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

DEPARTMENTOFCOMPUTERSCIENCEANDENGINEERINGLABMANUAL

CS23431–OPERATINGSYSTEMS

(REGULATION 2023)

RAJALAKSHMIENGINEERINGCOLLEGE
Thandalam,Chennai-602015

Name: . . . . . SRI RAM G P . . . . . .

Register No. : . ... .231501511 . . ..

Year/Branch/Section:.....II/B.Tech/AIML-‘FB’.....

Semester: . . . . . .4. . . .

AcademicYear: .. .. 2024-2025. .. . .

INDEX

8	7/3/25	ProducerConsumerusingSemaphores	60-62	
9	19/3/25	BankersDeadlockAvoidancealgorithms	63-66	

10a	19/3/25	BestFit	67-68	
10b	19/3/25	First Fit	69-71	
11a	26/3/25	FIFO	72-77	
11b	26/3/25	LRU	78-81	
11c	26/3/25	Optimal	82-85	
12	28/3/25	FileOrganizationTechnique-singleandTwo level directory	86-96	

Ex No: 1a)

Date:24/1/25

INSTALLATIONANDCONFIGURATIONOFLINUX

Aim:

ToinstallandconfigureLinuxoperatingsysteminaVirtualMachine.Installation/Configuration Steps:

1. Installtherequiredpackagesforvirtualization
dnf install xen virt-manager qemu libvirt

2. Configure xend to start up on boot
systemctlenablevirt-manager.service

3. Rebootthemachine
Reboot

4. CreateVirtualmachinebyfirstrunningvirt-manager
virt-manager &

5. Click on File and then click to connect to localhost

6. Inthebasemenu,rightclickonthelocalhost(QEMU)tocreateanewVM7.SelectLinuxISO image

8. Choose puppy-linux.iso then kernel version

9. Select CPU and RAM limits

10. Create default disk image to 8 GB

11. Click finish for creating the new VM with Puppy Linux

Output:

Result :

Thus,installationand configuration of linux is done successfully.

Ex No: 1b)
Date:24/1/25

BASIC LINUX COMMANDS

GENERAL PURPOSE COMMANDS

1. The ‘date’ command:

Thedatecommanddisplaysthecurrentdatewithdayofweek,month,day,time(24hoursclock)and the year.

SYNTAX:$date

The date command can also be used with following format.

Format	Purpos	Example
+%m	Todisplayonlymonth	$date+%m
+%h	Todisplaymonthname	$date+%h
+%d	Todisplaydayofmonth	$date+%d
+%y	Todisplaylasttwodigitsoftheyear	$date+%y
+%H	TodisplayHours	$date+%H
+%M	TodisplayMinutes	$date+%M
+%S	TodisplaySeconds	$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:

Thecal command displays the specified month or year calendar.

SYNTAX:$cal[month][year]

EXAMPLE: $ cal Jan 2012

4. The ‘bc’ command:

Unixoffers anonline calculatorand can beinvoked bythe command bc.

SYNTAX: $ bc

EXAMPLE:bc–l

16/4

5/2

5. The ‘who’ command

Thewhocommandisusedtodisplaythedataaboutalltheuserswhoarecurrentlyloggedinto the system.

SYNTAX:$who

6. The ‘who am i’ command

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

SYNTAX:$whoami

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,whichdisplaysinformationaboutprocessthatarealivewhenyourunthe command.

'ps;'produces a snapshot of machine activity.

SYNTAX: $ ps

EXAMPLE:$ps

$ ps –e

$ps -aux

12. The ‘uname’ command

Theunamecommandisusedtodisplayrelevantdetailsabouttheoperatingsystemonthe 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

DIRECTORYCOMMANDS

1. The ‘pwd’ command:

Thepwd(printworkingdirectory)commanddisplaysthecurrentworkingdirectory.SYNTAX:

$ pwd

2. The‘mkdir’command:

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

SYNTAX:$mkdirdirname

EXAMPLE: $ mkdir receee

3. The‘rmdir’command:

Thermdirisusedtoremoveadirectoryfromthedisk.Beforeremovingadirectory,the directory must be empty (no files and directories).

SYNTAX:$rmdirdirname

EXAMPLE: $ rmdir receee

4. The ‘cd’ command:

Thecd command is used to move from one directory to another.

SYNTAX:$cddirname

EXAMPLE: $ cd receee

5. The ‘ls’ command:

Thels command displays the list of files in the current working directory.

SYNTAX:$ls

EXAMPLE: $ ls

$ ls –l

$ ls –a

FILE HANDLING COMMANDS

1. The ‘cat’ command:

   Thecatcommandisusedtocreateafile.

   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:$catfilename

3. The ‘cp’ command:

   Thecpcommandisusedtocopythecontentsofonefiletoanotherandcopiesthefilefromone place to another.

   SYNTAX:$cpoldfilenewfile

   EXAMPLE: $ cp cse ece

4. The ‘rm’ command:

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

   SYNTAX:$rmfilename

   EXAMPLE: $ rm rec

   $ rm –f rec

   Useoption–frtodeleterecursivelythecontentsofthedirectoryanditssubdirectories.5.The ‘mv’ command:

   Themvcommandisusedtomoveafilefromoneplacetoanother.Itremovesaspecifiedfile from its original location and places it in specified location.

   SYNTAX:$mvoldfilenewfile

   EXAMPLE: $ mv cse eee

6. The ‘file’ command:

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

   SYNTAX:$filefilename

   EXAMPLE: $ file receee

7. The ‘wc’ command:

Thewc 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:

Thelscommandliststhefilesontheterminal(screen).Usingtheredirectionoperator‘>’wecan 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 mechanismbywhichtheoutputofonecommandcanbechanneledintotheinputofanother command.SYNTAX: $ command1 | command2

EXAMPLE: $ who | wc -l

10. The ‘tee’ command:

Whileusingpipes,wehavenotseenanyoutputfromacommandthatgetspipedintoanother command.Tosavetheoutput,whichisproducedinthemiddleofapipe,theteecommandis 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 othercharactersinUnix.Theshellunderstandsandinterpretsthesemetacharactersinaspecial 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-Displaysthefileswhicharehaving‘kkk’,fromthesecondcharactersirrespectiveof the first character.

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

$ls[!a-m]–Listsallfilesotherthanfileswhosenamesbeginsalphabetsfrom‘a’to‘m’12. The ‘File permissions’ command:

File permission is the way of controlling the accessibility of file for each of three usersnamely

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.

Firstthreebits	Ownerofthefile
Nextthree bits	Grouptowhichownerofthefilebelongs
Lastthree bits	Others

EXAMPLE:

$ ls college

-rwxr-xr--1Lakstd1525jan1012:10college 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 Isthe Group Owner of the file.

r--Indicatesreadpermissionsforothers.

13. The 'chmod' command:

Thechmodcommandisusedtosettheread,writeandexecutepermissionsforallcategoriesof users for file.

SYNTAX:

$ chmod category operation permission file

Categor	Operation	permission
u-users	+assign	r-read

g-group	-Remove	w-write
o-others	=assignabsolutely	x-execute
a-all		

EXAMPLE:

$ chmod u –wx college

Removes write & execute permission for users for ‘college’ file.

$chmodu+rw,g+rwcollege

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

$ chmod g=wx college

Assignsabsolutepermissionforgroupsofallread,writeandexecutepermissionsfor‘college’ file.

14. The ‘Octal Notations’ command:

Thefilepermissionscanbechangedusingoctalnotationsalso.Theoctalnotationsforfile permission are

Readpermission	4
Writepermissior	2

EXAMPLE:

$ chmod 761 college

Execute permission	1

Assignsallpermissiontotheowner,readandwritepermissionstothegroupandonlyexecutable permission to the others for ‘college’ file.

GROUPING COMMANDS

1. The ‘semicolon’ command:

Thesemicolon(;)commandisusedtoseparatemultiplecommandsatthecommandline.
SYNTAX: $ command1;command2;command3… .....;commandn

EXAMPLE: $ who;date

2. The '&&' operator:

The'&&'operatorsignifiesthelogicalANDoperationinbetweentwoormorevalidUnix commands.It means that only if the first command is successfully executed, then the next commandwill 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.Itmeans,thatonlyifthefirstcommandwillhappentobeunsuccessfully,itwill continueto execute next commands.

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

EXAPLE: $ who || date

FILTERS

1. The head filter

It displays the first ten lines of a file.

SYNTAX:$headfilename

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:$tailfilename

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 thestandard input and displaythoselinesonthestandardoutput.“Grep”standsfor“globalsearchforregularexpression.”

SYNTAX:$grep[pattern][file_name]
EXAMPLE: $ cat> student

Aruncse

Ramece

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:$sortfilename

EXAMPLE:$sortcollege

OPTIONS:

Command	Purpose
Sort–rcollege	Sortsanddisplaysthefilecontentsinreverseorder
Sort–ccollege	Checkifthefileissorted
Sort–ncollege	Sortsnumerically
Sort–mcollege	Sortsnumericallyinreverseorder

Sort–ucollege	Removeduplicaterecords
Sort–lcollege	Skipthecolumnwith+1(one)option.Sortsaccordingto second column

6. The ‘nl’ command:

Thenlfilteraddslinesnumberstoafileanditdisplaysthefileandnotprovidesaccesstoeditbut simply displays the contents on the screen.

SYNTAX:$nlfilename

EXAMPLE: $ nl college

7. The ‘cut’ command:

Wecanselectspecifiedfieldsfromalineoftextusingcutcommand.

SYNTAX:$cut-cfilename

EXAMPLE:$cut-ccollege

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

totalused freeshared buff/cache availableMem: 4044380605464 2045080148820 1393836 3226708

Swap: 2621436 0 2621436

Total:66658166054644666516

2. top

Itprovidesadynamicreal-timeviewofprocessesinthesystem.

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

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

KiBMem:4044380total,2052960free,600452used,1390968buff/cacheKiBSwap:2621436total, 2621436 free, 0 used. 3234820 avail Mem PID USER PR NI VIRT RESSHR S %CPU %MEM TIME+ COMMAND

1105root20 017500875700 51264S1.7 1.90:20.46Xorg 2529root20 0804443264024796S 1.0 0.8 0:02.47 gnome-term

3. ps

Itreportsthesnapshotofcurrentprocesses

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

Itreportsvirtualmemorystatistics

synopsis- vmstat [options]

example

[root@localhost ~]# vmstat

procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu------ rbswpdfreebuffcache si

sobi bo incs us syid wa st0 0 0187936816041487116 0 064 7 72140 1 097 1 0

5. df

Itdisplaystheamountofdiskspaceavailableinfile-system.

Synopsis- df [options]

example

[root@localhost ~]# df

Filesystem1K-blocksUsedAvailableUse%Mountedon

devtmpfs 2010800 0 2010800 0% /dev tmpfs 2022188 148 2022040 1% /dev/shmtmpfs 2022188 1404 2020784 1% /run /dev/sda6 487652 168276 289680 37% /boot

6. ping

Itisusedverifythatadevicecancommunicatewithanotheronnetwork.PINGstandsforPacket Internet Groper.

synopsis- ping [options]

[root@localhost~]#ping172.16.4.1

PING 172.16.4.1 (172.16.4.1) 56(84) bytes of data.
64bytesfrom172.16.4.1:icmp_seq=1ttl=64time=0.328ms 64
bytes from 172.16.4.1: icmp_seq=2 ttl=64 time=0.228 ms

18
64bytesfrom172.16.4.1:icmp_seq=3ttl=64time=0.264ms64bytesfrom172.16.4.1:icmp_seq=4 ttl=64 time=0.312 ms^C
--- 172.16.4.1 ping statistics ---
4packetstransmitted,4received,0%packetloss,time3000msrttmin/avg/max/mdev=
0.228/0.283/0.328/0.039 ms

7. ifconfig

Itisusedconfigurenetworkinterface.

synopsis- ifconfig [options]

example

[root@localhost ~]# ifconfig

enp2s0:flags=4163<UP,BROADCAST,RUNNING,MULTICAST>mtu1500inet172.16.6.102
netmask255.255.252.0broadcast172.16.7.255inet6fe80::4a0f:cfff:fe6d:6057prefixlen64scopeid 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)
TXerrors0dropped0overruns0carrier0collisions08.

traceroute

It tracks the route the packet takes to reach the destination.synopsis- traceroute [options]

example

[root@localhost ~]# traceroute www.rajalakshmi.org
traceroutetowww.rajalakshmi.org(220.227.30.51),30hopsmax,60bytepackets1gateway
(172.16.4.1) 0.299 ms 0.297 ms 0.327 ms2 220.225.219.38 (220.225.219.38) 6.185 ms 6.203 ms
6.18ms

Result:

Thus ,the basic linux commands program is executed successfully

Ex. no: 2a)

Date:5/2/25

SHELL SCRIPT

Aim:

TowriteaShellscripttotodisplaybasiccalculator.Program:

```
#!/bin/bash

echo"Enterfirstnumber:" read

a

echo"Entersecondnumber:"

read b

echo"Selectoperation:"

echo "1. Addition"

echo "2. Subtraction"

echo "3. Multiplication"

echo "4. Division"

echo"5.Modulus"
read choice
case $choice in

1) result=$((a + b))

echo "Addition = $result";;

2) result=$((a - b))

echo "Subtraction = $result";;

3) result=$((a * b))

echo "Multiplication = $result";;

4) if[$b-ne0]

then
```

result=$((a/b))

echo"Division=$result"el

se

echo"Divisionbyzeronotallowed" fi

;;

5) result=$((a % b))

echo "Modulus = $result";;

*)echo"Invalidchoice";; Esac

Sample Input and Output
Run the program using the below command [REC@local host~]$ sh arith.sh

Entertwono
5
10
add 15
sub -5
mul 50
div 0
mod 5c"

Result:

Thus the basic calculator program is executed successfully

Ex. no: 2b)

Date:5/2/25

SHELL SCRIPT

Aim:
TowriteaShellscripttotestgivenyearisleapornotusingconditionalstatement

Program:

```
#!/bin/bash

echo"Enterayear:"
read year

if (( year % 400 == 0 )); then
echo "$year is a Leap Year"
elif((year%100==0));then
echo"$yearisNOTaLeapYear"
elif (( year % 4 == 0 )); thenecho
"$year is a Leap Year"
else
echo"$yearisNOTaLeapYear"fi
```

Sample Input and Output
Run the program using the below command [REC @ local host~]$ sh leap.sh

```
enternumber
12
leap year
```

Result:

Thus the leap year program using linux commands is executed successfully

Ex.No.:3a)

Date:7/2/25

Shell Script – Reverse of Digit

Aim:

TowriteaShellscripttoreverseagivendigitusingloopingstatement.

Program:

```
#!/bin/bash
echo"Enteranumber:" read
num
reverse=0
while[$num-gt0] do
remainder=$((num % 10))
reverse=$((reverse*10+remainder))
num=$((num / 10))
done
echo "Reversed number: $reverse"
```

Sample Input and Output

Run the program using the below command [REC@local host~]$sh indhu.sh

enternumber
123
321

Result:

Thus the Shell script to reverse a given digit using loopingis executed successfully

Ex. No.: 3b)

Date: 7/2/25

Shell Script – Fibbonacci Series

Aim:

TowriteaShellscripttogenerateaFibonacciseriesusingfor loop.

Program:

```
#!/bin/bash
echo"Enterthenumberofterms:" read
n
a=0
b=1
echo"Fibonacciseries:"
for (( i=0; i<n; i++ ))
do
echo-n"$a"
fn=$((a+b))
a=$b
b=$fn
done
echo
```

Sample Input and Output

Run the program using the below command [REC@local host~]$sh indhu.sh

```
enternumber
21
fibonacciseries
0
1
1
2
3
5
8
13
21
34
55
89
144
233
377
```

Result:

Thus the fibonacci program using linux is executed successfully

Ex. No.: 4a)

Date:12/2/25

EMPLOYEEAVERAGEPAY

Aim:

Tofindouttheaveragepayofallemployeeswhosesalaryismorethan6000andno.ofdaysworkedis more than 4.

Algorithm:

1. Createaflatfileemp.datforemployeeswiththeirname,salaryperdayandnumberofdaysworked and save it.
2. Create an awk script emp.awk
3. For each employee record do
a. IfSalaryisgreaterthan6000andnumberofdaysworkedismorethan4,thenprintnameandsalary earned
b. Compute total pay of employee
4. Print the total number of employees satisfying the criteria and their average pay.

ProgramCode:

emp.data

```
JOE80005
 RAM 6000 5
 TIM 5000 6
 BEN 7000 7
 AMY 6500 6
```

emp.awk

```
BEGIN{total=0;count=0}
$2>6000 && $3>4 {
pay=$2*$3
print$1,pay
```

```
total+=pay

count+=1

}

END {

print"noofemployeesare=",count print

"total pay=", total

if(count>0)

print"averagepay=",total/count else

print "average pay= 0"

}
```

Sample Input:

//emp.dat–Col1isname,Col2isSalaryPerDayandCol3is//no.ofdaysworked JOE 8000

5
RAM 6000 5
TIM 5000 6
BEN 7000 7
AMY 6500 6

Output:
Run the program using the below commands
[student@localhost ~]$ vi emp.dat
[student@localhost ~]$ vi emp.awk
[student@localhost~]$gawk-femp.awkemp.dat.

EMPLOYEESDETAILS
JOE 40000

BEN 49000
AMY 39000
noofemployeesare=3
total pay= 128000
average pay= 42666.7
[student@localhost ~]$

Result:

Thus the program to find out the average pay of all employees whose salary is more than 6000 andno. ofdays worked is more than 4 is executed successfully

Ex. No.: 4b)

Date:12/2/25

RESULTSOFEXAMINATION

Aim:

Toprintthepass/failstatusofastudentinaclass.

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

ProgramCode:

```
//marks.awk

BEGIN{print"NAME SUB1 SUB2 SUB3 SUB4 SUB5 SUB6 STATUS"}

{

status="PASS"

for(i=2;i<=7;i++)

if($i<45) {status="FAIL";break}

print $1,$2,$3,$4,$5,$6,$7,status

}
```

Input:

//marks.dat
//Col1-name,Col2toCol7–marksinvarioussubjects BEN
40 55 66 77 55 77
TOM6067849290 60
RAM 90 95 84 87 56 70
JIM 60 70 65 78 90 87

Output:
Run the program using the below command
[root@localhoststudent]#gawk-fmarks.awkmarks.dat

NAMESUB-1 SUB-2 SUB-3 SUB-4 SUB-5 SUB-6 STATUS

BEN405566775577FAILTOM606784929060PASSRAM909584875670PASSJIM6070
65 78 90 87 PASS

Result:

Thus,the program to print the pass/fail status of a student in a class is executed successfully

Ex. No.: 5
Date:12/2/25

System Calls Programming

Aim:Toexperimentsystemcallsusingfork(),execlp()andpid()functions.

Algorithm:

1. Start
o Include the required header files (stdio.h and stdlib.h).
2. VariableDeclaration
o Declare an integer variable pid to hold the process ID.
3. Create a Process
o Callthefork()functiontocreateanewprocess.Storethereturnvalueinthepidvariable:▪If fork() returns:
▪ -1: Forking failed (child process not created).
▪ 0: Process is the child process.
▪ Positive integer: Process is the parent process.
4. Print Statement Executed Twice
o Print the statement:

scss
Copy code
THIS LINE EXECUTED TWICE

(This line is executed by both parent and child processes after fork()).

5. Check for Process Creation Failure
o If pid == -1:
▪ Print:

Copy code
CHILD PROCESS NOT CREATED

▪ Exit the program using exit(0).
6. Child Process Execution
o If pid == 0 (child process):
▪ Print:
▪ Process ID of the child process using getpid().
▪ Parent process ID of the child process using getppid().
7. Parent Process Execution
o If pid > 0 (parent process):
▪ Print:
▪ Process ID of the parent process using getpid().
▪ Parent's parent process ID using getppid().
8. Final Print Statement
o Print the statement:

Objective
Copycode
IT CAN BE EXECUTED TWICE

(This line is executed by both parent and child processes).

9. End

Program:

```
#include <stdio.h>
#include <unistd.h>
#include<sys/types.h>
#include <sys/wait.h>

int main() {
pid_tpid = fork();// Create a new process

if (pid < 0) {
// If fork fails
perror("Forkfailed");
return 1;
}

if (pid == 0) {
// Child process
printf("Child process: PID = %d, Parent PID = %d\n", getpid(), getppid());

// Execute a command using execlp
execlp("ls", "ls", "-l", NULL);// List files in the current directory
// If execlp fails
perror("execlpfailed");
return 1;
} else {
// Parent process
wait(NULL);//Waitforchildprocessto complete
printf("Parent process: PID = %d, Child PID = %d\n", getpid(), pid);
}

return 0;
}
```

Output:

Childprocess:PID=12345,ParentPID=12344

total 4

drwxrwxrwx 2 user user 4096 Apr 25 12:00 folder1

-rwxrwxrwx1useruser1732Apr2512:00testfile Parent

process: PID = 12344, Child PID = 12345

Result:

Thus the System Calls Programming using linux is executed successfully

Ex. No.: 6a)
Date:14/2/25

FIRST COME FIRST SERVE

Aim:
ToimplementFirst-comeFirst-serve(FCFS)scheduling technique

Algorithm:

1. Getthenumberofprocessesfromtheuser.

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&bursttimeforeachprocess.6.Displaythetotalwaitingtime,averagewaitingtime,turnaround time

ProgramCode:

```
#include <stdio.h>

struct Process {
int pid;            //Process ID
int arrival_time; // Arrival time of the process
intburst_time;//Bursttime(timeneededbytheprocesstocomplete) int
waiting_time; // Waiting time for the process
intturn_around_time;//Turnaroundtime(waitingtime+bursttime)
};

voidcalculate_waiting_time(structProcess[],int,int);
void calculate_turnaround_time(struct Process[], int);
void find_average_times(struct Process[], int);

intmain(){
int n;

// Get the number of processes
printf("Enterthenumberofprocesses:");
```

```
scanf("%d", &n);

struct Process p[n];

// Input process
detailsfor(inti=0;i<n;i++)
{
printf("\nEnterdetailsforprocess%d:\n",i+1);
p[i].pid = i + 1;// Process ID
printf("Arrival time: ");
scanf("%d",&p[i].arrival_time);
printf("Burst time: ");
scanf("%d", &p[i].burst_time);
}

// FCFS Scheduling
calculate_waiting_time(p, n, 0);// Calculate waiting times
calculate_turnaround_time(p, n);        //Calculateturnaroundtimes
find_average_times(p, n);       // Find and print average times

return 0;
}

voidcalculate_waiting_time(structProcessp[],intn,intstart_time){
p[0].waiting_time = 0; // First process has no waiting time

//Calculatewaitingtimeforeachprocess for
(int i = 1; i < n; i++) {
p[i].waiting_time = p[i - 1].waiting_time + p[i - 1].burst_time;
}
}
```

```
void calculate_turnaround_time(struct Process p[], int n) {
//Calculateturnaroundtimeforeachprocess for
(int i = 0; i < n; i++) {
p[i].turn_around_time = p[i].waiting_time + p[i].burst_time;
}
}

void find_average_times(struct Process p[], int n)
{floattotal_waiting_time=0,total_turnaround_time=0;

// Display individual process times and calculate totals
printf("\nProcess\tArrival Time\tBurst Time\tWaiting Time\tTurnaround Time\n");
for (int i = 0; i < n; i++) {
printf("%d\t%d\t\t%d\t\t%d\t\t%d\n",p[i].pid,p[i].arrival_time,p[i].burst_time,
p[i].waiting_time, p[i].turn_around_time);
total_waiting_time += p[i].waiting_time;
total_turnaround_time+=p[i].turn_around_time;
}

// Calculate and display average waiting time and turnaround time
printf("\nAverage waiting time: %.2f", total_waiting_time / n);
printf("\nAverageturnaroundtime:%.2f",total_turnaround_time/n);
}
```

Sample Output:

```
Enter the number of process:
3
Enter the burst time of the processes:
24 3 3

ProcessBurstTimeWaitingTimeTurnAroundTime024024132427232730 Average waiting

time is: 17.0
AverageTurnaroundTimeis:19
```

Result:

Thus the linux program to implement First-come First- serve (FCFS) scheduling technique is executed successfully

Ex. No.: 6b)
Date:5/3/25

SHORTESTJOBFIRST

Aim:
ToimplementtheShortestJobFirst(SJF)schedulingtechniqueAlgorithm:

1. Declare the structure and its elements.
2. Get number of processes as input from the user.
3. Read the process name, arrival time and burst time
4. Initializewaitingtime,turnaroundtime&flagofreadprocessestozero.5.Sortbasedonbursttime ofallprocessesinascendingorder6.Calculatethewaitingtimeandturnaroundtimeforeachprocess.
7. Calculate the averagewaiting time and average turnaround time. 8. Display the results.

ProgramCode:

```
#include <stdio.h>



struct Process {

int pid;            //Process ID

int arrival_time; // Arrival time of the process

intburst_time;//Bursttime(timeneededbytheprocesstocomplete) int

waiting_time; // Waiting time for the process

intturn_around_time;//Turnaroundtime(waitingtime+bursttime)

};



// Function prototypes

void calculate_waiting_time(struct Process[], int);
```

```
voidcalculate_turnaround_time(structProcess[],int);

void find_average_times(struct Process[], int);

void sort_by_burst_time(struct Process[], int);


intmain(){

int n;


// Get the number of processes

printf("Enterthenumberofprocesses:");

scanf("%d", &n);


struct Process p[n];


// Input process

detailsfor(inti=0;i<n;i++)

{

printf("\nEnterdetailsforprocess%d:\n",i+1);

p[i].pid = i + 1;// Process ID

printf("Arrival time: ");
```

```
scanf("%d",&p[i].arrival_time);
```

```
printf("Burst time: ");
scanf("%d",&p[i].burst_time);
}

//Sortprocessesbybursttime(non-preemptiveSJF) sort_by_burst_time(p,
n);

//Calculatewaitingandturnaroundtimes
calculate_waiting_time(p, n);
calculate_turnaround_time(p, n);

//Findanddisplayaveragetimes
find_average_times(p, n);

return 0;
}

//Sortprocessesbybursttime(non-preemptiveSJF)
void sort_by_burst_time(struct Process p[], int n) {
```

```
struct Process temp;

for(inti=0;i<n-1;i++){ for

(int j = i + 1; j < n; j++) {

if (p[i].burst_time > p[j].burst_time) {

//Swaptheprocesses

temp = p[i];

p[i] = p[j];

p[j] = temp;

}

}

}

}



// Calculate waiting time for each process

void calculate_waiting_time(struct Process p[], int n) {

p[0].waiting_time=0;//Firstprocesshasnowaitingtime



// Calculate waiting time for each process

for (int i = 1; i < n; i++) {
```

```
p[i].waiting_time = p[i - 1].waiting_time + p[i - 1].burst_time;

}

}


// Calculate turnaround time for each process

void calculate_turnaround_time(struct Process p[], int n) {

//Calculateturnaroundtimeforeachprocess for

(int i = 0; i < n; i++) {

p[i].turn_around_time = p[i].waiting_time + p[i].burst_time;

}

}


//Calculateanddisplayaveragewaitingandturnaroundtimes void

find_average_times(struct Process p[], int n) {

float total_waiting_time = 0, total_turnaround_time = 0;


// Display individual process times and calculate totals

printf("\nProcess\tArrivalTime\tBurstTime\tWaitingTime\tTurnaroundTime\n");

for (int i = 1; i < n; i++) {
```

```
printf("%d\t%d\t\t%d\t\t%d\t\t%d\n",p[i].pid,p[i].arrival_time,p[i].burst_time,

p[i].waiting_time, p[i].turn_around_time);

total_waiting_time += p[i].waiting_time;

total_turnaround_time+=p[i].turn_around_time;

}



// Calculate and display average waiting time and turnaround time

printf("\nAverage waiting time: %.2f", total_waiting_time / n);

printf("\nAverageturnaroundtime:%.2f",total_turnaround_time/n);

}
```

Sample Output:

Enter the number of process:
4
Enter the burst time of the processes:
8 4 9 5

ProcessBurstTimeWaitingTimeTurnAroundTime 2 4

0 44 5 4 91 8 9 173 9 17 26

Average waiting time is: 7.5
AverageTurnAroundTimeis:13.0

Result: ThustheprogramtoimplementtheShortestJobFirst(SJF)schedulingtechniqueisexecuted successful

Ex. No.: 6c) **PRIORITY SCHEDULING**
Date:5/3/25

Aim:
Toimplementpriorityschedulingtechnique

Algorithm:
1. Get the number of processes from the user.
2. Read the process name, burst time and priority of process.
3. Sort based on burst time of all processes in ascending order based 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. Displaythetotalwaitingtime,averagewaitingtime,turnaround Time

ProgramCode:

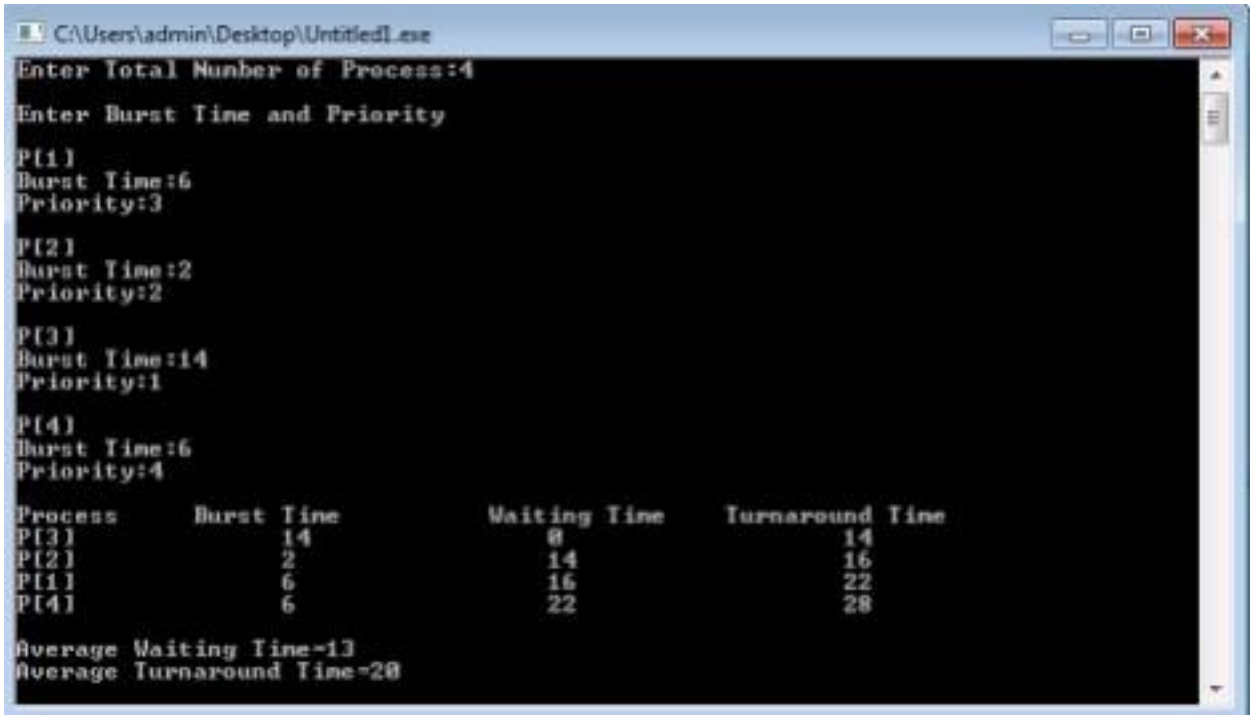
```
#include<stdio.h>
struct Process {
int pid;          //Process ID
int burst_time;          //Bursttimeoftheprocess
int priority;     // Priority of the process
int waiting_time;         //Waitingtimeoftheprocess
intturn_around_time;//Turnaroundtimeoftheprocess
};

void calculate_waiting_time(struct Process[], int);
voidcalculate_turnaround_time(structProcess[],int);
void find_average_times(struct Process[], int);
void sort_by_priority(struct Process[], int);

intmain(){
int n;

// Get the number of processes
printf("Enterthenumberofprocesses:");
scanf("%d", &n);
```

```
struct Process p[n];

// Input process
detailsfor(inti=0;i<n;i++)
{
printf("\nEnterdetailsforprocess%d:\n",i+1);
p[i].pid = i + 1;// Process ID
printf("Burst time: ");
scanf("%d",&p[i].burst_time);
printf("Priority: ");
scanf("%d", &p[i].priority);
}

//Sortprocessesbypriority(highestpriorityfirst)
sort_by_priority(p, n);

//Calculatewaitingandturnaroundtimes
calculate_waiting_time(p, n);
calculate_turnaround_time(p, n);

//Findanddisplayaveragetimes
find_average_times(p, n);

return 0;
}

//Sortprocessesbypriority(higherpriorityfirst)
void sort_by_priority(struct Process p[], int n) {
struct Process temp;
for(inti=0;i<n-1;i++){ for
(int j = i + 1; j < n; j++) {
if (p[i].priority > p[j].priority) {
//Swapprocessesifthepriorityofthefirstislower(higherpriorityvalue) temp =
p[i];
p[i] = p[j];
p[j] = temp;
}
}
}
}

// Calculate waiting time for each process
void calculate_waiting_time(struct Process p[], int n) {
p[0].waiting_time=0;//Firstprocesshasnowaitingtime

//Calculatewaitingtimeforeachprocess for
```

```
(int i = 1; i < n; i++) {
```

```
p[i].waiting_time = p[i - 1].waiting_time + p[i - 1].burst_time;
}
}

// Calculate turnaround time for each process
void calculate_turnaround_time(struct Process p[], int n) {
//Calculateturnaroundtimeforeachprocess for
(int i = 0; i < n; i++) {
p[i].turn_around_time = p[i].waiting_time + p[i].burst_time;
}
}

//Calculateanddisplayaveragewaitingandturnaroundtimes void
find_average_times(struct Process p[], int n) {
float total_waiting_time = 0, total_turnaround_time = 0;

// Display individual process times and calculate totals
printf("\nProcess\tBurst Time\tPriority\tWaiting Time\tTurnaround Time\n");
for (int i = 0; i < n; i++) {
printf("%d\t%d\t\t%d\t\t%d\t\t%d\n",p[i].pid,p[i].burst_time,p[i].priority,
p[i].waiting_time, p[i].turn_around_time);
total_waiting_time += p[i].waiting_time;
total_turnaround_time+=p[i].turn_around_time;
}

// Calculate and display average waiting time and turnaround time
printf("\nAverage waiting time: %.2f", total_waiting_time / n);
printf("\nAverageturnaroundtime:%.2f",total_turnaround_time/n);
}
```

Sample Output:



Result: Thus the linux programming for priority scheduling is executed

Ex. No.: 6d)
Date : 5/3/25 **ROUNDROBIN SCHEDULING**

Aim:

ToimplementtheRoundRobin(RR)scheduling technique

Algorithm:

1. Declare the structure and its elements.
2. GetnumberofprocessesandTime quantumasinputfromthe user.
3. Read the process name, arrival time and burst time
4. Createanarrayrem_bt[]tokeeptrackofremainingbursttimeofprocesseswhichisinitiallycopyof bt[] (burst times array)
5. Createanotherarraywt[]tostorewaitingtimesofprocesses.Initializethisarrayas0.6.Initialize time : t = 0
7. Keeptraversingtheallprocesseswhileallprocessesarenotdone.Dofollowingfori'thprocessifit 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.

ProgramCode:

```
#include<stdio.h>
struct Process {

int pid;          // Process ID

int burst_time;          // Burst time of the process

intremaining_time;//Remainingtimefortheprocess int

waiting_time;          // Waiting time of the process

intturn_around_time;//Turnaroundtimeoftheprocess

};
```

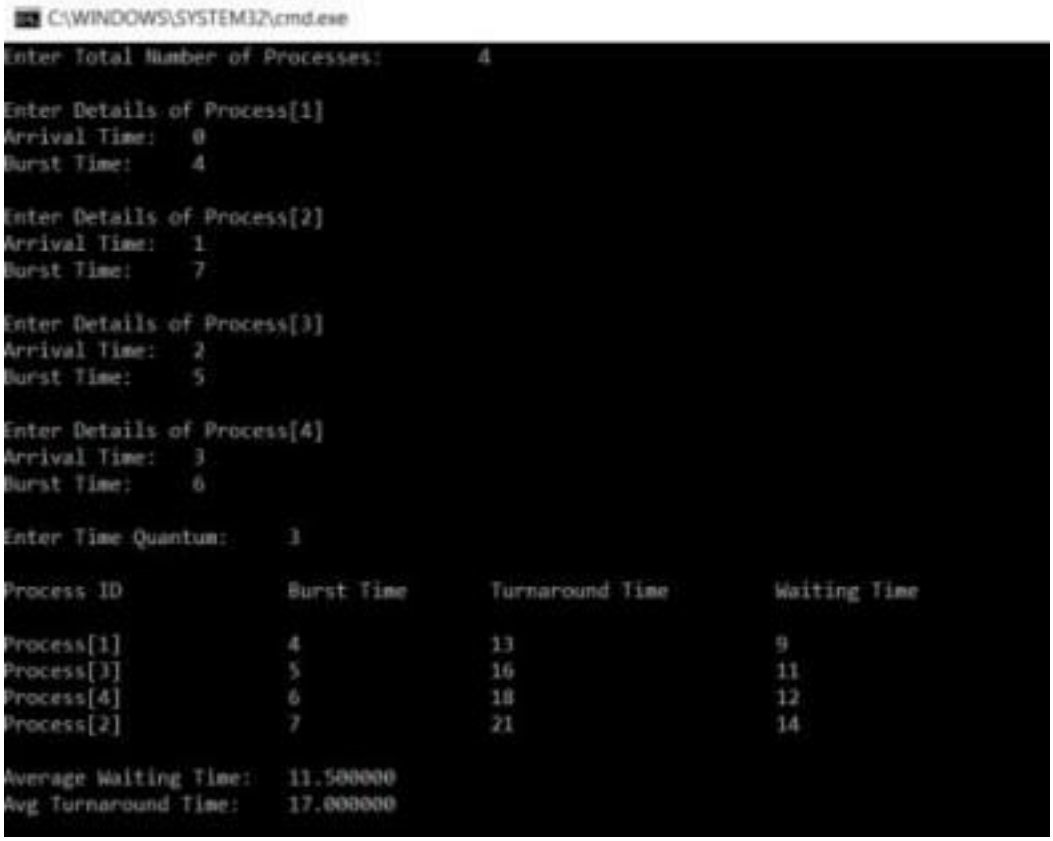
```
voidcalculate_waiting_time(structProcess[],int,int);
void calculate_turnaround_time(struct Process[], int);
void find_average_times(struct Process[], int);

int main() {
int n, quantum;

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

struct Process p[n];

// Input process
detailsfor(inti=0;i<n;i++)
{
printf("\nEnterdetailsforprocess%d:\n",i+1);
p[i].pid = i + 1;// Process ID
printf("Burst time: ");
scanf("%d",&p[i].burst_time);
p[i].remaining_time=p[i].burst_time;//Initially,remainingtimeisthebursttime
}
```

```
//Calculatewaitingandturnaroundtimes
calculate_waiting_time(p, n, quantum);
calculate_turnaround_time(p, n);

//Findanddisplayaveragetimes
find_average_times(p, n);

return 0;
}

// Calculate waiting time for each process
voidcalculate_waiting_time(structProcessp[],intn,intquantum){ int
time = 0;
int remaining_processes = n;

while(remaining_processes>0){ for
(int i = 0; i < n; i++) {
if (p[i].remaining_time > 0) {
// If the process has remaining
timeif(p[i].remaining_time>quantu
m){ time += quantum;
p[i].remaining_time -= quantum;
} else {
//Iftheprocesscanfinishwithinthequantum time
+= p[i].remaining_time;
```

```
p[i].waiting_time=time-p[i].burst_time;//Calculatewaitingtime
p[i].remaining_time = 0;
remaining_processes--;
}
}
}
}
}

// Calculate turnaround time for each process
voidcalculate_turnaround_time(structProcessp[],intn){ for
(int i = 0; i < n; i++) {
p[i].turn_around_time = p[i].waiting_time + p[i].burst_time;
}
}

//Calculateanddisplayaveragewaitingandturnaroundtimes void
find_average_times(struct Process p[], int n) {
float total_waiting_time = 0, total_turnaround_time = 0;

// Display individual process times and calculate totals
printf("\nProcess\tBurst Time\tWaiting Time\tTurnaround Time\n");
for (int i = 0; i < n; i++) {
printf("%d\t%d\t\t%d\t\t%d\n", p[i].pid, p[i].burst_time, p[i].waiting_time,
p[i].turn_around_time);
```

```
total_waiting_time += p[i].waiting_time;

total_turnaround_time+=p[i].turn_around_time;

}


// Calculate and display average waiting time and turnaround time

printf("\nAverage waiting time: %.2f", total_waiting_time / n);

printf("\nAverageturnaroundtime:%.2f",total_turnaround_time/n);

}
```

Sample Output:



Result:

Thus the round robin program is executed successfully

Ex. No.: 7

Date:7/3/25 **IPCUSING SHARED MEMORY**

Aim:

TowriteaCprogramtodoInterProcessCommunication(IPC)usingsharedmemorybetweensender process and 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. Writeastringtothesharedmemorysegmentusingsprintf
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. Printthesharedmemorycontentssentbythesender process.
5. Detach shared memory segment using shmdt

ProgramCode:
Sender.c

#include

<stdio.h>#include

<sys/ipc.h>

#include<sys/shm.h>

#include <string.h>

#include <stdlib.h>

#defineSHM_SIZE 1024// Size of shared memory segment

```
int main() {

key_tkey=1234;//Uniquekeyforsharedmemory int

shmid;

char *shm_ptr;


// Create shared memory segment

shmid=shmget(key,SHM_SIZE,0666|IPC_CREAT);//Creatingsharedmemory if

(shmid == -1) {

perror("shmgetfailed");

exit(1);

}


//Attachsharedmemorysegmenttosenderprocess shm_ptr

= shmat(shmid, NULL, 0);

if(shm_ptr==(char*)-1){

perror("shmat failed");

exit(1);

}
```

```
printf("Sender: Enter a message to send: ");

fgets(shm_ptr,SHM_SIZE,stdin);//Writingmessagetoshared memory


// Print confirmation that the message is written to shared memory

printf("Sender:Message written to shared memory: %s", shm_ptr);


//Detachsharedmemory

shmdt(shm_ptr);


return 0;

}
```

Receiver.c

```
// receiver.c#include

<stdio.h> #include

<sys/ipc.h>

#include<sys/shm.h>

#include <stdlib.h>


#defineSHM_SIZE 1024// Size of shared memory segment
```

```
int main() {

key_tkey=1234;//Uniquekeyforsharedmemory int

shmid;

char *shm_ptr;


// Access shared memory segment

shmid=shmget(key,SHM_SIZE,0666); if

(shmid == -1) {

perror("shmgetfailed");

exit(1);

}


//Attachsharedmemorysegmenttoreceiverprocess

shm_ptr = shmat(shmid, NULL, 0);

if(shm_ptr==(char*)-1){

perror("shmat failed");

exit(1);

}


// Read and print the message from shared memory
```

```
printf("Receiver: Message from shared memory: %s", shm_ptr);



//Detachsharedmemory

shmdt(shm_ptr);



return 0;

}
```

Sample Output
Terminal1
[root@localhoststudent]#gccsender.c-osender[root@localhoststudent]#./sender

Terminal2
[root@localhoststudent]#gccreceiver.c-oreceiver[root@localhoststudent]#./receiver Message Received: Welcome to Shared Memory[root@localhost student]#

Result:

Thus IPC using shared memory is executed successfully

Ex. No.: 8
Date:7/3/25

PRODUCER CONSUMER USING SEMAPHORES

Aim:
Towriteaprogramtoimplementsolutiontoproducerconsumerproblemusingsemaphores.

Algorithm:
1. Initializesemaphoreempty,fullandmutex.
2. Create two threads- producer thread and consumer thread.
3. Waitfortargetthreadtermination.
4. Call sem_wait on empty semaphore followed by mutex semaphore beforeentry into critical section.
5. Produce/Consume the item in critical section.
6. Call sem_post on mutex semaphore followed by full semaphore
7. before exiting critical section.
8. Allow the other thread to enter its critical section.
9. TerminateafterloopingtentimesinproducerandconsumerThreadseach.

ProgramCode:

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

#define SIZE 5// Size of the buffer int

buffer[SIZE];              //Sharedbuffer
int in = 0, out = 0; // Indexes for producer and consumer

// Semaphores
sem_tempty;//Countsemptyslots
sem_t full;// Counts full slots
pthread_mutex_t mutex; // Mutual exclusion for accessing buffer

void*producer(void*arg){ int
item;
for (int i = 1; i <= 10; i++) {
item = rand() % 100;// Produce an item
sem_wait(&empty);     // Decrease empty count
pthread_mutex_lock(&mutex);//Lockbufferaccess

//Additemtobuffer buffer[in]
= item;
printf("Producerproduced:%datposition%d\n",item,in); in =
(in + 1) % SIZE;
```

```
pthread_mutex_unlock(&mutex);//Unlockbuffer
sem_post(&full);                    // Increase full count

sleep(1); // Simulate production time
}
pthread_exit(NULL);
}

void*consumer(void*arg){
int item;
for (int i = 1; i <= 10; i++) {
sem_wait(&full);         //Decreasefullcount
pthread_mutex_lock(&mutex);//Lockbufferaccess

//Removeitemfrombuffer
item = buffer[out];
printf("Consumerconsumed:%dfromposition%d\n",item,out); out =
(out + 1) % SIZE;

pthread_mutex_unlock(&mutex); // Unlock buffer
sem_post(&empty);                //Increaseemptycount

sleep(2); // Simulate consumption time
}
pthread_exit(NULL);
}

int main() {
pthread_tprod,cons;

//Initializesemaphoresandmutex
sem_init(&empty, 0, SIZE);
sem_init(&full, 0, 0);
pthread_mutex_init(&mutex,NULL);

// Create producer and consumer threads
pthread_create(&prod, NULL, producer, NULL);
pthread_create(&cons,NULL,consumer,NULL);

//Waitforboththreadstofinish
pthread_join(prod, NULL);
pthread_join(cons, NULL);

// Destroy semaphores and mutex
sem_destroy(&empty);
sem_destroy(&full);
pthread_mutex_destroy(&mutex);

return 0;
```

}

Sample Output:

1. Producer
2.Consumer
3.Exit
Enter your choice:1
Producerproducestheitem1Enteryourchoice:2
Consumer consumes item1 Enter your choice:2
Buffer is empty!!
Enter your choice:1
Producerproducestheitem1Enteryourchoice:1
Producerproducestheitem2Enteryourchoice:1
Producerproducestheitem3Enteryourchoice:1
Buffer is full!!
Enter your choice:3

Result:

ThustheProducer-Consumerusing Semaphoresisexecuted successfully

Ex. No.: 9
Date:19/3/25

DEADLOCK AVOIDANCE

Aim:
TofindoutasafesequenceusingBanker'salgorithmfordeadlock avoidance.

Algorithm:
1. Initialize work=available and finish[i]=false for all values of i
2. Find an i such that both:
finish[i]=falseandNeedi<=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

ProgramCode:

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

#defineP5//Numberofprocesses
#define R 3 // Number of resources

int main() {
int allocation[P][R] = {
{0, 1, 0},
{2, 0, 0},
{3, 0, 2},
{2, 1, 1},
{0, 0, 2}
};

int max[P][R] = {
{7, 5, 3},
{3, 2, 2},
{9, 0, 2},
{2, 2, 2},
{4, 3, 3}
```

```
};

int available[R] = {3, 3, 2};

int need[P][R];
boolfinished[P]={false}; int
safeSequence[P];
int count = 0;

// Calculate need matrix
for(inti=0;i<P;i++){ for
(int j = 0; j < R; j++) {
need[i][j] = max[i][j] - allocation[i][j];
}
}

while(count<P){
bool found = false;
for(intp=0;p<P;p++){ if
(!finished[p]) {
bool canAllocate = true;
for(intr=0;r<R;r++){
if(need[p][r]>available[r]){
canAllocate = false;
break;
}
}

if (canAllocate) {
for (int r = 0; r < R; r++) {
available[r]+=allocation[p][r];
}
safeSequence[count++]=p;
finished[p] = true;
```

```
found = true;
}
}
}

if (!found) {
printf("Systemisnotinasafestate.\n");
return -1;
}
}

// Print safe sequence
printf("Systemisinasafestate.\nSafesequenceis:"); for
(int i = 0; i < P; i++) {
printf("P%d ", safeSequence[i]);
}
printf("\n");

return 0;
}
```

Sample Output:

The SAFE Sequence isP1 -> P3 -> P4 -> P0 ->P2

Result: Thus the deadlock avoidance program is executed successfully

Ex. No.: 10a) **BEST FIT**
Date:19/3/25

Aim:
ToimplementBestFitmemoryallocationtechniqueusing Python.

Algorithm:
1. Input memory blocks and processes with sizes
2. Initialize all memory blocks as free.
3. Startbypickingeachprocessandfindtheminimumblocksizethatcanbeassignedtocurrent process
4. If found then assign it to the current process.
5. Ifnotfoundthenleavethatprocessandkeepcheckingthefurtherprocesses.
Program Code:

```
def best_fit(block_size, process_size):
n = len(block_size)
m = len(process_size)
allocation = [-1] * m# Stores index of block allocated to process

for i in range(m):
best_idx = -
1forjinrange(n):
if block_size[j] >= process_size[i]:
ifbest_idx==-1orblock_size[j]<block_size[best_idx]: best_idx
= j

if best_idx != -1:
allocation[i] = best_idx
block_size[best_idx]-=process_size[i]

print("\nProcessNo.\tProcessSize\tBlockNo.")
for i in range(m):
print(f"{i+1}\t\t{process_size[i]}\t\t",end="") if
allocation[i] != -1:
print(f"{allocation[i]+1}")
else:
print("Not Allocated")


# Example Inputs
block_size = [100, 500, 200, 300, 600]
process_size=[212,417,112,426]

# Run Best Fit Allocation
best_fit(block_size,process_size)
```

Sample Output:

ProcessNo. ProcessSize Block no.1 212 42 417 2311234426 5

Result:

Thus the best fit program using python is execxuted successfully

Ex. No.: 10b) **FIRST FIT**
Date:19/3/25

Aim:
TowriteaCprogramforimplementationmemoryallocationmethodsforfixedpartitionusingfirstfit.

Algorithm:

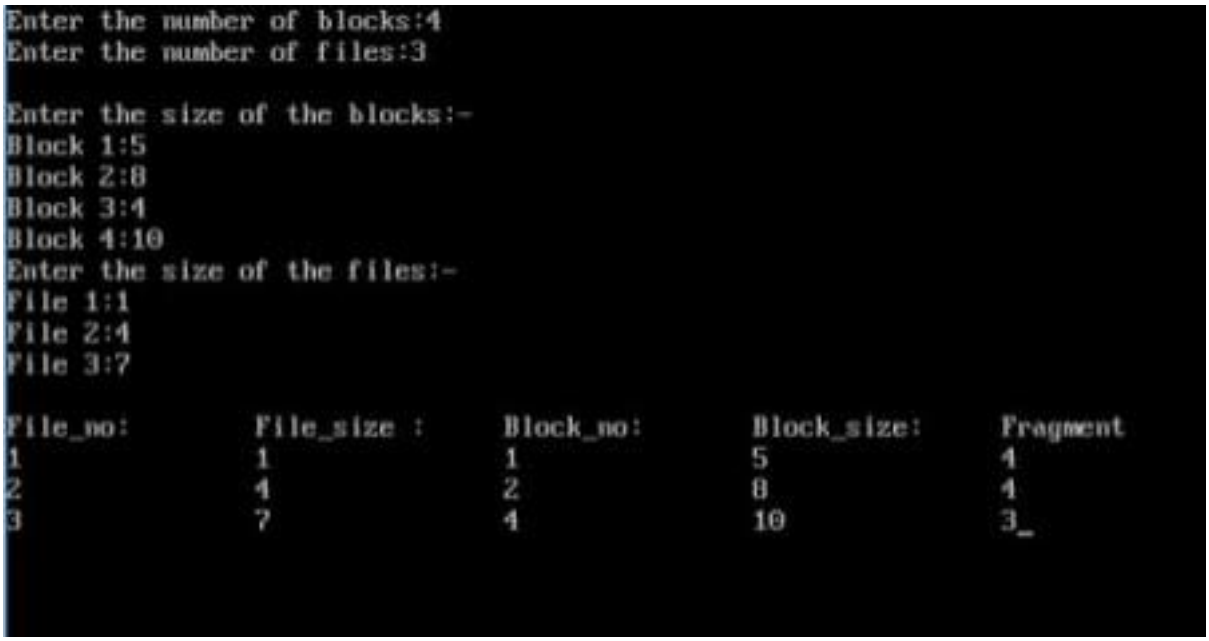
1. Define the max as 25.
2:Declarethevariablefrag[max],b[max],f[max],i,j,nb,nf,temp,highest=0,bf[max],ff[max].3:Getthe number of blocks,files,size of the blocks using for loop.
4:Inforloopcheckbf[j]!=1,ifsotemp=b[j]-f[i] 5:
Check highest

ProgramCode:

```
#include <stdio.h>

#defineMAX_PARTITIONS10
#define MAX_PROCESSES 10

int main() {
int partitions[MAX_PARTITIONS], processes[MAX_PROCESSES];
int partitionCount, processCount;
int allocation[MAX_PROCESSES];

printf("Enternumberofmemorypartitions:");
scanf("%d", &partitionCount);

printf("Entersizeofeachpartition:\n");
for (int i = 0; i < partitionCount; i++) {
printf("Partition %d: ", i + 1);
scanf("%d", &partitions[i]);
}

printf("Enternumberofprocesses:"); scanf("%d",
&processCount);
```

```
printf("Entersizeofeachprocess:\n"); for
(int i = 0; i < processCount; i++) {
printf("Process %d: ", i + 1);
scanf("%d", &processes[i]);
allocation[i] = -1; // initially not allocated
}

// First Fit Allocation
for (int i = 0; i < processCount; i++) {
for(intj=0;j<partitionCount;j++){ if
(partitions[j] >= processes[i]) {
allocation[i] = j;
partitions[j]-=processes[i];//reducepartitionsize
break;
}
}
}

// Display Allocation
printf("\nProcessNo.\tProcessSize\tPartitionNo.\n");
for (int i = 0; i < processCount; i++) {
printf("%d\t\t%d\t\t", i + 1, processes[i]);
if (allocation[i] != -1)
printf("%d\n",allocation[i]+1);
else
printf("Not Allocated\n");
}

return 0;
}
```

Sample Output:



Result: Thus the first fit program is executed successful

Ex.No.: 11a) **FIFOPAGEREPLACEMENT**
Date:26/3/25

Aim:
Tofindoutthenumberofpagefaultsthatoccurusing First-in First-out(FIFO)pagereplacement technique.

Algorithm:
1. Declare the size with respect to page length
2. Check the need of replacement from the page to memory
3. Checktheneedofreplacementfromoldpagetonewpageinmemory4.Formaqueuetoholdall pages
5. Insert the page require 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

ProgramCode:

```
#include <stdio.h>

#include<stdbool.h>



int main() {

int frames, pages;



printf("Enternumberofframes:");

scanf("%d", &frames);



printf("Enternumberofpages:");

scanf("%d", &pages);
```

```
int page[pages];

printf("Enterthepagereferencestring:\n"); for

(int i = 0; i < pages; i++) {

scanf("%d", &page[i]);

}


int memory[frames];

for (int i = 0; i < frames; i++)

memory[i]= -1;// initialize frame content


intpageFaults=0; int

pointer = 0;


printf("\nPage\tFrames\t\tStatus\n");

for (int i = 0; i < pages; i++) {

bool found = false;


//Checkifpageisalreadyinmemory for

(int j = 0; j < frames; j++) {

if (memory[j] == page[i]) {
```

```
found=true;
break;
}
}

if (!found) {
memory[pointer]=page[i];
pointer=(pointer+1)%frames;
pageFaults++;

printf("%d\t", page[i]);
for(intk=0;k<frames;k++){ if
(memory[k] == -1)
printf("-");
else
printf("%d ", memory[k]);
}
printf("\tPage Fault\n");
} else {
printf("%d\t", page[i]);
```

```
for(intk=0;k<frames;k++){ if

(memory[k] == -1)

printf("-");

else

printf("%d ", memory[k]);

}

printf("\tNo Fault\n");

}

}



printf("\nTotalPageFaults=%d\n",pageFaults);



return 0;

}
```

Sample Output:

[root@localhost student]# python fifo.py

Enterthesizeofreferencestring:20Enter[1]:7 Enter [2]
: 0
Enter [3] : 1
Enter [4] : 2
Enter [5] : 0
Enter [6] : 3
Enter [7] : 0
Enter [8] : 4
Enter [9] : 2
Enter [10] : 3
Enter[11]:0
Enter [12] : 3
Enter [13] : 2
Enter [14] : 1
Enter [15] : 2
Enter [16] : 0
Enter [17] : 1
Enter [18] : 7
Enter [19] : 0
Enter [20] : 1

Enterpageframesize:3 7 -

> 7 - -
0 -> 7 0 -
1 -> 7 0 1
2 -> 2 0 1
0 -> No Page Fault

3 -> 2 3 1
0 -> 2 3 0
4 -> 4 3 0
2 -> 4 2 0
3 -> 4 2 3
0 -> 0 2 3
3->NoPageFault 2
-> No Page Fault
1 -> 0 1 3
2 -> 0 1 2
0 -> No Page Fault
1->NoPageFault 7
-> 7 1 2
0 -> 7 0 2

1 -> 7 0 1
Totalpagefaults:15.[root@localhost student]#

Result: Thus the FIFO program is executed successfully

Ex.No.: 11b) **LRU**
Date:26/3/25

Aim:
TowriteacprogramtoimplementLRUpagereplacementalgorithm.

Algorithm:
1:Starttheprocess 2:
Declare the size
3:Getthenumberofpagestobeinserted 4:
Get the value
5: Declare counter and stack
6:Selecttheleastrecentlyusedpagebycountervalue 7:
Stack them according the selection.
8:Displaythevalues 9:
Stop the process

ProgramCode:

```
#include <stdio.h>

intfindLRU(inttime[],intn){ int
min = time[0], pos = 0;
for(inti=1;i<n;++i){ if
(time[i] < min) {
min=time[i];
pos = i;
}
}
return pos;
}

int main() {
int frames, pages;

printf("Enternumberoffframes:");
scanf("%d", &frames);

printf("Enter number of pages: ");
```

```
scanf("%d", &pages);

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

int memory[frames];
inttime[frames];//Totracklastusedtime int
count = 0, pageFaults = 0;
intcurrentTime=0;

for(inti=0;i<frames;i++){
memory[i] = -1;
time[i] = 0;
}

printf("\nPage\tFrames\t\tStatus\n");

for(inti=0;i<pages;i++){ int
flag = 0;

for(intj=0;j<frames;j++){ if
(memory[j] == page[i]) {
currentTime++;
time[j]=currentTime;
flag = 1;
break;
}
}

if(!flag){
int pos;
```

```
if(count<frames){
pos = count;
count++;
} else {
pos = findLRU(time, frames);
}
memory[pos] = page[i];
currentTime++;
time[pos]=currentTime;
pageFaults++;
}

printf("%d\t", page[i]);
for(intk=0;k<frames;k++){ if
(memory[k] != -1)
printf("%d",memory[k]); else
printf("- ");
}

if (!flag)
printf("\tPageFault\n");
else
printf("\tNo Fault\n");
}

printf("\nTotalPageFaults=%d\n",pageFaults); return

0;
}
```

Sample Output :

Enter number of frames: 3Enter number of pages: 6Enter reference string: 5 7 5 6 7 35 -1 -1
5 7 -1
5 7 -1
5 7 6
5 7 6
3 7 6
TotalPageFaults=4

Result:

Thus the LRU program has been executed successfully

Ex. No.: 11c) **Optimal**
Date: 26/3/25

Aim:
TowriteacprogramtoimplementOptimalpagereplacementalgorithm.

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>

intsearch(intpage,intframe[],intn){ for
(int i = 0; i < n; i++) {
if(frame[i]==page)
return 1;
}
return 0;
}

intpredict(intpages[],intframe[],intn,intindex,inttotal_pages){ int res
= -1, farthest = index;
for(inti=0;i<n;i++){ int j;
for(j=index;j<total_pages;j++){ if
(frame[i] == pages[j]) {
if (j > farthest) {
```

```
farthest=j;
res = i;
}
break;
}
}

if (j == total_pages)
return i; // If not found in future, return immediately
}

return (res == -1) ? 0 : res;
}

voidoptimalPageReplacement(intpages[],inttotal_pages,intcapacity){ int
frame[capacity];
int count = 0, page_faults = 0;

for(inti=0;i<capacity;i++) frame[i] =
-1;

for (int i = 0; i < total_pages; i++) {
if(search(pages[i],frame,capacity)){
printf("Page %d -> HIT\n", pages[i]);
continue;
}

if (count < capacity) {
frame[count++]=pages[i];
} else {
intpos=predict(pages,frame,capacity,i+1,total_pages);
frame[pos] = pages[i];
}

page_faults++;
printf("Page%d->FAULT\tFrames:",pages[i]);
```

```
for(intj=0;j<capacity;j++)
printf("%d ", frame[j]);
printf("\n");
}

printf("\nTotalPageFaults=%d\n",page_faults);
}

int main() {
int pages[] = {7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2};
inttotal_pages=sizeof(pages)/sizeof(pages[0]); int
capacity = 4;

optimalPageReplacement(pages, total_pages, capacity);

return 0;
}
```

Output:

```
Page 7 ->FAULT       Frames: 7 -1 -1 -1
Page 0 ->FAULT       Frames: 7 0 -1 -1
Page 1 ->FAULT       Frames: 7 0 1 -1
Page 2 -> FAULT      Frames:7012
Page 0 -> HIT
Page 3 ->FAULT       Frames: 3 0 1 2
...
TotalPageFaults=X
```

Result:

Thus,a c program to implement Optimal page replacement is executed successfully

Ex. No.: 12 FileOrganizationTechnique-SingleandTwoleveldirectory
Date:28/3/25

AIM:

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

a. Single Level

Directory

ALGORITHM

1. Start

2. Declarethenumber,namesandsizeofthedirectoriesandfilenames.3.Getthevaluesforthe declared

variables.

4. Display the files that are available in the directories.

5. Stop.

PROGRAM:

```
#include <stdio.h>

#include <string.h>


struct Directory {

charfilename[20][20];

int file_count;

};


voidsingleLevelDirectory(){

struct Directory dir;

dir.file_count = 0;

int choice;
```

```
char name[20];

printf("Single Level Directory Implementation\n");

while (1) {
printf("\n1.CreateFile\n2.DeleteFile\n3.SearchFile\n4.ListFiles\n5.Exit\nEnterchoice:");
scanf("%d", &choice);

switch(choice){
case 1:
printf("Enterfilenametocreate:");
scanf("%s", name);
int found = 0;
for(inti=0;i<dir.file_count;i++){
if(strcmp(name,dir.filename[i])==0){
found = 1;
break;
}
}
if (found)
printf("Filealreadyexists!\n");
else {
strcpy(dir.filename[dir.file_count], name);
dir.file_count++;
printf("File created successfully.\n");
}
break;
case 2:
printf("Enterfilenametodelete:");
scanf("%s", name);
```

```
found = 0;
for(inti=0;i<dir.file_count;i++){
if(strcmp(name,dir.filename[i])==0){
found = 1;
for (int j = i; j < dir.file_count - 1; j++) {
strcpy(dir.filename[j],dir.filename[j+1]);
}
dir.file_count--;
printf("Filedeletedsuccessfully.\n");
break;
}
}
if (!found)
printf("Filenotfound!\n");
break;
case 3:
printf("Enterfilenametosearch:");
scanf("%s", name);
found = 0;
for(inti=0;i<dir.file_count;i++){
if(strcmp(name,dir.filename[i])==0){
found = 1;
printf("Filefound!\n");
break;
}
}
if (!found)
printf("Filenotfound!\n");
break;
case 4:
```

```
printf("Files:\n");
for(inti=0;i<dir.file_count;i++){ printf("%s\n",
dir.filename[i]);
}
break;
case 5:
return;
default:
printf("Invalid choice.\n");
}
}
}
```

b. Two-leveldirectoryStructure

ALGORITHM:

1. Start
2. Declarethenumber,names andsizeof thedirectoriesandsubdirectories andfilenames.
3. Get the values for the declared variables.
4. Display the files that are available in the directories and subdirectories. 5. Stop.

PROGRAM:

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

structUserDirectory{
char username[20];
char files[10][20];
int file_count;
};
```

```
void twoLevelDirectory() {
structUserDirectoryusers[5];
int user_count = 0;
int choice;
charusername[20],filename[20];
int uIndex = -1;

printf("TwoLevelDirectoryImplementation\n");

while (1) {
printf("\n1.CreateUserDirectory\n2.CreateFile\n3.DeleteFile\n4.ListFiles\n5.Exit\nEnterchoice: ");
scanf("%d", &choice);

switch(choice){
case 1:
printf("Enternewusername:");
scanf("%s", username);
int exists = 0;
for (int i = 0; i < user_count; i++) {
if(strcmp(username,users[i].username)==0){
exists = 1;
break;
}
}
if (exists)
printf("User already exists!\n");
```

```
else {
strcpy(users[user_count].username,username);
users[user_count].file_count = 0;
user_count++;
printf("User directory created.\n");
}
break;
case 2:
printf("Enterusername:");
scanf("%s", username);
uIndex = -1;
for (int i = 0; i < user_count; i++) {
if(strcmp(username,users[i].username)==0){
uIndex = i;
break;
}
}
if (uIndex == -1) {
printf("Usernotfound.\n");
break;
}
printf("Enterfilenametocreate:");
scanf("%s", filename);
int fExists = 0;
for (int j = 0; j < users[uIndex].file_count; j++) {
if(strcmp(filename,users[uIndex].files[j])==0){
fExists = 1;
```

```
break;
}
}
if (fExists)
printf("Filealreadyexists.\n");
else {
strcpy(users[uIndex].files[users[uIndex].file_count],filename);
users[uIndex].file_count++;
printf("File created.\n");
}
break;
case 3:
printf("Enterusername:");
scanf("%s", username);
uIndex = -1;
for (int i = 0; i < user_count; i++) {
if(strcmp(username,users[i].username)==0){
uIndex = i;
break;
}
}
if (uIndex == -1) {
printf("Usernotfound.\n");
break;
}
printf("Enterfilenametodelete:");
scanf("%s", filename);
```

```
int fIndex = -1;
for (int j = 0; j < users[uIndex].file_count; j++) {
if(strcmp(filename,users[uIndex].files[j])==0){
fIndex = j;
break;
}
}
if (fIndex == -1)
printf("Filenotfound.\n");
else {
for(intj=fIndex;j<users[uIndex].file_count-1;j++){
strcpy(users[uIndex].files[j], users[uIndex].files[j + 1]);
}
users[uIndex].file_count--;
printf("File deleted.\n");
}
break;
case 4:
for (int i = 0; i < user_count; i++) {
printf("User:%s\n",users[i].username);++
for(intj=0;j<users[i].file_count;j++){
printf("- %s\n", users[i].files[j]);
}
}
break;
case5:
return;
```

```
default:
printf("Invalid choice.\n");
}
}
}
```

Result:

Thus,theFileOrganizationTechnique-SingleandTwoleveldirectoryisexecutedsuccessfully