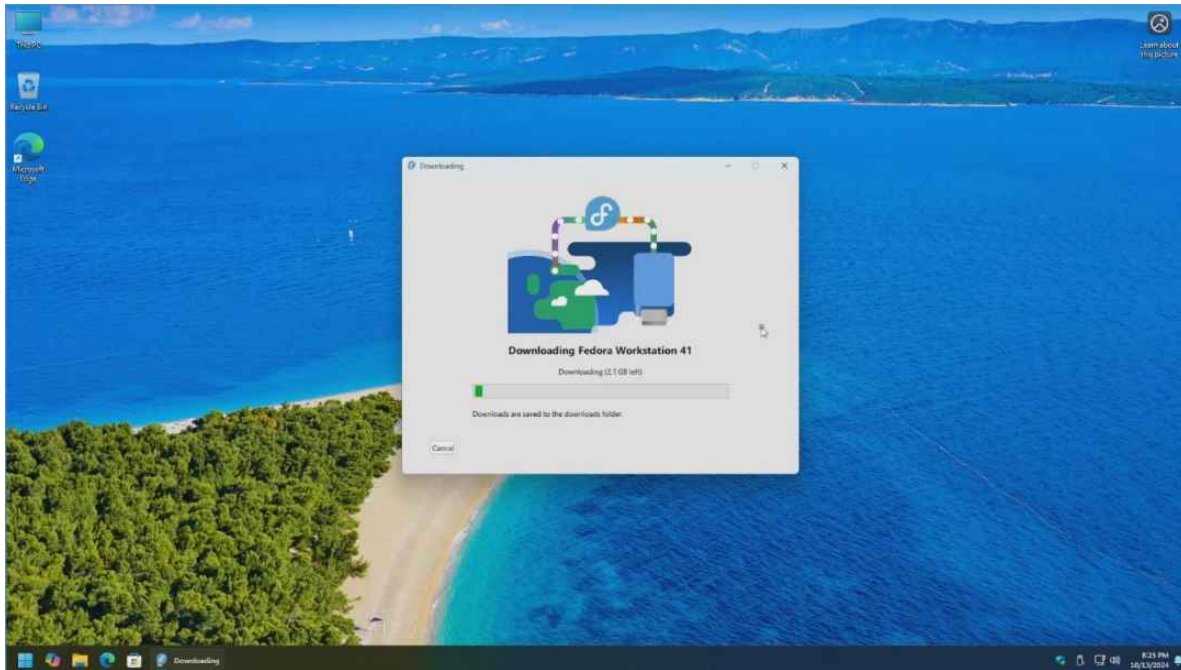
1. Install the required packages for virtualization
`dnf install xen virt-manager qemu libvirt`
2. Configure xend to start up on boot
`systemctl enable virt-manager. service`
3. Reboot the machine
Reboot
4. Create a Virtual machine by first running virt-manager
virt-manager &
5. Click on File and then click to connect to localhost
6. In the base menu, right-click on the localhost (QEMU) to create a new VM
7. Select Linux ISO image
8. Choose puppy-linux.iso then the kernel version
9. Select CPU and RAM limits
10. Create default disk image to 8 GB
11. Click finish to create the new VM with PuppyLinux.

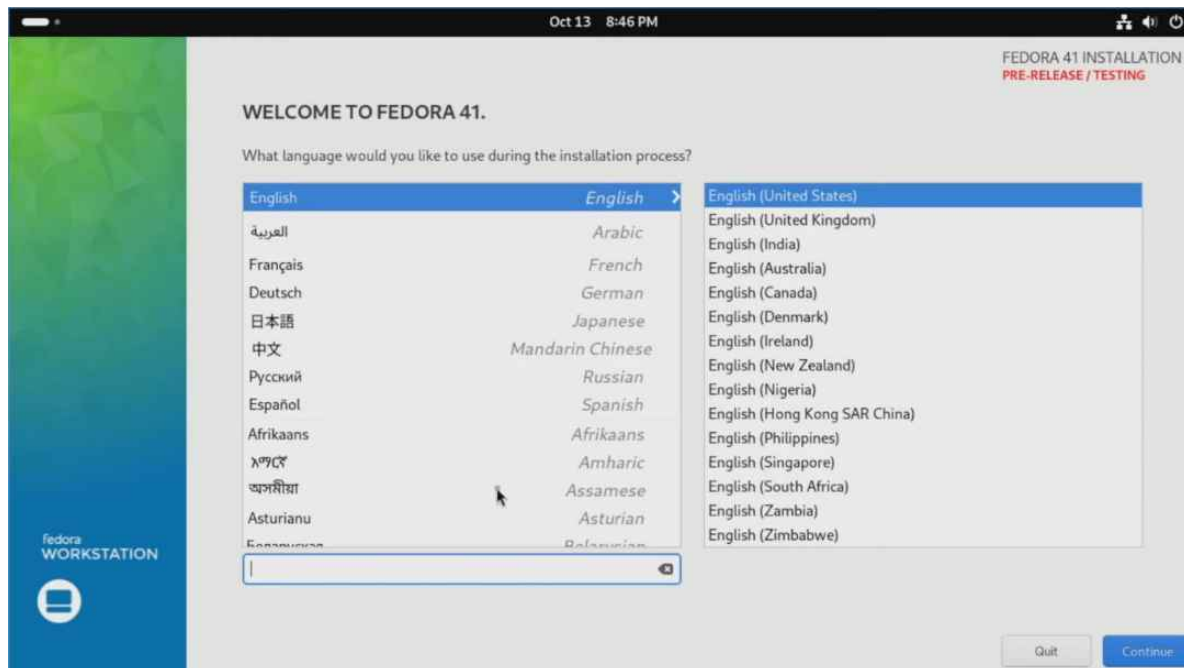
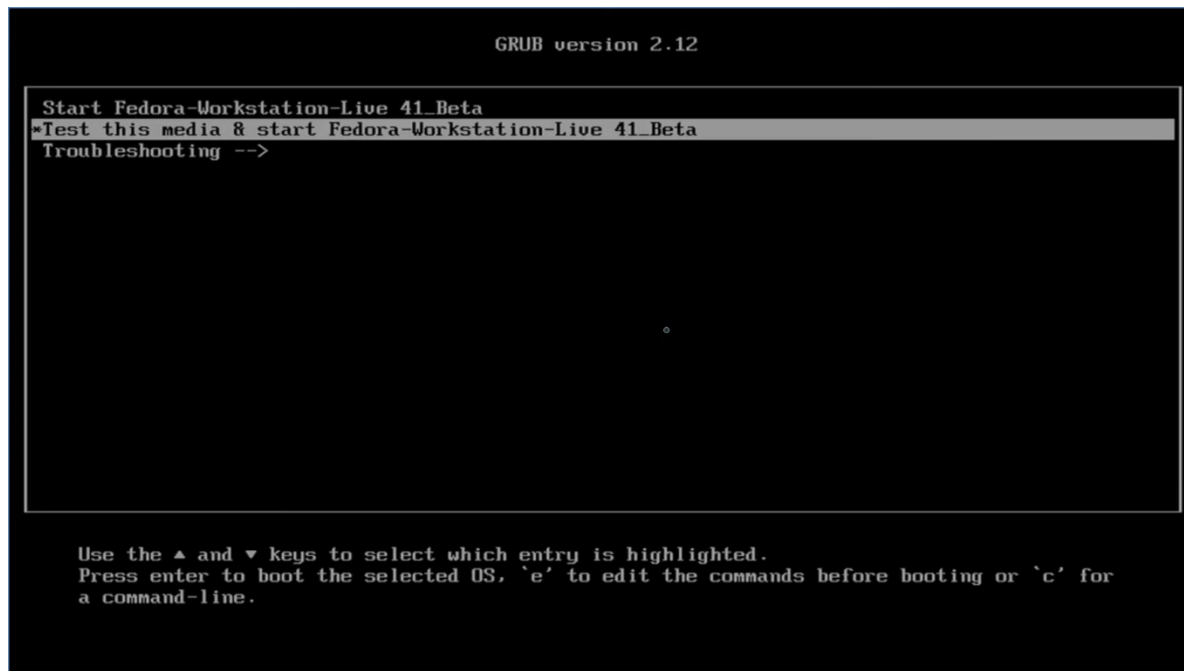
OUTPUT:

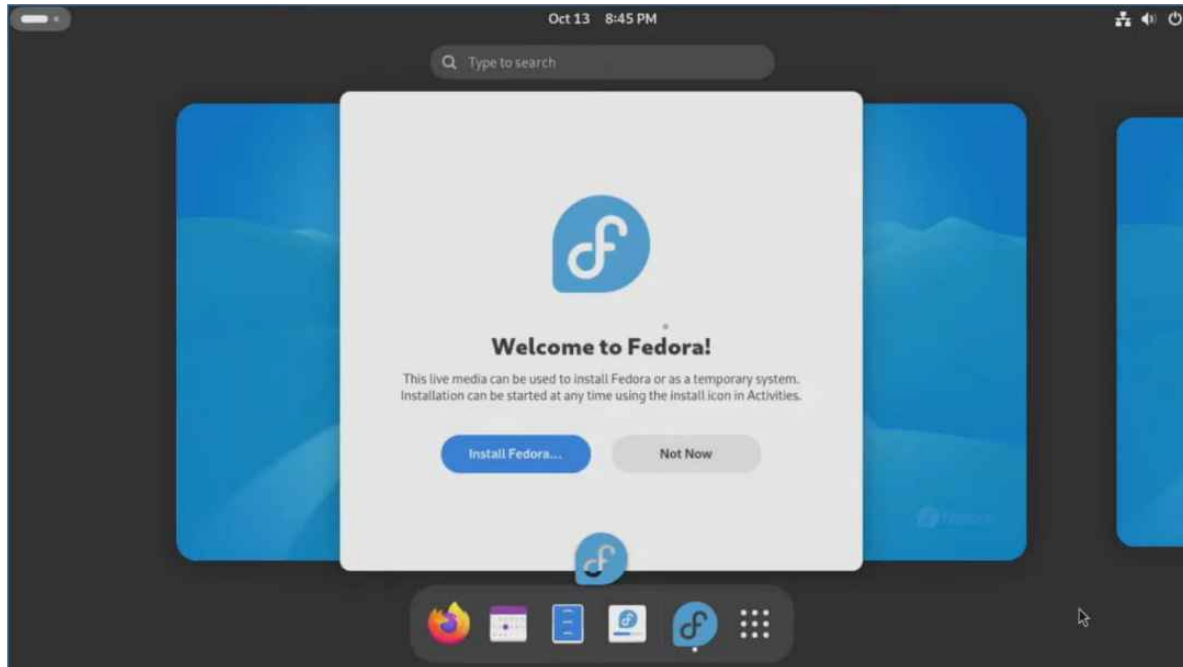


2116231801135









RESULT:

The Linux OS is Installed and Configured.

2116231801135

Ex No: 1b
Date: 21/1/2025

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:

2116231801135

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. 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)

2116231801135

-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:

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

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

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

9. The '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:

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 the 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.

13. The 'chmod' command:

2116231801135

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.

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

2116231801135

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 be 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 unsuccessful,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

2116231801135

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 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.

2116231801135

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.

```
2116231801135
```

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

syntax- 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
```

```
--- 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 to configure network interface.

syntax- 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. syntax- 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
```

OUTPUT:

```
[student@localhost ~]$ date +%m
01
[student@localhost ~]$ date +%h
Jan
[student@localhost ~]$ date +%d
25
[student@localhost ~]$ date +%y
25
[student@localhost ~]$ date +%H
09
[student@localhost ~]$ date +%M
21
[student@localhost ~]$ date +%S
26
[student@localhost ~]$ echo "Hello World"
Hello World
[student@localhost ~]$ echo "Hi"
Hi
[student@localhost ~]$ bc
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'.
15*25
375
524*965
505660
quit
[student@localhost ~]$ who
student pts/0      2025-01-25 08:12 (:0)
student pts/1      2025-01-25 09:20 (:0)
[student@localhost ~]$ who am i
student pts/1      2025-01-25 09:20 (:0)
[student@localhost ~]$ id
uid=1000(student) gid=1000(student) groups=1000(student) context=unconfined_u:unconfined_r:unconfined_t:s0-s0:c0.c1023
[student@localhost ~]$ tty
/dev/pts/1
[student@localhost ~]$ man
what manual page do you want?
[student@localhost ~]$ ps
  PID TTY          TIME CMD
 2125 pts/1    00:00:00 bash
2161 pts/1    00:00:00 ps
```

```
[student@localhost ~]$ ps
  PID TTY          TIME CMD
 2125 pts/1    00:00:00 bash
2161 pts/1    00:00:00 ps
[student@localhost ~]$ ps -e
  PID TTY          TIME CMD
   1 ?        00:00:01 systemd
   2 ?        00:00:00 kthreadd
   4 ?        00:00:00 kworker/0:0H
   6 ?        00:00:00 mm_percpu_wq
   7 ?        00:00:00 ksoftirqd/0
   8 ?        00:00:00 rcu_sched
   9 ?        00:00:00 rcu_bh
  10 ?       00:00:00 migration/0
  11 ?       00:00:00 watchdog/0
  12 ?       00:00:00 cpuhp/0
  13 ?       00:00:00 cpuhp/1
  14 ?       00:00:00 watchdog/1
  15 ?       00:00:00 migration/1
  16 ?       00:00:00 ksoftirqd/1
  18 ?       00:00:00 kworker/1:0H
  19 ?       00:00:00 cpuhp/2
  20 ?       00:00:00 watchdog/2
  21 ?       00:00:00 migration/2
  22 ?       00:00:00 ksoftirqd/2
  24 ?       00:00:00 kworker/2:0H
  25 ?       00:00:00 cpuhp/3
  26 ?       00:00:00 watchdog/3
  27 ?       00:00:00 migration/3
  28 ?       00:00:00 ksoftirqd/3
  30 ?       00:00:00 kworker/3:0H
  31 ?       00:00:00 kdevtmpfs
  32 ?       00:00:00 netns
  34 ?       00:00:01 kworker/2:1
  35 ?       00:00:00 oom_reaper
  36 ?       00:00:00 writeback
  37 ?       00:00:00 kcompactd0
  38 ?       00:00:00 ksm
  39 ?       00:00:00 crypto
  40 ?       00:00:00 kintegrityd
  41 ?       00:00:00 bioset
  42 ?       00:00:00 khlockd
  44 ?       00:00:01 kworker/0:1
  48 ?       00:00:00 kworker/0:1
```

2116231801135

```
44 ? 00:00:01 kworker/0:1
45 ? 00:00:00 ata_sff
46 ? 00:00:00 md
47 ? 00:00:00 devfreq_wq
48 ? 00:00:00 watchdogd
50 ? 00:00:00 kauditd
51 ? 00:00:00 kswapd0
52 ? 00:00:00 bioset
99 ? 00:00:00 kthrotld
100 ? 00:00:00 acpi_thermal_pm
101 ? 00:00:00 scsi_ah_0
102 ? 00:00:00 scsi_tm_0
103 ? 00:00:00 scsi_ah_1
104 ? 00:00:00 scsi_tm_1
105 ? 00:00:00 scsi_ah_2
106 ? 00:00:00 scsi_tm_2
107 ? 00:00:00 scsi_ah_3
108 ? 00:00:00 scsi_tm_3
109 ? 00:00:00 scsi_ah_4
110 ? 00:00:00 scsi_tm_4
115 ? 00:00:00 dm_bufio_cache
116 ? 00:00:00 ipv6_addrconf
150 ? 00:00:00 bioset
151 ? 00:00:00 bioset
209 ? 00:00:01 kworker/1:2
355 ? 00:00:00 kworker/0:1H
357 ? 00:00:00 kworker/1:1H
363 ? 00:00:00 kworker/3:1H
366 ? 00:00:00 i915/signal:0
367 ? 00:00:00 i915/signal:1
368 ? 00:00:00 i915/signal:2
369 ? 00:00:00 i915/signal:4
391 ? 00:00:00 kworker/2:1H
428 ? 00:00:00 kdmflush
429 ? 00:00:00 bioset
441 ? 00:00:00 kdmflush
442 ? 00:00:00 bioset
459 ? 00:00:00 jbd2/dm-0-8
460 ? 00:00:00 ext4-rsv-conver
544 ? 00:00:00 systemd-journal
573 ? 00:00:00 systemd-udev
612 ? 00:00:00 irq/32-mei_me
652 ? 00:00:00 jbd2/sda6-8
```

```
460 ? 00:00:00 ext4-rsv-conver
544 ? 00:00:00 systemd-journal
573 ? 00:00:00 systemd-udev
612 ? 00:00:00 irq/32-mei_me
652 ? 00:00:00 jbd2/sda6-8
653 ? 00:00:00 ext4-rsv-conver
656 ? 00:00:00 kdmflush
658 ? 00:00:00 bioset
660 ? 00:00:00 jbd2/dm-2-8
669 ? 00:00:00 ext4-rsv-conver
692 ? 00:00:00 rpciod
693 ? 00:00:00 xprtiod
695 ? 00:00:00 auditd
714 ? 00:00:00 alsactl
715 ? 00:00:00 mcelog
716 ? 00:00:00 Modemanager
718 ? 00:00:00 sssd
719 ? 00:03:15 avahi-daemon
720 ? 00:00:00 irqbalance
721 ? 00:00:00 dbus-daemon
723 ? 00:00:00 avahi-daemon
727 ? 00:00:00 gssproxy
735 ? 00:00:00 rsyslogd
736 ? 00:00:00 smartd
738 ? 00:00:00 firewallld
743 ? 00:00:00 rtkit-daemon
748 ? 00:00:00 abrttd
751 ? 00:00:00 chronyd
764 ? 00:00:00 sssd-be
768 ? 00:00:00 abrt-dump-journ
769 ? 00:00:00 abrt-dump-journ
770 ? 00:00:00 abrt-dump-journ
771 ? 00:00:00 sssd-nss
772 ? 00:00:00 accounts-daemon
773 ? 00:00:00 systemd-logind
788 ? 00:00:00 NetworkManager
789 ? 00:00:00 polkitd
820 ? 00:00:00 crond
821 ? 00:00:00 atd
823 ? 00:00:00 sddm
884 tty1 00:00:13 Xorg
1011 ? 00:00:01 udiskd
1019 ? 00:00:00 upowerd
1080 ? 00:00:00 ddp-buscon
```

```
1013 ? 00:00:01 udiskd
1019 ? 00:00:00 upowerd
1058 ? 00:00:00 sddm-helper
1062 ? 00:00:00 systemd
1064 ? 00:00:00 (sd-pam)
1075 ? 00:00:00 kwalllet5
1078 ? 00:00:00 startkde
1097 ? 00:00:00 dbus-daemon
1102 ? 00:00:00 ssh-agent
1143 ? 00:00:00 start_kdeinit
1144 ? 00:00:00 kdeinit5
1145 ? 00:00:00 klauncher
1148 ? 00:00:01 kded5
1161 ? 00:00:00 kaccess
1166 ? 00:00:00 kwrapper5
1171 ? 00:00:00 dconf-service
1173 ? 00:00:00 kmserver
1178 ? 00:00:00 kglobalaccel5
1183 ? 00:00:00 mission-control
1185 ? 00:00:00 colord
1191 ? 00:00:13 kwin_x11
1205 ? 00:00:00 kscreen_backend
1210 ? 00:00:00 baloo_file
1212 ? 00:00:00 kdeconnectd
1214 ? 00:00:01 krunner
1216 ? 00:00:15 plasmashell
1217 ? 00:00:00 polkit-kde-auth
1218 ? 00:00:00 xembedsniproxy
1269 ? 00:00:01 kworker/3:0
1279 ? 00:00:00 pulseaudio
1296 ? 00:00:00 abrt-applet
1298 ? 00:00:00 korgac
1299 ? 00:00:00 org_kde_powerde
1328 ? 00:00:00 kactivitymanage
1371 ? 00:00:00 at-spi-bus-laun
1381 ? 00:00:00 dbus-daemon
1386 ? 00:00:00 at-spi2-registr
1446 ? 00:00:00 abrt-dbus
1452 ? 00:00:00 akonadi_control
1456 ? 00:00:00 akonadiServer
1459 ? 00:00:02 mysqld
1499 ? 00:00:00 akonadi_akonote
1500 ? 00:00:00 akonadi_archive
```

```
1499 ? 00:00:00 akonadi_akonote
1500 ? 00:00:00 akonadi_archive
1501 ? 00:00:00 akonadi_birthday
1502 ? 00:00:00 akonadi_contact
1503 ? 00:00:00 akonadi_follow
1504 ? 00:00:00 akonadi_ical_re
1507 ? 00:00:00 akonadi_indexin
1510 ? 00:00:00 akonadi_maildir
1529 ? 00:00:00 akonadi_maildis
1530 ? 00:00:00 akonadi_mailfil
1531 ? 00:00:00 akonadi_migrati
1532 ? 00:00:00 akonadi_newmail
1533 ? 00:00:00 akonadi_sendlat
1601 ? 00:00:00 kuiserver5
1605 ? 00:00:00 cupsd
1607 ? 00:00:00 packagekitd
1818 ? 00:00:00 kworker/3:1
1845 ? 00:00:00 kworker/2:2
1939 ? 00:00:00 kworker/u4:0
1942 ? 00:00:00 kworker/u4:3
1952 ? 00:00:00 kworker/0:2
1960 ? 00:00:00 kworker/u4:1
2004 ? 00:00:13 amarok
2008 ? 00:00:00 kdeinit4
2010 ? 00:00:00 klauncher
2012 ? 00:00:00 kded4
2014 ? 00:00:00 gam_server
2057 ? 00:00:00 knotify4
2087 ? 00:00:00 kio_http_cache_
2114 ? 00:00:00 kworker/1:0
2121 ? 00:00:00 konsole
2125 pts/1 00:00:00 bash
2158 ? 00:00:00 kworker/1:1
2162 pts/1 00:00:00 ps

[student@localhost ~]$ ps -aux
USER      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.1 32260 10376 ?        Ss   08:01   0:01 /usr/lib/systemd/systemd --switched-root --system --deserialize 24
root         2  0.0  0.0      0   0 ?        S    08:01   0:00 [kthreadd]
root         4  0.0  0.0      0   0 ?        S<   08:01   0:00 [kworker/0:0H]
root         6  0.0  0.0      0   0 ?        S<   08:01   0:00 [mm_percpu_wq]
root         7  0.0  0.0      0   0 ?        S    08:01   0:00 [ksoftirqd/0]
root         8  0.0  0.0      0   0 ?        S    08:01   0:00 [rcu_sched]
root         9  0.0  0.0      0   0 ?        S    08:01   0:00 [rcu_bh]
```

USER	PID	PCPU	MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.1	32260	10376	?	Ss	00:01	0:01	/usr/lib/systemd/systemd --switched-root --system --deserialize 24
root	2	0.0	0.0	0	0	?	S	00:01	0:00	[kthreadd]
root	4	0.0	0.0	0	0	?	Sc	00:01	0:00	[kworker/0:0H]
root	6	0.0	0.0	0	0	?	Sc	00:01	0:00	[mm_percpu_wq]
root	7	0.0	0.0	0	0	?	S	00:01	0:00	[ksoftirqd/0]
root	8	0.0	0.0	0	0	?	S	00:01	0:00	[rcu_sched]
root	9	0.0	0.0	0	0	?	S	00:01	0:00	[rcu_bh]
root	10	0.0	0.0	0	0	?	S	00:01	0:00	[migration/0]
root	11	0.0	0.0	0	0	?	S	00:01	0:00	[watchdog/0]
root	12	0.0	0.0	0	0	?	S	00:01	0:00	[cpuhp/0]
root	13	0.0	0.0	0	0	?	S	00:01	0:00	[cpuhp/1]
root	14	0.0	0.0	0	0	?	S	00:01	0:00	[watchdog/1]
root	15	0.0	0.0	0	0	?	S	00:01	0:00	[migration/1]
root	16	0.0	0.0	0	0	?	S	00:01	0:00	[ksoftirqd/1]
root	18	0.0	0.0	0	0	?	Sc	00:01	0:00	[kworker/1:0H]
root	19	0.0	0.0	0	0	?	S	00:01	0:00	[cpuhp/2]
root	20	0.0	0.0	0	0	?	S	00:01	0:00	[watchdog/2]
root	21	0.0	0.0	0	0	?	S	00:01	0:00	[migration/2]
root	22	0.0	0.0	0	0	?	S	00:01	0:00	[ksoftirqd/2]
root	24	0.0	0.0	0	0	?	Sc	00:01	0:00	[kworker/2:0H]
root	25	0.0	0.0	0	0	?	S	00:01	0:00	[cpuhp/3]
root	26	0.0	0.0	0	0	?	S	00:01	0:00	[watchdog/3]
root	27	0.0	0.0	0	0	?	S	00:01	0:00	[migration/3]
root	28	0.0	0.0	0	0	?	S	00:01	0:00	[ksoftirqd/3]
root	30	0.0	0.0	0	0	?	Sc	00:01	0:00	[kworker/3:0H]
root	31	0.0	0.0	0	0	?	S	00:01	0:00	[kdevtmpfs]
root	32	0.0	0.0	0	0	?	Sc	00:01	0:00	[netns]
root	34	0.0	0.0	0	0	?	S	00:01	0:01	[kworker/2:1]
root	35	0.0	0.0	0	0	?	S	00:01	0:00	[oom_reaper]
root	36	0.0	0.0	0	0	?	Sc	00:01	0:00	[writeback]
root	37	0.0	0.0	0	0	?	S	00:01	0:00	[kcompactd]
root	38	0.0	0.0	0	0	?	Sc	00:01	0:00	[kswapd]
root	39	0.0	0.0	0	0	?	Sc	00:01	0:00	[crypto]
root	40	0.0	0.0	0	0	?	Sc	00:01	0:00	[kintegrityd]
root	41	0.0	0.0	0	0	?	Sc	00:01	0:00	[blosel]
root	42	0.0	0.0	0	0	?	Sc	00:01	0:00	[kblockd]
root	44	0.0	0.0	0	0	?	S	00:01	0:01	[kworker/0:1]
root	45	0.0	0.0	0	0	?	Sc	00:01	0:00	[ata_sff]
root	46	0.0	0.0	0	0	?	Sc	00:01	0:00	[md]
root	47	0.0	0.0	0	0	?	Sc	00:01	0:00	[devfreq_wq]
root	48	0.0	0.0	0	0	?	Sc	00:01	0:00	[watchdog]
root	50	0.0	0.0	0	0	?	S	00:01	0:00	[kauditd]
root	81	0.0	0.0	0	0	?	S	00:01	0:00	[dmesg]

root	50	0.0	0.0	0	0	?	S	00:01	0:00	[kauditd]
root	51	0.0	0.0	0	0	?	S	00:01	0:00	[kswapd]
root	52	0.0	0.0	0	0	?	Sc	00:01	0:00	[blosel]
root	99	0.0	0.0	0	0	?	Sc	00:01	0:00	[kthrotld]
root	100	0.0	0.0	0	0	?	Sc	00:01	0:00	[acpi_thermal_pm]
root	101	0.0	0.0	0	0	?	S	00:01	0:00	[scsi_oh_0]
root	102	0.0	0.0	0	0	?	Sc	00:01	0:00	[scsi_tmf_0]
root	103	0.0	0.0	0	0	?	S	00:01	0:00	[scsi_oh_1]
root	104	0.0	0.0	0	0	?	Sc	00:01	0:00	[scsi_tmf_1]
root	105	0.0	0.0	0	0	?	S	00:01	0:00	[scsi_oh_2]
root	106	0.0	0.0	0	0	?	Sc	00:01	0:00	[scsi_tmf_2]
root	107	0.0	0.0	0	0	?	S	00:01	0:00	[scsi_oh_3]
root	108	0.0	0.0	0	0	?	Sc	00:01	0:00	[scsi_tmf_3]
root	109	0.0	0.0	0	0	?	S	00:01	0:00	[scsi_oh_4]
root	110	0.0	0.0	0	0	?	Sc	00:01	0:00	[scsi_tmf_4]
root	115	0.0	0.0	0	0	?	Sc	00:01	0:00	[dm_bufio_cache]
root	116	0.0	0.0	0	0	?	Sc	00:01	0:00	[ipv6_addrconf]
root	150	0.0	0.0	0	0	?	Sc	00:01	0:00	[blosel]
root	151	0.0	0.0	0	0	?	Sc	00:01	0:00	[blosel]
root	209	0.0	0.0	0	0	?	S	00:01	0:01	[kworker/1:2]
root	355	0.0	0.0	0	0	?	Sc	00:01	0:00	[kworker/0:1H]
root	357	0.0	0.0	0	0	?	Sc	00:01	0:00	[kworker/1:1H]
root	363	0.0	0.0	0	0	?	Sc	00:01	0:00	[kworker/1:1H]
root	366	0.0	0.0	0	0	?	S	00:01	0:00	[i915/signal:0]
root	367	0.0	0.0	0	0	?	S	00:01	0:00	[i915/signal:1]
root	368	0.0	0.0	0	0	?	S	00:01	0:00	[i915/signal:2]
root	369	0.0	0.0	0	0	?	S	00:01	0:00	[i915/signal:4]
root	391	0.0	0.0	0	0	?	Sc	00:01	0:00	[kworker/2:1H]
root	428	0.0	0.0	0	0	?	Sc	00:01	0:00	[kdmflush]
root	429	0.0	0.0	0	0	?	Sc	00:01	0:00	[blosel]
root	441	0.0	0.0	0	0	?	Sc	00:01	0:00	[kdmflush]
root	442	0.0	0.0	0	0	?	Sc	00:01	0:00	[blosel]
root	459	0.0	0.0	0	0	?	S	00:01	0:00	[jbd2/dm-0-B]
root	460	0.0	0.0	0	0	?	Sc	00:01	0:00	[ext4-rsv-conver]
root	544	0.0	0.1	42756	9240	?	Ss	00:01	0:00	/usr/lib/systemd/systemd-journald
root	573	0.0	0.0	23956	8028	?	Ss	00:01	0:00	/usr/lib/systemd/systemd-udev
root	612	0.0	0.0	0	0	?	S	00:01	0:00	[irq/32-mei_me]
root	652	0.0	0.0	0	0	?	S	00:01	0:00	[jbd2/sdab-0]
root	653	0.0	0.0	0	0	?	Sc	00:01	0:00	[ext4-rsv-conver]
root	656	0.0	0.0	0	0	?	Sc	00:01	0:00	[kdmflush]
root	658	0.0	0.0	0	0	?	Sc	00:01	0:00	[blosel]
root	668	0.0	0.0	0	0	?	S	00:01	0:00	[jbd2/dm-2-B]
root	669	0.0	0.0	0	0	?	Sc	00:01	0:00	[ext4-rsv-conver]

```

root 693 0.0 0.0 0 0 ? 5c 00:01 0:00 [xperiod]
root 695 0.0 0.0 20388 1908 ? 5c1 00:01 0:00 /shin/auditd
root 714 0.0 0.0 4112 1384 ? 5M6 00:01 0:00 /usr/shin/alsactl -s -n 19 -c -E ALSA_CONFIG_PATH=/etc/alsa/alsactl.conf --initfile=/lib/alsa/init/00main ndaemo
root 715 0.0 0.0 11100 2080 ? 5s 00:01 0:00 /usr/shin/mclog --ignoreudev --daemon --foreground
root 716 0.0 0.1 50948 8392 ? 5s1 00:01 0:00 /usr/shin/ModemManager
root 718 0.0 0.1 3088 8420 ? 5s 00:01 0:00 /usr/shin/sssd -i -f
root 719 0.0 0.0 14632 7480 ? 5s 00:01 3:15 avahi-daemon: running [linux-2.local]
root 720 0.0 0.0 14192 1384 ? 5s1 00:01 0:00 /usr/shin/irqbalance --foreground
dbus 721 0.0 0.0 40312 5520 ? 5s1 00:01 0:00 /usr/bin/dbus-daemon --system --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
avahi 723 0.0 0.0 31204 276 ? 5 00:01 0:00 avahi-daemon: chroot helper
root 727 0.0 0.0 40856 3352 ? 5s1 00:01 0:00 /usr/shin/gssproxy -D
root 735 0.0 0.0 66848 5808 ? 5s1 00:01 0:00 /usr/shin/rsyslogd -n
root 736 0.0 0.0 5972 3896 ? 5s 00:01 0:00 /usr/shin/smartd -n -q never
root 738 0.0 0.3 43212 26920 ? 5s1 00:01 0:00 /usr/bin/python3 -Es /usr/sbin/firewalld --nofork --nopid
rtkit 743 0.0 0.0 24164 3300 ? 5M1 00:01 0:00 /usr/libexec/rtkit-daemon
root 748 0.0 0.1 63192 8468 ? 5s1 00:01 0:00 /usr/shin/abrt-d -s
chrony 753 0.0 0.0 22396 3356 ? 5 00:01 0:00 /usr/shin/chronyd
root 764 0.0 0.1 38936 9236 ? 5 00:01 0:00 /usr/libexec/sss/sss_be --domain implicit_files --uid 0 --gid 0 --debug-to-files
root 768 0.0 0.1 70260 9576 ? 5s1 00:01 0:00 /usr/bin/abrt-dump-journal-oops -xtd
root 769 0.0 0.1 20224 8068 ? 5s 00:01 0:00 /usr/bin/abrt-dump-journal-xorg -xtd
root 770 0.0 0.1 70224 9412 ? 5s 00:01 0:00 /usr/bin/abrt-dump-journal-core -D -i -f -e
root 771 0.0 0.1 44304 32760 ? 5 00:01 0:00 /usr/libexec/sss/sss_nss --uid 0 --gid 0 --debug-to-files
root 772 0.0 0.1 66456 8548 ? 5s1 00:01 0:00 /usr/libexec/accounts-daemon
root 773 0.0 0.0 20680 8196 ? 5s 00:01 0:00 /usr/lib/systemd/systemd-logind
root 788 0.0 0.2 83628 17480 ? 5s1 00:01 0:00 /usr/shin/NetworkManager --no-daemon
polkitd 789 0.0 0.1 104460 15588 ? 5s1 00:01 0:00 /usr/lib/polkit-1/polkitd --no-debug
root 820 0.0 0.0 14776 7616 ? 5s 00:01 0:00 /usr/shin/cron -n
root 821 0.0 0.0 18104 2360 ? 5s 00:01 0:00 /usr/shin/atd -f
root 823 0.0 0.1 73100 13760 ? 5s1 00:01 0:00 /usr/bin/sddm
root 884 0.2 0.6 103248 56960 tty1 5s1+ 00:01 0:13 /usr/libexec/Xorg -nolisten tcp -auth /var/run/sddm/{28a38881-c800-485f-a7fb-244d759105bf} -background none -nor
root 1013 0.0 0.1 68208 10052 ? 5s1 00:01 0:01 /usr/libexec/udisks2/udisksd
root 1019 0.0 0.0 46296 6272 ? 5s1 00:01 0:00 /usr/libexec/fupowerd
root 1054 0.0 0.1 61492 14076 ? 5s1 00:12 0:00 /usr/libexec/sdow-helper --socket /tmp/sddm-auth8563439-eed-4e7b-a691-1f0713da042 --id 1 --start /usr/bin/sta
student 1062 0.0 0.0 20164 7616 ? 5s 00:12 0:00 /usr/lib/systemd/systemd --user
student 1064 0.0 0.0 51884 2480 ? 5 00:12 0:00 (sd-pam)
student 1075 0.0 0.4 139196 33828 ? 5s1 00:12 0:00 /usr/bin/kwalletd5 --pam-login 4 17
student 1078 0.0 0.0 5784 3092 ? 5s 00:12 0:00 /bin/sh /usr/bin/startkde
student 1097 0.0 0.0 33988 5888 ? 5s1 00:12 0:00 /usr/bin/dbus-daemon --session --address=systemd: --nofork --nopidfile --systemd-activation --syslog-only
student 1102 0.0 0.0 10644 520 ? 5s 00:12 0:00 /usr/bin/ssh-agent /bin/sh -c exec -l /bin/bash -c "/usr/bin/startkde"
student 1143 0.0 0.0 4468 128 ? 5s 00:12 0:00 /usr/libexec/kfs/start_kdeinit --kded *kcmint_startup
student 1144 0.0 0.1 64224 8232 ? 5s 00:12 0:00 kdeinit5; Running...
student 1145 0.0 0.3 122860 32328 ? 5s1 00:12 0:00 /usr/libexec/kfs/klauncher --fd=9
student 1148 0.0 0.7 277396 58832 ? 5s1 00:12 0:01 kded5 [kdeinit5]
student 1161 0.0 0.3 143244 31340 ? 5s1 00:13 0:00 /usr/bin/kdeconnect

student 1173 0.0 0.4 146560 38380 ? 5s1 00:12 0:00 /usr/bin/kmservice
student 1178 0.0 0.3 123464 31244 ? 5s1 00:12 0:00 /usr/bin/kglobalaccel5
student 1183 0.0 0.1 50876 11780 ? 5s1 00:12 0:00 /usr/libexec/mission-control-5
colord 1185 0.0 0.1 5816 1464 ? 5s1 00:12 0:00 /usr/libexec/colord
student 1191 0.1 0.9 316792 78436 ? 5s1 00:12 0:13 bdlc_x11
student 1205 0.0 0.2 80824 17476 ? 5s1 00:12 0:00 /usr/libexec/kfs/kscreen_backend_launcher
student 1210 0.0 0.2 1125568 17704 ? 5M1 00:12 0:00 /usr/bin/baloo_file
student 1212 0.0 0.4 151940 40124 ? 5s1 00:12 0:00 /usr/libexec/kdeconnectd
student 1214 0.0 1.2 1385220 99496 ? 5s1 00:12 0:01 /usr/bin/krunner
student 1216 0.3 2.5 1018156 210240 ? 5s1 00:12 0:15 /usr/bin/plasmashell
student 1217 0.0 0.4 157936 35264 ? 5s1 00:12 0:00 /usr/libexec/kfs/polkit-kde-authentication-agent-1
student 1218 0.0 0.3 129832 30464 ? 5s1 00:12 0:00 /usr/bin/xembeddiproxy
root 1260 0.0 0.0 0 0 ? 5 00:12 0:01 [kworker/3:0]
student 1279 0.0 0.1 1404256 10852 ? 5c1 00:12 0:00 /usr/bin/pulseaudio --start --log-target=syslog
student 1296 0.0 0.2 78068 23864 ? 5s1 00:12 0:00 /usr/bin/abrt-applet
student 1298 0.0 0.7 349124 65560 ? 5s1 00:12 0:00 /usr/bin/korgac
student 1299 0.0 0.4 151000 35844 ? 5s1 00:12 0:00 /usr/libexec/org.kde.powersdevil
student 1328 0.0 0.4 103268 35060 ? 5s1 00:12 0:00 /usr/bin/ktactivitymanagerd start-daemon
student 1371 0.0 0.0 48172 7160 ? 5s1 00:12 0:00 /usr/libexec/at-spi-bus-launcher
student 1381 0.0 0.0 33240 4668 ? 5s1 00:12 0:00 /bin/dbus-daemon --config-file=/usr/share/defaults/at-spi2/accessibility.conf --nofork --print-address 3
student 1386 0.0 0.0 30076 6560 ? 5s1 00:12 0:00 /usr/libexec/at-spi2-registryd --use-gnome-session
root 1446 0.0 0.0 55488 8116 ? 5s1 00:12 0:00 /usr/shin/abrt-dbus -t133
student 1452 0.0 0.3 130748 31828 ? 5s1 00:12 0:00 /usr/bin/akonadi_control
student 1456 0.0 0.3 241808 28160 ? 5s1 00:12 0:00 akonadi-server
student 1459 0.0 0.0 526992 95356 ? 5s1 00:12 0:02 /usr/libexec/mysql -defaults-file=/home/student/.local/share/akonadi/mysql.conf --datadir=/home/student/.local
student 1499 0.0 0.4 145168 36240 ? 5s1 00:12 0:00 /usr/bin/akonadi_akonotes_resource --identifier akonadi_akonotes_resource_0
student 1500 0.0 0.7 345544 64868 ? 5s1 00:12 0:00 /usr/bin/akonadi_archivemail_agent --identifier akonadi_archivemail_agent
student 1501 0.0 0.4 150136 38044 ? 5s1 00:12 0:00 /usr/bin/akonadi_birthdays_resource --identifier akonadi_birthdays_resource
student 1502 0.0 0.4 144812 36384 ? 5s1 00:12 0:00 /usr/bin/akonadi_contacts_resource --identifier akonadi_contacts_resource_0
student 1503 0.0 0.4 160816 39700 ? 5s1 00:12 0:00 /usr/bin/akonadi_followupreminder_agent --identifier akonadi_followupreminder_agent
student 1504 0.0 0.4 151472 38064 ? 5s1 00:12 0:00 /usr/bin/akonadi_ical_resource --identifier akonadi_ical_resource_0
student 1507 0.0 0.4 154560 40440 ? 5M1 00:12 0:00 /usr/bin/akonadi_indexing_agent --identifier akonadi_indexing_agent
student 1510 0.0 0.4 145168 35732 ? 5s1 00:12 0:00 /usr/bin/akonadi_maildir_resource --identifier akonadi_maildir_resource_0
student 1529 0.0 0.4 155640 37248 ? 5s1 00:12 0:00 /usr/bin/akonadi_maildispatcher_agent --identifier akonadi_maildispatcher_agent
student 1530 0.0 0.8 351476 67004 ? 5s1 00:12 0:00 /usr/bin/akonadi_mailfilter_agent --identifier akonadi_mailfilter_agent
student 1531 0.0 0.4 144376 36524 ? 5s1 00:12 0:00 /usr/bin/akonadi_migration_agent --identifier akonadi_migration_agent
student 1532 0.0 0.7 317728 58360 ? 5s1 00:12 0:00 /usr/bin/akonadi_newmailnotifier_agent --identifier akonadi_newmailnotifier_agent
student 1533 0.0 0.7 343160 63856 ? 5s1 00:12 0:00 /usr/bin/akonadi_sendlater_agent --identifier akonadi_sendlater_agent
student 1601 0.0 0.3 130504 32776 ? 5s1 00:12 0:00 /usr/bin/kuiservers
root 1605 0.0 0.0 31532 7876 ? 5s 00:12 0:00 /usr/shin/cupsd -l
root 1607 0.1 1.0 172724 88596 ? 5s1 00:12 0:06 /usr/libexec/packagekitd
root 1818 0.0 0.0 0 0 ? 5 00:49 0:00 [kworker/3:1]
root 1845 0.0 0.0 0 0 ? 5 00:50 0:00 [kworker/2:2]

```

```

root      1607  0.1  1.0 122724 88596 ?        Ssl  08:12  0:00 /usr/libexec/packagekitd
root      1838  0.0  0.0   0   0 ?        S    08:49  0:00 [kworker/3:1]
root      1845  0.0  0.0   0   0 ?        S    08:50  0:00 [kworker/2:2]
root      1939  0.0  0.0   0   0 ?        S    09:09  0:00 [kworker/u8:0]
root      1942  0.0  0.0   0   0 ?        S    09:10  0:00 [kworker/u8:1]
root      1952  0.0  0.0   0   0 ?        S    09:11  0:00 [kworker/0:2]
root      1960  0.0  0.0   0   0 ?        S    09:15  0:00 [kworker/u8:1]
student   2004  1.6  2.3 1321396 193748 ?        Sl   09:16  0:13 /usr/bin/amarok
student   2008  0.0  0.1 83180 15240 ?        Ss   09:16  0:00 kdeinit4: kdeinit4 Running...
student   2010  0.0  0.2 89312 19168 ?        S    09:16  0:00 kdeinit4: klauncher [kdeinit] --fd=9
student   2012  0.0  0.3 108896 27292 ?        S    09:16  0:00 kdeinit4: kded4 [kdeinit]
student   2014  0.0  0.0 12508 2700 ?        S    09:16  0:00 /usr/libexec/gam_server
student   2057  0.0  0.5 426024 45072 ?        Sl   09:17  0:00 /usr/bin/knottyfa
student   2087  0.0  0.2 88256 22272 ?        S    09:17  0:00 /usr/libexec/kde4/kio_http_cache_cleaner
root      2114  0.0  0.0   0   0 ?        S    09:17  0:00 [kworker/1:0]
student   2121  0.3  0.6 172348 56672 ?        Rl   09:20  0:00 /usr/bin/konsole
student   2125  0.0  0.0 14580 3996 pts/1    Ss   09:20  0:00 /bin/bash
root      2158  0.0  0.0   0   0 ?        S    09:22  0:00 [kworker/1:1]
student   2163  0.0  0.0 16672 3616 pts/1    R+   09:23  0:00 ps -aux
[student@localhost ~]$ uname -n
i686
[student@localhost ~]$ uname -m
localhost.localdomain
[student@localhost ~]$ uname -r
4.11.0-300.fc26.i686.PAE
[student@localhost ~]$ uname -s
Linux
[student@localhost ~]$ uname -v
#1 SMP Thu Jun 29 20:38:21 UTC 2017
[student@localhost ~]$ uname -a
Linux localhost.localdomain 4.11.0-300.fc26.i686.PAE #1 SMP Thu Jun 29 20:38:21 UTC 2017 i686 i686 i386 GNU/Linux
[student@localhost ~]$ pwd
/home/student
[student@localhost ~]$ ls
Desktop  Documents  Downloads  filename.sh  gowtham  karthi79  'lab 2 05.txt'  Music  os.txt  Pictures  Public  stu  Templates  Videos  wx  wxcollege
[student@localhost ~]$ mv os.txt karthi79
[student@localhost ~]$ cat karthi79
cat: karthi79: Is a directory
[student@localhost ~]$ ls karthi79
os.txt
[student@localhost ~]$ cat os.txt
cat: os.txt: No such file or directory
[student@localhost ~]$ cd karthi79
[student@localhost karthi79]$ cat os.txt

```

```

Linux localhost.localdomain 4.11.0-300.fc26.i686.PAE #1 SMP Thu Jun 29 20:38:21 UTC 2017 i686 i686 i386 GNU/Linux
[student@localhost ~]$ pwd
/home/student
[student@localhost ~]$ ls
Desktop  Documents  Downloads  filename.sh  gowtham  karthi79  'lab 2 05.txt'  Music  os.txt  Pictures  Public  stu  Templates  Videos  wx  wxcollege
[student@localhost ~]$ mv os.txt karthi79
[student@localhost ~]$ cat karthi79
cat: karthi79: Is a directory
[student@localhost ~]$ ls karthi79
os.txt
[student@localhost ~]$ cat os.txt
cat: os.txt: No such file or directory
[student@localhost ~]$ cd karthi79
[student@localhost karthi79]$ cat os.txt
Hi hello, how are you?
Good Bye
[student@localhost karthi79]$ cd -
/home/student
[student@localhost ~]$ cat os.txt
cat: os.txt: No such file or directory
[student@localhost ~]$ cd karthi79
[student@localhost karthi79]$ wc os.txt
 2  7 32 os.txt
[student@localhost karthi79]$ cd -
/home/student
[student@localhost ~]$ top gowtham
top: unknown option 'g'
Usage:
  top -hv [-bcHIOSS -d secs -n max -u[u] user -p pid(s) -o field -w [cols]
[student@localhost ~]$ head gowtham
r
e
c
c
o
l
l
e
e
[student@localhost ~]$ tail gowtham
e
t

```



```

o
l
l
e
g
e
[student@localhost ~]$ tail gowtham
e
t
h
a
n
d
a
l
a
m
[student@localhost ~]$ ping gowtham
ping: gowtham: Name or service not known
[student@localhost ~]$ cd karthi79
[student@localhost karthi79]$ ping os.txt
ping: os.txt: Name or service not known
[student@localhost karthi79]$ cd -
/home/student
[student@localhost ~]$ ifconfig
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.8.29 netmask 255.255.252.0 broadcast 172.16.11.255
    inet6 fe80::354c:ba27:ebcc:5d62 prefixlen 64 scopeid 0x20<link>
    ether f8:bc:12:90:45:7e txqueuelen 1000 (Ethernet)
    RX packets 409135 bytes 342188533 (326.3 MiB)
    RX errors 0 dropped 109 overruns 0 frame 0
    TX packets 7862 bytes 474073 (462.9 KiB)
    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 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
[student@localhost ~]$ cd karthi79

```

```

[student@localhost karthi79]$ cd -
/home/student
[student@localhost ~]$ ifconfig
enp2s0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 172.16.8.29 netmask 255.255.252.0 broadcast 172.16.11.255
    inet6 fe80::354c:ba27:ebcc:5d62 prefixlen 64 scopeid 0x20<link>
    ether f8:bc:12:90:45:7e txqueuelen 1000 (Ethernet)
    RX packets 409135 bytes 342188533 (326.3 MiB)
    RX errors 0 dropped 109 overruns 0 frame 0
    TX packets 7862 bytes 474073 (462.9 KiB)
    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 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[student@localhost ~]$ cd karthi79
[student@localhost karthi79]$ sort -r os.txt
Hi hello, how are you?
Good Bye
[student@localhost karthi79]$ sort -n os.txt
Good Bye
Hi hello, how are you?
[student@localhost karthi79]$ sort -m os.txt
Hi hello, how are you?
Good Bye
[student@localhost karthi79]$ grep "h" os.txt
bash: grep: command not found
[student@localhost karthi79]$ grep "h" os.txt
Hi hello, how are you?
[student@localhost karthi79]$ tail os.txt
Hi hello, how are you?
Good Bye
[student@localhost karthi79]$ who;date
student pts/0      2025-01-25 08:12 (:0)
student pts/1      2025-01-25 09:20 (:0)
Sat Jan 25 09:31:14 TST 2025
[student@localhost karthi79]$ who&&date

```

```

inlet 172.16.0.29 netmask 255.255.252.0 broadcast 172.16.0.31
ineto fe80::35dc:ba27:ebcc:5d02 prefixlen 64 scopeid 0x20<link>
ether fa:bc:12:90:45:7e txqueuelen 1000 (Ethernet)
RX packets 409135 bytes 342108532 (326.3 MiB)
RX errors 0 dropped 109 overruns 0 frame 0
TX packets 7862 bytes 474073 (462.9 KiB)
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
ineto ::1 prefixlen 128 scopeid 0x10<host>
loop txqueuelen 1000 (local loopback)
RX packets 0 bytes 0 (0.0 B)
RX errors 0 dropped 0 overruns 0 frame 0
TX packets 0 bytes 0 (0.0 B)
TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

[student@localhost ~]$ cd karthi79
[student@localhost karthi79]$ sort -r os.txt
Hi hello, how are you?
Good Bye
[student@localhost karthi79]$ sort -n os.txt
Good Bye
Hi hello, how are you?
[student@localhost karthi79]$ sort -m os.txt
Hi hello, how are you?
Good Bye
[student@localhost karthi79]$ grep "h" os.txt
bash: grep: command not found
[student@localhost karthi79]$ grep "h" os.txt
Hi hello, how are you?
[student@localhost karthi79]$ tail os.txt
Hi hello, how are you?
Good Bye
[student@localhost karthi79]$ who;date
student pts/0 2025-01-25 08:12 (:0)
student pts/1 2025-01-25 09:20 (:0)
Sat Jan 25 09:31:14 IST 2025
[student@localhost karthi79]$ who&&date
student pts/0 2025-01-25 08:12 (:0)
student pts/1 2025-01-25 09:20 (:0)
Sat Jan 25 09:31:31 IST 2025
[student@localhost karthi79]$

```

RESULT:

Thus, the program of basic Linux commands has been executed and the output has been verified.
2116231801135

Ex. No: 2a
Date: 24/1/25

Shell Script

AIM:

To write a Shell script to display a basic calculator.

PROGRAM:

```
#!/bin/bash

while true; do
    echo "=====
    echo "  Basic Calculator"
    echo "=====
    echo "1. Addition"
    echo "2. Subtraction"
    echo "3. Multiplication"
    echo "4. Division"
    echo "5. Exit"
    echo -n "Choose an option (1-5): "
    read choice

    if [[ $choice -eq 5 ]]; then
        echo "Exiting Calculator. Goodbye!"
        exit 0
    fi

    echo -n "Enter first number: "
    read num1
    echo -n "Enter second number: "
    read num2

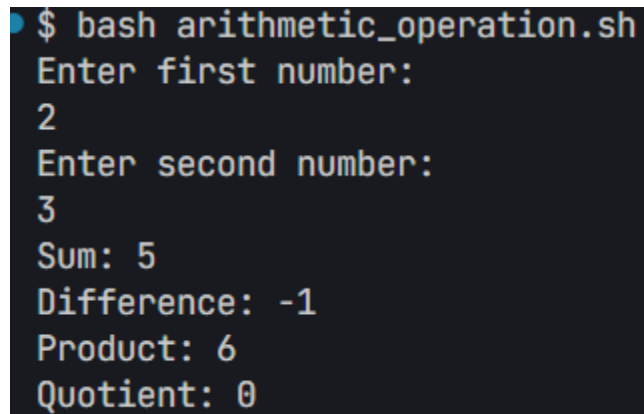
    case $choice in
        1) result=$((num1 + num2))
            echo "Result: $num1 + $num2 = $result"
            ;;
        2) result=$((num1 - num2))
            echo "Result: $num1 - $num2 = $result"
            ;;
        3) result=$((num1 * num2))
            echo "Result: $num1 * $num2 = $result"
            ;;
```

2116231801135

```
4) if [[ $num2 -eq 0 ]]; then
    echo "Error: Division by zero is not allowed!"
else

    result=$(awk "BEGIN {print $num1 / $num2}")
    echo "Result: $num1 / $num2 = $result"
fi
;;
*) echo "Invalid option! Please choose between 1-5."
;;
esac
echo "-----"
echo ""
done
```

OUTPUT:



```
$ bash arithmetic_operation.sh
Enter first number:
2
Enter second number:
3
Sum: 5
Difference: -1
Product: 6
Quotient: 0
```

RESULT:

Thus, the basic calculator program was successfully implemented using shell scripting.

Ex. No: 2b

Date: 24/1/25

Shell Script

AIM:

To write a Shellscrip to test given year is leap or not using conditional statement

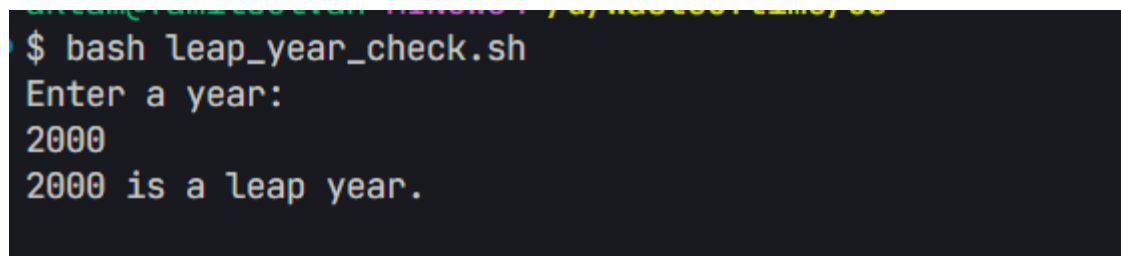
PROGRAM:

```
#!/bin/bash

read -p "Enter year: " year

if (( year % 4 == 0 && year % 100 != 0 )) || (( year % 400 == 0 )); then
    echo "$year is a Leap Year"
else
    echo "$year is not a Leap Year"
fi
```

OUTPUT:



```
$ bash leap_year_check.sh
Enter a year:
2000
2000 is a leap year.
```

RESULT:

Thus, the leap year program was successfully implemented using shell scripting.

Ex. No: 3a
Date: 28/1/25

Shell Script – Reverse of Digit

AIM:

To write a Shell script to reverse a given digit using looping statement.

PROGRAM:

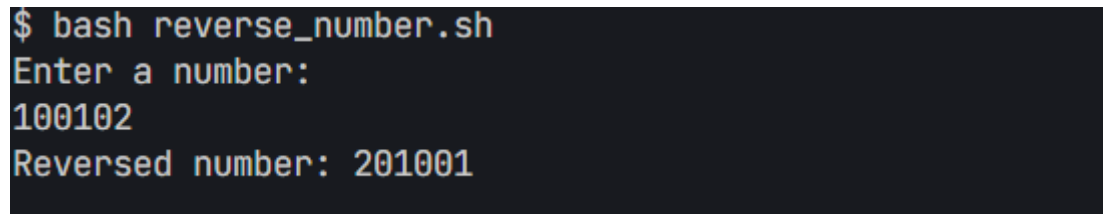
```
#!/bin/bash

read -p "Enter a number: " num

reverse=0
while [ $num -gt 0 ]; do
    digit=$(( num % 10 ))
    reverse=$(( reverse * 10 + digit ))
    num=$(( num / 10 ))
done

echo "Reversed number: $reverse"
```

OUTPUT:



```
$ bash reverse_number.sh
Enter a number:
100102
Reversed number: 201001
```

RESULT:

The shell script to reverse a given digit is successfully implemented.

Ex. No: 3b
Date: 28/1/25

Shell Script – Fibonacci Series

AIM:

To write a Shell script to generate a Fibonacci series using a for loop.

PROGRAM:

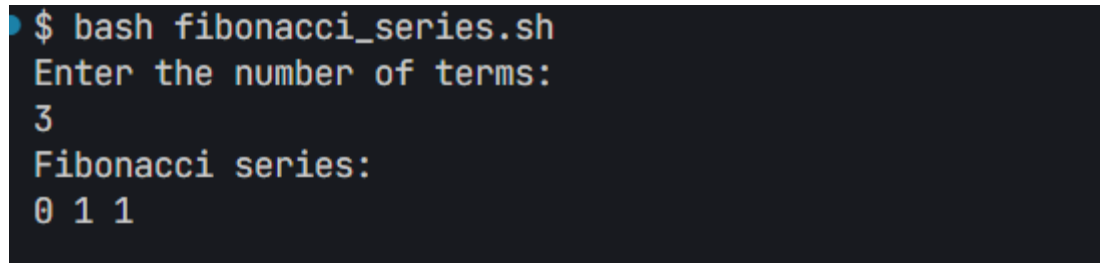
```
#!/bin/bash

read -p "Enter the number of terms: " n
a=0
b=1

echo "Fibonacci Series:"
for (( i=0; i<n; i++ )); do
    echo -n "$a "
    temp=$((a + b))
    a=$b
    b=$temp
done

echo
```

OUTPUT



```
$ bash fibonacci_series.sh
Enter the number of terms:
3
Fibonacci series:
0 1 1
```

RESULT:

The Shell Script to generate the Fibonacci series is successfully implemented.

Ex. No: 4a
Date: 3/2/25

EMPLOYEE AVERAGE PAY

AIM:

To find out the average pay of all employees whose salary is more than 6000 and no. of days worked is more than 4.

ALGORITHM:

1. Create a flat file emp.dat for employees with their name, salary per day and number of days worked and save it.
2. Create an awk script emp.awk
3. For each employee record do
 - a. If the Salary is greater than 6000 and number of days worked is more than 4, then print the name and salary earned
 - b. Compute total pay of employee
4. Print the total number of employees satisfying the criteria and their average pay.

PROGRAM:

```
#!/usr/bin/awk -f
```

```
BEGIN {  
    count = 0;  
    total_pay = 0;  
}  
  
{  
    salary = $2;  
    days = $3;  
  
    if (salary > 6000 && days > 4) {  
        pay = salary * days;  
        print "Employee:", $1, "Total Pay:", pay;  
        total_pay += pay;  
        count++;  
    }  
}  
  
END {  
    if (count > 0) {  
        avg_pay = total_pay / count;  
        print "\nTotal Employees:", count;  
        print "Total Pay:", total_pay;  
        print "Average Pay:", avg_pay;  
    } else {  
        print "No employees satisfy the criteria.";  
    }  
}
```

2116231801135

```
}  
}
```

INPUT:

John 7000 10
Alice 5000 12
Bob 8000 9
Mike 6500 6

OUTPUT:

```
$ gawk -f emp.awk emp.dat  
Employee: John Total Pay: 70000  
Employee: Bob Total Pay: 72000  
Employee: Mike Total Pay: 39000  
  
Total Employees: 3  
Total Pay: 181000  
Average Pay: 60333.3
```

RESULT:

To find the average salary whose salary is above 6000 is successfully implemented.

Ex. No: 4b
Date: 3/2/25

RESULTS OF EXAMINATION

AIM:

To print the pass/fail status of a student in a class.

ALGORITHM:

1. Read the data from file
2. Get a data from each column
3. Compare the all subject marks column
 - a. If marks less than 45 then print Fail
 - b. else print Pass

PROGRAM:

```
//marks.awk
#!/usr/bin/gawk -f
{
    name = $1;
    pass = 1;
    for (i = 2; i <= NF; i++) {
        if ($i < 45) {
            pass = 0;
            break;}
    }
    if(pass) {
        print name, "Pass";
    } else {
        print name, "Fail";}
}
```

INPUT:

```
//marks.dat
John 50 60 45 70 80
Alice 40 55 30 65 75
Bob 80 85 90 78 88
Mike 35 40 50 60 45
```

OUTPUT:

```
$ awk -f emp.awk emp.dat
awk -f pass_fail.awk results.dat
Jane 42000
Alice 56000
Bob 31000
Total employees: 3
Average pay: 43000
Name Pass
Alice Pass
Bob Fail
Charlie Pass
```

RESULT:

To print the Pass/Fail Status of a student in a class is successfully implemented.

Ex. No: 5

Date: 8/2/25

System Calls Programming

AIM:

To experiment system calls using fork(), execlp() and pid() functions.

ALGORITHM:

1. **Start**
2. **Include Header Files**
 - Include stdio.h for input/output functions
 - Include stdlib.h for general utility functions
3. **Variable Declaration**
 - Declare an integer variable pid to store the process ID returned by fork()
4. **Create a New Process**
 - Call the fork() function and assign its return value to pid
 - If fork() returns:
 - -1: Process creation failed
 - 0: This is the **child** process
 - A positive integer: This is the **parent** process
5. **Print Statement Executed by Both Processes**
 - Print: "THIS LINE EXECUTED TWICE"
6. **Check for Process Creation Failure**
 - If pid == -1:
 - Print: "CHILD PROCESS NOT CREATED"
 - Exit the program using exit(0)
7. **Child Process Execution Block**
 - If pid == 0:
 - Print:
 - "Process ID of child: " followed by getpid()
 - "Parent Process ID of child: " followed by getppid()
8. **Parent Process Execution Block**
 - If pid > 0:
 - Print:
 - "Process ID of parent: " followed by getpid()
 - "Parent's Parent Process ID: " followed by getppid()

9. Final Print Statement (Executed by Both Processes)

- Print: objectives

IT CAN BE EXECUTED TWICE

10. End

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    int pid;
    pid = fork();
    printf("This Line Executed Twice\n");

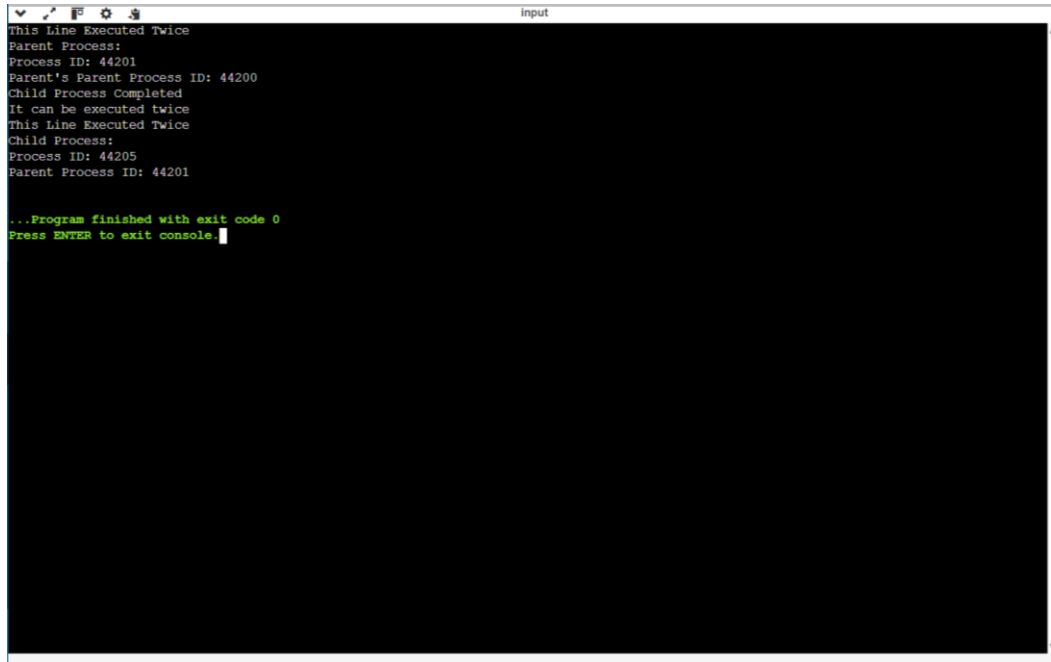
    if (pid < 0) {
        printf("Child Process Not Created\n");
        exit(1);
    }

    if (pid == 0) {
        printf("Child Process:\n");
        printf("Process ID: %d\n", getpid());
        printf("Parent Process ID: %d\n", getppid());
        execlp("/bin/ls", "ls", NULL);
        perror("execlp failed");
        exit(1);
    } else { // Parent process
        printf("Parent Process:\n");
        printf("Process ID: %d\n", getpid());
        printf("Parent's Parent Process ID: %d\n", getppid());
        printf("Child Process Completed\n");
    }

    printf("It can be executed twice\n");

    return 0;
}
```

OUTPUT:

A screenshot of a terminal window titled 'input'. The terminal has a black background with white text. The output shows a parent process (ID: 44201) executing a child process (ID: 44205). The child process prints 'This Line Executed Twice' and 'It can be executed twice'. The parent process prints 'Parent Process:' and 'Child Process Completed'. The terminal ends with a green message: '...Program finished with exit code 0' and 'Press ENTER to exit console.' with a cursor.

```
input
This Line Executed Twice
Parent Process:
Process ID: 44201
Parent's Parent Process ID: 44200
Child Process Completed
It can be executed twice
This Line Executed Twice
Child Process:
Process ID: 44205
Parent Process ID: 44201

...Program finished with exit code 0
Press ENTER to exit console.
```

RESULT:

To Program is implemented using fork(),execlp() and pid() Functions.

Ex. No: 6a

Date: 15/2/25

FIRST COME FIRST SERVE

AIM:

To implement First-come First- serve (FCFS) scheduling technique

ALGORITHM:

1. Get the number of processes from the user.
2. Read the process name and burst time.
3. Calculate the total process time.
4. Calculate the total waiting time and total turnaround time for each process
5. Display the process name & burst time for each process.
6. Display the total waiting time, average waiting time, turnaround time.

PROGRAM:

```
#include <stdio.h>

int main() {
    int pid[15], bt[15], wt[15], n;
    float twt = 0, ttat = 0;

    printf("Enter the number of processes: ");
    scanf("%d", &n);

    printf("Enter process ID of all the processes:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &pid[i]);
    }

    printf("Enter burst time of all the processes:\n");
    for (int i = 0; i < n; i++) {
        scanf("%d", &bt[i]);
    }

    wt[0] = 0;
    // Calculate waiting time for all other processes
    for (int i = 1; i < n; i++) {
        wt[i] = wt[i - 1] + bt[i - 1];
    }

    printf("\nProcess ID\tBurst Time\tWaiting Time\tTurnaround Time\n");
```

2116231801135


```

for (int i = 0; i < n; i++) {
    int tat = bt[i] + wt[i];
    twt += wt[i];
    ttat += tat;

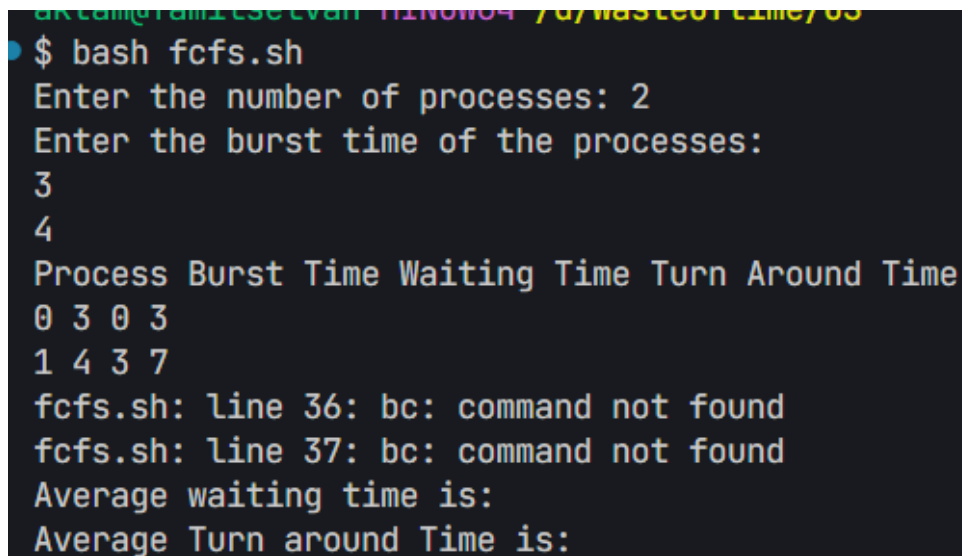
    printf("%d\t%d\t%d\t%d\n", pid[i], bt[i], wt[i], tat);
}

printf("\nAverage waiting time = %.2f\n", twt / n);
printf("Average turnaround time = %.2f\n", ttat / n);

return 0;
}

```

OUTPUT:



```

$ bash fcfs.sh
Enter the number of processes: 2
Enter the burst time of the processes:
3
4
Process Burst Time Waiting Time Turn Around Time
0 3 0 3
1 4 3 7
fcfs.sh: line 36: bc: command not found
fcfs.sh: line 37: bc: command not found
Average waiting time is:
Average Turn around Time is:

```

RESULT:

The Program of first come first serve is successfully implemented.

Ex. No: 6b

Date: 15/2/25

SHORTEST JOB FIRST

AIM:

To implement the Shortest Job First (SJF) scheduling technique

ALGORITHM:

1. Declare the structure and its elements.
2. Get a number of processes as input from the user.
3. Read the process name, arrival time and burst time
4. Initialize waiting time, turnaround time & flag of read processes to zero.
5. Sort based on the burst time of all processes in ascending order.
6. Calculate the waiting time and turnaround time for each process.
7. Calculate the average waiting time and average turnaround time.
8. Display the results.

PROGRAM:

```
#include <stdio.h>
```

```
int main() {
    int A[100][4]; // A[i][0]=PID, A[i][1]=BT, A[i][2]=WT, A[i][3]=TAT
    int i, j, n, total = 0, index, temp;
    float avg_wt, avg_tat;

    printf("Enter number of processes: ");
    scanf("%d", &n);

    printf("Enter Burst Time:\n");
    for (i = 0; i < n; i++) {
        printf("P%d: ", i + 1);
        scanf("%d", &A[i][1]);
        A[i][0] = i + 1; // Assign process ID
    }

    for (i = 0; i < n; i++) {
        index = i;
        for (j = i + 1; j < n; j++) {
            if (A[j][1] < A[index][1])
                index = j;
        }
    }
}
```

```

    temp = A[i][1];
    A[i][1] = A[index][1];
    A[index][1] = temp;

    temp = A[i][0];
    A[i][0] = A[index][0];
    A[index][0] = temp;
}

A[0][2] = 0;
for (i = 1; i < n; i++) {
    A[i][2] = 0;
    for (j = 0; j < i; j++) {
        A[i][2] += A[j][1];
    }
    total += A[i][2];
}
avg_wt = (float) total / n;

total = 0;
printf("\nProcess\tBT\tWT\tTAT\n");
for (i = 0; i < n; i++) {
    A[i][3] = A[i][1] + A[i][2]; // TAT = BT + WT
    total += A[i][3];
    printf("P%d\t%d\t%d\t%d\n", A[i][0], A[i][1], A[i][2], A[i][3]);
}
avg_tat = (float) total / n;

printf("\nAverage Waiting Time = %.2f", avg_wt);
printf("\nAverage Turnaround Time = %.2f\n", avg_tat);

return 0;
}

```

OUTPUT:

```

$ bash sjf.sh
Enter the number of processes: 2
Enter the burst time of the processes:
1
2
Process Burst Time Waiting Time Turn Around Time
1 1 0 1
2 2 1 3

```

RESULT:

The Program Shortest Job First is successfully implemented.

Ex. No: 6c

Date: 16/2/25

PRIORITY SCHEDULING

AIM:

To implement a priority scheduling technique

ALGORITHM:

1. Get the number of processes from the user.
2. Read the process name, burst time and priority of the process.
3. Sort based on burst time of all processes in ascending order based on priority
4. Calculate the total waiting time and total turnaround time for each process
5. Display the process name & burst time for each process.
6. Display the total waiting time, average waiting time, turnaround time.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

void swap(int *a, int *b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

int main() {
    int n;
    printf("Enter number of processes: ");
    scanf("%d", &n);

    int *burst = (int*)malloc(n * sizeof(int));
    int *priority = (int*)malloc(n * sizeof(int));
    int *pid = (int*)malloc(n * sizeof(int));
    int total_wait = 0, total_turnaround = 0;

    for (int i = 0; i < n; i++) {
        printf("Enter Burst Time and Priority for Process %d: ", i + 1);
        scanf("%d %d", &burst[i], &priority[i]);
        pid[i] = i + 1;
    }

    for (int i = 0; i < n - 1; i++) {
        for (int j = i + 1; j < n; j++) {
            if (priority[j] > priority[i]) {
                swap(&priority[i], &priority[j]);
                swap(&burst[i], &burst[j]);
                swap(&pid[i], &pid[j]);
            }
        }
    }
}
```

2116231801135

```

}

int wait_time = 0;
printf("\nProcess  Burst Time  Wait Time  Turnaround Time\n");

for (int i = 0; i < n; i++) {
    int turnaround_time = wait_time + burst[i];
    total_wait += wait_time;
    total_turnaround += turnaround_time;

    printf("P%d      %d      %d      %d\n", pid[i], burst[i], wait_time, turnaround_time);

    wait_time += burst[i];
}

printf("\nAverage Waiting Time: %.2f\n", (float)total_wait / n);
printf("Average Turnaround Time: %.2f\n", (float)total_turnaround / n);

free(burst);
free(priority);
free(pid);

return 0;
}

```

OUTPUT:

```

$ bash priority_scheduling.sh
Enter the number of processes: 2
Enter process name, burst time, and priority (space separated): 2
Enter process name, burst time, and priority (space separated): 1
Process Burst Time Priority Waiting Time Turn Around Time
1      0
2      0 0

```

RESULT:

The Program of Priority scheduling is successfully implemented.

Ex. No: 6d
Date: 16/2/25

ROUND ROBIN SCHEDULING

AIM:

To implement the round-robin (RR) scheduling technique

ALGORITHM:

1. Declare the structure and its elements.
2. Get a number of processes and Time quantum as input from the user.
3. Read the process name, arrival time and burst time
4. Create an array rem_bt[] to keep track of the remaining burst time of processes which is initially copy of bt[] (burst times array)
5. Create another array wt[] to store waiting times of processes. Initialize this array as 0.
6. Initialize time : t = 0
7. Keep traversing all processes while all processes are not done. Do the following for i'th process if it is not done yet.
 - a- If rem_bt[i] > quantum
 - (i) t = t + quantum
 - (ii) bt_rem[i] -= quantum;
 - b- Else // Last cycle for this process
 - (i) t = t + bt_rem[i];
 - (ii) wt[i] = t - bt[i]
 - (iii) bt_rem[i] = 0; // This process is over
8. Calculate the waiting time and turnaround time for each process.
9. Calculate the average waiting time and average turnaround time.
10. Display the results.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

int main() {
    int n, time_quantum;
    printf("Enter number of processes: ");
    scanf("%d", &n);

    int *arrival = (int*)malloc(n * sizeof(int));
    int *burst = (int*)malloc(n * sizeof(int));
    int *remaining = (int*)malloc(n * sizeof(int));
    int wait_time = 0, turnaround_time = 0, total = 0, x = n;

    for (int i = 0; i < n; i++) {
        printf("Enter arrival time and burst time for process %d: ", i + 1);
        scanf("%d %d", &arrival[i], &burst[i]);
        remaining[i] = burst[i];
    }

    2116231801135
```

```

}

printf("Enter time quantum: ");
scanf("%d", &time_quantum);printf("\nProcess\tBurst\tTurnaround\tWaiting\n");

for (int i = 0; x != 0;) {
    if (remaining[i] > 0) {
        if (remaining[i] <= time_quantum) {
            total += remaining[i];
            remaining[i] = 0;
            x--;
            printf("P%d\t%d\t%d\t%d\n", i + 1, burst[i], total - arrival[i], total - arrival[i] - burst[i]);
            wait_time += total - arrival[i] - burst[i];
            turnaround_time += total - arrival[i];
        } else {
            remaining[i] -= time_quantum;
            total += time_quantum;
        }
    }
}

i = (i + 1) % n;
}

printf("\nAverage Waiting Time: %.2f", (float)wait_time / n);
printf("\nAverage Turnaround Time: %.2f\n", (float)turnaround_time / n);

free(arrival);
free(burst);
free(remaining);

return 0;
}

```

OUTPUT:

```

$ bash round_robin.sh
Enter the number of processes: 2
Enter process name and burst time (space separated): 1
Enter process name and burst time (space separated): 1
Enter Time Quantum: 2
round_robin.sh: line 31: [: -gt: unary operator expected
round_robin.sh: line 31: [: -gt: unary operator expected
Process Burst Time Waiting Time Turn Around Time
1 0 0
1 0 0
round_robin.sh: line 62: bc: command not found
round_robin.sh: line 63: bc: command not found
Average waiting time is:
Average Turn Around Time is:

```


RESULT:

The Program of Round Robin Scheduling is successfully implemented.

Ex. No: 7

Date: 22/2/25

IPC USING SHARED MEMORY

AIM:

To write a C program to do Inter-Process Communication (IPC) using shared memory between the sender process and the receiver process.

ALGORITHM:

sender

1. Set the size of the shared memory segment
2. Allocate the shared memory segment using shmget
3. Attach the shared memory segment using shmat
4. Write a string to the shared memory segment using sprintf
5. Set delay using sleep
6. Detach shared memory segment using shmdt

receiver

1. Set the size of the shared memory segment
2. Allocate the shared memory segment using shmget
3. Attach the shared memory segment using shmat
4. Print the shared memory contents sent by the sender process.
5. Detach shared memory segment using shmdt

PROGRAM:

SENDER

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>

#define SHMSIZE 1024

typedef struct {
    int ready;
    char message[SHMSIZE];
} SharedMemory;

2116231801135
```

```

int main() {
    key_t key = ftok("sender.c", 65);
    int shmid;
    SharedMemory *shm;

    shmid = shmget(key, sizeof(SharedMemory), 0666 | IPC_CREAT);
    if (shmid == -1) {
        perror("shmget failed");
        exit(1);
    }

    shm = (SharedMemory *)shmat(shmid, NULL, 0);
    if (shm == (SharedMemory *)-1) {
        perror("shmat failed");
        exit(1);
    }

    printf("Sender: Enter a message to send to receiver: ");
    fgets(shm->message, SHMSIZE, stdin);

    shm->message[strcspn(shm->message, "\n")] = '\0';

    shm->ready = 1;

    sleep(5);

    if (shmdt(shm) == -1) {
        perror("shmdt failed");
        exit(1);
    }

    return 0;
}

```

RECEIVER

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <unistd.h>

#define SHMSIZE 1024

typedef struct {
    int ready;
    char message[SHMSIZE];
} SharedMemory;

2116231801135

```

```

int main() {
    key_t key = ftok("sender.c", 65);
    int shmid;
    SharedMemory *shm;

    shmid = shmget(key, sizeof(SharedMemory), 0666 | IPC_CREAT);
    if (shmid == -1) {
        perror("shmget failed");
        exit(1);
    }

    shm = (SharedMemory *)shmat(shmid, NULL, 0);
    if (shm == (SharedMemory *)-1) {
        perror("shmat failed");
        exit(1);
    }

    while (shm->ready == 0) {
        sleep(1);
    }

    printf("Receiver: Message received from sender: %s\n", shm->message);

    if (shmdt(shm) == -1) {
        perror("shmdt failed");
        exit(1);
    }

    if (shmctl(shmid, IPC_RMID, NULL) == -1) {
        perror("shmctl failed");
        exit(1);
    }

    return 0;
}

```

OUTPUT:

```

sender: Enter a message to send to receiver: Hi helloo!...

```

```

receiver: Message received from sender: Hi helloo!...

```

RESULT:

The IPC Program with Shared Memory is Successfully Implemented.

2116231801135

Ex. No: 8

Date: 22/2/25

PRODUCER CONSUMER USING SEMAPHORES

AIM:

To write a program to implement solutions to producer consumer problem using semaphores.

ALGORITHM:

1. Initialize semaphore empty, full and mutex.
2. Create two threads- the producer thread and the consumer thread.
3. Wait for target thread termination.
4. Call sem_wait on empty semaphore followed by mutex semaphore before entry into critical section.
5. Produce/Consume the item in the critical section.
6. Call sem_post on mutex semaphore followed by full semaphore
7. before exiting the critical section.
8. Allow the other thread to enter its critical section.
9. Terminate after looping ten times in producer and consumer Threads each.

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>

int mutex = 1;
int full = 0;
int empty = 10, x = 0;

pthread_mutex_t lock;

void *producer(void *arg)
{
    pthread_mutex_lock(&lock);

    if (empty != 0) {
        --mutex;
        ++full;
        --empty;
        x++;
        printf("\nProducer produces item %d\n", x);
        ++mutex;
    } else {
        printf("Buffer is full!\n");
    }

    pthread_mutex_unlock(&lock);
    return NULL;
}

2116231801135
```

```

}

void *consumer(void *arg)
{
    pthread_mutex_lock(&lock);

    if (full != 0) {
        --mutex;
        --full;
        ++empty;
        printf("\nConsumer consumes item %d\n", x);
        x--;
        ++mutex;
    } else {
        printf("Buffer is empty!\n");
    }

    pthread_mutex_unlock(&lock);
    return NULL;
}

int main()
{
    int n, i;
    pthread_t prod_thread, cons_thread;

    pthread_mutex_init(&lock, NULL);

    printf("\n1. Press 1 for Producer"
           "\n2. Press 2 for Consumer"
           "\n3. Press 3 for Exit\n");

    for (i = 1; i > 0; i++) {
        printf("\nEnter your choice: ");
        scanf("%d", &n);

        switch (n) {
            case 1:
                if (mutex == 1 && empty != 0) {
                    pthread_create(&prod_thread, NULL, producer, NULL);
                    pthread_join(prod_thread, NULL);
                } else {
                    printf("Buffer is full!\n");
                }
                break;

            case 2:
                if (mutex == 1 && full != 0) {
                    pthread_create(&cons_thread, NULL, consumer, NULL);
                    pthread_join(cons_thread, NULL);
                } else {
                    printf("Buffer is empty!\n");
                }
            }
        }
    }
}

```

```

    }
    break;

case 3:
    pthread_mutex_destroy(&lock);
    exit(0);
    break;
default:
    printf("Invalid choice! Please enter a valid option.\n");
}
}

return 0;
}

```

OUTPUT:

```

input
1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice: 1
Producer produces item 1
Enter your choice: 2
Consumer consumes item 1
Enter your choice: 2
Buffer is empty!
Enter your choice: 1
Producer produces item 1
Enter your choice: 1
Producer produces item 2
Enter your choice: 1
Producer produces item 3
Enter your choice: 1
Producer produces item 4
Enter your choice: 1
Buffer is full!
Enter your choice:

```

RESULT:

The Producer Consumer Program using Semaphore is Successfully Implemented.

2116231801135

Ex. No.: 9

Date: 22/2/25

DEADLOCK AVOIDANCE

AIM:

To find out a safe sequence using Banker's algorithm for deadlock avoidance.

ALGORITHM:

1. Initialize work=available and finish[i]=false for all values of i
2. Find an i such that both:
finish[i]=false and Needi<= work
3. If no such i exists go to step 6
4. Compute work=work+allocationi
5. Assign finish[i] to true and go to step 2
6. If finish[i]==true for all i, then print safe sequence
7. Else print there is no safe sequence.

PROGRAM:

```
#include <stdio.h>
#include <stdbool.h>

#define MAX 10

void findSafeSequence(int n, int m, int available[], int max[][MAX], int allocation[][MAX]) {
    int work[MAX], finish[MAX] = {0}, safeSeq[MAX], need[MAX][MAX];
    for (int i = 0; i < m; i++) work[i] = available[i];
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++)
            need[i][j] = max[i][j] - allocation[i][j];

    int count = 0;
    while (count < n) {
        bool found = false;
        for (int i = 0; i < n; i++) {
            if (!finish[i]) {
                bool canAllocate = true;
                for (int j = 0; j < m; j++)
                    if (need[i][j] > work[j]) { canAllocate = false; break; }
                if (canAllocate) {
                    for (int j = 0; j < m; j++) work[j] += allocation[i][j];
                    safeSeq[count++] = i;
                    finish[i] = 1;
                    found = true;
                }
            }
        }
    }
}
```

2116231801135

```

        if (!found) { printf("No safe sequence.\n"); return; }
    }
    printf("Safe sequence: ");
    for (int i = 0; i < n; i++) printf("P%d ", safeSeq[i]);
    printf("\n");
}

int main() {
    int n, m, available[MAX], max[MAX][MAX], allocation[MAX][MAX];

    printf("Enter processes and resources: ");
    scanf("%d %d", &n, &m);
    while (getchar() != '\n');

    printf("Enter available resources: ");
    for (int i = 0; i < m; i++) scanf("%d", &available[i]);
    while (getchar() != '\n');

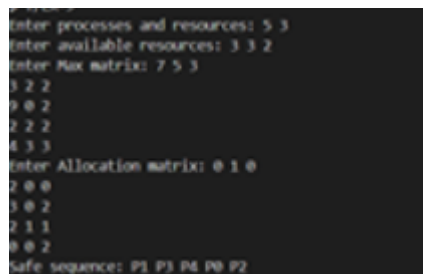
    printf("Enter Max matrix: \n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++) scanf("%d", &max[i][j]);
    while (getchar() != '\n');

    printf("Enter Allocation matrix: \n");
    for (int i = 0; i < n; i++)
        for (int j = 0; j < m; j++) scanf("%d", &allocation[i][j]);
    while (getchar() != '\n');

    findSafeSequence(n, m, available, max, allocation);
    return 0;
}

```

OUTPUT:



```

Enter processes and resources: 5 3
Enter available resources: 3 3 2
Enter Max matrix: 7 5 3
3 2 2
9 0 2
2 2 2
4 3 3
Enter Allocation matrix: 0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Safe sequence: P1 P3 P4 P0 P2

```

RESULT:

The Safe Sequence is found using Banker's Algorithm for Deadlock Avoidance.

Ex. No: 10a

Date: 7/3/25

BEST FIT

2116231801135

AIM:

To implement Best Fit memory allocation technique using Python.

ALGORITHM:

1. Input memory blocks and processes with sizes
2. Initialize all memory blocks as free.
3. Start by picking each process and find the minimum block size that can be assigned to current process
4. If found then assign it to the current process.
5. If not found then leave that process and keep checking the further processes.

PROGRAM:

```
def best_fit(blocks, processes):
    allocation = [-1] * len(processes)

    for i in range(len(processes)):
        best_index = -1

        for j in range(len(blocks)):
            if blocks[j] >= processes[i]:
                if best_index == -1 or blocks[j] < blocks[best_index]:
                    best_index = j

        if best_index != -1:
            allocation[i] = best_index
            blocks[best_index] -= processes[i]

    print("\nProcess No.\tProcess Size\tBlock No.")
    for i in range(len(processes)):
        print(f"{i + 1}\t\t{processes[i]}\t\t{allocation[i] + 1 if allocation[i] != -1 else 'Not Allocated'}")

if __name__ == "__main__":
    num_blocks = int(input("Enter number of memory blocks: "))
    blocks = list(map(int, input(f"Enter sizes of {num_blocks} memory blocks (space-separated): ").split()))

    num_processes = int(input("\nEnter number of processes: "))
    processes = list(map(int, input(f"Enter sizes of {num_processes} processes (space-separated): ").split()))

    best_fit(blocks, processes)
```

OUTPUT:

```
Enter number of processes: 4
Enter sizes of 4 processes (space-separated): 212 417 112 426

Process No.   Process Size   Block No.
1             212         4
2             417         2
3             112         3
4             426         5
```

RESULT:

Thus, the Best Fit Memory allocation technique is implemented successfully using Python.

Ex. No: 10b

Date: 7/3/25

FIRST FIT**AIM:**

To write a C program for the implementation of memory allocation methods for a fixed

2116231801135

partition using the first fit.

ALGORITHM:

1. Define the max as 25.
2. Declare the variable frag[max],b[max],f[max],i,j, nb,nf, temp, highest=0, bf[max],ff[max].
3. Get the number of blocks, files, size of the blocks using a for loop.
4. In for loop check bf[j]!=1, if so temp=b[j]-f[i]
5. Check the highest.

PROGRAM:

```
#include <stdio.h>
#define MAX 25

int main() {
    int frag[MAX], b[MAX], f[MAX], i, j, nb, nf, temp;
    static int bf[MAX], ff[MAX];

    printf("Enter the number of blocks: ");
    scanf("%d", &nb);

    printf("Enter the number of files: ");
    scanf("%d", &nf);

    printf("Enter the size of the blocks:\n");
    for (i = 0; i < nb; i++) {
        printf("Block %d: ", i + 1);
        scanf("%d", &b[i]);
    }

    printf("Enter the size of the files:\n");
    for (i = 0; i < nf; i++) {
        printf("File %d: ", i + 1);
        scanf("%d", &f[i]);
    }

    for (i = 0; i < nf; i++) {
        for (j = 0; j < nb; j++) {
            if (bf[j] != 1) {
                temp = b[j] - f[i];
                if (temp >= 0) {
                    ff[i] = j;
                    bf[j] = 1;
                    frag[i] = temp;
                    break;
                }
            }
        }
    }
}
```

```

printf("\nFile No.\tFile Size\tBlock No.\tBlock Size\tFragment\n");
for (i = 0; i < nf; i++) {
    if (bf[ff[i]] == 1)
        printf("%d\t%d\t%d\t%d\t%d\n", i + 1, f[i], ff[i] + 1, b[ff[i]], frag[i]);
    else
        printf("%d\t%d\t\t\t\tNot Allocated\n", i + 1, f[i]);
}

return 0;
}

```

OUTPUT:

```

Enter the number of blocks: 5
Enter the number of files: 4
Enter the size of the blocks:
Block 1: 100
Block 2: 500
Block 3: 200
Block 4: 300
Block 5: 600
Enter the size of the files:
File 1: 232
File 2: 437
File 3: 132
File 4: 426

File No.      File Size      Block No.      Block Size      Fragment
1             232            2              100             288
2             437            5              600             183
3             132            3              200              88
4             426            Not Allocated

```

RESULT:

Thus, the First Fit allocation technique is implemented successfully using C.

Ex. No: 11a

Date: 21/3/25

FIFO PAGE REPLACEMENT

AIM:

To find out the number of page faults that occur using the First-in First-out (FIFO) page replacement technique.

ALGORITHM:

1. Declare the size with respect to page length

2116231801135

2. Check the need for replacement from the page to memory
3. Check the need for replacement from the old page to the new page in memory
4. Form a queue to hold all pages
5. Insert the page required memory into the queue
6. Check for bad replacement and page fault
7. Get the number of processes to be inserted
8. Display the values.

PROGRAM:

```
def fifo_page_replacement(pages, frame_size):
    frames = []
    page_faults = 0
    front = 0

    print("\nPage Replacement Process:")

    for page in pages:
        if page not in frames:
            if len(frames) < frame_size:
                frames.append(page)
            else:
                frames[front] = page
                front = (front + 1) % frame_size
            page_faults += 1
            print(f"Page {page} => {frames} *Page Fault*")
        else:
            print(f"Page {page} => {frames}")

    print(f"\nTotal Page Faults = {page_faults}")

if __name__ == "__main__":
    n = int(input("Enter the number of pages: "))
    pages = []
    print("Enter the page numbers one by one:")
    for i in range(n):
        page = int(input(f"Page {i+1}: "))
        pages.append(page)
    frame_size = int(input("Enter the number of frames: "))

    fifo_page_replacement(pages, frame_size)
```

OUTPUT:


```

$ python Ex-11a.py
Enter the number of pages: 5
Enter the page numbers one by one:
Page 1: 1
Page 2: 2
Page 3: 3
Page 4: 4
Page 5: 5
Enter the number of frames: 3

Page Replacement Process:
Page 1 => [1] "Page Fault"
Page 2 => [1, 2] "Page Fault"
Page 3 => [1, 2, 3] "Page Fault"
Page 4 => [4, 2, 3] "Page Fault"
Page 5 => [4, 5, 3] "Page Fault"

Total Page Faults = 5

```

RESULT:

The Fifo Page Replacement is Successfully Implemented using Python.

Ex. No: 11b

Exp 11 b-LRU

Date: 25/3/25

LRU

AIM:

To write a C program to implement LRU page replacement algorithm.

ALGORITHM:

2116231801135

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least recently used page by counter value
7. Stack them according the selection.
8. Display the values
9. Stop the process

PROGRAM:

```
#include <stdio.h>
```

```
int main() {
    int pages[50], frames[10], counter[10];
    int n, frameSize, i, j, k, flag, least, time = 0, faults = 0;
    printf("Enter the number of frames: ");
    scanf("%d", &frameSize);

    printf("Enter the number of pages: ");
    scanf("%d", &n);

    printf("Enter the page reference string: ");
    for(i = 0; i < n; i++) {
        scanf("%d", &pages[i]);
    }

    for(i = 0; i < frameSize; i++) {
        frames[i] = -1;
        counter[i] = 0;
    }

    for(i = 0; i < n; i++) {
        flag = 0;

        for(j = 0; j < frameSize; j++) {
            if(frames[j] == pages[i]) {
                counter[j] = ++time;
                flag = 1;
                break;
            }
        }

        if(flag == 0) {
            int pos = -1, min = 9999;

            for(j = 0; j < frameSize; j++) {
                if(frames[j] == -1) {
                    pos = j;
                    break;
                } else if(counter[j] < min) {
```

```

        min = counter[j];
        pos = j;
    }
}

frames[pos] = pages[i];
counter[pos] = ++time;
faults++;
}
printf("Frames after inserting %d: ", pages[i]);
for(k = 0; k < frameSize; k++) {
    if(frames[k] != -1)
        printf("%d ", frames[k]);
    else
        printf("- ");
}
printf("\n");
}

printf("\nTotal Page Faults: %d\n", faults);
return 0;
}

```

OUTPUT:

```

$ bash lru_page.sh
Enter number of frames: 2
Enter number of pages: 1
Enter page reference string (space-separated): 3

Page Replacement Process:
Page 3 -> [ 3 - ] (Page Fault)

Total Page Faults: 1
lru_page.sh: line 106: bc: command not found
Hit Ratio: %
lru_page.sh: line 108: bc: command not found
Miss Ratio: %

```

RESULT:

The LRU Program is Successfully Implemented using C.

Ex. No: 11c
Date: 25/3/25

Optimal

AIM:

To write a c program to implement the Optimal page replacement algorithm

ALGORITHM:

1. Start the process
2. Declare the size
3. Get the number of pages to be inserted
4. Get the value
5. Declare counter and stack
6. Select the least frequently used page by counter value.
7. Stack them according the selection.
8. Display the values
9. Stop the process

PROGRAM:

```
#include <stdio.h>
#include <stdlib.h>

int isInFrame(int frame[], int count, int page) {
    for (int i = 0; i < count; i++)
        if (frame[i] == page) return 1;
    return 0;
}

int predict(int pages[], int frame[], int n, int index, int count) {
    int farthest = index, res = -1;
    for (int i = 0; i < count; i++) {
        int j;
        for (j = index; j < n; j++) {
            if (frame[i] == pages[j]) {
                if (j > farthest) {
                    farthest = j;
                    res = i;
                }
                break;
            }
        }
        if (j == n) return i; // If page not found in future
    }
    return (res == -1) ? 0 : res;
}

int main() {
    int n, frameCount, pageFaults = 0, filled = 0;
```

2116231801135

```

printf("Enter number of pages: ");
scanf("%d", &n);
int* pages = malloc(n * sizeof(int));

printf("Enter the page numbers:\n");
for (int i = 0; i < n; i++)
    scanf("%d", &pages[i]);

printf("Enter number of frames: ");
scanf("%d", &frameCount);
int* frame = malloc(frameCount * sizeof(int));
for (int i = 0; i < frameCount; i++)
    frame[i] = -1;

for (int i = 0; i < n; i++) {
    if (!isInFrame(frame, frameCount, pages[i])) {
        if (filled < frameCount)
            frame[filled++] = pages[i];
        else
            frame[predict(pages, frame, n, i, frameCount)] = pages[i];
        pageFaults++;
    }

    printf("Frame: ");
    for (int j = 0; j < frameCount; j++)
        frame[j] == -1 ? printf("- ") : printf("%d ", frame[j]);
    printf("\n");
}

printf("\nTotal Page Faults = %d\n", pageFaults);
free(pages);
free(frame);
return 0;

```

OUTPUT:

```
$ bash optimal_page.sh
Enter number of frames: 1
Enter number of pages: 1
Enter page reference string (space-separated): 1

Page Replacement Process:
Page 1 -> [ 1 ] (Page Fault)

Total Page Faults: 1
optimal_page.sh: line 98: bc: command not found
Hit Ratio: %
optimal_page.sh: line 100: bc: command not found
Miss Ratio: %
```

RESULT:

The Optimal page replacement Program is Successfully Implemented using C.

RESULT:

The Optimal page replacement Program is Successfully Implemented using C.

Ex. No: 12
Date: 1/4/25

File Organization Technique- Single- and Two-level directory

AIM:

To implement File Organization Structures in C are

- a. Single Level Directory
- b. Two-Level Directory
- c. Hierarchical Directory Structure
- d. Directed Acyclic Graph Structure

A. SINGLE LEVEL DIRECTORY

ALGORITHM:

1. Start
2. Declare the number, names and size of the directories and file names. 3. Get the values for the declared variables.
4. Display the files that are available in the directories.
5. Stop.

PROGRAM:

```
#include <stdio.h>
#include <string.h>

struct File {
    char name[20];
};

int main() {
    int n, i;
    struct File files[10];

    printf("Enter the number of files: ");
    scanf("%d", &n);

    if (n <= 0 || n > 10) {
        printf("Please enter a valid number of files (1-10).\n");
        return 1;
    }

    for (i = 0; i < n; i++) {
        printf("Enter the file %d: ", i + 1);
```

2116231801135

```

        scanf("%s", files[i].name);
    }

    printf("\n\nRoot Directory\n");
    printf("\n");

    for (i = 0; i < n; i++) {
        printf("|-- %s\n", files[i].name);
    }

    return 0;
}

```

OUTPUT:

```

Single Level Directory Operations
1. Create File
2. List Files
3. Delete File
4. View File
5. Exit
Enter choice: 1
Enter file name: 2
Enter file content: Hi hellow
File created successfully

Single Level Directory Operations
1. Create File
2. List Files
3. Delete File
4. View File
5. Exit
Enter choice: █

```

B. TWO-LEVEL DIRECTORY STRUCTURE

ALGORITHM:

1. Start
2. Declare the number, names and size of the directories and subdirectories and file names.
3. Get the values for the declared variables.
4. Display the files that are available in the directories and subdirectories. 5. Stop.

PROGRAM:

2116231801135

```

#include <stdio.h>
    Implemented using C.
#include <string.h>

struct File {
    char name[20];
};

struct SubDirectory {
    char name[20];
    struct File files[10];
    int fileCount;
};

struct Directory {
    char name[20];
    struct SubDirectory subDirs[10];
    int subDirCount;
};

int main() {
    struct Directory dir;
    int i, j;

    printf("Enter root directory name: ");
    scanf("%s", dir.name);

    printf("How many subdirectories in '%s'? ", dir.name);
    scanf("%d", &dir.subDirCount);

    for (i = 0; i < dir.subDirCount; i++) {
        printf("\nEnter name of subdirectory %d under '%s': ", i + 1, dir.name);
        scanf("%s", dir.subDirs[i].name);

        printf("How many files in '%s'? ", dir.subDirs[i].name);
        scanf("%d", &dir.subDirs[i].fileCount);

        for (j = 0; j < dir.subDirs[i].fileCount; j++) {
            printf("Enter file %d in '%s': ", j + 1, dir.subDirs[i].name);
            scanf("%s", dir.subDirs[i].files[j].name);
        }
    }

    printf("\nDirectory Structure:\n");
    printf("NULL\n");
    printf("|__ %s\n", dir.name);

    for (i = 0; i < dir.subDirCount; i++) {
        printf("    |__ %s\n", dir.subDirs[i].name);
        for (j = 0; j < dir.subDirs[i].fileCount; j++) {
            printf("        |__ %s\n", dir.subDirs[i].files[j].name);
        }
    }
}

```

2116231801135

```
    }  
}  
  
return 0;  
}
```

OUTPUT:

```
Single Level Directory Operations  
1. Create File  
2. List Files  
3. Delete File  
4. View File  
5. Exit  
Enter choice: 1  
Enter file name: 2  
Enter file content: Hi hellow  
File created successfully  
  
Single Level Directory Operations  
1. Create File  
2. List Files  
3. Delete File  
4. View File  
5. Exit  
Enter choice: █
```

RESULT:

2116231801135

The File Organization Technique-Single and Two-Level Directory Program is Successfully Implemented using C.