BASIC COMMANDS IN UNIX

1. GENERAL PURPOSE COMMANDS

i) To display the present working directory

Syntax: $ pwd

ii) To clear the screen

Syntax: $ clear

iii) To calculate the values

Syntax: $ bc

iv) Uname: To know your machine name

-n: Tells machine name in network

Syntax: $ uname –n

v) To display the version number of the OS

Syntax: $ uname –r

vi) To display the current date

Syntax: $ date

vii) To display the list of users logged in

Syntax: $ who

viii) To display the current status details of our login

Syntax: $ who am i

ix) To compile and run shell program

Syntax: $ sh filename

Example: $ sh odd.sh

x) To compile a C program

Syntax: $ cc –o filename.c

xi) To run a C program

Syntax: $ ./a.out

xii) To display the calendar.

Syntax: $ cal

xiii) To display the previous, current and next month.

Syntax: $ cal -3

xiv) To display the current month starting from Sunday.

Syntax: $ cal –s

xv) To display month only.

Syntax: $ date+%m

Output: 11

xvi) To display month name and month

Syntax: $date +%h%m

Output: Nov 11

xvii) To display month name

Syntax: $ date+%h

Output: Nov

xviii) To display the time in hours

Syntax: $ date+%H

Output: 16

xix) To display the time in minutes

Syntax: $ date+%M

Output: 50

xx) To display the time in AM or PM

Syntax: $ date+%r

Output: 16: 50:40 PM

xxi) To display date of month

Syntax: $ date+%d

Output: 25

xxii) It is used to view more details of all the commands

Syntax: $ man command_name

Example: $ man date

Output: It displays more details of date command

xxiii) To display my login id

Syntax: $ log name

Output: cs1010

2. DIRECTORY COMMANDS

i) To create a directory

Syntax: $ mkdirdirname

Example: $ mkdir peacock

ii) To change the name of the directory

Syntax: $ cd dirname

Example: $ cd flower

iii) To remove the directory

Syntax: $ rmdirdirname

Example: $ rmdir flower

3. FILE COMMANDS

i) To create a file.

Syntax: $ cat>filename

Example: $ cat>ex1

ii) To view the content of the file.

Syntax: $ cat filename

Example: $ cat ex1

iii) To append some details with the existing details in the file

Syntax: $ cat>>filename

Example: $ cat>>ex1

iv) To concatenate multiple files

Syntax: $ cat file1 file2 > file3

Example: $ cat computer compiler>world

Output: The contents of computer and compiler are merged into a single file called world.

v) To know the list of all files in directory

Syntax: $ ls

vi) To copy the file to another file

Syntax: $ cp source destination

Example: $ cp ex1 ex2

vii) To rename the file

Syntax: $ mv oldfilenewfile

Example: $ mv ex1 ex3

viii) To delete a file

Syntax: $ rm filename

Example: $ rm ex1

ix) To delete all files

Syntax: $ rm *

x) To display the filename starting with single letter

Syntax: $ echo?

xi) To display the filename starting with two letters

Syntax: $ echo??

xii) To display the filename starting with the letter f

Syntax: $ echo f*

xiii) To display the filename ending with letter f.

Syntax: $ echo *f

xiv) To display first 10 lines

Syntax: $ head [count][filename]

Example: $ head 10 ex1

xv) To display first 6 characters

Syntax: $ head -6c filename

Example: $ head -6c ex1

xvi) To display 5 lines from 2 files

Syntax: $ head -5 file1 file2

Example: $ head -5 ex1 ex2

xvii) To display last 10 lines

Syntax: $ tail [count][filename]

Example: $ tail 10 ex3

xviii) To display the number of words in a file

Syntax: $ wc filename

Example: $ wc ex1

xix) To display the number of characters in a file

Syntax: $ wc –c filename

Example: $ wc –c ex1

xx) To display the number of lines

Syntax: $ wc –l filename

Example: $ wc –l ex3

xxi) To display number of lines with numbers

Syntax: $ nl filename

Example: $ nl ex1

xxii) To provide the line number starting from s

Syntax: $ nl –v3 filename

Example: $ nl –v3 ex3

xxiii) To increment the line number by 5

Syntax: $ nl –i5 filename

Example: $ nl –i5 ex3

xxiv) To reverse and sort the content of file

Syntax: $ sort –r filename

Example: $ sort –r ex1

xxv) To sort the content of the file

Syntax: $ sort filename

Example: $ sort ex1

xxvi) To sort and remove the duplicate

Syntax: $ sort –u filename

Example: $ sort –u ex1

xxvii) To display file contents page by page

Syntax: $ more filename

APPLICATION I

Create a directory called stud, change to the stud directory. Verify whether you have changed to stud directory. Return to your original directory

APPLICATION II

Create a file called top display the first three and last three lines of a file top in the directory

APPLICATION III

Create a parent directory dept. Create sub-directory and files using the following tree diagram and perform the following task.

Dept

CSE	IT	ECE	EEE
STUDENT	STUDENT	STUDENT	STUDENT
STAFF	STAFF	STAFF	STAFF

1. Rename the directory
2. Create a new directory and copy the content of CSE to new directory
3. Delete all directories and files

SHELL PROGRAMMING

GREATEST OF TWO NUMBERS

PROGRAM:

```
echo "Enter the two numbers"
read a b
if [ $a -gt $b ]
then
echo "$a is greater"
else
echo "$b is greater"
fi
```

OUTPUT:

```
[cse2a   @localhost~]$sh greater.sh
Enter the two numbers
10 40
40 is greater
```

FIBONACCI SERIES

PROGRAM:

```
echo "Enter the value of n"
read n
f1=-1
f2=1
i=1
while [ $i -le $n ]
do
f3=`expr $f1 + $f2`
echo $f3
f1=$f2
f2=$f3
i=`expr $i + 1`
done
```

OUTPUT:

```
[cse2a    @localhost~]$sh fibo.sh
Enter the value of n
5
0
1
1
2
3
```

FACTORIAL OF A GIVEN NUMBER

PROGRAM:

```
echo "Enter the number"
read num
fact=1
i=1
while [ $i -le $num ]
do
fact=`expr $fact \* $i`
i=`expr $i + 1`
done
echo "The factorial of $num=$fact"
```

OUTPUT:

```
[cse2a    @localhost~]$sh FACT.sh
Enter the number
5
The factorial of 5 =120
```

SUM OF n NUMBERS

PROGRAM:

```
echo "Enter the number"
read num
i=1
sum=0
while [ $i -le $num ]
do
sum=`expr $sum + $i`
i=`expr $i + 1`
done
echo "The sum is $sum"
```

OUTPUT:

```
[cse2a    @localhost~]$sh sum.sh
Enter the number
10
The sum is 55
```

SORTING OF n NUMBERS

PROGRAM:

```
echo "Sorting of n numbers"
echo "Enter the numbers"
cat>p
echo "ascending order"
sort -n p
echo "Descending Order"
sort -nr p
```

OUTPUT:

```
[cse2a   @localhost~]$sh sort.sh
Sorting of n numbers
Enter the numbers
12
23
25
24
26
15
14
16
13 (Ctrl+D)
ascending order
12
13
14
15
16
23
24
25
```

26

Descending Order

26

25

24

23

16

15

14

13

1

CHECKING FOR A PRIME NUMBER

PROGRAM:

```
echo “Enter the number”
read num
k=2
i=0
c=num/2
while[$k –lt $c]
do
r=$[n%k]
if[$r –eq $i]
then
echo “Composite”
exit
else
k=$[k+1]
fi
done
echo “Prime no”
```

OUTPUT:

```
[cse2a@localhost~]$ sh prime.sh
Enter the number
11
Prime no
25
Composite
```

MENU DRIVEN DIRECTORY ACTIONS

PROGRAM:

```
echo "1.Display a long list of file"
echo "Menu"
echo "2.Delete files from the directory"
echo "3.Append a file"
read choice
case $choice in
1) ls -l;;
2) echo "Enter the file to be deleted"
read file
rm $file
echo "The file has been deleted";;
3) echo "Enter the name of the file to be inserted"
read new
echo "Enter the text"
cat>>new
echo "The file has been appended";;
esac
```

OUTPUT:

```
 [cse2a    @localhost~]$ cat>new
 Good Morning
 [cse2a    @localhost~]$sh menu.sh
Menu
1.Display a long list of file
2.Delete files from the directory
3.Append a file
1
total 100
-rw-rw-r--. 1 netlabnetlab   236 2013-01-05 19:54 FACT.sh
-rw-rw-r--. 1 netlabnetlab   149 2013-01-05 19:52 FACT.sh~
```

-rw-rw-r--. 1 netlabnetlab 215 2013-01-05 19:49 fibo.sh

-rw-rw-r--. 1 netlabnetlab 137 2013-01-05 19:48 fibo.sh~

-rw-rw-r--. 1 netlabnetlab 193 2013-01-05 19:40 greater.sh

-rw-rw-r--. 1 netlabnetlab 110 2013-01-05 19:39 greater.sh~

-rw-rw-r--. 1 netlabnetlab 378 2013-01-05 20:06 menu.sh

-rw-rw-r--. 1 netlabnetlab 13 2013-01-05 20:07 new

-rw-rw-r--. 1 netlabnetlab 32801 2013-01-05 19:20 OPERATING SYSTEM LAB EXERCISES.odt

-rw-rw-r--. 1 netlabnetlab 13979 2013-01-05 19:55 OPERATING SYSTEM LAB program.odt

-rw-rw-r--. 1 netlabnetlab 0 2013-01-05 20:06 p

-rw-rw-r--. 1 netlabnetlab 326 2013-01-05 20:01 sort.sh

-rw-rw-r--. 1 netlabnetlab 127 2013-01-05 20:00 sort.sh~

-rw-rw-r--. 1 netlabnetlab 212 2013-01-05 19:58 sum.sh

-rw-rw-r--. 1 netlabnetlab 133 2013-01-05 19:56 sum.sh~

output:

[cse2a @localhost]$sh menu.sh

Menu

1.Display a long list of file

2.Delete files from the directory

3.Append a file

3

Enter the name of the file to be inserted

new

Enter the text

Have a Nice Day

The file has been appended

[cse2a @localhost]$ cat new

Good Morning

Have a Nice Day

[cse2a @localhost]$sh menu.sh

Menu

1.Display a long list of file

2.Delete files from the directory

3.Append a file

2

Enter the file to be deleted

hello

rm: cannot remove `hello': No such file or directory

The file has been deleted

IMPLEMENTATION OF FORK(),GETPID(), EXECLP()

PROGRAM:

```
#include<stdio.h>
#include<sys/types.h>
#define MAX_COUNT 5
void parentprocess();
void childprocess();
void main()
{
        pid_tpid;
        pid=fork();
        if(pid==0)
                childprocess();
        else
                parentprocess();
}
void childprocess()
{
        printf("The output for execlpsyscall is:\n");
        execlp("/bin/ls","ls",NULL);
}
void parentprocess()
{
        int i,k;
        k=getpid();
        wait();
        for(i=1;i<=MAX_COUNT;i++)
                printf("The line from parent value=%d\n",i);
        printf("The parent process is done\n");
        printf("getpid() value of process id is %d\n",k);
        exit(0);
}
```

OUTPUT:

```
[cse2a @localhost~]$ cc system1.c
[cse2a @localhost~]$ ./a.out
The output for execlp syscall is:
a.out                    image processing4.pdf          priority.c
bestfit.c                image processing5.pdf          priority.c~
bestfit.c~               indexed.c                      Resume.doc
cloud1.pdf               indexed.c~                     Resume_new1.doc
contiguous1.c            index.ods                      round.c
contiguous.c             ipc1.c                         round.c~
contiguous.c~            ipc2.c                         semaphores.c
FACT.sh                  ipc3.c                         semaphores.c~
FACT.sh~                 ipc3.c~                        sjf.c
fcfs.c                   linked.c                       sjf.c~
fcfs.c~                  lrupaging.c                    sort.sh
fibo.sh                  lrupaging.c~                   sort.sh~
fibo.sh~                 ls.c                           sum.sh
fifopaging.c             ls.c~                          sum.sh~
fifopaging.c~            menu.sh                        system1.c
firstfit.c               menu.sh~                       system2.c
firstfit.c~              msgrec.c                       system2.c~
for.c                    msgsend.c                      system3.c
fork1.c                  new                            system3.c~
fork1.c~                 NEW1                           system4.c
greater.sh               NEW2                           system4.c~
greater.sh~              OPERATING SYSTEM LAB EXERCISES.odt  system5.c
grep.c                   OPERATING SYSTEM LAB program.  system5.c~
grep.c~                  OperatingSystems(OS)Lab.pdf    system calls-1.odt
image processing01.pdf   paging.c                       system calls.odt
image processing02.pdf   paging.c~                      worstfit.c
image processing3.pdf    pipe.c                         worstfit.c~
The line from parent value=1
The line from parent value=2
The line from parent value=3
The line from parent value=4
```

The line from parent value=5
The parent process is done
getpid() value of process id is 2147

IMPLEMENTATION OF OPENDIR(), CLOSEDIR()

PROGRAM:

```
#include<stdio.h>
#include<sys/types.h>
#include<dirent.h>
void main()
{
        DIR *dir;
        struct dirent *entry;
        int count;
        if((dir=opendir("/"))==NULL)
                perror("opendir()error");
        else
        {
                count=0;
                while((entry=readdir(dir))!=NULL)
                {
                        printf("Directory Entry %3d=%s\n",++count,entry->d_name);
                }
                closedir(dir);
        }
}
```

OUTPUT:

```
[cse2a    @localhost~]$ cc system2.c
[cse2a   @localhost~]$ ./a.out
Directory Entry   1=.dbus
Directory Entry   2=root
Directory Entry   3=etc
Directory Entry   4=..
Directory Entry   5=.pulse-cookie
Directory Entry   6=proc
```

Directory Entry 7=.pulse

Directory Entry 8=.autofsck

Directory Entry 9=sbin

Directory Entry 10=home

Directory Entry 11=usr

Directory Entry 12=dev

Directory Entry 13=lib

Directory Entry 14=media

Directory Entry 15=opt

Directory Entry 16=.

Directory Entry 17=var

Directory Entry 18=bin

Directory Entry 19=srv

Directory Entry 20=lost+found

Directory Entry 21=sys

Directory Entry 22=tmp

Directory Entry 23=boot

Directory Entry 24=mnt

Directory Entry 25=selinux

IMPLEMENTATION OF STAT()

PROGRAM:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<time.h>
struct stat nfile;
int main(int argc, char *argv[])
{
        char fname[20];
        printf("Enter the filename: ");
        scanf("%s",fname);
        stat (fname,&nfile);
        int flag;
        flag=stat(fname, &nfile);
        if (flag==-1)
        {
                printf("File does not exist");
        }
        else
        {
                printf("File Exists and Filename is given\n");
                printf("The information about the file %s\n\n",fname);
                printf("%s has %d link\n",fname,nfile.st_nlink);
                printf("%s has %d devices\n",fname,nfile.st_dev);
                printf("%s has %d inodes\n",fname,nfile.st_ino);
                printf("%s has %d inode devices\n",fname,nfile.st_rdev);
                printf("%s has %d size\n",fname,nfile.st_size);
                printf("%s has %d owner\n",fname,nfile.st_gid);
                printf("%s has %d block size\n",fname,nfile.st_blocks);
                printf("%s has %d time\n",fname,nfile.st_atime);
```

```
                printf("%s has %d time\n",fname,nfile.st_mtime);
        }
}
```

OUTPUT:

```
[cse2a   @localhost~]$ cc system3.c
[cse2a   @localhost~]$ ./a.out
Enter the filename: new
File Exists and Filename is given
The information about the file new

new has 1 link
new has 2054 devices
new has 3580183 inodes
new has 0 inode devices
new has 29 size
new has 502 owner
new has 8 block size
new has 1357553666 time
new has 1357396651 time
```

IMPLEMENTATION OF WAIT()

PROGRAM:

```
#include<stdio.h>
#include<unistd.h>
main()
{
        int id, cid, stat;
        if((id=fork())==0)
        {
                printf("\nParent Id is %d \nStatus Id is %d\n", getpid(), stat);
                execl("/bin/date", "date", 0);
        }
        cid=wait(&stat);
        printf("\n Id is %d Status is %d", cid, stat);
}
```

OUTPUT:

```
[cse2a   @localhost~]$ cc system4.c
[cse2a    @localhost~]$ ./a.out
Parent Id is 2235
Status Id is 0
Mon Jan  7 15:54:03 IST 2013
Id is 2235 Status is 0
```

INPUT OUTPUT SYSTEM CALL

PROGRAM:

```
#include<stdio.h>
#include<sys/types.h>
#include<sys/stat.h>
#include<fcntl.h>
#include<stdlib.h>
int main(int argc, char *argv[])
{
        int n, fd;
        char buff[50];
        if(argc<2)
        {
                printf("Input file not specified %s filename \n",argv[0]);
                exit(0);
        }
        fd=open(argv[1],0);
        if(fd!=-1)
        {
                while ((n=read(fd, buff, sizeof(buff)))>0)
                        write (1, buff, n);
        }
        else
        {
                char ch;
                printf ("File does not exist....\n");
                printf("Do u wish to create a new file\n");
                scanf ("%c", &ch);
                if(ch=='y' || ch=='Y')
                {
                        int fd1=creat(argv[1],2);
```

```
                        char attr[25]= "chmod +rw ";
                        strcat(attr, argv[1]);
                        system(attr);
                        n=read(1, buff, sizeof(buff));
                        write (fd1,buff, n);
                        close(fd1);
                        fd=open(argv[1],0);
                        printf("Contents\n");
                        while((n=read(fd1, buff, sizeof(buff)))>0)
                                write(1, buff, n);
                }
        }
}
```

OUTPUT:

```
[cse2b    @localhost~]$ cc system5.c
[cse2b    @localhost~]$ ./a.out new
Good Morning
Have a Nice Day

[cse2b    @localhost~]$ cc system5.c
[cse2a    @localhost~]$ ./a.out NEW1
[cse2a    @localhost~]$ ./a.out NEW2
File does not exist....
Do u wish to create a new file
y
Have a nice day
Contents
Have a nice day
[cse2b    @localhost~]$ cat NEW2
Have a nice day
```

CREATING A PROCESS USING FORK

PROGRAM:

```
#include<stdio.h>
#include<time.h>
int fib(int);
int fork(void);
void sleep(unsigned);
int main()
{
        int begin=time(NULL);
        int i;
        if(fork()==0)
                for(i=0;i<10;i++)
                        printf("File(%2d)=%d\n",i,fib(i));
        else
                for(i=0;i<10;i++)
                {
                        sleep(2);
                        printf("\nElapsed time=%d\n",time(NULL)-begin);
                }
        return(0);
}
int fib(int n)
{
        if(n<=1)
                return n;
        else
                return (fib(n-1)+fib(n-2));
}
```

OUTPUT:

```
[cse2a  @localhost~]$ cc fork1.c
[cse2a    @localhost~]$ ./a.out
File( 0)=0                File( 1)=1                File( 2)=1
File( 3)=2                File( 4)=3                File( 5)=5
File( 6)=8                File( 7)=13               File( 8)=21
File( 9)=34
Elapsed time=2            Elapsed time=4            Elapsed time=6
Elapsed time=8            Elapsed time=10           Elapsed time=12
Elapsed time=14           Elapsed time=16           Elapsed time=18
Elapsed time=20
```

SIMULATION OF LS COMMAND

PROGRAM:

```
#include<dirent.h>
#include<stdio.h>
int main()
{
        struct dirent **namelist;
        int n,i;
        char pathname[100];
        getcwd(pathname,100);
        n=scandir(pathname,&namelist,0,alphasort);
        if(n<0)
                printf("\nError\n");
        else
        {
                for(i=0;i<n;i++)
                        printf("%s\t",namelist[i]->d_name);
        }
}
```

OUTPUT:

```
[cse2a   @localhost~]$ cc ls.c
[cse2a   @localhost~]$ ./a.out
.       ..      .~lock.OPERATING SYSTEM LAB program.#   .~lock.index.ods#       FACT.sh
        FACT.sh~        NEW1  NEW2  OPERATING SYSTEM LAB EXERCISES.odt
OPERATING SYSTEM LAB program.   Resume.doc    Resume_new1.doc    a.out
fibo.shfibo.sh~         fork1.c fork1.c~        greater.sh      greater.sh~     index.ods       ls.c
        menu.sh         menu.sh~        new     sort.sh sort.sh~        sum.sh sum.sh~system
calls-1.odt     system calls.odt        system1.c       system2.c       system2.c~      system3.c
        system3.c~      system4.c       system4.c~      system5.c       system5.c~
```

SIMULATION OF GREP COMMAND

PROGRAM:

```
#include<stdio.h>
#include<string.h>
void print();
int count=0,count1=0,notfound=0,flag=0,linec=1;
char line[50];
int main()
{
        FILE *fp;
        char ch,s[50],line[50];
        int n,i;
        fp=fopen("for.c","r");
        printf("\nEnter the sequence ");
        scanf("%s",s);
        n=strlen(s);
        while(1)
        {
                ch=fgetc(fp);
                line[count1]=ch;
                count1=count1+1;
                if(ch=='\n'&&flag==1)
                        print();
                if(ch==s[count])
                {
                        count=count+1;
                        if(count==n)
                        {
                                flag=1;
                        }
                }
```

```
                else if(ch==EOF)
                        break;
                else if(ch=='\n')
                {
                        count1=0;
                        linec=linec+1;
                }
                else if(ch!=s[count])
                        count=0;
        }
        fclose(fp);
        if(notfound==0)
                printf("\nSequence not found in the line");
}
void print()
{
        int i;
        printf("\nSequence found in the line %d\n",linec);
        for(i=0;i<count1;i++)
                printf("%c",line[i]);
        flag=0;
        notfound=1;
        count=0;
        count1=0;
}
```

OUTPUT:

```
[cse2a    @localhost~]$ cat>for.c
Have A Nice Day
Be Happy Forever
Good Morning
```

Good Evening

Good Day

Good Bye

[cse2a @localhost~]$ cc grep.c

[cse2a @localhost~]$ ./a.out

Enter the sequence: Have

Sequence found in the line 1

[cse2a @localhost~]$ ./a.out

Enter the sequence: Good

Sequence found in the line 3

Sequence found in the line 4

Sequence found in the line 5

Sequence found in the line 6

FIRST COME FIRST SERVER (FCFS) SCHEDULING ALGORITHM

PROGRAM:

```
#include<stdio.h>
void main()
{
	int a[25],n,i,wt=0,swt=0,tt=0,stt=0;
	printf("\nFIRST COME FIRST SERVE SCHEDULING PROCESS\n");
	printf("Enter the no. of jobs: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nService Time or Burst Time for a process %d: ",i);
		scanf("%d",&a[i]);
	}
	printf("\nProcess\t\tBurst Time\t\tWaiting Time\t\tTurn Around Time");
	for(i=0;i<n;i++)
	{
		swt=swt+wt;
		tt=tt+a[i];
		stt=stt+tt;
		printf("\nP%d\t\t%d\t\t\t%d\t\t\t%d",i,a[i],wt,tt);
		wt=wt+a[i];
	}
	printf("\n\n\tAverage Waiting Time=%d",swt/n);
	printf("\n\n\tAverage Turn Around Time=%d",stt/n);
}
```

OUTPUT:

```
[cse2a    @localhost~]$ cc fcfs.c
[cse2a    @localhost~]$ ./a.out
```

FIRST COME FIRST SERVE SCHEDULING PROCESS

Enter the no. of jobs: 3

Service Time or Burst Time for a process 0: 24

Service Time or Burst Time for a process 1: 3

Service Time or Burst Time for a process 2: 3

Process	Burst Time	Waiting Time	Turn Around Time
P0	24	0	24
P1	3	24	27
P2	3	27	30

Average Waiting Time=17

Average Turn Around Time=27

SHORTEST JOB FIRST (SJF) SCHEDULING ALGORITHM

PROGRAM:

```
#include<stdio.h>
void main()
{
	int a[25],p[25],j,t,n,i,wt=0,swt=0,tt=0,stt=0;
	printf("\nSHORTEST JOB NEXT SCHEDULING PROCESS\n");
	printf("Enter the no. of jobs: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nService Time or Burst Time for a process %d: ",i);
		scanf("%d",&a[i]);
		p[i]=i;
	}
	for(i=0;i<n;i++)
	{
		for(j=i+1;j<n;j++)
		{
			if(a[i]>a[j])
			{
				t=a[i];
				a[i]=a[j];
				a[j]=t;
				t=p[i];
			}
		}
	}
	printf("\nProcess\t\tBurst Time\t\tWaiting Time\t\tTurn Around Time");
	for(i=0;i<n;i++)
	{
		swt=swt+wt;
```

```
            tt=tt+a[i];
            stt=stt+tt;
            printf("\nP%d\t\t%d\t\t\t%d\t\t\t%d",i,a[i],wt,tt);
            wt=wt+a[i];
        }
        printf("\n\n\tAverage Waiting Time=%d",swt/n);
        printf("\n\n\tAverage Turn Around Time=%d",stt/n);
}
```

OUTPUT:

```
cse2a  @localhost~]$ cc sjf.c
[cse2a    @localhost~]$ ./a.out
SHORTEST JOB NEXT SCHEDULING PROCESS
Enter the no. of jobs: 4
Service Time or Burst Time for a process 0: 5
Service Time or Burst Time for a process 1: 10
Service Time or Burst Time for a process 2: 8
Service Time or Burst Time for a process 3: 3
```

Process	Burst Time	Waiting Time	Turn Around Time
P0	3	0	3
P1	5	3	8
P2	8	8	16
P3	10	16	26

```
Average Waiting Time=6
Average Turn Around Time=13
```

PRIORITY SCHEDULING

PROGRAM:

```
#include<stdio.h>
void main()
{
	void interchange(int *,int *);
	int s[25],pos[25],p[25],n,j,i,wt=0,swt=0,tt=0,stt=0;
	printf("\nPRIORITY SCHEDULING PROCESS\n");
	printf("Enter the no. of jobs: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nService Time or Burst Time for a process %d: ",i);
		scanf("%d",&s[i]);
		printf("\nEnter the Priority %d: ",i);
		scanf("%d",&p[i]);
		pos[i]=i;
	}
	printf("\nProcess\tBurst Time\tPriority\tWaiting Time\tTurn Around Time");
	for(i=0;i<n;i++)
	{
		for(j=i+1;j<n;j++)
		{
			if(p[i]>p[j])
			{
				interchange(&s[i],&s[j]);
				interchange(&p[i],&p[j]);
				interchange(&pos[i],&pos[j]);
			}
		}
	}
	for(i=0;i<n;i++)
```

```
        {
                swt=swt+wt;
                tt=tt+s[i];
                stt=stt+tt;
                printf("\nP%d\t%d\t\t%d\t\t%d\t\t\t%d",i,s[i],p[i],wt,tt);
                wt=wt+s[i];
        }
        printf("\n\n\tAverage Waiting Time=%d",swt/n);
        printf("\n\n\tAverage Turn Around Time=%d",stt/n);
}

void interchange(int *a, int *b)
{
        int t;
        t=*a;
        *a=*b;
        *b=t;
}
```

OUTPUT:

```
[cse2a    @localhost~]$ cc priority.c
[cse2a    @localhost~]$ ./a.out

PRIORITY SCHEDULING PROCESS
Enter the no. of jobs: 3
Service Time or Burst Time for a process 0: 24
Enter the Priority 0: 2
Service Time or Burst Time for a process 1: 3
Enter the Priority 1: 1
Service Time or Burst Time for a process 2: 3
Enter the Priority 2: 3
Process         Burst Time      PriorityWaiting Time   Turn Around Time
```

P0	3	1	0	3
P1	24	2	3	27
P2	3	3	27	30

Average Waiting Time=10

Average Turn Around Time=20

ROUND ROBIN SCHEDULING ALGORITHM

PROGRAM:

```
#include<stdio.h>
int tt,i,j,temp;
void main()
{
	int btime[10],n,x,z;
	printf("\nROUND ROBIN SCHEDULING ALGORITHM\n");
	printf(  "\nEnter the no. of process: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("\nEnter Burst time for the process %d: ",i);
		scanf("%d",&btime[i]);
	}
	printf("Process Name\t\tBurst Time\n");
	for(i=0;i<n;i++)
		printf("P%d\t\t\t%d\n",i,btime[i]);
	z=1;
	while(z==1)
	{
		tt=0;
		printf("Press 1. Round Robin 2. Exit\n");
		scanf("%d",&x);
		switch(x)
		{
			case 1:
				rrobin(btime,n);break;
			case 2:
				exit(0);
			default:
				printf("Invalid Option");
```

```
                }
                printf("\n\nIf you want to continue press 1.");
                scanf("%d",&z);
        }
}
rrobin(int btime[],int n)
{
        int tslice,j=0;
        printf("Enter the Time Slicing: ");
        scanf("%d",&tslice);
        printf("Process Name\tRemaining Time\tTotal Time");
        while(j<n)
        {
                for(i=0;i<n;i++)
                {
                        if(btime[i]>0)
                        {
                                if(btime[i]>=tslice)
                                {
                                        tt+=tslice;
                                        btime[i]=btime[i]-tslice;
                                        printf("\nP%d\t\t%d\t\t%d",i,btime[i],tt);
                                }
                                else
                                {
                                        tt+=btime[i];
                                        btime[i]=0;
                                        printf("\nP%d\t\t%d\t\t%d",i,btime[i],tt);
                                }
                        }
                }
                j++;
```

```
        }
}
```

OUTPUT:

```
[cse2a    @localhost~]$ cc round.c
[cse2a    @localhost~]$ ./a.out
ROUND ROBIN SCHEDULING ALGORITHM
Enter the no. of process: 4
Enter Burst time for the process 0: 8
Enter Burst time for the process 1: 4
Enter Burst time for the process 2: 6
Enter Burst time for the process 3: 2
Process Name        Burst Time
P0                  8
P1                  4
P2                  6
P3                  2
Press 1. Round Robin 2. Exit
1
Enter the Time Slicing: 2
Process Name Remaining Time      Total Time
P0           6                   2
P1           2                   4
P2           4                   6
P3           0                   8
P0           4                   10
P1           0                   12
P2           2                   14
P0           2                   16
P2           0                   18
P0           0                   20
If you want to continue press 1.0
```

PRODUCER CONSUMER PROBLEM - SEMAPHORES

PROGRAM:

```
#include<stdio.h>
#include<string.h>
# define SIZE 10
struct process
{
	char a[10];
}
buffer[10];
int mutex=1, full=0, empty=SIZE, flag=0;
int wait(int);
int signal(int);
main()
{
	int ch,i;
	printf("\nPRODUCER - CONSUMER PROBLEM\n");
	while(1)
	{
		printf("\nChoices are\n");
		printf("1. Producer Routine\n");
		printf("2. Consumer Routine\n");
		printf("3. Display the contents of the buffer\n");
		printf("4. Exit\n");
		printf("Enter your choice: ");
		scanf("%d",&ch);
		switch(ch)
		{
			case 1:
				empty=wait(empty);
				mutex=wait(mutex);
				if(flag==0)
				{
					printf("\nEnter the item to be added: ");
					scanf("%s",&buffer[full]);
```

```
                printf("Item inserted successfully\n\n");
                full=signal(full);
        }
        else
        {
                printf("Buffer Full\n");
                flag=0;
        }
        mutex=signal(mutex);
        break;
case 2:
        full=wait(full);
        mutex=wait(mutex);
        if(flag==0)
        {
                printf("\n\nItem %s is successfullyconsumed\n",
                        buffer[0].a)
                for(i=0;i<SIZE;i++)
                        strcpy(buffer[i].a,buffer[i+1].a);
                        empty=signal(empty);
                }
        else
        {
                printf("No item is in buffer\n\n");
                flag=0;
        }
        mutex=signal(mutex);
        break;
case 3:
        if(full!=0)
        {
                for(i=0;i<full;i++)
                printf("%s\n", buffer[i].a);
        }
        else
```

```
                                printf("Buffer is empty\n\n");
                                break;

                        default:
                                printf("Please enter a valid option\n");
                }
        }
}

int wait(int s)
{
        if(s==0)
                flag=1;
        else
                s--;
        return s;
}

int signal(int s)
{
        s++;
        return s;
}
```

OUTPUT:

```
[cse2a@localhost]$ cc semaphores.c
[cse2a@localhost]$ ./a.out
PRODUCER - CONSUMER PROBLEM
Choices are
1. Producer Routine
2. Consumer Routine
3. Display the contents of the buffer
4. Exit
Enter your choice: 1
Enter the item to be added: p1
```

Item inserted successfully

Choices are

1. Producer Routine

2. Consumer Routine

3. Display the contents of the buffer

4. Exit

Enter your choice: 1

Enter the item to be added: p2

Item inserted successfully

Choices are

1. Producer Routine

2. Consumer Routine

3. Display the contents of the buffer

4. Exit

Enter your choice: 1

Enter the item to be added: p3

Item inserted successfully

Choices are

1. Producer Routine

2. Consumer Routine

3. Display the contents of the buffer

4. Exit

Enter your choice: 2

Item p1 is successfully consumed

Choices are

1. Producer Routine

2. Consumer Routine

3. Display the contents of the buffer

4. Exit

Enter your choice: 3

p2

p3

Choices are

1. Producer Routine

2. Consumer Routine

3. Display the contents of the buffer

4. Exit

Enter your choice: 4

INTERPROCESS COMMUNICATION – MESSAGE QUEUES

PROGRAM:

```
#include<stdio.h>
#include<string.h>
#include<sys/ipc.h>
#include<sys/msg.h>
char name[50];
int main()
{
        int pid,len,qid;
        struct
        {
                long mtype;
                char mtext[50];
        }message;
        system("clear");
        printf("\n Inter Process communication using message queue \n");
        qid = msgget((key_t)13,IPC_CREAT|0666);
        printf("\n\nMessage queue is created \n");
        if(qid==-1)
        {
                printf("\nMessage queue is not created \n");
                exit(1);
        }
        printf("\nThe value of qid is %d",qid);
        printf("\nEnter the message for communication: ");
        scanf("%s",name);
        strcpy(message.mtext,name);
        message.mtype=1;
        len=strlen(message.mtext);
        pid=fork();
```

```
        printf("\n The value of pid is %d \n",pid);
        if(pid==0)
        {
                msgsnd(qid,&message,len,IPC_NOWAIT);
                printf("\n\n MESSAGE SEND BY CHILD PROCESS\n");
        }
        if(pid>0)
        {
                wait();
                msgrcv(qid,&message,strlen(message.mtext),0,IPC_NOWAIT|
                MSG_NOERROR);
                printf("\n MESSAGE RECEIVED BY PARENT PROCESS\n");
                printf("\n Received message is %s \n",message.mtext);
        }
        if(pid==-1)
        {
                printf("\n Error in creating a child \n");
                exit(1);
        }
}
```

OUTPUT:

```
[cse2b    @localhost~]$ cc ipcmsg.c
[cse2b    @localhost~]$ ./a.out
Inter Process communication using message queue
Message queue is created
The value of qid is 0
Enter the message for communication: Hello
The value of pid is 8517
The value of pid is 0
MESSAGE SEND BY CHILD PROCESS
MESSAGE RECEIVED BY PARENT PROCESS
```

Received message is hello

INTERPROCESS COMMUNICATION – PIPES

PROGRAM:

```
#include<stdio.h>
main()
{
	int pid,ppid,pdes[2],n,a;
	printf("\nInterprocess communication using pipes");
	if(pipe(pdes)==-1)
	{
		printf("\n\tThere is no error");
		exit(1);
	}
	else
	{
		pid=fork();
		if(pid==-1)
			printf("\nThere is an error in the process");
		else if(pid)
		{
			close(pdes[0]);
			printf("\nParent is %d\n",ppid);
			printf("\nProcess is %d\n",pid);
			printf("\nEnterno.of process");
			scanf("%d",&a);
			write(pdes[1],&a,sizeof(a));
		}
		else
		{
			close(pdes[1]);
			open(pdes[0]);
			read(pdes[0],&n,sizeof(a));
```

```
                        if(n%2==0)
                                printf("\nThe number %d is even",n);
                }
        }
}
```

OUTPUT:

```
[cse2b    @localhost~]$ cc ipc3.c
[cse2b    @localhost~]$ ./a.out
Interprocess communication using pipes
Parent is 13860852
Process is 3976
Enter no.of process88
Interprocess communication using pipes
The number 88 is even
```

INTERPROCESS COMMUNICATION – SHARED MEMORY

PROGRAM:

```
#include<sys/types.h>
#include<sys/shm.h>
#include<sys/ipc.h>
main()
{       int shmid;
        key_t key=0x10;
        shmid=shmget(key,100,IPC_CREAT|0666);
        if( shmid<0 )
                printf("\nFirst SHMID failed\n");
        else
                printf("\nFirst SHMID Succeded id=%d \n",shmid);
        shmid=shmget(key,101,IPC_CREAT|0666);
        if(shmid<0)
                printf("\nSecond SHMID failed\n");
        else
                printf("\nSecond SHMID Succeded id=%d \n",shmid);
        shmid=shmget(key,90,IPC_CREAT|0666);
        if(shmid<0)
                printf("\nThird SHMID failed\n");
        else
                printf("\n Third SHMID Succeded id=%d \n",shmid);
}
```

OUTPUT:

```
[cse2b    @localhost~]$ cc ipcshared.c
[cse2b    @localhost~]$ ./a.out
First SHMID Succeded id=753682
Second SHMID failed
Third SHMID Succeded id=753682
```

DEAD LOCK AVOIDANCE

IMPLEMENT BANKERS ALGORITHM FOR DEAD LOCK AVOIDANCE

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
int main()
{
intMax[10][10], need[10][10], alloc[10][10], avail[10], completed[10], safeSequence[10];
int p, r, i, j, process, count;
count =0;
printf("Enter the no of processes : ");
scanf("%d",&p);
for(i =0; i< p; i++)
completed[i]=0;
printf("\n\nEnter the no of resources : ");
scanf("%d",&r);
printf("\n\nEnter the Max Matrix for each process : ");
for(i =0; i < p; i++)
{
printf("\nFor process %d : ", i +1);
for(j =0; j < r; j++)
scanf("%d",&Max[i][j]);
}
printf("\n\nEnter the allocation for each process : ");
for(i =0; i < p; i++)
{
printf("\nFor process %d : ",i +1);
for(j =0; j < r; j++)
scanf("%d",&alloc[i][j]);
}
printf("\n\nEnter the Available Resources : ");
for(i =0; i < r; i++)
scanf("%d",&avail[i]);
for(i =0; i < p; i++)
for(j =0; j < r; j++)
need[i][j]=Max[i][j]- alloc[i][j];
do
{
printf("\n Max matrix:\tAllocation matrix:\n");
for(i =0; i < p; i++)
{
for( j =0; j < r; j++)
printf("%d ",Max[i][j]);
printf("\t\t");
for( j =0; j < r; j++)
printf("%d ", alloc[i][j]);
printf("\n");
```

```
}
process =-1;
for(i =0; i < p; i++)
{
if(completed[i]==0)//if not completed
{
process = i ;
for(j =0; j < r; j++)
{
if(avail[j]< need[i][j])
{
process =-1;
break;
}
}
}
if(process !=-1)
break;
}
if(process !=-1)
{
printf("\nProcess %d runs to completion!", process +1);
safeSequence[count]= process +1;
count++;
for(j =0; j < r; j++)
{
avail[j]+= alloc[process][j];
alloc[process][j]=0;
Max[process][j]=0;
completed[process]=1;
}
}
}
while(count != p && process !=-1);
{
if(count == p)
{
printf("\nThe system is in a safe state!!\n");
printf("Safe Sequence : < ");
for( i =0; i < p; i++)
printf("%d ", safeSequence[i]);
printf(">\n");
}
else
printf("\nThe system is in an unsafe state!!");
}
}
```

OUTPUT:

```
[cse2a@local host~]# gcc bankerssafesequence.c
[cse2a@local host~]#./a.out
Enter the no of processes :5
Enter the no of resources :3
Enter the MaxMatrixfor each process :
For process 1:7
5
3
For process 2:3
2
2
For process 3:7
0
2
For process 4:2
2
2
For process 5:4
3
3
Enter the allocation for each process :
For process 1:0
1
0
For process 2:2
0
0
For process 3:3
0
2
For process 4:2
1
1
For process 5:0
0
2
Enter the AvailableResources:3
3
2
Max matrix:Allocation matrix:
753010
322200
702302
222211
433002
Process2 runs to completion!
Max matrix:Allocation matrix:
753010
```

000000
702302
222211
433002

Process3 runs to completion!
Max matrix:Allocation matrix:
753010
000000
000000
222211
433002

Process4 runs to completion!
Max matrix:Allocation matrix:
753010
000000
000000
000000
433002

Process1 runs to completion!
Max matrix:Allocation matrix:
000000
000000
000000
000000
433002

Process5 runs to completion!
The system is in a safe state!!
SafeSequence:<23415>

AN ALGORITHM TO IMPLEMENT DEAD LOCK DETECTION

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
void main()
{
int found,flag,l,p[4][5],tp,c[4][5],i,j,k=1,m[5],r[5],a[5],temp[5],sum=0;
clrscr();
printf("enter total no of processes");
scanf("%d",&amp;tp);
printf("enter clain matrix");
for(i=1;i&lt;=4;i++)
for(j=1;j&lt;=5;j++)
{
scanf("%d",&amp;c[i][j]);
}
printf("enter allocation matrix");
for(i=1;i&lt;=4;i++)
for(j=1;j&lt;=5;j++)
{
scanf("%d",&amp;p[i][j]);
}
printf("enter resource vector:\n");
for(i=1;i&lt;=5;i++)
{
scanf("%d",&amp;r[i]);
}
printf("enter availability vector:\n");
for(i=1;i&lt;=5;i++)
{
scanf("%d",&amp;a[i]);
temp[i]=a[i];
}
for(i=1;i&lt;=4;i++)
{
sum=0;
for(j=1;j&lt;=5;j++)
{
sum+=p[i][j];
}
if(sum==0)
{
m[k]=i;
k++;
}
}
for(i=1;i&lt;=4;i++)
{
```

```
for(l=1;l&lt;k;l++)
if(i!=m[l])
{
flag=1;
for(j=1;j&lt;=5;j++)
if(c[i][j]&gt;temp[j])
{
flag=0;
break;
}
}
if(flag==1)
{
m[k]=i;
k++;
for(j=1;j&lt;=5;j++)
temp[j]+=p[i][j];
}
}
printf("deadlock causing processes are:");
for(j=1;j&lt;=tp;j++)
{
found=0;
for(i=1;i&lt;k;i++)
{
if(j==m[i])
found=1;
}
if(found==0)
printf("%d\t",j);
}
getch();
}
```

INPUT:

```
Enter total no.of processes : 4
Enter chain matrix:
0 1 0 0 1
0 0 0 0 1
1 0 1 0 1
Enter allocation matrix:
1 0 1 1 0
1 1 0 0 0
0 0 0 1 0
0 0 0 0 0
Enter resource vector:
2 1 1 2 1
Enter availability vector:
```

0 0 0 0 1

OUTPUT:
Deadlock causing processes are: 1 2

IMPLEMENT THREADING &SYNCHRONIZATION APPLICATIONS

PROGRAM:

```
 #include<pthread.h>
#include<stdio.h>
#define NUM_THREADS 5
void *printhello(void *threadid)
{
long tid;
tid=(long)threadid;
printf("hello world ! it's me,thread#%d!\n",tid);
pthread_exit(NULL);
}
int main(int argc,char *argv[])
{
pthread_t threads[NUM_THREADS];
int rc;
long t;
for(t=0;t<NUM_THREADS;t++)
{
printf("in main: creating thread %ld\n",t);
rc=pthread_create(&threads[t],NULL,printhello,(void *)t);
if(rc)
printf("ERROR:return code from pthread_create() is %d \n",rc);
}
pthread_exit(NULL);
}
```

OUTPUT:

```
[cse2a@sys-d14 ~]$ cc -pthread threading.c
[cse2a@sys-d14 ~]$ ./a.out
in main: creating thread 0
in main: creating thread 1
hello world ! it's me,thread#1!
hello world ! it's me,thread#0!
in main: creating thread 2
in main: creating thread 3
hello world ! it's me,thread#2!
in main: creating thread 4
hello world ! it's me,thread#3!
hello world ! it's me,thread#4
```

MEMORY ALLOCATION METHODS

BEST FIT MEMORY MANAGEMENT

PROGRAM:

```
#include<stdio.h>
void main()
{
	int a[20],p[20],i,j,n,m,temp,b[20],temp1,temp2,c[20];
	printf("\nMEMORY MANAGEMENT SCHEME - BEST FIT \n");
	printf("Enter No. of Blocks: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("Enter the %dst block size: ",i);
		scanf("%d",&a[i]);
	}
	printf("Enter No. of Process: ");
	scanf("%d",&m);
	for(i=0;i<m;i++)
	{
		printf("Enter the size of %dst process: ",i);
		scanf("%d",&p[i]);
	}
	for(i=0;i<n;i++)
	{
		for(j=0;j<m;j++)
		{
			if(a[i]>a[j])
			{
				temp=a[i];
				a[i]=a[j];
				a[j]=temp;
```

```
                }
                if(p[i]>p[j])
                {
                        temp=p[i];
                        p[i]=p[j];
                        p[j]=temp;
                }
        }
    }
    printf("\nProcess\tBlock Size\n");
    for(i=0;i<n;i++)
        printf("%d\t%d\n",p[i],a[i]);
    printf("\n\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
                if(p[j]<=a[i])
                {
                        printf("\nThe process %d [size %d] allocated to block
                        %d\n", j,p[j],a[i]);
                        p[j]=10000;
                        break;
                }
        }
    }
    for(j=0;j<m;j++)
        if(p[j]!=10000)
                printf("\nThe process %d is not allocated\n",p[j]);
}
```

OUTPUT:

```
[cse2a@localhost~]$ cc bestfit.c
[cse2a@localhost~]$ ./a.out
MEMORY MANAGEMENT SCHEME - BEST FIT
Enter No. of Blocks: 5
Enter the 0st block size: 500
Enter the 1st block size: 120
Enter the 2st block size: 180
Enter the 3st block size: 250
Enter the 4st block size: 350
Enter No. of Process: 5
Enter the size of 0st process: 600
Enter the size of 1st process: 100
Enter the size of 2st process: 160
Enter the size of 3st process: 400
Enter the size of 4st process: 300
Process         Block Size
600                     500
400                     350
300                     250
160                     180
100                     120

The process 1 [size 400] allocated to block 500
The process 2 [size 300] allocated to block 350
The process 3 [size 160] allocated to block 250
The process 4 [size 100] allocated to block 180
The process 600 is not allocated
```

FIRST FIT MEMORY MANAGEMENT

PROGRAM:

```
#include<stdio.h>
void main()
{
	int a[20],p[20],i,j,n,m,temp,b[20],temp1,temp2,c[20];
	printf("\nMEMORY MANAGEMENT SCHEME - FIRST FIT \n");
	printf("Enter No. of Blocks: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("Enter the %dst block size: ",i);
		scanf("%d",&a[i]);
	}
	printf("Enter No. of Process: ");
	scanf("%d",&m);
	for(i=0;i<m;i++)
	{
		printf("Enter the size of %dst process: ",i);
		scanf("%d",&p[i]);
	}
	printf("\nProcess\tBlock Size\n");
	for(i=0;i<n;i++)
		printf("%d\t\t%d\n",p[i],a[i]);
	printf("\n\n");
	for(i=0;i<n;i++)
	{
		for(j=0;j<m;j++)
		{
			if(p[j]<=a[i])
			{
```

```
                        printf("The process %d [size %d]allocated to block
                        %d\n",j,p[j],a[i]);
                        p[j]=10000;
                                break;
                }
        }
}
for(j=0;j<m;j++)
        if(p[j]!=10000)
                printf("The process %d is not allocated\n",j);
}
```

OUTPUT:

```
[cse2a@localhost~]$ cc firstfit.c
[cse2a@localhost~]$ ./a.out
MEMORY MANAGEMENT SCHEME - FIRST FIT
Enter No. of Blocks: 5
Enter the 0st block size: 120
Enter the 1st block size: 230
Enter the 2st block size: 340
Enter the 3st block size: 450
Enter the 4st block size: 560
Enter No. of Process: 5
Enter the size of 0st process: 530
Enter the size of 1st process: 430
Enter the size of 2st process: 630
Enter the size of 3st process: 203
Enter the size of 4st process: 130
Process         Block Size
530             120
430             230
630             340
203             450
```

130 560

The process 3 [size 203]allocated to block 230

The process 4 [size 130]allocated to block 340

The process 1 [size 430]allocated to block 450

The process 0 [size 530]allocated to block 560

The process 2 is not allocated

WORST FIT MEMORY MANAGEMENT

PROGRAM:

```
#include<stdio.h>
void main()
{
	int a[20],p[20],i,j,n,m,temp,b[20],temp1,temp2,c[20];
	printf("\nMEMORY MANAGEMENT SCHEME - WORST FIT \n");
	printf("Enter No. of Blocks: ");
	scanf("%d",&n);
	for(i=0;i<n;i++)
	{
		printf("Enter the %dst block size: ",i);
		scanf("%d",&a[i]);
	}
	printf("Enter No. of Process: ");
	scanf("%d",&m);
	for(i=0;i<m;i++)
	{
		printf("Enter the size of %dst process: ",i);
		scanf("%d",&p[i]);
	}
	for(i=0;i<n;i++)
	{
		for(j=0;j<m;j++)
		{
			if(a[i]>a[j])
			{
				temp=a[i];
				a[i]=a[j];
				a[j]=temp;
			}
			if(p[i]<p[j])
```

```
                {
                        temp=p[i];
                        p[i]=p[j];
                        p[j]=temp;
                }
        }
    }
    printf("\nProcess\tBlock Size\n");
    for(i=0;i<n;i++)
        printf("%d\t%d\n",p[i],a[i]);
    printf("\n\n");
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            if(p[j]<=a[i])
            {
                printf("\nThe process %d [size %d] allocated to block
                %d\n",j,p[j],a[i]);
                p[j]=10000;
                break;
            }
        }
    }
    for(j=0;j<m;j++)
        if(p[j]!=10000)
            printf("\nThe process %d is not allocated\n",p[j]);
}
```

OUTPUT

```
[cse2a@localhost~]$ cc worstfit.c
[cse2a@localhost~]$ ./a.out
```

MEMORY MANAGEMENT SCHEME - WORST FIT

Enter No. of Blocks: 5

Enter the 0st block size: 520

Enter the 1st block size: 147

Enter the 2st block size: 236

Enter the 3st block size: 289

Enter the 4st block size: 600

Enter No. of Process: 5

Enter the size of 0st process: 196

Enter the size of 1st process: 245

Enter the size of 2st process: 570

Enter the size of 3st process: 145

Enter the size of 4st process: 600

Process	Block Size
145	600
196	520
245	289
570	236
600	147

The process 0 [size 145] allocated to block 600

The process 1 [size 196] allocated to block 520

The process 2 [size 245] allocated to block 289

The process 570 is not allocated

The process 600 is not allocated

PAGING TECHNIQUE OF MEMORY MANAGEMENT
PAGING

PROGRAM:

```
#include<stdio.h>
void main()
{
int imem[100][100],pmem[500],ptable[100],psize,n,i,j,phyadd;
printf("\nEnter the no. of pages: ");
scanf("%d",&n);
printf("\nEnter the page size: ");
scanf("%d",&psize);
printf("\nEnter the data values to be stored: ");
for(i=0;i<n;i++)
for(j=0;j<psize;j++)
scanf("%d",&imem[i][j]);
printf("\nEnter the base address for each page: ");
for(i=0;i<n;i++)
scanf("%d",&ptable[i]);
printf("\nLogical memory\n");
printf("\nPage no\tOffset\tValues\n");
for(i=0;i<n;i++)
for(j=0;j<psize;j++)
printf("\n\t%d\t%d\t%d\n",i,j,imem[i][j]);
printf("\nPage table\n");
printf("\nIndex\tbase address ");
for(i=0;i<n;i++)
printf("\n%d\t%d",i,ptable[i]);
printf("\nPhysical Memory\n");
printf("\nLocation\tvalue\tpage no\n");
for(i=0;i<n;i++)
for(j=0;j<psize;j++)
{
phyadd=(ptable[i]*psize)+j;
pmem[phyadd]=imem[i][j];
printf("\n%d\t%d\tpage%d",phyadd,imem[i][j],i);
}
}
```

OUTPUT:

```
[cse2a@localhost~]$ cc paging.c
[cse2a@localhost~]$ ./a.out

Enter the no. of pages: 3
Enter the page size: 3
Enter the data values to be stored: 1 2 3 4 5 6 7 8 9
Enter the base address for each page: 100 101 102
Logical memory

Page no Offset Values
```

0	0	1
0	1	2
0	2	3
1	0	4
1	1	5
1	2	6
2	0	7
2	1	8
2	2	9

Page table Index base address

0	100
1	101
2	102

Physical Memory

Location	value	page no
300	1	page0
301	2	page0
302	3	page0
303	4	page1
304	5	page1
305	6	page1
306	7	page2
307	8	page2
308	9	page2

IMPLEMENT E ALL PAGE REPLACEMENT ALGORITHMS
FIFO PAGE REPLACEMENT

PROGRAM:

```
#include<stdio.h>
void main()
{
int n,i,j,a[50],frame[10],no,k,avail,count=0;
printf("\nEnter the no. of page: ");
scanf("%d",&n);
printf("\nEnter the page no: ");
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\nEnter the no. of frame: ");
scanf("%d",&no);
for(i=0;i<no;i++)
frame[i]=-1;
j=0;
printf("\n Refernce string\t Page frame\n");
for(i=0;i<n;i++)
{
printf("%d\t",a[i]);
avail=0;
for(k=0;k<no;k++)
if(frame[k]==a[i])
avail=1;
if(avail==0)
{
frame[j]=a[i];
j=(j+1)%no;
count++;
for(k=0;k<no;k++)
printf("%d\t",frame[k]);
}
printf("\n");
}
printf("\nPage fault id %d",count);
}
```

OUTPUT:

```
[netlab@system-2 oslab]$ cc fifopaging.c
[netlab@system-2 oslab]$ ./a.out

Enter the no. of page: 20
Enter the page no: 3
5
1
2
1
```

7
0
6
0
7
8
9
4
1
5
6
1
4
7
5

Enter the no. of frame: 3

Refernce string			Page frame
3	3	-1	-1
5	3	5	-1
1	3	5	1
2	2	5	1
1			
7	2	7	1
0	2	7	0
6	6	7	0
0			
7			
8	6	8	0
9	6	8	9
4	4	8	9
1	4	1	9
5	4	1	5
6	6	1	5
1			
4	6	4	5
7	6	4	7
5	5	4	7

Page fault id 16

LRU PAGE REPLACEMENT

PROGRAM:

```
#include<stdio.h>
#define frame 10
#define framesize 10
int arr[frame][framesize];
int page[framesize];
int main()
{
int i,j,k,n,b;
int failure=0,found;
for(i=0;i<frame;i++)
for(j=0;j<framesize;j++)
arr[i][j]=-1;
printf("\nEnter the no. of frame available: ");
scanf("%d",&b);
printf("\nEnter the no. of pageframe to be called: ");
scanf("%d",&n);
printf("\nEnter the no. of page frame array: ");
for(i=0;i<n;i++)
scanf("%d",&page[i]);
arr[0][0]=page[0];
failure++;
for(i=1;i<n;i++)
{
for(j=0;j<b;j++)
{
found=0;
if(arr[i][i-1]==page[i])
{
found=1;
break;
}
}
if(found)
{
for(k=b;k>0;k--)
arr[k][i]=arr[k-1][i-1];
arr[k][i]=page[i];
failure++;
}
else
{
arr[0][i]=page[i];
for(k=1;k<b;k++)
if(k>j)arr[k][i]=arr[k][i-1];
else
arr[k][i]=arr[k-1][i-1];
```

```
}
}
printf("\n Page replacement using LRU\n");
for(i=0;i<b;i++)
{
for(j=0;j<n;j++)
if(arr[i][j]!=-1)
printf("%2d   ",arr[i][j]);
printf("\n");
}
printf("\nNo. of replacement failed: %d out of %d",failure,n);
printf("\nNo. of replacement success: %d out of %d",n-failure,n);
}
```

OUTPUT:

```
[cse2a@localhost~]$ cc lrupaging.c
[cse2a@localhost~]$ ./a.out

Enter the no. of frame available: 3
Enter the no. of pageframe to be called: 10
Enter the no. of page frame array: 7 0 1 2 0 3 0 4 5 6
Page replacement using LRU
7   0   1   2   0   3   0   4   5   6
7   0   1   2   0   3   0   4   5
7   0   1   2   0   3   0   4

No. of replacement failed: 1 out of 10
No. of replacement success: 9 out of 10
```

LFU PAGE REPLACEMENT

Program:

```
#include<stdio.h>
int main()
{
  int f,p;
  int pages[50],frame[10],hit=0,count[50],time[50];
  int i,j,page,flag,least,minTime,temp;

  printf("Enter no of frames : ");
  scanf("%d",&f);
  printf("Enter no of pages : ");
  scanf("%d",&p);

  for(i=0;i<f;i++)
  {
   frame[i]=-1;
  }
  for(i=0;i<50;i++)
  {
   count[i]=0;
  }
  printf("Enter page no : \n");
  for(i=0;i<p;i++)
  {
   scanf("%d",&pages[i]);
  }
  printf("\n");
  for(i=0;i<p;i++)
  {
   count[pages[i]]++;
   time[pages[i]]=i;
   flag=1;
   least=frame[0];
   for(j=0;j<f;j++)
   {
     if(frame[j]==-1 || frame[j]==pages[i])
     {
       if(frame[j]!=-1)
       {
          hit++;
       }
     flag=0;
     frame[j]=pages[i];
     break;
     }
     if(count[least]>count[frame[j]])
     {
```

```
            least=frame[j];
      }
    }
    if(flag)
    {
      minTime=50;
      for(j=0;j<f;j++)
      {
        if(count[frame[j]]==count[least] && time[frame[j]]<minTime)
        {
          temp=j;
          minTime=time[frame[j]];
        }
      }
      count[frame[temp]]=0;
      frame[temp]=pages[i];
    }
    for(j=0;j<f;j++)
    {
      printf("%d ",frame[j]);
    }
    printf("\n");
  }
  printf("Page hit = %d",hit);
  return 0;
}
```

Output:

```
[cse2a@localhost~]$ cc.lfupaging.c
[cse2a@localhost~]$./a.out
Enter the no.of frames:5
Enter no.of pages:20
Enter page no:3
5
1
2
1
7
0
6
0
7
8
9
4
1
5
6
1
```

4
7
5
3 -1 -1
3 5 -1
3 5 1
2 5 1
2 5 1
2 7 1
0 7 1
0 6 1
0 6 1
0 7 1
0 8 1
0 9 1
0 4 1
0 4 1
0 5 1
0 6 1
0 6 1
0 4 1
0 7 1
0 5 1
Page hit =4

SINGLE LEVEL DIRECTORY ORGANIZATION

program:

```
#include<stdio.h>
int main()
{
int master,s[20];
char f[20][20][20];
char d[20][20];
int i,j;
printf("Enter the number of directories:");
scanf("%d",&master);
printf("Enter names of directories:");
for(i=0;i<master;i++)
scanf("%s",&d[i]);
printf("Enter size of directories:");
for(i=0;i<master;i++)
scanf("%d",&s[i]);
printf("Enter the filenames:");
for(i=0;i<master;i++)
{
for(j=0;j<s[i];j++)
scanf("%s",&f[i][j]);
}
printf("\n");
printf("Directory \t size \t Filename\n");
printf("_____\n");
for(i=0;i<master;i++)
{
printf("%s \t\t  %2d  \t",d[i],s[i]);
for(j=0;j<s[i];j++)
printf("%s\n\t\t   ",f[i][j]);
printf("\n");
```

```
}
printf("\t\n");
}
```

OUTPUT:

```
[cse2a@localhost ~]$ cc sld.c
[cse2a@localhost ~]$ ./a.out
Enter the number of directories:2
Enter names of directories: bin
 root
Enter size of directories:2
2
Enter the filenames:tarzan
godzilla
eragon
VIP
Directory        size      Filename
_____________________________________
bin                2       tarzan
                           godzilla
root               2       eragon
                             VIP
```

TWO LEVEL DIRECTORY

program:

```
#include<stdio.h>
struct st
{
char dname[10];
char sdname[10][10];
char fname[10][10][10];
int ds,sds[10];
}dir[10];
int main()
{
int i,j,k,n;
printf("enter number of directories:");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("enter directory %d name:",i+1);
scanf("%s",&dir[i].dname);
printf("enter size of directory %d:",i+1);
scanf("%d",&dir[i].ds);
for(j=0;j<dir[i].ds;j++)
{
printf("enter subdirectory name and size:");
scanf("%s",&dir[i].sdname[j]);
scanf("%d",&dir[i].sds[j]);
for(k=0;k<dir[i].sds[j];k++)
{
printf("enter file names:");
scanf("%s",&dir[i].fname[j][k]);
} } }
printf("\n");
```

```
printf("\ndirname\t\tsize\tsubdirname\tsize\tfiles");
printf("\n*****************************************************\n");
for(i=0;i<n;i++)
{
printf("%s\t\t%d",dir[i].dname,dir[i].ds);
for(j=0;j<dir[i].ds;j++)
{
printf("\t%s\t\t%d\t",dir[i].sdname[j],dir[i].sds[j]);
for(k=0;k<dir[i].sds[j];k++)
printf("\n\t\t");
}
printf("\n");
printf("%s\t",dir[i].fname[j][k]);
}
}
```

OUTPUT:

```
[cse2a@localhost ~]$ cc tld.c
[cse2a@localhost ~]$ ./a.out
enter number of directories:2
enter directory 1 name:bin
enter size of directory 1:2
enter subdirectory name and size:branch 2
enter file names:roadrash
enter file names:tarzan
enter subdirectory name and size:branch1 1
enter file names:claw
enter directory 2 name:root
enter size of directory 2:1
enter subdirectory name and size:root1 1
enter file names:mortal

dirname       size     subdirname     size     files
```

****

bin	2	branch	2	roadrash	tarzan
		branch1	1	claw	
root	1	root1	1	mortal	

FILE ALLOCATION STRATEGIES

CONTIGUOUS ALLOCATION

PROGRAM:

```
#include<string.h>
#include<stdio.h>
struct block
{
        int b_id;
        int b_allocated;
};
struct block b[50];
int main()
{
        int no,i,j,n,sblock,eblock,count=0;
        int psize,flag=1,pname,bname;
        printf("\nEnter the no. of block: ");
        scanf("%d",&n);
        for(i=0;i<n;i++)
        {
                b[i].b_id=i;
                b[i].b_allocated=0;
        }
        printf("\nEnter the no. of block already exists: ");
        scanf("%d",&no);
        for(i=0;i<no;i++)
        {
                printf("\nEnter the block: ");
                scanf("%d",&bname);
                b[bname].b_allocated=100;
        }
        printf("\nEnter the process name: ");
```

```
scanf("%d",&pname);
printf("\nEnter the process size: ");
scanf("%d",&psize);
for(i=0;i<n;i++)
{
        if(b[i].b_allocated==0)
        {
                count=count+1;
                if(count==1)
                        sblock=i;
                if(count==psize)
                {
                        eblock=i;
                        for(j=sblock;j<=eblock;j++)
                                b[j].b_allocated=pname;
                        printf("\n\nProcess %d is allocated %d blocks from %d to
                        %d\n\n",pname,sblock,eblock,sblock);
                        i=n+1;
                        flag=1;
                }
        }
        else
        {
                count=0;
                flag=0;
        }
}
if(flag==0)
printf("\nProcess not allocated");
count=0;
for(i=0;i<n;i++)
{
```

```
                count=count+1;
                if(count<no)
                        printf("b[%d]->%d\t\t",b[i].b_id,b[i].b_allocated);
                else
                {
                        count=0;
                        printf("\n\n\nb[%d]-->%d\t\t",b[i].b_id,b[i].b_allocated);
                        count=count+1;
                }
        }
        return 0;
}
```

OUTPUT:

```
[cse2a@localhost~]$ cc contiguous.c
[cse2a@localhost~]$ ./a.out
Enter the no. of block: 4
Enter the no. of block already exists: 4
 Enter the block: 2
Enter the block: 3
Enter the block: 5
Enter the block: 6
Enter the process name: 2
Enter the process size: 2
Process 2 is allocated 0 blocks from 1 to 0
b[0]->2         b[1]->0         b[2]->100         b[3]-->100
```

LINKED ALLOCATION

PROGRAM:

```
#include<stdio.h>
void display();
struct block
{
        int p_id;
        int b_allocated;
        struct block *next;
}*sblock=NULL;
int n;
struct block b[50];
int main()
{
        int i,pname,psize,no,bname,freespace=0,count,previous;
        printf("\nEnter the no. of blocks: ");
        scanf("%d",&n);
        freespace=n;
        for(i=0;i<n;i++)
        {
                b[i].p_id=0;
                b[i].b_allocated=0;
        }
        printf("\nEnter the no. of blocks already allocated process name: ");
        scanf("%d",&no);
        for(i=0;i<no;i++)
        {
                printf("\nEnter the block: ");
                scanf("%d",&bname);
                b[bname].p_id=100;
                b[bname].b_allocated=1;
                freespace=freespace-1;
```

```
}
display();
printf("\nEnter process name: ");
scanf("%d",&pname);
printf("\nEnter process size: ");
scanf("%d",&psize);
count=0;
if(psize<=freespace)
{
        for(i=0;i<n;i++)
        {
                if(b[i].b_allocated==0)
                {
                        b[i].b_allocated=1;
                        b[i].p_id=pname;
                        count++;
                        if(count==1)
                                sblock=&b[i];
                        else
                                if(count>psize)
                                        break;
                                else
                                        b[previous].next=&b[i];
                        previous=i;
                }
        }
        display();
        printf("\nProcess name %d\nStarting address %u\nEnding address
        %u",pname,sblock,&b[i]);
}
else
        printf("\nProcess cannot be allocated");
```

```
}
void display()
{
	int i,count=0;
	printf("\nBlocks\n");
	for(i=0;i<n;i++)
	{
		if(count<2)
			printf("%u-->\tb[%2d]->\t%2d->\t%2d->\t
			%4u\t\t",&b[i],i,b[i].p_id,b[i].b_allocated,b[i].next);
		else
		{
			count=0;
			printf("\n%u-->\tb[%2d]->\t%2d->\t%2d->\t
			%4u\t\t",&b[i],i,b[i].p_id,b[i].b_allocated,b[i].next);
		}
		count=count+1;
	}
}
```

OUTPUT:

```
[cse2a@localhost~]$ cc linked.c
[cse2a@localhost~]$ ./a.out
Enter the no. of blocks: 8
Enter the no. of blocks already allocated process name: 3
Enter the block: 1
Enter the block: 3
Enter the block: 5
Blocks
134519744-->b[ 0]->0->0->0          134519756-->b[ 1]->100->1->0
134519768-->b[ 2]->0->0->0          134519780-->b[ 3]->100->1->0
134519792-->b[ 4]->0->0->0          134519804-->b[ 5]->100->1->0
134519816-->b[ 6]->0->0->0          134519828-->b[ 7]->0->0->0
```

Enter process name: 2

Enter process size: 2

Blocks

134519744-->b[0]->2->1->134519768 134519756-->b[1]->100->1->0

134519768-->b[2]->2->1->0 134519780-->b[3]->100->1->0

134519792-->b[4]->2->1->0 134519804-->b[5]->100->1->0

134519816-->b[6]->0->0->0 134519828-->b[7]->0->0->0

Process name 2

Starting address 134519744

Ending address 134519792

INDEXED ALLOCATION

PROGRAM:

```
#include<stdio.h>
int b_allocated[30];
void display(int);
void main()
{
        int
    j=0,index[50],indexloc,i,n,no,psize,pname,block,freespace=0,count=0;
    printf("\nEnter the number of blocks: ");
    scanf("%d",&n);
    freespace=n;
    printf("\nEnter the number of blocks that are already allocated: ");
    scanf("%d",&no);
    for(i=0;i<no;i++)
        b_allocated[i]=0;
    printf("\nEnter the blocks allocated: ");
    for(i=0;i<no;i++)
    {
        scanf("%d",&block);
        b_allocated[block]=1;
        freespace=freespace-1;
    }
    printf("\nFreespace=%d\n",freespace);
    display(n);
    printf("\nEnter the name of the process: ");
    scanf("%d",&pname);
    printf("\nEnter the size of the process: ");
    scanf("%d",&psize);
    if(psize<freespace)
    {
        for(i=0;i<n;i++)
        {
            if(b_allocated[i]==0&&count<=psize)
            {
```

```
                        if(count==psize)
                        {
                                b_allocated[i]=pname;
                                indexloc=i;
                                break;
                        }
                        else
                        {
                                b_allocated[i]=1;
                                index[j]=0;
                                index[j]=i;
                                j=j+1;
                                count=count+1;
                        }
                }
        }
        display(n);
        printf("\nBlocks stored in index are: \n\n");
        for(i=0;i<j;i++)
                printf("%d",index[i]);
        printf("\nDirectory Structure\n\n");
        printf("\nProcess\t\tIndex\n");
        printf("\n%d\t\t%d",pname,indexloc);
    }
}
void display(int n)
{
    int i,count=0;
    printf("\n\n");
    for(i=0;i<n;i++)
        if(count<3)
        {
                printf("\t\tb[%2d]-->%d",i,b_allocated[i]);
                count=count+1;
        }
```

```
                else
                {
                        printf("\n\n\t\tb[%2d]--%d",i,b_allocated[i]);
                        count=1;
                }
}
```

OUTPUT:

```
[cse2a@localhost~]$ cc indexed.c
[cse2a@localhost~]$ ./a.out
Enter the number of blocks: 8
Enter the number of blocks that are already allocated: 4
Enter the blocks allocated: 1 3 2 5
Freespace=4
            b[ 0]-->0            b[ 1]-->1            b[ 2]-->1

            b[ 3]--1             b[ 4]-->0            b[ 5]-->1

            b[ 6]--0             b[ 7]-->0
Enter the name of the process: 3
Enter the size of the process: 2
            b[ 0]-->1            b[ 1]-->1            b[ 2]-->1

            b[ 3]--1             b[ 4]-->1            b[ 5]-->1

            b[ 6]--3             b[ 7]-->0
Blocks stored in index are:
04
Directory Structure
Process        Index

3              6
```