Ex No: 1

BASIC UNIX COMMANDS

1.1 GENERAL PURPOSE COMMANDS

1. The 'date' command:

The date command display the current date with day of week, month, day, time (24 hours clock) and the year.

SYNTAX: $ date

The date command can also be used with following format.

Format	Purpose	Example
+ %m	To display only month	$ date + %m
+ %h	To display month name	$ date + %h
+ %d	To display day of month	$ date + %d
+ %y	To display last two digits of the year	$ date + %y
+ %H	To display Hours	$ date + %H
+ %M	To display Minutes	$ date + %M
+ %S	To display Seconds	$ date + %S

2. The echo'command:

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

SYNTAX: $ echo

EXAMPLE: $ echo "God is Great"

3. The 'cal' command:

The cal command displays the specified month or year calendar.

SYNTAX: $ cal [month] [year]

EXAMPLE: $ cal Jan 2012

4. The 'bc' command:

Unix offers an online calculator and can be invoked by the command bc.

SYNTAX: $ bc

5. The 'who' command

The who command is used to display the data about all the users who are currently logged into the system.

SYNTAX: $ who

6. The ‘who am i’ command

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

SYNTAX: $ who am i

7. The ‘id’ command

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

SYNTAX: $ id

8. The ‘tty’ command

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

SYNTAX: $ tty

9. The ‘clear’ command

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

SYNTAX: $ clear

10. The ‘man’ command

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

SYNTAX: $ man [command]

11. The ‘ps’ command

The ps command is used to the process currently alive in the machine with the 'ps' (process status) command, which displays information about process that are alive when you run the command. 'ps;' produces a snapshot of machine activity.

SYNTAX: $ ps

12. The ‘uname’ command

The uname command is used to display relevant details about the operating system on the standard output.

-m -> Displays the machine id (i.e., name of the system hardware)

-n -> Displays the name of the network node. (host name)

-r -> Displays the release number of the operating system.

-s -> Displays the name of the operating system (i.e.. system name)

-v -> Displays the version of the operating system.

-a -> Displays the details of all the above five options.

SYNTAX: $ uname [option]

13. The ‘finger’ command

The finger command with an argument gives you more information about the user. The finger command followed by an argument can give complete information for a user who is not logged onto the system.

SYNTAX: $ finger [user-name]

EXAMPLE: $ finger eee61

1.2 DIRECTORY COMMANDS

1. The 'pwd' command:

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

SYNTAX: $ pwd

2. The 'mkdir' command:

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

SYNTAX: $ mkdir dirname

EXAMPLE: $ mkdir receee

3. The 'rmdir' command:

The rmdir is used to remove a directory from the disk. Before removing a directory, the directory must be empty (no files and directories).

SYNTAX: $ rmdir dirname

EXAMPLE: $ rmdir receee

4. The 'cd' command:

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

SYNTAX: $ cd dirname

EXAMPLE: $ cd receee

5. The 'ls' command:

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

SYNTAX: $ ls

1.3 FILE HANDLING COMMANDS

1. The 'cat' command:

The cat command is used to create a file.

SYNTAX: $ cat > filename

EXAMPLE: $ cat > rec

2. The 'Display contents of a file' command:

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

SYNTAX: $ cat filename

3. The ‘cp’ command:

The cp command is used to copy the contents of one file to another and copies the file from one place to another.

SYNTAX: $ cp oldfile newfile

EXAMPLE: $ cp cse ece

4. The ‘rm’ command:

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

SYNTAX: $ rm filename

EXAMPLE: $ rm rec

5. The ‘mv’ command:

The mv command is used to move a file from one place to another. It removes a specified file from its original location and places it in specified location.

SYNTAX: $ mv oldfile newfile

EXAMPLE: $ mv cse eee

6. The ‘file’ command:

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

SYNTAX: $ file filename

EXAMPLE: $ file receee

7. The ‘wc’ command:

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

SYNTAX: $ wc filename

EXAMPLE: $ wc receee

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

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

SYNTAX: $ ls > filename

EXAMPLE: $ ls > cseeee

9. The ‘pipes’ command:

The Unix allows us to connect two commands together using these pipes. A pipe (|) is an mechanism by which the output of one command can be channeled into the input of another command.

SYNTAX: $ command1 | command2

EXAMPLE: $ who | wc –l

10. The 'tee' command:

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

SYNTAX: $ command | tee filename

EXAMPLE: $ who | tee sample | wc -l

11. The 'Metacharacters of unix' command:

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

* - Specifies number of characters

?- Specifies a single character

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

! – Used to Specify Not

EXAMPLE:

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

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

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

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

12. The 'File permissions' command:

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

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

r-read **w-write** **x-execute**

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

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

EXAMPLE: $ ls college

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

Where,

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

Lak Specifies Owner of the file.

r-x Indicates the absence of the write permission by the Group owner of the file.

Std Is the Group Owner of the file.

r-- Indicates read permissions for others.

13. The 'chmod' command:

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

SYNTAX: $ chmod category operation permission file

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

EXAMPLE:

$ chmod u –wx college

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

$ chmod u +rw, g+rw college

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

$ chmod g=wx college

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

14. The 'Octal Notations' command:

The file permissions can be changed using octal notations also. The octal notations for file permission are

Read permission	4
Write permission	2

Execute permission	1

EXAMPLE:

$ chmod 761 college

Assigns all permission to the owner, read and write permissions to the group and only executable permission to the others for 'college' file.

1.4 GROUPING COMMANDS

1. The 'semicolon' command:

The semicolon(;) command is used to separate multiple commands at the command line.

SYNTAX: $ command1;command2;command3…….;commandn

EXAMPLE: $ who;date

2. The '&&' operator:

The '&&' operator signifies the logical AND operation in between two or more valid Unix commands.It means that only if the first command is successfully executed, then the next command will executed.

SYNTAX: $ command1 && command && command3…….&&commandn

EXAMPLE: $ who && date

3. The '||' operator:

The '||' operator signifies the logical OR operation in between two or more valid Unix commands.It means, that only if the first command will happen to be un successfully,it will continue to execute next commands.

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

EXAMPLE: $ who || date

1.5 FILTERS

1. The head filter

It displays the first ten lines of a file.

SYNTAX: $ head filename

EXAMPLE: $ head college Display the top ten lines.

$ head -5 college Display the top five lines.

2. The tail filter

It displays ten lines of a file from the end of the file.

SYNTAX: $ tail filename

EXAMPLE: $ tail college Display the last ten lines.

$tail -5 college Display the last five lines.

3. The pg filter:

The pg command shows the file page by page.

SYNTAX: $ ls –l | pg

4. The 'grep' command:

This command is used to search for a particular pattern from a file or from the standard input and display those lines on the standard output. "Grep" stands for "global search for regular expression."

SYNTAX: $ grep [pattern] [file_name]

EXAMPLE: $ cat > student

Arun cse

Ram ece

Kani cse

$ grep "cse" student

Arun cse

Kani cse

5. The 'sort' command:

The sort command is used to sort the contents of a file. The sort command reports only to the screen, the actual file remains unchanged.

SYNTAX: $ sort filename

EXAMPLE: $ sort college

OPTIONS:

Command	Purpose
Sort –r college	Sorts and displays the file contents in reverse order
Sort –c college	Check if the file is sorted
Sort –n college	Sorts numerically
Sort –m college	Sorts numerically in reverse order
Sort –u college	Remove duplicate records
Sort –l college	Skip the column with +1 (one) option.Sorts according to second column

6. The ‘uniq’ command:

In any enterprise data processing environment, there is often a duplicate entry creeping in due to faulty data entry. We just saw how sort removes them the –u option. Unix offers a special tool to handle these records

SYNTAX: $ uniq filename

EXAMPLE: $ cat > student

Arun cse

Arun cse

Ram ece

Ram ece

Kani cse

San cse

$ uniq student

Arun cse

Ram ece

Kani cse

San cse

7. The ‘nl’ command:

The nl filter adds lines numbers to a file and it displays the file and not provides access to edit but simply displays the contents on the screen.

SYNTAX: $ nl filename

EXAMPLE: $ nl college

8. The ‘cut’ command:

We can select specified fields from a line of text using cut command.

SYNTAX: $ cut -c filename

EXAMPLE: $ cut -c college

OPTION:

-c – Option cut on the specified character position from each line.

BASIC LINUX COMMANDS

1. DATE COMMAND

   1) Syntax: $date
      Output: Thu Jan 8 08:23:01 IST 2015

   2) Syntax: $date +%m
      Output: 01

   3) Syntax: $date +%h
      Output: Jan

   4) Syntax: $date +%d
      Output: 08

   5) Syntax: $date +%H
      Output: 08

   6) Syntax: $date +%M
      Output: 25

   7) Syntax: $date +%S
      Output: 42

   8) Syntax: $date +%Y
      Output: 15

2. ECHO COMMAND

   Syntax: $echo rec
   Output: rec

3. CALENDAR COMMAND

   Syntax: $cal 10 1995

   Output:

```
   Su Mo Tu We Th Fr Sa
                  1  2  3
    4  5  6  7  8  9  10
   11 12 13 14 15 16 17
   18 19 20 21 22 23 24
   25 26 27 28 29 30 31
```

4. ONLINE CALCULATOR

   Syntax: $bc

   Output: bc 1.06.95
   Copyright 1991-1994, 1997, 1998, 2000, 2004, 2006 Free Software Foundation, Inc.This is free software with ABSOLUTELY NO WARRANTY. For details type `warranty'

```
5 + 7
12
5 * 7
35
```

5. WHO

```
Syntax: $who
Output: rec      :0           2015-01-08 08:08 (:0)
        rec      pts/0        2015-01-08 08:13 (:0)
```

6. WHO AM I

```
Syntax: $who am i
Output: rec      pts/0        2015-01-08 08:13 (:0)
```

7. ID

```
Synatx: $id
Output: uid=1640(rec) gid=1640(rec) groups=1640(rec)
        context=unconfined_u:unconfined_r:unconfined_t:
        s0-s0:c0.c1023
```

8. TTY

```
Syntax: $tty
Output: /dev/pts/0
```

9. CLEAR

```
Syntax: $clear
Output: clears the screen
```

10. MAN

```
Syntax: $man who

Output:

NAME
who - show who is logged on

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

DESCRIPTION
 Print information about users who are currently logged in.

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

-b, --boot
   time of last system boot

-d, --dead
   print dead processes

-H, --heading
   print line of column headings
```

11. PROCESS STATUS

```
Syntax: $ps
Output: PID   TTY          TIME  CMD
        2774  pts/0    00:00:00 bash
        3821  pts/0    00:00:00 ps
```

12. UNAME COMMANDS

1) Syntax: $uname –m
   Output: i686

2) Syntax: $uname –n
   Output: hdc1306028

3) Syntax: $uname –r
   Output: 3.11.10-301.fc20.i686+PAE

4) Syntax: $uname –s
   Output: Linux

5) Syntax: $uname –v
   Output: #1 SMP Thu Dec 5 14:12:06 UTC 2013

6) Syntax: $uname –a
   Output: Linux hdc1306028 3.11.10-301.fc20.i686+PAE #1 SMP Thu
   Dec 5 14:12:06 UTC 2013 i686 i686 i386 GNU/Linux

13. FINGER COMMAND

Syntax: $finger
Output:

```
Login  Name    Tty      Idle  Login Time   Office  Office Phone   Host
rec    rec     *:0            Jan 8 08:08                         (:0)
rec    rec    pts/0           Jan 8 08:13                         (:0)
```

14. DIRECTORY COMMANDS

1) Syntax: $mkdir folder name
   Output: The directory is created.

2) Syntax: $cd folder name
   Output: It has entered into the directory.

3) Syntax: $cd\
   Output: It has come out of the directory.

4) Syntax: $ls
   Output: Desktop Downloads Pictures Templates
   Documents Music Public Videos

5) Syntax: $rmdir folder name
   Output: The directory is removed.

15. FILE COMMANDS

1) Syntax: $cat > filename
   Output: The file is created.

2) Syntax: $vi filename
   Output: hi am aa...
   ~
   ~
   "aa1" 1L, 12C

3) Syntax: $cp [source][destination]
   Output: copied to newfile.
```
hi am aa...
~
~
~
~
~
"aa2" 1L, 12C
```

4) Syntax: $cat [source][destination]
   Output:
```
 hi am aa...
 how r u
 hi am aa...
```

5) Syntax: $cat [source][destination]> filename
   Output:
```
hi am aa...
how r u
hi am aa...
```

6) Syntax: $mv [source][destination]
   Output:
```
 hi am aa...
 how r u
```

7) Syntax: $rm filename
   Output: The file is removed

8) Syntax: $file [filename]
   Output: aa: ASCII text

9) Syntax: $ls|wc
   Output: 13   13   89

10) Syntax: $wc filename
    Output: 10   10   74

11) Syntax: $ls|tee filename
    Output:
```
Desktop
 Documents
 Downloads
 Music
 Pictures
 Public
 Templates
 Videos
```

16. META CHARACTER COMMANDS

1) Syntax: $ls D*
   Output:
```
Desktop
Documents
Downloads
```

2) Syntax: $ls aa?
   Output:
```
 aa1
 aa2
```

3) Syntax: $ls[a-f]
   Output: b
           c

4) Syntax: $ls[a-f]*
   Output: Desktop
           Documents
           Downloads
5) Syntax: $ls ![a-f]*
   Output: No such files/directory.

17. PERMISSION COMMANDS

1) Syntax: Esc & i
   Output: To insert into a file.

2) Syntax: Esc & :wq / Esc & :zz
   Output: To save the contents and quit.

3) Syntax: Esc & :q!
   Output: Quits with a warning message.

4) Syntax: Esc & :q
   Output: Quits without saving.

5) Syntax: $who;ls
   Output: rec   :0   2015-01-08 08:08 (:0)
            rec   pts/0   2015-01-08 09:27 (:0)

            Desktop   Downloads Pictures aa   Templates
            Documents Music   Public   aa2 Videos

6) Syntax: $who&&ls
   Output: rec   :0   2015-01-08 08:08 (:0)
            rec   pts/0   2015-01-08 09:27 (:0)

            Desktop   Downloads Pictures aa   Template
            Documents Music   Public   aa2 Videos

7) Syntax: $who||ls
   Output: rec   :0   2015-01-08 08:08 (:0)
           rec   pts/0   2015-01-08 09:27 (:0)

8) Syntax: $head -2 filename
   Output: hi am aa...
           how r u

9) Syntax: $tail -2 filename
   Output: This is rec
           Welcomes you..

10) Syntax: $grep how filename
    Output: **how** r u

11) Syntax: $sort filename
    Output: hi am aa...
            how r u
            This is rec
            Welcomes you..

12) Syntax: $uniq filename
    Output: hi am aa...
            how r u
            This is rec
            Welcomes you..

13) Syntax: $nl filename
    Output: 1. hi am aa...
              2. how r u
              3. This is rec
              4. Welcomes you..

14) Syntax: $ls-l
    Output: drwxr-xr-x. 2 rec rec 4096 Jan 8 10:12 Desktop
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Documents
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Downloads
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Music
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Pictures
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Public
            drwxrwxr-x. 2 rec rec 4096 Jan 8 09:08 aa
            -rw-rw-r--. 1 rec rec   30 Jan 8 10:13 aa2
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Templates
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Videos

15) Syntax: $chmod g –w aa2
    Output: drwxr-xr-x. 2 rec rec 4096 Jan 8 10:16 Desktop
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Documents
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Downloads
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Music
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Pictures
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Public
            drwxrwxr-x. 2 rec rec 4096 Jan 8 09:08 aa
            -r--r--r--. 1 rec rec   30 Jan 8 10:13 aa2
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Templates
            drwxr-xr-x. 2 rec rec 4096 Jan 8 08:08 Videos

SHELL PROGRAMMING

Greatest among three numbers

PROGRAM

```
echo "Enter three number"
read a
read b
read c
if [ $a -gt $b ]
then
if [ $a -gt $c ]
then
echo "$a is greater"
else
echo "$c is greater"
fi
else
if [ $b -gt $c ]
then
echo "$b is greater"
else
echo "$c is greater"
fi
fi
```

OUTPUT

```
Enter three numbers
5
1
9
9 is greater
```

Reverse and Sum of the Digits

PROGRAM

```
echo "enter the number"
read n
rn=0
sum=0
while [ $n -gt 0 ]
do
d=`expr $n % 10`
rn=`expr $rn \* 10 + $d`
sum=`expr $sum + $d`
n=`expr $n / 10`
done
echo "reverse of the digit is: $rn"
echo "sum of the digit is: $sum"
```

OUTPUT

```
enter the number
612
reverse of the digit is: 216
sum of the digit is: 9
```

Implementation of Unix Command Using Case Statement

PROGRAM

```
echo "MENUS"
echo "1.Todays date"
echo "2.List of files"
echo "3.Who am i"
echo "4.Quit"
echo "Enter your choice"
read a
case $a in
1)date ;;
2)ls ;;
3)who am i ;;
4)exit ;;
esac
```

OUTPUT

```
MENUS
1.Todays date
2.List of files
3.Who am i
4.Quit
Enter your choice
1
Thu Jan 26 22:35:47 IST 2012
2
1    arith      Desktop  hai    odd   power     student      swap
123  big        fact     hello  pali  sample    sumeven      test3
area college    fib      neven  peri  san       sumodd       uni
ari  countdigit great    nodd   pos   sanmkdir  sumrevdigit  wc-l

3
receee   pts/1        2012-01-26 22:24 (:0.0)
```

ROUND ROBIN SCHEDULING

Program:

```
#include<stdio.h>
struct round
{
 int flag;
 int nts,bt,at,wt,tat;
 char pname[3];
 int st,et;
 int tbt;
}f[10],rd[15];
main()
{ int tempo;
 int ts;
 int max,i,j,n,k=0,m;
 int twt=0,ttat=0;
 float awt,atat;
 printf("\n enter the no of processes: ");
 scanf("%d",&n);
 printf("\n enter the time slice: ");
 scanf("%d",&ts);
 for(i=0;i<n;i++)
{
  f[i].at=0;
  f[i].flag=0;
   printf("\n enter the process name: ");
  scanf("%s",f[i].pname);
  printf("\n enter the burst time: ");
  scanf("%d",&f[i].bt);
  f[i].tbt=f[i].bt;
  f[i].nts=f[i].bt/ts;
  if(f[i].bt%ts!=0)
  f[i].nts=f[i].nts+1;
}
```

```
max=f[0].nts;
for(i=1;i<n;i++)
if(max<f[i].nts)
max=f[i].nts;
printf("\n max=%d",max);
for(j=0;j<max;j++)
{ for(i=0;i<n;i++)
  {
   if(f[i].nts>0&&f[i].flag==0)
    {
       if(f[i].tbt>ts)
       {
        f[i].tbt-=ts;
        rd[k]=f[i];
        rd[k].bt=ts;
          f[i].nts--;
        k++;
       }
       else
       {
         rd[k]=f[i];
         rd[k].bt=f[i].tbt;
         k++;
         f[i].nts--;
       }
     }
   else if(f[i].nts==0)
   f[i].flag=1;
   }
}
rd[0].st=0;
rd[0].et=rd[0].bt;
tempo=rd[0].bt;
for(i=1;i<k;i++)
```

```
{
 rd[i].st=tempo;
 tempo+=rd[i].bt;
 rd[i].et=tempo;
}
printf("\n READY QUEUE: ");
printf("\nPNAME\tST\tET\n");
for(i=0;i<k;i++)
printf("%s\t%d\t%d\n",rd[i].pname,rd[i].st,rd[i].et);
for(i=0;i<n;i++)
{
 f[i].wt=rd[i].st;
m=rd[i].et;
for(j=i+1;j<k;j++)
{
 if(strcmp(f[i].pname,rd[j].pname)==0)
 {
   f[i].wt+=(rd[j].st-m);
    m=rd[j].et;
}
}
twt+=f[i].wt;
}
awt=(float)twt/n;
printf("\n pname\tbt\twt\n");
for(i=0;i<n;i++)
printf("%s\t%d\t%d\n",f[i].pname,f[i].bt,f[i].wt);
printf("twt=%d\tawt=%f",twt,awt);
}
```

OUTPUT:

enter the no of processes: 3

enter the time slice: 3

enter the process name: p1

enter the burst time: 11

enter the process name: p2

enter the burst time: 4

enter the process name: p3

enter the burst time: 3

max=4

READY QUEUE:

PNAME	ST	ET
p1	0	3
p2	3	6
p3	6	9
p1	9	12
p2	12	13
p1	13	16
p1	16	18

pname	bt	wt
p1	11	7
p2	4	9
p3	3	6

twt=22 awt=7.333333

SHORTEST JOB FIRST SCHEDULING

PROGRAM:

```
#include<stdio.h>
#include<string.h>
struct sjf{
int flag;
int bt,wt,tat,at;
char pname[3];
}f[10],temp,inter[10],final[10];

main()
{
int count=0;
 int i,j,n,m=0,k=0,p,q;
 int twt=0,ttat=0,totbt=0,tot;
 float awt,atat;
 printf("\n enter the no of processes: ");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 { f[i].flag=0;
  printf("\n enter the process name: ");
  scanf("%s",f[i].pname);
  printf("\n enter the burst time: ");
  scanf("%d",&f[i].bt);
  printf("\n enter the arrival time: ");
  scanf("%d",&f[i].at);
 }
  for(i=0;i<n;i++)
  { if(f[i].at==0)
     count++;
   }
for(i=0;i<count;i++)
```

```
{ for(j=i+1;j<count;j++)
   { if(f[i].bt>f[j].bt)
     { temp=f[i];
       f[i]=f[j];
       f[j]=temp;
     }
 }
inter[m]=f[i];
m++;
}
if(count==n)
   {
    for(i=0;i<n;i++)
    { final[i]=inter[i];
      final[i].flag=1;
    }
   }
   else
   { final[k]=inter[0];
    f[0].flag=1;
     k++;
     tot=final[0].bt;
for(i=1;i<n;i++)
{     m=0;
       for(j=1;j<n;j++)
       {if(tot>=f[j].at&&f[j].flag==0)
            { inter[m]=f[j];
              m++;
            }
       }
       for(p=0;p<m;p++)
       {
            for(q=p+1;q<m;q++)
            { if(inter[p].bt>inter[q].bt)
```

```
            {
             temp=inter[p];
             inter[p]=inter[q];
             inter[q]=temp;
             }
            }
   }
for(p=0;p<n;p++)
if(strcmp(f[p].pname,inter[0].pname)==0)
f[p].flag=1;
final[k]=inter[0];
tot+=final[k].bt;
k++;
}
     }
printf("\n order of execution: ");
for(i=0;i<k;i++)
{
    printf("%s ",final[i].pname);
}
final[0].wt=0;
  final[0].tat=final[0].bt;
  for(i=1;i<n;i++)
  {totbt+=final[i-1].bt;
   final[i].wt=totbt-final[i].at;
   final[i].tat=final[i].bt+final[i].wt;
   twt+=final[i].wt;
   ttat+=final[i].tat;
  }
  awt=(float)twt/n;
  ttat+=final[0].tat;
  atat=(float)ttat/n;
   printf("\npname\tbt\tat\twt\ttat\n");
  for(i=0;i<n;i++)
```

```
 { printf("%s\t%d\t%d\t%d\t%d\n",final[i].pname,final[i].bt,final[i].at,final[i].wt,final[i].tat);
 }
printf("\ntwt=%d\tttat=%d\nawt=%f\tatat=%f",twt,ttat,awt,atat);
 }
```

OUTPUT:

```
enter the no of processes: 6
enter the process name: P1
enter the burst time: 2
enter the arrival time: 0
enter the process name: P2
enter the burst time: 2
enter the arrival time: 1
enter the process name: P3
enter the burst time: 5
enter the arrival time: 1
enter the process name: P4
enter the burst time: 2
enter the arrival time: 2
enter the process name: P5
enter the burst time: 3
enter the arrival time: 2
enter the process name: P6
enter the burst time: 1
enter the arrival time: 4
order of execution: P1 P2 P6 P4 P5 P3
pname   bt    at      wt      tat
P1          2       0       0       2
P2          2       1       1       3
P6          1       4       0       1
P4          2       2       3       5
P5          3       2       5       8
P3          5       1       9       14
twt=18  ttat=33
awt=3.000000    atat=5.500000
```

FIRST COME FIRST SERVE (FCFS) SCHEDULING

PROGRAM:

```
#include<stdio.h>
struct fcfs
{
char name[5];
int at,wt,bt,tat;
}f1[10];
main()
{
int,n;
int totbt=0;
printf("\nenter the no. of processes:");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
printf("\nprocess name?");
scanf("%s",f1[i].name);
f1[i].at=f1[i].wt=0;
printf("\nburst time?");
scanf("%d",&f1[i].bt);
printf("\narrival time?");
scanf("%d",&f1[i].at);
}
f1[1].wt=0;
f1[1].tat=f1[1].bt+f1[1].wt;
f1[1].at=0;
for(i=1;i<=n;i++)
{totbt+=f[i-1].wt;
f1[i].wt=totbt-f[i].at;
f1[i].tat=f1[i].wt+f1[i].bt;
}
for(i=1;i<=n;i++)
```

```
 f1[i].wt=f1[i].wt-f1[i].at;
 printf("\npname\tbt\tat\twt\ttat\n");
 for(i=1;i<=n;i++)
{
 printf("%s\t%d\t%d\t%d\t%d\n",f1[i].name,f1[i].bt,f1[i].at,f1[i].wt,f1[i].tat);
}
}
```

Output:

```
enter the no. of processes:4
process name?p1
burst time?10
arrival time?0
process name?p2
burst time?4
arrival time?1
process name?p3
burst time?2
arrival time?2
process name?p4
burst time?7
arrival time?2
```

pname	bt	at	wt	tat
p1	10	0	0	10
p2	4	1	9	14
p3	2	2	12	16
p4	7	2	14	23

PRIORITY SCHEDULING

PROGRAM:

```
 #include<stdio.h>
struct prior{
int pri;
int count;
int bt,wt,tat,at;
char pname[3];
}f[10],temp;
main()
{
 int i,j,n;
 int twt=0,ttat=0,totbt=0;
 float awt,atat;
 printf("\n enter the no of processes: ");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {        f[i].at=0;
          f[i].count=i+1 ;
          printf("\n enter the process name: ");
          scanf("%s",f[i].pname);
          printf("\n enter the burst time: ");
          scanf("%d",&f[i].bt);
          printf("\n enter the priority ");
          scanf("%d",&f[i].pri);
 }
 for(i=0;i<n;i++)
 {
  for(j=i+1;j<n;j++)
  {
   if(f[i].pri>f[j].pri)
   { temp=f[i];
     f[i]=f[j];
```

```
   f[j]=temp;
  }
  if(f[i].pri==f[j].pri)
  { if(f[i].count>f[j].count)
    {  temp=f[i];
          f[i]=f[j];
          f[j]=temp;
    }
  }
 }
 }
 printf("\n order of execution: ");
 for(i=0;i<n;i++)
 printf("%s ",f[i].pname);
 f[0].wt=0;
 f[0].tat=f[0].bt;
 for(i=1;i<n;i++)
 {totbt+=f[i-1].bt;
  f[i].wt=totbt-f[i].at;
  f[i].tat=f[i].bt+f[i].wt;
  twt+=f[i].wt;
  ttat+=f[i].tat;
 }
 awt=(float)twt/n;
 ttat+=f[0].tat;
 atat=(float)ttat/n;
  printf("\npname\tbt\tat\tprior\twt\ttat\n");
 for(i=0;i<n;i++)
{
   printf("%s\t%d\t%d\t%d\t%d\t%d\n",f[i].pname,f[i].bt,f[i].at,f[i].pri,f[i].wt,
            f[i].tat);
}
printf("\ntwt=%d\tttat=%d\nawt=%f\tatat=%f",twt,ttat,awt,atat);
}
```

OUTPUT:

enter the no of processes:3

enter the process name:P1

enter the burst time: 10

enter the priority 3

enter the process name: P2

enter the burst time: 2

enter the priority 1

enter the process name: P3

enter the burst time: 4

enter the priority 3

order of execution: P2 P1 P3

pname	bt	at	prior	wt	tat
P2	2	0	1	0	2
P1	10	0	3	2	12
P3	4	0	3	12	16

twt=14 ttat=30

awt=4.666667 atat=10.000000

SEQUENTIAL FILE ALLOCATION

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
main()
{
 int n,i,j,b[20],sb[20],t[20],x,c[20][20];
 clrscr();
 printf("Enter no.of files:");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
        printf("Enter no. of blocks occupied by file%d",i+1);
        scanf("%d",&b[i]);
        printf("Enter the starting block of file%d",i+1);
        scanf("%d",&sb[i]);
        t[i]=sb[i];
        for(j=0;j<b[i];j++)
                c[i][j]=sb[i]++;
        }
 printf("Filename\tStart block\tlength\n");
 for(i=0;i<n;i++)
        printf("%d\t  %d \t%d\n",i+1,t[i],b[i]);
 printf("Enter file name:");
 scanf("%d",&x);
 printf("File name is:%d",x);
 printf("length is:%d",b[x-1]);
 printf("blocks occupied:");
 for(i=0;i<b[x-1];i++)
        printf("%4d",c[x-1][i]);
 getch();
}
```

Output:

Enter no.of files: 2

Enter no. of blocks occupied by file1 4

Enter the starting block of file1 2

Enter no. of blocks occupied by file2 10

Enter the starting block of file2 5

Filename	Start block	length
1	2	4
2	5	10

Enter file name: rajesh

File name is:12803 length is:0blocks occupied

INDEXED FILE ALLOCATION

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
main()
{
 int n,m[20],i,j,sb[20],s[20],b[20][20],x;
 clrscr();
 printf("Enter no. of files:");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {      printf("Enter starting block and size of file%d:",i+1);
        scanf("%d%d",&sb[i],&s[i]);
        printf("Enter blocks occupied by file%d:",i+1);
        scanf("%d",&m[i]);
        printf("enter blocks of file%d:",i+1);
        for(j=0;j<m[i];j++)
                scanf("%d",&b[i][j]);
 } printf("\nFile\t index\tlength\n");
 for(i=0;i<n;i++)
 {
        printf("%d\t%d\t%d\n",i+1,sb[i],m[i]);
 }printf("\nEnter file name:");
 scanf("%d",&x);
 printf("file name is:%d\n",x);
 i=x-1;
 printf("Index is:%d",sb[i]);
 printf("Block occupied are:");
 for(j=0;j<m[i];j++)
        printf("%3d",b[i][j]);
 getch();
}
```

Output:

Enter no. of files:2

Enter starting block and size of file1: 2 5

Enter blocks occupied by file1:10

enter blocks of file1:3

2 5 4 6 7 2 6 4 7

Enter starting block and size of file2: 3 4

Enter blocks occupied by file2:5

enter blocks of file2: 2 3 4 5 6

File index length

1 2 10

2 3 5

Enter file name: venkat

file name is:12803

Index is:0Block occupied are:

LINKED FILE ALLOCATION

PROGRAM:

```
#include<stdio.h>
#include<conio.h>
struct file
{
 char fname[10];
 int start,size,block[10];
}f[10];
main()
{
 int i,j,n;
 clrscr();
 printf("Enter no. of files:");
 scanf("%d",&n);
 for(i=0;i<n;i++)
 {
 printf("Enter file name:");
 scanf("%s",&f[i].fname);
 printf("Enter starting block:");
 scanf("%d",&f[i].start);
 f[i].block[0]=f[i].start;
 printf("Enter no.of blocks:");
 scanf("%d",&f[i].size);
 printf("Enter block numbers:");
 for(j=1;j<=f[i].size;j++)
 {
        scanf("%d",&f[i].block[j]);
 }
 }
 printf("File\tstart\tsize\tblock\n");
 for(i=0;i<n;i++)
 {
        printf("%s\t%d\t%d\t",f[i].fname,f[i].start,f[i].size);
```

```
        for(j=1;j<=f[i].size-1;j++)
                printf("%d--->",f[i].block[j]);
        printf("%d",f[i].block[j]);
        printf("\n");
 }
 getch();
}
```

Output:

```
Enter no. of files:2
Enter file name:venkat
Enter starting block:20
Enter no.of blocks:6
Enter block numbers: 4
12
15
45
32
25
Enter file name:rajesh
Enter starting block:12
Enter no.of blocks:5
Enter block numbers:6
5
4
3
2
File    start   size    block
venkat  20      6       4--->12--->15--->45--->32--->25
rajesh  12      5       6--->5--->4--->3--->2
```

PRODUCER-CONSUMER PROBLEM USING SEMAPHORES

PROGRAM:

Producer.c

```
#include"prodcons.h"
void producer()
{
int i=0,n;
MEM *s=memory();
while(1)
{
i++;
sem_wait(&s->empty);
sem_wait(&s->mutex);
sem_getvalue(&s->full,&n);
(s->buff)[n]=i;
printf("[producer] placed item [%d] \n",i);
sem_post(&s->mutex);
sem_post(&s->full);
sleep(PRODUCER_SLEEP_SEC);
}
}
main()
{
init();
producer();
}
```

Consumer.c

```
#include"prodcons.h"
void consumer()
{
int n;
MEM *s=memory();
```

```
while(1)
{
sem_wait(&s->full);
sem_wait(&s->mutex);
sem_getvalue(&s->full,&n);
printf("[consumer] removed item [%d]\n",(s->buff)[n]);
sem_post(&s->mutex);
sem_post(&s->empty);
sleep(CONSUMER_SLEEP_SEC);
}
}
main()
{
consumer();
}
```

Prodcons.h

```
#include<stdio.h>
#include<semaphore.h>
#include<sys/types.h>
#include<sys/ipc.h>
#include<fcntl.h>
#define BUFFER_SIZE 10
#define CONSUMER_SLEEP_SEC 3
#define PRODUCER_SLEEP_SEC 1
#define KEY 1010
typedef struct
{
int buff[BUFFER_SIZE];
sem_t mutex,empty,full;
}MEM;
MEM  *memory()
{
key_t key=KEY;
int shmid;
```

```
shmid=shmget(key,sizeof(MEM),IPC_CREAT|0666);
return(MEM *)shmat(shmid,NULL,0);
}
void init()
{
MEM *M=memory(0);
sem_init(&M->mutex,1,1);
sem_init(&M->empty,1,BUFFER_SIZE);
sem_init(&M->full,1,0);
}
```

Output:

```
[os32@localhost ~]$ gcc producer.c -lrt -lpthread -o prod
[os32@localhost ~]$ gcc consumer.c -lrt -lpthread -o cons

Terminal 1
[os32@localhost ~]$ ./prod
[producer] placed item [1]
[producer] placed item [2]
[producer] placed item [3]
[producer] placed item [4]

Terminal 2
[os32@localhost ~]$ ./cons
 [consumer] removed item [4]
[consumer] removed item [3]
[consumer] removed item [2]
[consumer] removed item [1]
```

BANKER'S ALGORITHM

PROGRAM:

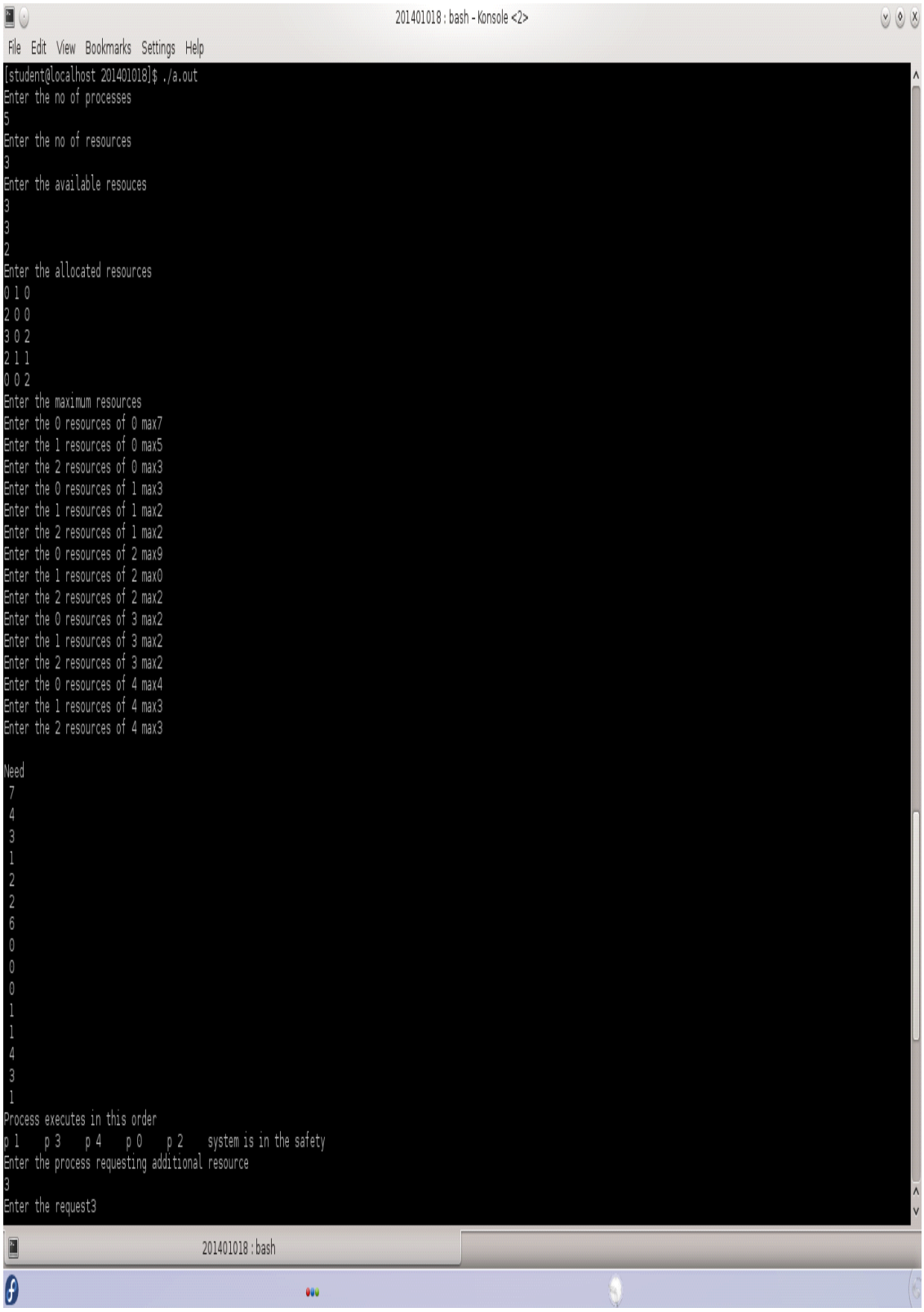
```
#include<stdio.h>
#include<stdlib.h>
int main()
{
int alloc[20][20],max[20][20],avail[20],need[20][20];
int work[20]={0};
int newavail[20],req[20]={0},check=0,check2=0,cond=0,p;
int i=0,j=0,m=0,n=0,t=0,x=0,c[20]={0},k=0,count,count2,a[20],b;
int x2=0,c2[20];
printf("Enter the no of processes\n");
scanf("%d",&n);
printf("Enter the no of resources\n");
scanf("%d",&m);
printf("Enter the available resouces\n");
for(j=0;j<m;j++)
{
scanf("%d",&avail[j]);
work[j]=avail[j];
}
printf("Enter the allocated resources\n");
for(i=0;i<n;i++)
{
 for(j=0;j<m;j++)
 scanf("%d",&alloc[i][j]);
}
printf("Enter the maximum resources\n");
for(i=0;i<n;i++)
{
 for(j=0;j<m;j++)
 {
  printf("Enter the %d resources of %d max",j,i);
```

```
  scanf("%d",&max[i][j]);
  need[i][j]=max[i][j]-alloc[i][j];
 }
}
printf("\nNeed\n");
for(i=0;i<n;i++)
{
 for(j=0;j<m;j++)
 {
  printf("%2d",need[i][j]);
  printf("\n");
 }}
printf("Process executes in this order\n");
do
{
 for(i=0;i<n;i++)
 {
  count=0;
  if(c[i]!=i+1)
  {
   for(j=0;j<m;j++)
   {
     if(need[i][j]<=work[j])
     count=count+1;
   }
   if(count==m)
   {
    printf("p %d\t",i);
    c[i]=i+1;
    x=x+1;
    for(j=0;j<m;j++)
    work[j]=work[j]+alloc[i][j];
   }
  }
```

```
 }
check=check+1;
}while(x<n&&check<=n);
if(x==n)
 printf("system is in the safety\n");
else
 printf("System is not in the safety\n");
printf("Enter the process requesting additional resource\n");
scanf("%d",&p);
printf("Enter the request");
for(j=0;j<m;j++)
scanf("%d",&req[j]);
for(j=0;j<m;j++)
{
 if(req[j]<=avail[j]&&req[j]<=need[p][j])
 cond=cond+1;
}
if(cond==m)
{
 for(j=0;j<m;j++)
{
 alloc[p][j]=alloc[p][j]+req[j];
 avail[j]=avail[j]-req[j];
 need[p][j]=need[p][j]-req[j];
}}
else
{
printf("Request is not satisfied\n");
exit(0);
}
do
{
for(i=0;i<n;i++)
{
```

```
 count2=0;
 if(c2[i]!=i+1)
 {
  for(j=0;j<m;j++)
  {
   if(need[i][j]<=avail[j]);
   count2=count2+1;
  }
 if(count2==m)
 {
  printf("p%d\t",i);
  c2[i]=i+1;
  x2=x2+1;
  for(j=0;j<m;j++)
  avail[j]=avail[j]+alloc[i][j];
 }
}
}
check2=check2+1;
}while((x2<n)&&(check2<=n));
if(x2==n)
printf("\n System is in safestate we can grant the request");
else
printf("\n System is in unsafe state we cannot grant the request");
return 0;
}
```

OUTPUT:


201401018 : bash - Konsole <2>
File Edit View Bookmarks Settings Help
[student@localhost 201401018]$ ./a.out
Enter the no of processes
5
Enter the no of resources
3
Enter the available resouces
3
3
2
Enter the allocated resources
0 1 0
2 0 0
3 0 2
2 1 1
0 0 2
Enter the maximum resources
Enter the 0 resources of 0 max7
Enter the 1 resources of 0 max5
Enter the 2 resources of 0 max3
Enter the 0 resources of 1 max3
Enter the 1 resources of 1 max2
Enter the 2 resources of 1 max2
Enter the 0 resources of 2 max9
Enter the 1 resources of 2 max0
Enter the 2 resources of 2 max2
Enter the 0 resources of 3 max2
Enter the 1 resources of 3 max2
Enter the 2 resources of 3 max2
Enter the 0 resources of 4 max4
Enter the 1 resources of 4 max3
Enter the 2 resources of 4 max3
Need
7
4
3
1
2
2
6
0
0
0
1
1
4
3
1
Process executes in this order
p 1 p 3 p 4 p 0 p 2 system is in the safety
Enter the process requesting additional resource
3
Enter the request3
201401018 : bash

DEADLOCK DETECTION ALGORITHM

PROGRAM:

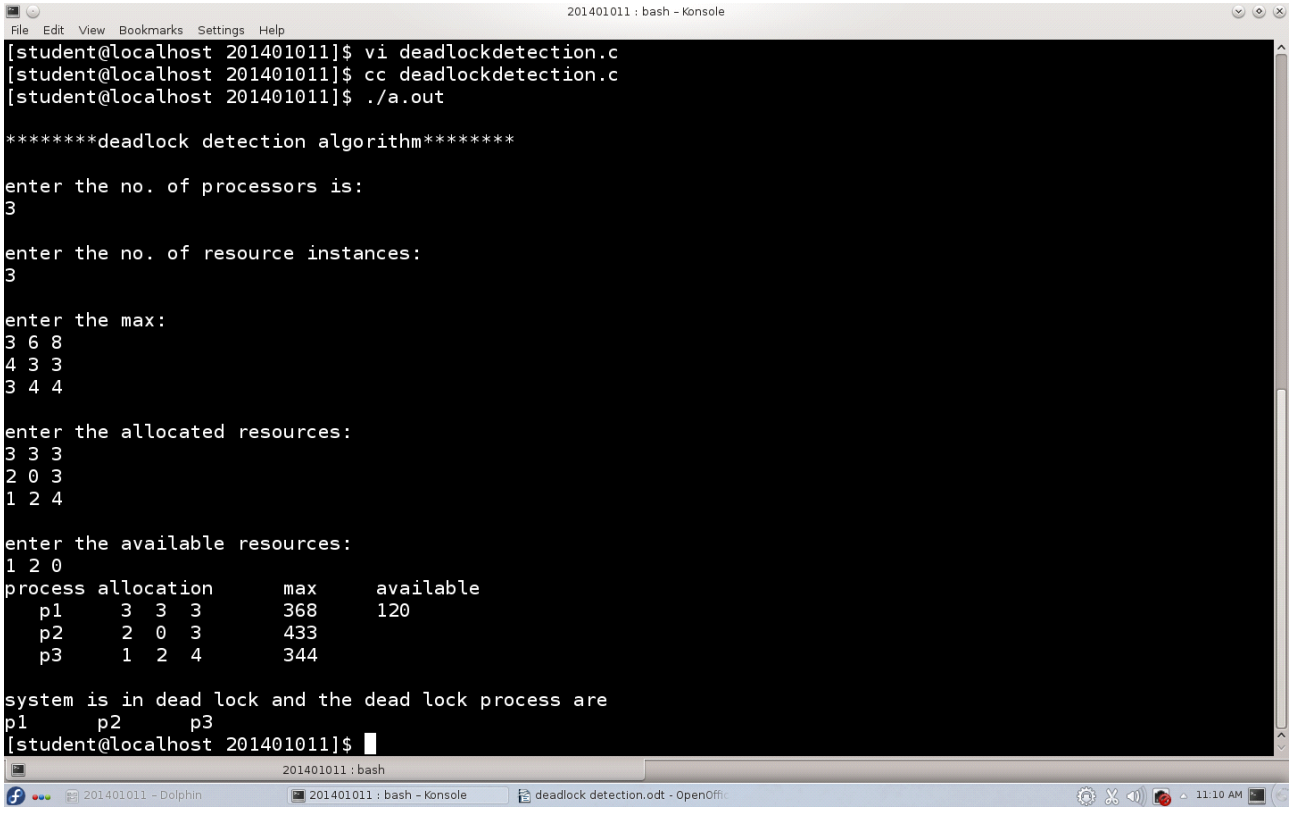
```
#include<stdio.h>
int max[100][100];
int alloc[100][100];
int need[100][100];
int avail[100];
int n,r;
void input();
void show();
void cal();
int main()
{
int i,j;
printf("\n********deadlock detection algorithm********\n");
input();
show();
cal();
return 0;
}
void input()
{
int i,j;
printf("\nenter the no. of processors is:\n");
scanf("%d",&n);
printf("\nenter the no. of resource instances:\n");
scanf("%d",&r);
printf("\nenter the max:\n");
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
scanf("%d",&max[i][j]);
}
printf("\nenter the allocated resources:\n");
```

```
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
scanf("%d",&alloc[i][j]);
}
printf("\nenter the available resources:\n");
for(j=0;j<r;j++)
scanf("%d",&avail[j]);
}
void show()
{
int i,j;
printf("process\tallocation\tmax\tavailable\t");
for(i=0;i<n;i++)
{
printf("\n   p%d\t",i+1);
for(j=0;j<r;j++)
printf("  %d",alloc[i][j]);
printf("\t");
for(j=0;j<r;j++)
printf("%d",max[i][j]);
printf("\t");
if(i==0)
{
for(j=0;j<r;j++)
printf("%d",avail[j]);
}
}
}
void cal()
{
int finish[100],temp,need[100][100],flag=1,k,cl=0;
int dead[100],safe[100],j,i;
for(i=0;i<n;i++)
```

```
finish[i]=0;
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
need[i][j]=max[i][j]-alloc[i][j];
}
while(flag)
{
flag=0;
for(i=0;i<n;i++)
{
int c=0;
for(j=0;j<r;j++)
{
if((finish[i]==0)&&(need[i][j]<=avail[j]))
{
c++;
if(c==r)
{
for(k=0;k<r;k++)
{
avail[k]+=alloc[i][j];
finish[i]=1;
flag=1;
}
printf("\np%d\n",i);
if(finish[i]==1)
i=n;
}}}}}
j=0;
flag=0;
for(i=0;i<n;i++)
{
if(finish[i]==0)
```

```
{
dead[j]=i;
j++;
flag=1;
}}
if(flag==1)
{
printf("\n\nsystem is in dead lock and the dead lock process are\n");
for(i=0;i<n;i++)
printf("p%d\t",dead[i]+1);
printf("\n");
}
else
printf("\n no deadlock occur\n");
}
```

OUTPUT:



FIFO REPLACEMENT ALGORITHM

PROGRAM:

```
#include<stdio.h>
main()
{
	int nframes,len,i,j,k,flag,pf=0,frame[20],s[20];
	printf("enter the number of frames\n");
	scanf("%d",&nframes);
	for(i=1;i<=nframes;i++)
		frame[i]=0;
	fflush(stdin);
	printf("enter the length of the reference string\n");
	scanf("%d",&len);
	printf("enter the reference string\n");
	for(i=1;i<=len;i++)
		scanf("%d",&s[i]);
	j=1;
	for(i=1;i<=len;i++)
	{	flag=0;
		for(k=1;k<=nframes;k++)
		{
			if(s[i]==frame[k])
			{
				flag=1;
				break;}
		}
		if(flag!=1)	{
			pf++;
			frame[j++]=s[i];
			if(j>nframes)
				j=1;
		}
		for(k=1;k<=nframes;k++)
			printf("%d\t",frame[k]);
```

```
            printf("\n");
        }
        printf("no. of page faults is %d",pf);
}
```

OUTPUT:

```
[root@localhost student]# gcc fifo.c
[root@localhost student]# ./a.out
enter the number of frames
3
enter the length of the reference string
5
enter the reference string
1 2 3 1 3
1       0       0
1       2       0
1       2       3
1       2       3
1       2       3
no. of page faults is 3
```

LRU REPLACEMENT ALGORITHM

PROGRAM:

```
#include<stdio.h>
main()
{
	int nframes,len,t,u,min,rep,pos[20],i,j=1,k,m,flag,pf=0,frame[20],s[20];
	printf("enter the number of frames\n");
	scanf("%d",&nframes);
	for(i=1;i<=nframes;i++)
		frame[i]=0;
	fflush(stdin);
	printf("enter the length of the reference string\n");
	scanf("%d",&len);
	printf("enter the reference string\n");
	for(i=1;i<=len;i++)
		scanf("%d",&s[i]);
	for(i=1;i<=len;i++)
	{
		flag=0;
		for(k=1;k<=nframes;k++)
		{
			if(s[i]==frame[k])
			{
				flag=1;
				break;
			}
		}
		if(flag!=1)
		{
			pf++;
			if(j>nframes)
			{
				for(u=1;u<=nframes;u++)
				{
```

```
                    m=i-1;
                    while(frame[u]!=s[m])
                    {
                        m--;
                    }
                    pos[frame[u]]=m;
                }
                min=pos[frame[1]];
                rep=1;
                for(u=2;u<=nframes;u++)
                    if(pos[frame[u]]<min)
                    {
                        min=pos[frame[u]];
                        rep=u;
                    }
                frame[rep]=s[i];
            }
            else
                frame[j++]=s[i];
        }
        for(k=1;k<=nframes;k++)
            printf("%d\t",frame[k]);
        printf("\n");
    }
    printf("no. of page faults is %d",pf);
}
```

OUTPUT:

```
[root@localhost student]# ./a.out
enter the number of frames
3
enter the length of the reference string
5
enter the reference string
1 2 4 3 2
1       0       0
1       2       0
1       2       4
3       2       4
3       2       4
        no. of page faults is 4
```

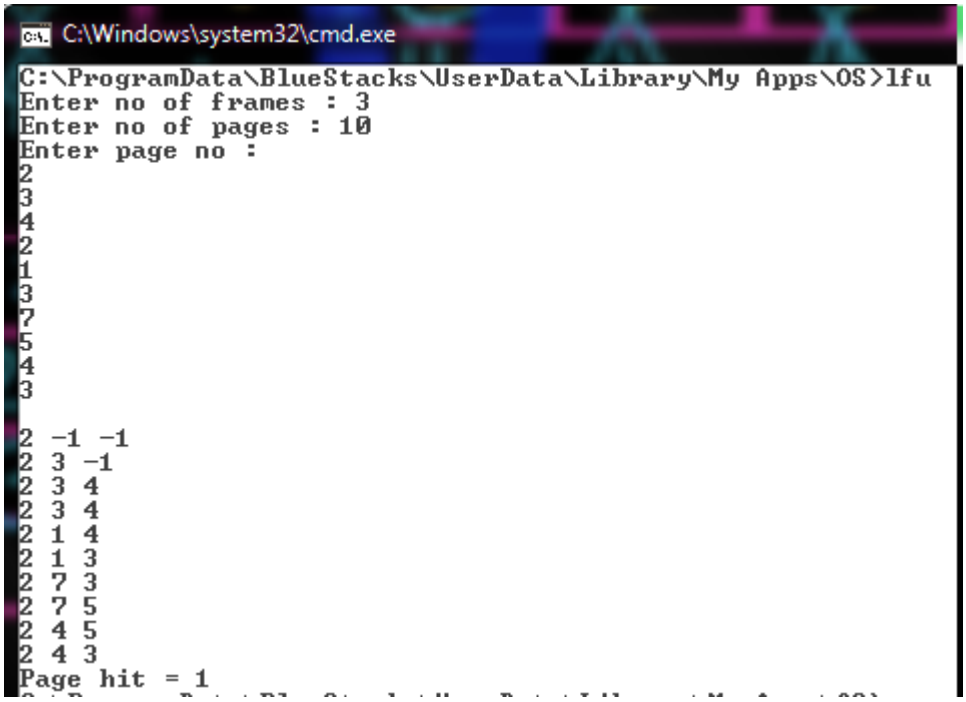
LFU (LEAST FREQUENTLY USED) PAGE REPLACEMENT ALGORITHMS

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;
}
```



IPC USING SHARED MEMORY

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<sys/ipc.h>
#include<sys/shm.h>
#include<unistd.h>
main()
{
 int child,id,n,i;
,0);
 char *shmptr;
 int td;
 printf("\n enter the no of items: ");
 scanf("%d",&n);
 child=fork();
 if(!child) {
 id=shmget(2000,32,0666|IPC_CREAT);
 shmptr=shmat(id,0
 printf("\n child is adding item to buffer: ");
 for(i=0;i<n;i++) {
  shmptr[i]='A'+i;
  printf("\nadded item%d: %c",i+1,shmptr[i]);
 } exit(0);}
 else {
  wait(&td);
  id=shmget(2000,32,0666|IPC_CREAT);
  shmptr=shmat(id,0,0);
  printf("\n parent is now consuming:\n");
  for(i=0;i<n;i++)
  putchar(shmptr[i]);
  shmdt(NULL);
  shmctl(id,IPC_RMID,NULL);
}}
```

OUTPUT:

enter the no of items: 3

child is adding item to buffer:

added item1: A

added item2: B

added item3: C

parent is now consuming:

ABC

PAGING

PROGRAM:

```
#include <stdio.h>
#include <conio.h>
struct pstruct
{       int fno;
        int pbit;
}ptable[10];
int pmsize,lmsize,psize,frame,page,ftable[20],frameno;
void info()
{
        printf("\n\nMEMORY MANAGEMENT USING PAGING\n\n");
        printf("\n\nEnter the Size of Physical memory: ");
        scanf("%d",&pmsize);
        printf("\n\nEnter the size of Logical memory: ");
        scanf("%d",&lmsize);
        printf("\n\nEnter the partition size: ");
        scanf("%d",&psize);
        frame = (int) pmsize/psize;
        page = (int) lmsize/psize;
        printf("\nThe physical memory is divided into %d no.of frames\n",frame);
        printf("\nThe Logical memory is divided into %d no.of pages",page);
}
void assign()
{
        int i;
        for (i=0;i<page;i++)
        {
        ptable[i].fno = -1;
        ptable[i].pbit= -1;
        }
        for(i=0; i<frame;i++)
                ftable[i] = 32555;
        for (i=0;i<page;i++)
```

```
        {
        printf("\n\nEnter the Frame number where page %d must be placed: ",i);
                scanf("%d",&frameno);
                ftable[frameno] = i;
                if(ptable[i].pbit == -1)
                {
                        ptable[i].fno = frameno;
                        ptable[i].pbit = 1;
                }
        }
        getch();
    //  clrscr();
        printf("\n\nPAGE TABLE\n\n");
        printf("PageAddress  FrameNo. PresenceBit\n\n");
        for (i=0;i<page;i++)
                printf("%d\t\t%d\t\t%d\n",i,ptable[i].fno,ptable[i].pbit);
        printf("\n\n\n\tFRAME TABLE\n\n");
        printf("FrameAddress   PageNo\n\n");
        for(i=0;i<frame;i++)
                printf("%d\t\t%d\n",i,ftable[i]);
}
void cphyaddr()
{
        int laddr,paddr,disp,phyaddr,baddr;
        getch();
    //  clrscr();
        printf("\n\n\n\tProcess to create the Physical Address\n\n");
        printf("\nEnter the Base Address: ");
        scanf("%d",&baddr);
        printf("\nEnter theLogical Address: ");
        scanf("%d",&laddr);

        paddr = laddr / psize;
        disp = laddr % psize;
```

```
        if(ptable[paddr].pbit == 1 )
                phyaddr = baddr + (ptable[paddr].fno*psize) + disp;
        printf("\nThe Physical Address where the instruction present: %d",phyaddr);
}
void main()
{
        clrscr();
        info();
        assign();
        cphyaddr();
        getch();
}
```

OUTPUT:

```
Enter the Size of Physical memory: 16
Enter the size of Logical memory: 8
Enter the partition size: 2
The physical memory is divided into 8 no.of frames
The Logical memory is divided into 4 no.of pages
Enter the Frame number where page 0 must be placed: 5
Enter the Frame number where page 1 must be placed: 6
Enter the Frame number where page 2 must be placed: 7
Enter the Frame number where page 3 must be placed: 2
```

PAGE TABLE

PageAddress	FrameNo.	PresenceBit
0	5	1
1	6	1
2	7	1
3	2	1

FRAME TABLE

FrameAddress	PageNo
0	32555
1	32555
2	3
3	32555
4	32555
5	0
6	1
7	2

Process to create the Physical Address

Enter the Base Address: 1000

Enter theLogical Address: 3

The Physical Address where the instruction present: 1013

READER WRITER PROBLEM USING THREADS AND SEMAPHORES

PROGRAM:

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

sem_t readCountAccess;
sem_t databaseAccess;
int readCount=0;

void *Reader(void *arg);
void *Writer(void *arg);
int main()
{
 int i=0,NumberofReaderThread=0,NumberofWriterThread;
 sem_init(&readCountAccess,0,1);
 sem_init(&databaseAccess,0,1);
  pthread_t Readers_thr[100],Writer_thr[100];
 printf("\nEnter number of Readers thread(MAX 10)");
 scanf("%d",&NumberofReaderThread);
 printf("\nEnter number of Writers thread(MAX 10)");
 scanf("%d",&NumberofWriterThread);
  for(i=0;i<NumberofReaderThread;i++)
 {
  pthread_create(&Readers_thr[i],NULL,Reader,(void *)i);
 }
 for(i=0;i<NumberofWriterThread;i++)
 {
  pthread_create(&Writer_thr[i],NULL,Writer,(void *)i);
 }
 for(i=0;i<NumberofWriterThread;i++)
 {
  pthread_join(Writer_thr[i],NULL);
```

```
 }
 for(i=0;i<NumberofReaderThread;i++)
 {
  pthread_join(Readers_thr[i],NULL);
 }
 sem_destroy(&databaseAccess);
 sem_destroy(&readCountAccess);
 return 0;
}
void * Writer(void *arg)
{
  sleep(1);
 int temp=(int)arg;
 printf("\nWriter %d is trying to enter into database for modifying the data",temp);
 sem_wait(&databaseAccess);
 printf("\nWriter %d is writting into the database",temp);
 printf("\nWriter %d is leaving the database");
 sem_post(&databaseAccess);
}
void *Reader(void *arg)
{
 sleep(1);
 int temp=(int)arg;
 printf("\nReader %d is trying to enter into the Database for reading the data",temp);
 sem_wait(&readCountAccess);
 readCount++;
 if(readCount==1)
 {
  sem_wait(&databaseAccess);
  printf("\nReader %d is reading the database",temp);
 }
 sem_post(&readCountAccess);
 sem_wait(&readCountAccess);
 readCount--;
```

```
 if(readCount==0)
 {
  printf("\nReader %d is leaving the database",temp);
  sem_post(&databaseAccess);
 }
 sem_post(&readCountAccess);
}
```

OUTPUT:

```
Enter number of Readers thread(MAX 10)2
Enter number of Writers thread(MAX 10)1
Reader 0 is trying to enter into the Database for reading the data
Reader 0 is reading the database
Reader 0 is leaving the database
Reader 1 is trying to enter into the Database for reading the data
Reader 1 is reading the database
Reader 1 is leaving the database
Writer 0 is trying to enter into database for modifying the data
Writer 0 is writting into the database
Writer 0 is leaving the database
```

MEMORY MANAGEMENT TECHNIQUES

FIRST FIT

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#include<string.h>
int size;
char pid[4];
int occupied;
struct node * next;
};
void createnode();
void displaynode(struct node *);
void firstfit();
char pid[4];
struct node *p,*cur=NULL;
int main()
{
int option;
do
{
printf("\n press 1 to create node");
printf("\n press 4 to quit");
printf("\n your option");
scanf("%d",&option);
switch(option)
{
case 1:
createnode();
break;
case 2:
displaynode(p);
break;
```

```
case 3:
firstfit();
break;
case 4:break;
default:printf("press proper option");
}
}while(option!=4);

return 0;
}
void createnode()
{
int i,n,flag=1;
struct node * newnode;
printf("enter total no.of nodes to be created");
scanf("%d",&n);
for(i=0;i<n;i++)
{
newnode=(struct node*) malloc (sizeof(struct node));
newnode->next=NULL;
printf("\n enter starting address, ending address, process id,
 size of process,occupied");
scanf("%d %d %s %d %d",&newnode->start,&newnode->end,&newnode-
>pid,&newnode->size,&newnode->occupied);
if(flag==1)
{
cur=newnode;
p=cur;
cur->next=NULL;
flag=0;
}

else
{
```

```
cur->next=newnode;
cur=cur->next;
}
}
}
void displaynode(struct node *r)
{
while(r!=NULL)
{
printf("\n start address,end address,pid,size, occupied : %d %
d %s %d %d \n",r->start,r->end,r->pid,r->size,r->occupied);
r=r->next;
}
}
void firstfit()
{
struct node *q;
int psize,hole;
q=p;
printf("enter process id and size of fit");
scanf("%s %d",&pid,&psize);
while(q!=NULL)
{
hole=q->end-q->start;
if((q->occupied==0)&&(psize<=hole))
{
strcpy(q->pid,pid);
q->occupied=1;
q->size=psize;
return;
}
q=q->next;
}
}
```

OUTPUT:

```
 [os32@localhost ~]$ gcc first.c
[os32@localhost ~]$ ./a.out
 press 1 to create node
 press 2 to display node
 press 3 to allocate firstfitnode
 press 4 to quit
 your option1
enter total no.of nodes to be created:6

 enter starting address, ending address, pid, size of process, occupied 100 120 p
3 20 1
 enter starting address, ending address, process id, size of process,occupied 121 200
hole 79 0
 enter starting address, ending address, process id, size of process,occupied201 220 p
7 20 1
 enter starting address, ending address, process id, size of process,occupied311 410 p
55 99 1
 enter starting address, ending address, process id, size of process,occupied411 475 h
ole 64 0
 enter starting address, ending address, process id, size of process,occupied511 575 h
ole 56 0
 press 1 to create node
 press 2 to display node
 press 3 to allocate firstfitnode
 press 4 to quit
 your option2
 start address,end address,pid,size, occupied : 100 120 p3 20 1
 start address,end address,pid,size, occupied : 121 200 hole 79 0
 start address,end address,pid,size, occupied : 201 220 p7 20 1
 start address,end address,pid,size, occupied : 311 410 p55 99 1
 start address,end address,pid,size, occupied : 411 475 hole 64 0
 start address,end address,pid,size, occupied : 511 575 hole 56 0
```

press 1 to create node

press 2 to display node

press 3 to allocate firstfitnode

press 4 to quit

your option3

enter process id and size of fit p99 20

press 1 to create node

press 2 to display node

press 3 to allocate firstfitnode

press 4 to quit

your option2

start address,end address,pid,size, occupied : 100 120 p3 20 1

start address,end address,pid,size, occupied : 121 200 p99 20 1

start address,end address,pid,size, occupied : 201 220 p7 20 1

start address,end address,pid,size, occupied : 311 410 p55 99 1

start address,end address,pid,size, occupied : 411 475 hole 64 0

start address,end address,pid,size, occupied : 511 575 hole 56 0

BEST FIT

PROGRAM:

```
#include<stdio.h>
#include<stdlib.h>
#include<malloc.h>
#include<string.h>
struct node{
char pid[4];
int occupied;
struct node * next;
};
void createnode();
void displaynode(struct node *);
void searchbestnode();
void bestalloc(int,int);
int n;
char pid[4];
struct node *p,*cur=NULL;
int main(){
int option;
do{
printf("\n press 1 to create node");
printf("\n press 4 to quit");
printf("\n your option");
scanf("%d",&option);
switch(option){
case 1:
createnode();
break;
case 2:
displaynode(p);
break;
case 3:
```

```
searchbestnode();
break;
case 4:break;
default:printf("press proper option");
}
}while(option!=4);
return 0;
}
void createnode(){
int i,flag=1;
struct node * newnode;
printf("enter total no.of nodes to be created");
scanf("%d",&n);
for(i=0;i<n;i++){
newnode=(struct node*) malloc (sizeof(struct node));
newnode->next=NULL;
newnode->occupied=0;
if(flag==1){
cur=newnode;
p=cur;
cur->next=NULL;
flag=0;
}
else{
cur->next=newnode;
cur=cur->next;
}}}
void displaynode(struct node *r){
while(r!=NULL){
printf("\n start address,end address,pid,size, occupied : %d %
d %s %d %d \n",r->start,r->end,r->pid,r->size,r->occupied);
r=r->next;
}
}
```

```
void searchbestnode(){
struct node *q;
int psize,hole,bestaddr,besthole=9999;
q=p;
printf("enter process id,size to fit");
scanf("%s %d",&pid,&psize);
bestaddr=q->start;
while(q!=NULL){
if(q->occupied==0){
hole=q->end-q->start;
if(hole>psize)
if(hole<besthole){
besthole=hole;
bestaddr=q->start;
}
}q=q->next;
}
bestalloc(bestaddr,psize);
}
void bestalloc(int bestaddr,int psize){
struct node *t;
t=p;
while(t!=NULL){
if(t->start==bestaddr)
{
strcpy(t->pid,pid);
t->occupied=1;
t->size=psize;
}t=t->next;
}
}
```

OUTPUT:

```
[os32@localhost ~]$ gcc best.c
[os32@localhost ~]$ ./a.out
 press 1 to create node
 press 2 to display node
 press 3 to allocate firstfitnode
 press 4 to quit
 your option1
enter total no.of nodes to be created:6
 enter starting address, ending address, pid, size of process, occupied 100 120 p
3 20 1
 enter starting address, ending address, process id, size of process,occupied 121 200
hole 79 0
 enter starting address, ending address, process id, size of process,occupied201 220 p
7 20 1
 enter starting address, ending address, process id, size of process,occupied311 410 p
55 99 1
 enter starting address, ending address, process id, size of process,occupied411 475 h
ole 64 0
 enter starting address, ending address, process id, size of process,occupied511 575 h
ole 56 0
 press 1 to create node
 press 2 to display node
 press 3 to allocate bestfitnode
 press 4 to quit
 your option2
 start address,end address,pid,size, occupied : 100 120 p3 20 1
 start address,end address,pid,size, occupied : 121 200 hole 79 0
 start address,end address,pid,size, occupied : 201 220 p7 20 1
 start address,end address,pid,size, occupied : 311 410 p55 99 1
 start address,end address,pid,size, occupied : 411 475 hole 64 0
start address,end address,pid,size, occupied : 511 575 hole 56 0

 press 1 to create node
```

press 2 to display node

press 3 to allocate bestfitnode

press 4 to quit

your option3

enter process id and size of fit p99 20

press 1 to create node

press 2 to display node

press 3 to allocate bestfitnode

press 4 to quit

your option2

start address,end address,pid,size, occupied : 100 120 p3 20 1

start address,end address,pid,size, occupied : 121 200 hole 79 0

start address,end address,pid,size, occupied : 201 220 p7 20 1

start address,end address,pid,size, occupied : 311 410 p55 99 1

start address,end address,pid,size, occupied : 411 475 hole 64 0

start address,end address,pid,size, occupied : 511 575 p99 20 0

SINGLE LEVEL DIRECTORY

```
#include<stdio.h>
#include<conio.h>
main()
{
int master,s[20];
char f[20][20][20];
char d[20][20];
int i,j;
clrscr();
printf("enter number of directorios:");
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 file names :");
for(i=0;i<master;i++)
for(j=0;j<s[i];j++)
scanf("%s",&f[i][j]);
printf("\n");
printf(" directory\tsize\tfilenames\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\t",f[i][j]);
printf("\n");
}
printf("\t\n");
getch();}
```

TWO LEVEL DIRECTORY

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

```
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("%s\t",dir[i].fname[j][k]);
printf("\n\t\t");
}
printf("\n"); }
getch(); }
```