RNS INSTITUTE OF TECHNOLOGY

Dr. VISHNUVARDHAN ROAD, CHANNASANDRA, BENGALURU - 560 098

Department of Information Science & Engineering

OPERATING SYSTEMS LABORATORY MANUAL

III Semester

BCS303

Faculty-in-charge

Dr. Hema N

Dr. Bhagyashree Ambore

Ms. Ashwini R

RNS INSTITUTE OF TECHNOLOGY

Dr. VISHNUVARDHAN ROAD, CHANNASANDRA, BENGALURU -560 098

Department of Information Science and Engineering

VISION of the College

Building RNSIT into a World - Class Institution

MISSION of the College

To impart high quality education in Engineering, Technology and Management with a difference, enabling students to excel in their career by

1. Attracting quality Students and preparing them with a strong foundation in fundamentals so as *to achieve distinctions in various walks of life* leading to outstanding contributions.
2. Imparting value based, need based, and choice based and skill based professional education to the aspiring youth and *carving them into disciplined, World class Professionals* with *social responsibility.*
3. Promoting excellence in Teaching, Research and Consultancy that galvanizes academic consciousness among Faculty and Students.
4. Exposing Students to emerging frontiers of knowledge in various domains and make them suitable for Industry, Entrepreneurship, Higher studies, and Research & Development.
5. Providing freedom of action and choice for all the Stake holders with better visibility.

VISION of the Department

Building Information Technology Professionals by Imparting Quality Education and Inculcating Key Competencies

MISSION of the Department

1. Provide strong fundamentals through learner centric approach.
2. Instill technical, interpersonal, interdisciplinary skills and logical thinking for holistic development.
3. Train to excel in higher education, research, and innovation with global perspective.
4. Develop leadership and entrepreneurship qualities with societal responsibilities.

PROGRAM EDUCATIONAL OBJECTIVES (PEOs)

ISE Graduates within three-four years of graduation will have

- **PEO1**: Acquired the fundamentals of computers and applied knowledge of Information Science & Engineering and continue to develop their technical competencies by problem solving using programming.
- **PEO2**: Ability to formulate problems, attained the Proficiency to develop system/application software in a scalable and robust manner with various platforms, tools and frameworks to provide cost effective solutions.
- **PEO3**: Obtained the capacity to investigate the necessities of the software Product, adapt to technological advancement, promote collaboration and interdisciplinary activities, Protecting Environment and developing Comprehensive leadership.
- **PEO4**: Enabled to be employed and provide innovative solutions to real-world problems across different domains.
- **PEO5**: Possessed communication skills, ability to work in teams, professional ethics, social responsibility, entrepreneur and management, to achieve higher career goals, and pursue higher studies.

PROGRAM OUTCOMES (POs)

Engineering Graduates will be able to:

- **PO1: Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization for the solution of complex engineering problems
- **PO2: Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences and engineering sciences.
- **PO3: Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for public health and safety, and cultural, societal, and environmental considerations.
- **PO4: Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
- **PO5: Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools, including prediction and modeling to complex engineering activities, with an understanding of the limitations.
- **PO6: The engineer and society:** Apply reasoning informed by the contextual knowledge to assess Societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
- **PO7: Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

- **PO8: Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
- **PO9: Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
- **PO10: Communication:** Communicate effectively on complex engineering activities with the engineering community and with the society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
- **PO11: Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
- **PO12: Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

PROGRAM SPECIFIC OUTCOMES (PSOs)

ISE Graduates will have

- **PSO1 – Problem Solving Abilities:** Ability to demonstrate the fundamental and theoretical concepts, analyze the real time problems and develop customized software solutions by applying the knowledge of mathematics and algorithmic techniques.
- **PSO2 – Applied Engineering Skills:** Enable creative thinking, Ability to apply standard practices and strategies, technical skills in software design, development, integration of systems and management for improving the security, reliability and survivability of the infrastructure.
- **PSO3 – General Expertise and Higher Learning:** Ability to exchange knowledge effectively demonstrate the ability of team work, documentation skills, professional ethics, entrepreneurial skills and continuing higher education in the field of Information technology.

Course Outcomes (CO's):

SUBJECT NAME: OPERATING SYSTEMS LAB
CODE: BCS303 SEMESTER: III

CO No.	Course Outcome	Taxonomy Level
BCS303.1	Evaluate the performance of different types of CPU scheduling algorithms.	Evaluating
BCS303.2	Implement producer – consumer problem, reader – writers problem, Dining philosopher's problem.	Applying
BCS303.3	Simulate Banker's algorithm for deadlock avoidance.	Creating
BCS303.4	Implement paging replacement and disk scheduling techniques.	Applying
BCS303.5	Use different system calls for writing application programs.	Applying
BCS303.6	Ability to implement inter process communication between two processes.	Applying

GENERAL LABORATORY INSTRUCTIONS

1. Students are advised to come to the laboratory at least 5 minutes before (to starting time),those who come after 5 minutes will not be allowed into the lab.
2. Plan your task properly much before to the commencement, come prepared to the lab withthe program / experiment details.
3. Student should enter into the laboratory with:

    a. Laboratory observation notes with all the details (Problem statement, Aim, Algorithm,Procedure, Program, Expected Output, etc.,) filled in for the lab session.
    b. Laboratory Record updated up to the last session experiments.
    c. Formal dress code and Identity card.

4. Sign in the laboratory login register, write the TIME-IN, and occupy the computer system allotted to you by the faculty.
5. Execute your task in the laboratory, and record the results / output in the lab observation note book, and get certified by the concerned faculty.
6. All the students should be polite and cooperative with the laboratory staff, must maintain the discipline and decency in the laboratory.
7. Computer labs are established with sophisticated and highend branded systems, which should be utilized properly.
8. Students / Faculty must keep their mobile phones in SWITCHED OFF mode during the lab sessions. Misuse of the equipment, misbehaviors with the staff and systems etc., will attract severe punishment.
9. Students must take the permission of the faculty in case of any urgency to go out. If anybody found loitering outside the lab / class without permission during working hours will be treated seriously and punished appropriately.
10. Students should SHUT DOWN the computer system before he/she leaves the lab after completing the task (experiment) in all aspects. He/she must ensure the system / seat is kept properly.

Head of the Department **Principal**

CODE OF CONDUCT FOR THE LABORATORY

- All students must observe the dress code while in the laboratory
- Footwear is NOT allowed
- Foods, drinks and smoking are NOT allowed
- All bags must be left at the indicated place
- The lab timetable must be strictly followed
- Be PUNCTUAL for your laboratory session
- All programs must be completed within the given time
- Noise must be kept to a minimum
- Workspace must be kept clean and tidy at all time
- All students are liable for any damage to system due to their own negligence
- Students are strictly PROHIBITED from taking out any items from the laboratory
- Report immediately to the lab programmer if any damages to equipment

BEFORE LEAVING LAB:

- Arrange all the equipment and chairs properly.
- Turn off / shut down the systems before leaving.
- Please check the laboratory notice board regularly for updates.

Lab Write-up and Execution Rubrics (Max: 15 marks)

		Above Average	Average	Below Average
a.	**Understanding of problem and approach to solve. (5 Marks)**	Able to analyze the given problem and efficiently implement using suitable language.(5-4)	Able to analyze the problem and moderate understanding of test cases. (3-2)	No program write-up. (1-0)
b.	**Execution and Testing (5 Marks)**	Program is executed for varied and invalid inputs with valid results.(5-4)	Program is executed for some inputs. (3-2)	No Execution. (1-0)
c.	**Results and Documentation (5 Marks)**	Program and results obtained is well documented(5-4)	Program and results obtained is acceptably documented(3-2)	No Proper results and poor documentation. (1-0)

LAB Internal Assessment rubrics (Max: 10 marks)

		Above Average	Average	Below Average
a.	**Write-up (4 Marks)**	Able to write the complete code (4)	Able to write the code with few errors. (3-2)	Unable to write. (1-0)
b.	**Execution (4 Marks)**	Executed successfully for all the input set given (4)	Obtained Partially correct results. (3-2)	No Execution. (1-0)
c.	**Viva (2 Marks)**	Able to answer all the questions correctly (2)	Able to answer few questions (1)	Not answered any. (0)

Linux CASE STUDY: Perform a case study by installing and exploring various types of operating systems on a physical or logical (virtual) machine. (Linux Installation).

Instructions to Install Ubuntu 12.04 (LTS) along with Windows

Back Up Your Existing Data

This is highly recommended that you should take backup of your entire data before start with the installation process.

Obtaining System Installation Media

Download latest Desktop version of Ubuntu from this link:http://www.ubuntu.com/download/desktop

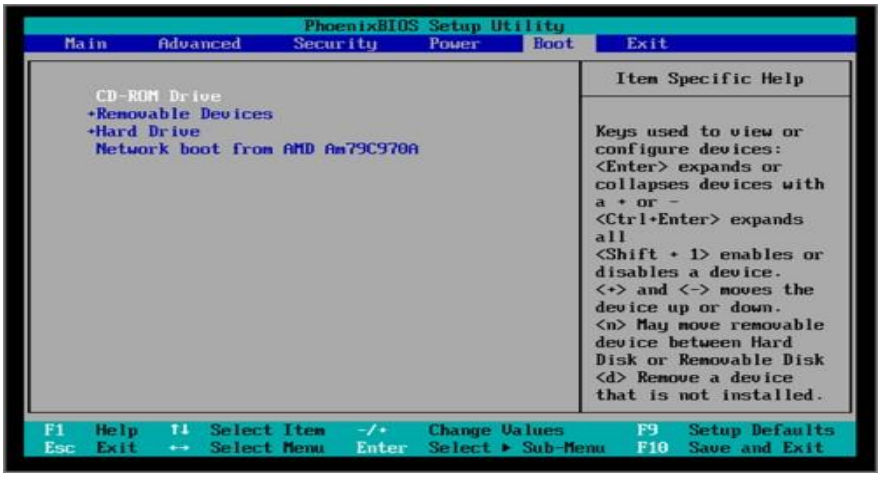
Booting the Installation System

There are several ways to boot the installation system. Some of the very popular ways are ,Booting from a CD ROM, Booting from a USB memory stick, and Booting from TFTP. Here we will learn how to boot installation system using a CD ROM. Before booting the installation system, one need to change the boot order and set CD-ROMas first boot device.

Changing the Boot Order of a Computers

As your computer starts, press the DEL, ESC, F1, F2, F8 or F10 during the initial startup screen. Depending on the BIOS manufacturer, a menu may appear. However, consult the hardware documentation for the exact key strokes. In my machine, its DEL key as shown in following screen-shot.

2. Find the Boot option in the setup utility. Its location depends on your BIOS. Select the Boot option from the menu, you can now see the options Hard Drive, CD-ROMDrive, Removable Devices Disk etc.
3. Change the boot sequence setting so that the CD-ROM is first. See the list of “Item SpecificHelp” in right side of the window and find keys which is used to toggle to change the boot sequence.



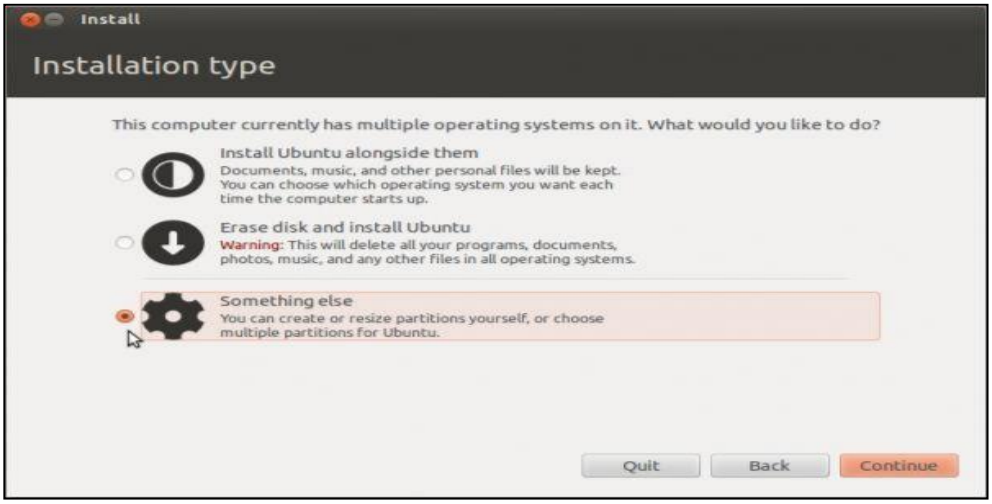
4. Insert the Ubuntu Disk in CD/DVD drive.
5. Save your changes. Instructions on the screen tell you how to save the changes on your computer. The computer will restart with the changed settings.

Machine should boot from CD ROM, Wait for the CD to load...

In a few minutes installation wizard will be started. Select your language and click the "Install Ubuntu" button to continue...

Optionally, you can choose to download updates while installing and/or install third party software, such as MP3 support. Be aware, though, that if you select those options, the entire installation process will be longer!

Since we are going to create partitions manually, select **Something else**, then click **Continue**. Keep in mind that even if you do not want to create partitions manually, it is better to select the same option as indicated here. This would insure that the installer will not overwrite yourWindows, which will destroy your data. The assumption here is that **sdb** will be used just for Ubuntu 12.04, and that there are no valuable data on it.



Where are you? Select your location and Click the "Continue" button.

Keyboard layout

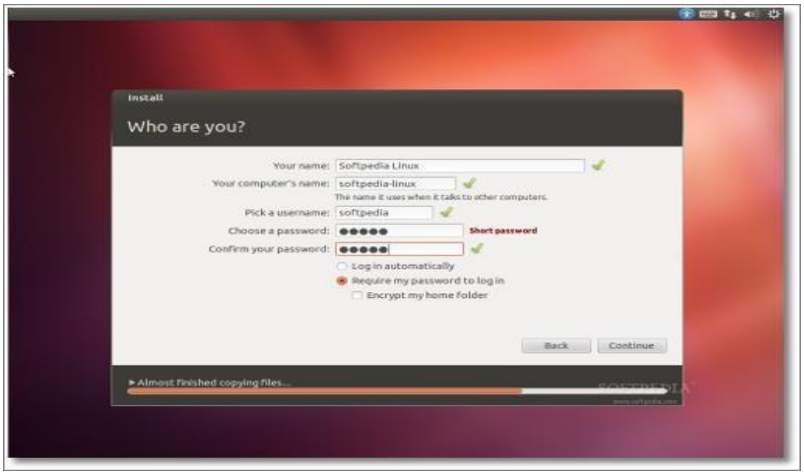
Select your keyboard layout and UK (English) and Click on “Continue” button.

Who are you?

Fill in the fields with your real name, the name of the computer (automatically generated,but can be overwritten), username, and the password.

Also at this step, there's an option called "Log in automatically." If you check it, you willautomatically be logged in to the Ubuntu desktop without giving the password.

Option "Encrypt my home folder," will encrypt your home folder. Click on the "Continue"button to continue...

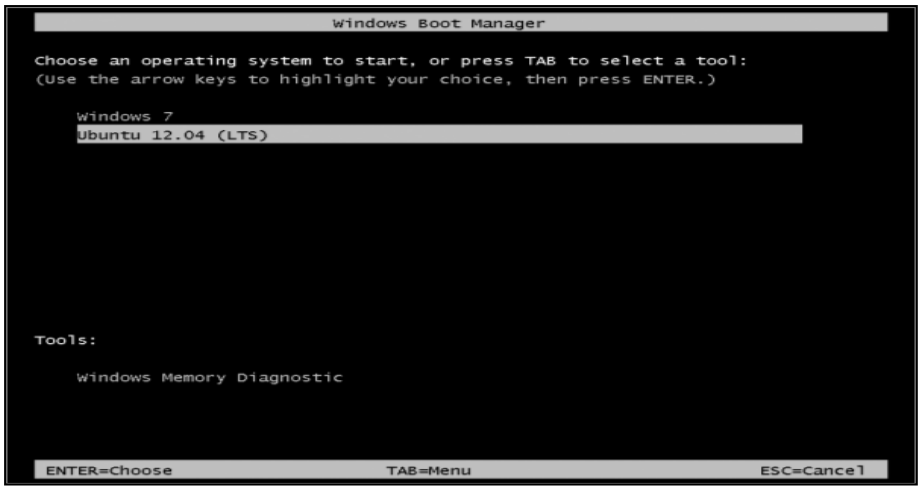


Now Ubuntu 12.04 LTS (Precise Pangolin) operating system will be installed.

It will take approximately 10-12 minutes (depending on computer's speed), a pop-up window will appear, notifying you that the installation is complete, and you'll need to restart thecomputer in order to use the newly installed Ubuntu operating system. Click the "Restart Now" button.

Please remove the CD and press the "Enter" key to reboot. The computer will be restarted. In a few seconds, you should see Windows 7′s boot menu with two entires listed – Windows 7 andUbuntu 12.04 (LTS).

Then you may choose to boot into Windows 7 or Ubuntu 12.04 using the UP/Down arrow key.



Please select Ubuntu 12.04 (LTS) and press Enter to boot the machine in Ubuntu 12.04 Linux.

Here you can see the users on the machine, Click on the user name and enter the password and press Enter key to login.

We have successfully install and login to Ubuntu 12.04 LTS.

PROGRAM 1: Develop a c program to implement the Process system calls (fork (), exec(), wait(), create process, terminate process)

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

int main() {
   // Variable to store the process ID.
   pid_t child_pid; char ch[3];
   // Create a child process using fork().
   child_pid = fork();
   if (child_pid == -1) {
      // Error occurred during fork.
      perror("fork");
      exit(EXIT_FAILURE);
   }
  if (child_pid == 0) {
      // This code executes in the child process.
      printf("Child process (PID: %d) is running.\n", getpid());
      // Execute a different program using exec().
      //abort(); child terminated abnormally by sending signal number 6 to the parent
         // return(-1); child terminates with exit status 255

         execl("/bin/date","date", NULL);

         exit(0);
   }
         else {
      // This code executes in the parent process.
      printf("Parent process (PID: %d) is waiting for the child to terminate.\n", getpid());
      // Wait for the child process to terminate using wait().
      int status;

         wait(&status);
         printf("parent resumes\n");
      if (WIFEXITED(status)) {
         printf("\nChild process (PID: %d) terminated with status %d.\n", child_pid,
WEXITSTATUS(status));
      } else if (WIFSIGNALED(status)) {
         printf("\nChild process (PID: %d) terminated due to signal %d.\n", child_pid,
WTERMSIG(status));
      } else {
         printf("\nChild process (PID: %d) terminated abnormally.\n", child_pid);}
         }

   return 0;
}
```

output

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc pgm1.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Parent process (PID: 11284) is waiting for the child to terminate.
Child process (PID: 11285) is running.
parent resumes

Child process (PID: 11285) terminated with status 255.

guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc pgm1.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Parent process (PID: 11356) is waiting for the child to terminate.
Child process (PID: 11357) is running.
parent resumes

Child process (PID: 11357) terminated due to signal 6.

guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc pgm1.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Parent process (PID: 11433) is waiting for the child to terminate.
Child process (PID: 11434) is running.
Tue Oct 10 10:50:07 IST 2023
parent resumes

Child process (PID: 11434) terminated with status 0.
```

PROGRAM 2: Simulate the following CPU scheduling algorithms to find turnaround time and waiting time a) FCFS b) SJF c) Round Robin d) Priority.

//FCFS

```
#include<stdio.h>
int main()
{
int bt[20], wt[20], tat[20], i, n;
 float wtavg, tatavg;

printf("\nEnter the number of processes -- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
printf("\nEnter Burst Time for Process %d -- ", i);
scanf("%d", &bt[i]);
}
wt[0] = wtavg = 0;
tat[0] = tatavg = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] +bt[i-1];
tat[i] = tat[i-1] +bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
```

```
}
printf("\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");
for(i=0;i<n;i++)
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", i, bt[i], wt[i], tat[i]);
printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);
}
```

output:

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc fcfs.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out

Enter the number of processes -- 3

Enter Burst Time for Process 0 -- 24

Enter Burst Time for Process 1 -- 3

Enter Burst Time for Process 2 -- 3
        PROCESS     BURST TIME          WAITING TIME       TURNAROUND TIME

        P0              24              0               24
        P1              3               24              27
        P2              3               27              30
Average Waiting Time -- 17.000000
Average Turnaround Time -- 27.000000
```

//SJF

```
#include<stdio.h>
int main()
{
int p[20], bt[20], wt[20], tat[20], i, k, n, temp;
float wtavg, tatavg;
printf("\nEnter the number of processes -- ");
scanf("%d", &n);
for(i=0;i<n;i++)
{
p[i]=i;
printf("Enter Burst Time for Process %d -- ", i);
scanf("%d", &bt[i]);
}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(bt[i]>bt[k])
{
temp=bt[i];
 bt[i]=bt[k];
bt[k]=temp;
```

```
temp=p[i];
p[i]=p[k];
p[k]=temp;
}
wt[0] = wtavg = 0;


tat[0] = tatavg = bt[0];

for(i=1;i<n;i++)
{
wt[i] = wt[i-1] +bt[i-1];
tat[i] = tat[i-1] +bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}

printf("\n\t PROCESS \tBURST TIME \t WAITING TIME\t TURNAROUND TIME\n");

for(i=0;i<n;i++)
printf("\n\t P%d \t\t %d \t\t %d \t\t %d", p[i], bt[i], wt[i], tat[i]);
printf("\nAverage Waiting Time -- %f", wtavg/n);
printf("\nAverage Turnaround Time -- %f", tatavg/n);
}
```

output:

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc sjf.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out

Enter the number of processes -- 3
Enter Burst Time for Process 0 -- 10
Enter Burst Time for Process 1 -- 8
Enter Burst Time for Process 2 -- 7
```

PROCESS	BURST TIME	WAITING TIME	TURNAROUND TIME
P2	7	0	7
P1	8	7	15
P0	10	15	25

```
Average Waiting Time -- 7.333333
Average Turnaround Time -- 15.666667
```

//Round Robin

```
#include<stdio.h>
int main()
{
int i,j,n,bu[10],wa[10],tat[10],t,ct[10],max;
float awt=0,att=0,temp=0;
printf("Enter the no of processes -- ");
scanf("%d",&n);
for(i=0;i<n;i++)
```

```
{
printf("\nEnter Burst Time for process %d -- ", i+1);
scanf("%d",&bu[i]);
ct[i]=bu[i];
}
printf("\nEnter the size of time slice -- ");
scanf("%d",&t);
max=bu[0];

for(i=1;i<n;i++)
if(max<bu[i])
max=bu[i];

for(j=0;j<(max/t)+1;j++)
for(i=0;i<n;i++)
if(bu[i]!=0)
if(bu[i]<=t)
{
   tat[i]=temp+bu[i];
    temp=temp+bu[i];
    bu[i]=0;
}

else
{
bu[i]=bu[i]-t;
temp=temp+t;
}
for(i=0;i<n;i++)
{
wa[i]=tat[i]-ct[i];
att+=tat[i];
awt+=wa[i];
}

printf("\nThe Average Turnaround time is -- %f",att/n);
printf("\nThe Average Waiting time is -- %f ",awt/n);
printf("\n\tPROCESS\t BURST TIME \t WAITING TIME\tTURNAROUND TIME\n");
for(i=0;i<n;i++)
printf("\t%d \t %d \t\t %d \t\t %d \n",i+1,ct[i],wa[i],tat[i]);

}
```

output:

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc rr.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Enter the no of processes -- 3

Enter Burst Time for process 1 -- 10

Enter Burst Time for process 2 -- 8

Enter Burst Time for process 3 -- 7
```

```
Enter the size of time slice -- 5

The Average Turnaround time is -- 22.666666
The Average Waiting time is -- 14.333333
        PROCESS       BURST TIME          WAITING TIME      TURNAROUND TIME
              1              10                  10                20
              2              8                   15                 23
              3              7                   18                 25
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Enter the no of processes -- 3

Enter Burst Time for process 1 -- 10

Enter Burst Time for process 2 -- 8

Enter Burst Time for process 3 -- 7

Enter the size of time slice -- 4

The Average Turnaround time is -- 22.666666
The Average Waiting time is -- 14.333333
        PROCESS       BURST TIME          WAITING TIME      TURNAROUND TIME
        1       10          15          25
        2       8           12          20
        3       7           16          23
```

//Priority

```
#include<stdio.h>
int main()
{
int p[20],bt[20],pri[20], wt[20],tat[20],i, k, n, temp;
float wtavg, tatavg;
printf("Enter the number of processes --- ");
scanf("%d",&n);
for(i=0;i<n;i++)
{
p[i] = i;
printf("Enter only positive numbers\n");
printf("Enter the Burst Time & Priority of Process %d --- ",i);
scanf("%d %d",&bt[i], &pri[i]);
}
for(i=0;i<n;i++)
for(k=i+1;k<n;k++)
if(pri[i] > pri[k])
{
temp=p[i];
p[i]=p[k];
p[k]=temp;

temp=bt[i];
```

```
bt[i]=bt[k];
bt[k]=temp;

temp=pri[i];
pri[i]=pri[k];
pri[k]=temp;
}
wtavg = wt[0] = 0;
tatavg = tat[0] = bt[0];
for(i=1;i<n;i++)
{
wt[i] = wt[i-1] + bt[i-1];
tat[i] = tat[i-1] + bt[i];
wtavg = wtavg + wt[i];
tatavg = tatavg + tat[i];
}

printf("\nPROCESS\t\tPRIORITY\tBURST TIME\tWAITING TIME\tTURNAROUND TIME");
for(i=0;i<n;i++)
printf("\n%d \t\t %d \t\t %d \t\t %d \t\t %d ",p[i],pri[i],bt[i],wt[i],tat[i]);
printf("\nAverage Waiting Time is --- %f",wtavg/n);
printf("\nAverage Turnaround Time is --- %f",tatavg/n);

}
```

output:

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Enter the number of processes --- 3
Enter the Burst Time & Priority of Process 0 --- 10 3
Enter the Burst Time & Priority of Process 1 --- 8 1
Enter the Burst Time & Priority of Process 2 --- 7 2

PROCESS         PRIORITY     BURST TIME    WAITING TIME      TURNAROUND TIME
1               1            8             0                 8
2               2            7             8                 15
0               3            10            15                25
Average Waiting Time is --- 7.666667
Average Turnaround Time is --- 16.000000
```

PROGRAM 3: Develop a C program to simulate producer-consumer problem using semaphores.

```
//producer consumer

#include<stdio.h>
#include <stdlib.h>
int mutex = 1;
 int full = 0;
int empty = 3, x = 0;
void producer()
{
```

```
    --mutex;
    ++full;
    --empty;
    x++;
    printf("\nProducer produces item %d",x);
    ++mutex;
}

void consumer()
{
    --mutex;
    --full;
    ++empty;
    printf("\nConsumer consumes item %d",x);
    x--;
    ++mutex;
}
int main()
{
    int n, i;
    printf("\n1. Press 1 for Producer"
           "\n2. Press 2 for Consumer"
           "\n3. Press 3 for Exit");

#pragma omp critical

    for (i = 1; i > 0; i++) {

        printf("\nEnter your choice:");
        scanf("%d", &n);
        // Switch Cases
        switch (n) {
        case 1:
            if ((mutex == 1) && (empty != 0))
            {
                producer();
            }

            else
            {
                printf("Buffer is full!");
            }
            break;

        case 2:
            if ((mutex == 1)&& (full != 0))
            {
                consumer();
            }

            else
            {
                printf("Buffer is empty!");
```

```
            }
            break;
        case 3:
            exit(0);
            break;
        }
    }
}
```

output:

guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out

1. Press 1 for Producer
2. Press 2 for Consumer
3. Press 3 for Exit
Enter your choice:1

Producer produces item 1
Enter your choice:1

Producer produces item 2
Enter your choice:1

Producer produces item 3
Enter your choice:1
Buffer is full!
Enter your choice:2

Consumer consumes item 3
Enter your choice:2

Consumer consumes item 2
Enter your choice:2

Consumer consumes item 1
Enter your choice:2
Buffer is empty!
Enter your choice:3

PROGRAM 4: Develop a C program which demonstrates interprocess communication between a reader process and a writer process. Use mkfifo, open, read, write and close APIs in your program.

```
// Create seperate files for reader and writer processes
// two terminals to run the program
// unless the writer process writes into the buffer, reader cannot read
/*Writer Process*/
#include <stdio.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
```

```
int main()
{
 int fd;
 char buf[1024]="Hello RNSIT";
 /* create the FIFO (named pipe) */
 char * myfifo = "/tmp/guest-uivabk/Desktop/tmp";
 mkfifo(myfifo, 0666);
 printf("Run Reader process to read the FIFO File\n");
 fd = open(myfifo, O_WRONLY);
 write(fd,buf,sizeof(buf));
 close(fd);
 unlink(myfifo); /* remove the FIFO */
 return 0;
}


/* Reader Process */
 #include <fcntl.h>
 #include <sys/stat.h>
 #include <sys/types.h>
 #include <unistd.h>
 #include <stdio.h>
 #define MAX_BUF 1024
 int main()
{
 int fd;
 /* A temp FIFO file is not created in reader */
 char *myfifo = "/tmp/guest-uivabk/Desktop/tmp";
 char buf[MAX_BUF];
 /* open, read, and display the message from the FIFO */
 fd = open(myfifo, O_RDONLY);
 read(fd, buf, MAX_BUF);
 printf("Reader process has read : %s\n", buf);
 close(fd);
 return 0;
}
```

********output ************

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc writer.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Run Reader process to read the FIFO File


guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc reader.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Reader process has read : Hello RNSIT
```

PROGRAM 5: Develop a C program to simulate Bankers Algorithm for DeadLock Avoidance.

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

```
int all[10];
int max[10];
int need[10];
int flag;
};
void main()
{
struct file f[10]; int fl;
int i, j, k, p, b, n, r, g, cnt=0, id, newr;
int avail[10],seq[10];

printf("Enter number of processes -- ");
 scanf("%d",&n);
printf("Enter number of resources -- ");
 scanf("%d",&r);
for(i=0;i<n;i++)
{
printf("Enter details for P%d",i);
printf("\nEnter allocation\t -- \t");
for(j=0;j<r;j++)
scanf("%d",&f[i].all[j]);
printf("Enter Max\t\t -- \t");
for(j=0;j<r;j++)
scanf("%d",&f[i].max[j]);
f[i].flag=0;
}

printf("\nEnter Available Resources\t -- \t");
for(i=0;i<r;i++)
scanf("%d",&avail[i]);
for(i=0;i<n;i++)
{
for(j=0;j<r;j++)
{
 f[i].need[j]=f[i].max[j]-f[i].all[j];
if(f[i].need[j]<0)
f[i].need[j]=0;
}
}
cnt=0; fl=0;

while(cnt!=n)
{
g=0;
for(j=0;j<n;j++)
{
if(f[j].flag==0)
{
b=0;
for(p=0;p<r;p++)
{
if(avail[p]>=f[j].need[p])
    b=b+1;
```

```
else
   b=b-1;
}
if(b==r)
{
printf("\nP%d is visited",j);
seq[fl++]=j;
f[j].flag=1;
for(k=0;k<r;k++)
avail[k]=avail[k]+f[j].all[k];
cnt=cnt+1;
printf("(");
for(k=0;k<r;k++)
printf("%3d",avail[k]);
printf(")");
g=1;
}
}
}
if(g==0)
{
printf("\n REQUEST NOT GRANTED -- DEADLOCK OCCURRED");
printf("\n SYSTEM IS IN UNSAFE STATE");
goto y;
}
}
printf("\nSYSTEM IS IN SAFE STATE");
printf("\nThe Safe Sequence is -- (");
for(i=0;i<fl;i++)
printf("P%d ",seq[i]);
printf(")");
y: printf("\nProcess\t\tAllocation\t\tMax\t\t\tNeed\n");
for(i=0;i<n;i++)
{
printf("P%d\t",i);
for(j=0;j<r;j++)
printf("%6d",f[i].all[j]);
for(j=0;j<r;j++)
printf("%6d",f[i].max[j]);
for(j=0;j<r;j++)
printf("%6d",f[i].need[j]); printf("\n");
}

}
```

*******output*******

```
guest-yk70yi@student-OptiPlex-3040:~$ cc bankers.c
guest-yk70yi@student-OptiPlex-3040:~$ ./a.out
Enter number of processes -- 5
Enter number of resources -- 3
Enter details for P0
Enter allocation        --      0 1 0
Enter Max               --      7 5 3
```

```
Enter details for P1
Enter allocation        --      2 0 0
Enter Max               --      3 2 2
Enter details for P2
Enter allocation        --      3 0 2
Enter Max               --      9 0 2
Enter details for P3
Enter allocation        --      2 1 1
Enter Max               --      2 2 2
Enter details for P4
Enter allocation        --      0 0 2
Enter Max               --      4 3 3

Enter Available Resources       --      3 3 2

P1 is visited(  5  3  2)
P3 is visited(  7  4  3)
P4 is visited(  7  4  5)
P0 is visited(  7  5  5)
P2 is visited( 10  5  7)
SYSTEM IS IN SAFE STATE
The Safe Sequence is -- (P1 P3 P4 P0 P2 )
```

Process	Allocation			Max			Need		
P0	0	1	0	7	5	3	7	4	3
P1	2	0	0	3	2	2	1	2	2
P2	3	0	2	9	0	2	6	0	0
P3	2	1	1	2	2	2	0	1	1
P4	0	0	2	4	3	3	4	3	1

PROGRAM 6: Develop a C program to simulate the following contiguous memory allocation Techniques: a) Worst fit b) Best fit c) First fit.

```
//**********FIRST FIT**********

#include<stdio.h>

#define max 25

void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - First Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
```

```
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i]; if(temp>=0)
{
ff[i]=j;
break;
}
}
}
frag[i]=temp; bf[ff[i]]=1;
}
printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
//getch();
}
```

***********OUTPUT**************

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out

        Memory Management Scheme - First Fit
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               1               1               5               4
2               4               3               7               3


guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out

        Memory Management Scheme - First Fit
Enter the number of blocks:3
Enter the number of files:2
```

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:4
File 2:1

File_no:	File_size :	Block_no:	Block_size:	Fragement
1	4	1	5	1
2	1	2	2	1

//*****************BEST FIT*****************************

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

void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,lowest=10000;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - Best Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{
for(j=1;j<=nb;j++)
{
if(bf[j]!=1)
{
temp=b[j]-f[i]; if(temp>=0)
if(lowest>temp)
{
ff[i]=j;
lowest=temp;
}
}
```

```
}
frag[i]=lowest; bf[ff[i]]=1; lowest=10000;
}
printf("\nFile No\tFile Size \tBlock No\tBlock Size\tFragment");
for(i=1;i<=nf && ff[i]!=0;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);
//getch();
}
```

OUTPUT:

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc worstfit.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out

        Memory Management Scheme - Best Fit
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4
```

File No	File Size	Block No	Block Size	Fragment
1	1	2	2	1
2	4	1	5	1

```
// WORST FIT
#include<stdio.h>
#define max 25
void main()
{
int frag[max],b[max],f[max],i,j,nb,nf,temp,highest=0;
static int bf[max],ff[max];
printf("\n\tMemory Management Scheme - Worst Fit");
printf("\nEnter the number of blocks:");
scanf("%d",&nb);
printf("Enter the number of files:");
scanf("%d",&nf);
printf("\nEnter the size of the blocks:-\n");
for(i=1;i<=nb;i++)
{
printf("Block %d:",i);
scanf("%d",&b[i]);
}
printf("Enter the size of the files :-\n");
```

```
for(i=1;i<=nf;i++)
{
printf("File %d:",i);
scanf("%d",&f[i]);
}
for(i=1;i<=nf;i++)
{

for(j=1;j<=nb;j++)
{
if(bf[j]!=1) //if bf[j] is not allocated
{
temp=b[j]-f[i]; if(temp>=0)
if(highest<temp)
{
 ff[i]=j; highest=temp;
}
}
}
frag[i]=highest;
bf[ff[i]]=1;
highest=0;
}

printf("\nFile_no:\tFile_size :\tBlock_no:\tBlock_size:\tFragement");
for(i=1;i<=nf;i++)
printf("\n%d\t\t%d\t\t%d\t\t%d\t\t%d",i,f[i],ff[i],b[ff[i]],frag[i]);

}
```

OUTPUT:

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc worstfit.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out

        Memory Management Scheme - Worst Fit
Enter the number of blocks:3
Enter the number of files:2

Enter the size of the blocks:-
Block 1:5
Block 2:2
Block 3:7
Enter the size of the files :-
File 1:1
File 2:4

File_no:        File_size :     Block_no:       Block_size:     Fragement
1               1               3               7               6
2               4               1               5               1
```

PROGRAM 7: Develop a C program to simulate page replacement algorithms: a) FIFO b) LRU

```
//FIFO
#include<stdio.h>
void main()
{
int i, j, k, f, pf=0, count=0, rs[25], m[10], n;
printf("\n Enter the length of reference string -- ");
scanf("%d",&n);
printf("\n Enter the reference string -- ");
for(i=0;i<n;i++)
scanf("%d",&rs[i]);
printf("\n Enter no. of frames -- ");
scanf("%d",&f);
for(i=0;i<f;i++)
m[i]=-1;

printf("\n The Page Replacement Process is -- \n");
for(i=0;i<n;i++)
{
for(k=0;k<f;k++)
{
if(m[k]==rs[i])
break;
}
if(k==f)
{
m[count++]=rs[i];
pf++;
}

for(j=0;j<f;j++)
printf("\t%d",m[j]);
if(k==f)
printf("\tPF No. %d",pf);
printf("\n");
if(count==f)
count=0;
}
printf("\n The number of Page Faults using FIFO are %d",pf);
}
```

OUTPUT:

```
guest-yk70yi@student-OptiPlex-3040:~$ ./a.out

 Enter the length of reference string -- 20

 Enter the reference string -- 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1

 Enter no. of frames -- 3

 The Page Replacement Process is --
         7       -1      -1      PF No. 1
```

```
7       0       -1      PF No. 2
7       0       1       PF No. 3
2       0       1       PF No. 4
2       0       1
2       3       1       PF No. 5
2       3       0       PF No. 6
4       3       0       PF No. 7
4       2       0       PF No. 8
4       2       3       PF No. 9
0       2       3       PF No. 10
0       2       3
0       2       3
0       1       3       PF No. 11
0       1       2       PF No. 12
0       1       2
0       1       2
7       1       2       PF No. 13
7       0       2       PF No. 14
7       0       1       PF No. 15
```

The number of Page Faults using FIFO are 15

//LRU

```
#include<stdio.h>
int main()
{
int i, j , k, min, rs[25], m[10], count[10], flag[25], n, f, pf=0, next=1;

printf("Enter the length of reference string -- ");
scanf("%d",&n);
printf("Enter the reference string -- ");
for(i=0;i<n;i++)
{
scanf("%d",&rs[i]);
flag[i]=0;
}
printf("Enter the number of frames -- ");
scanf("%d",&f);
for(i=0;i<f;i++)
{
count[i]=0;
m[i]=-1;
}
printf("\nThe Page Replacement process is -- \n");
for(i=0;i<n;i++)
{
for(j=0;j<f;j++)
{
if(m[j]==rs[i])
{
flag[i]=1;
 count[j]=next;
```

```
next++;
}
}
if(flag[i]==0)
{
if(i<f)
{
m[i]=rs[i];
count[i]=next;
next++;
}
else
{
min=0;
for(j=1;j<f;j++)
if(count[min] > count[j])
   min=j;

m[min]=rs[i];
count[min]=next;
next++;

}
pf++;
}

for(j=0;j<f;j++)
printf("%d\t", m[j]);
if(flag[i]==0)
printf("PF No. -- %d" , pf);
printf("\n");
}
printf("\nThe number of page faults using LRU are %d",pf);
}
```

OUTPUT:

```
guest-yk70yi@student-OptiPlex-3040:~$ ./a.out
Enter the length of reference string -- 20
Enter the reference string -- 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Enter the number of frames -- 3

The Page Replacement process is --
7       -1      -1      PF No. -- 1
7       0       -1      PF No. -- 2
7       0       1       PF No. -- 3
2       0       1       PF No. -- 4
2       0       1
2       0       3       PF No. -- 5
2       0       3
4       0       3       PF No. -- 6
4       0       2       PF No. -- 7
4       3       2       PF No. -- 8
```

```
0       3       2       PF No. -- 9
0       3       2
0       3       2
1       3       2       PF No. -- 10
1       3       2
1       0       2       PF No. -- 11
1       0       2
1       0       7       PF No. -- 12
1       0       7
1       0       7
```

The number of page faults using LRU are 12

PROGRAM 8: Simulate following File Organization Techniques a) Single level directory b) Two level directory

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define MAX_FILES 10
#define MAX_FILENAME_LENGTH 10
char directoryName[MAX_FILENAME_LENGTH];
char filenames[MAX_FILES][MAX_FILENAME_LENGTH];
int fileCount = 0;
void createFile() {
if (fileCount < MAX_FILES) {
char filename[MAX_FILENAME_LENGTH];
printf("Enter the name of the file: ");
scanf("%s", filename);
strcpy(filenames[fileCount], filename);
fileCount++;
printf("File %s created.\n", filename);
} else {
printf("Directory is full, cannot create more files.\n");
}
}
void deleteFile() {
if (fileCount == 0) {
printf("Directory is empty, nothing to delete.\n");
return;
}
char filename[MAX_FILENAME_LENGTH];
printf("Enter the name of the file to delete: ");
scanf("%s", filename);
int found = 0;
for (int i = 0; i < fileCount; i++) {
if (strcmp(filename, filenames[i]) == 0) {
found = 1;
printf("File %s is deleted.\n", filename);
strcpy(filenames[i], filenames[fileCount - 1]);
fileCount--;
```

```
break;
}
}
if (!found) {
printf("File %s not found.\n", filename);
}
}
void displayFiles() {
if (fileCount == 0) {
printf("Directory is empty.\n");
} else {
printf("Files in the directory %s:\n", directoryName);
for (int i = 0; i < fileCount; i++) {
printf("\t%s\n", filenames[i]);
}
}
}
int main() {
printf("Enter the name of the directory: ");
scanf("%s", directoryName);
while (1) {
printf("\n1. Create File\t2. Delete File\t3. Search File\n4. Display Files\t5. Exit\nEnter your choice:");
int choice;
scanf("%d", &choice);
switch (choice) {
case 1:
createFile();
break;
case 2:
deleteFile();
break;
case 3:
printf("Enter the name of the file: ");
char searchFilename[MAX_FILENAME_LENGTH];
scanf("%s", searchFilename);
int found = 0;
for (int i = 0; i < fileCount; i++) {
if (strcmp(searchFilename, filenames[i]) == 0) {
found = 1;
printf("File %s is found.\n", searchFilename);
break;
}
}
if (!found) {
printf("File %s not found.\n", searchFilename);
}
break;
case 4:
displayFiles();
break;
case 5:
exit(0);
default:
```

```
printf("Invalid choice, please try again.\n");
}
}
return 0;
}
```

OUTPUT:

```
guest-yk70yi@student-OptiPlex-3040:~$ ./a.out
Enter the name of the directory: rns

1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice:1
Enter the name of the file: cse
File cse created.

1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice:1
Enter the name of the file: ise
File ise created.

1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice:4
Files in the directory rns:
        cse
        ise


1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice:3
Enter the name of the file: ai
File ai not found.

1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice:3
Enter the name of the file: ise
File ise is found.

1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice:2
Enter the name of the file to delete: ise
File ise is deleted.

1. Create File  2. Delete File  3. Search File
4. Display Files        5. Exit
Enter your choice:4
Files in the directory rns:
```

```
1. Create File  2. Delete File  3. Search File
4. Display Files          5. Exit
Enter your choice:5
*****************************************************
// Two-level
```

PROGRAM 9: Develop a C program to simulate the Linked file allocation strategies.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
   int f[50], p, i, st, len, j, c, k, a, fn=0;

   for (i = 0; i < 50; i++)
      f[i] = 0;
  /* printf("Enter how many blocks already allocated: ");
   scanf("%d", &p);
   printf("Enter blocks already allocated: ");
   for (i = 0; i < p; i++) {
      scanf("%d", &a);
      f[a] = 1;
   }*/
x:
         fn=fn+1;
         printf("Enter index starting block and length: ");
    scanf("%d%d", &st, &len);
   k = len;

  if (f[st] == 0)
   {
      for (j = st; j < (st + k); j++){
         if (f[j] == 0)
         {
            f[j] = fn;
            printf("%d-------->%d\n", j, f[j]);
         }
         else{
            printf("%d Block is already allocated \n", j);
            k++;
         }
      }
   }
   else
      printf("%d starting block is already allocated \n", st);
   printf("Do you want to enter more file(Yes - 1/No - 0)");
   scanf("%d", &c);
   if (c == 1)

    goto x;
   else
```

```
        exit(0);
    return 0;
}
```

OUTPUT:

*****only allocation is simulated*********

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ cc pgm9.c
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
Enter index starting block and length: 1 3
1-------->1
2-------->1
3-------->1
Do you want to enter more file(Yes - 1/No - 0)1
Enter index starting block and length: 5 3
5-------->2
6-------->2
7-------->2
Do you want to enter more file(Yes - 1/No - 0)1
Enter index starting block and length: 4 5
4-------->3
5 Block is already allocated
6 Block is already allocated
7 Block is already allocated
8-------->3
9-------->3
10-------->3
11-------->3
Do you want to enter more file(Yes - 1/No - 0)0
```

PROGRAM 10: Develop a C program to simulate SCAN disk scheduling algorithm.

```
// head movement towards left

#include<stdio.h>
int main()
{
int t[20], d[20], h, i, j, n, temp, k, atr[20], tot, p, sum=0;

printf("enter the no of tracks to be traveresed");
scanf("%d'",&n);
printf("enter the position of head");
scanf("%d",&h);
t[0]=0;
t[1]=h;
printf("enter the tracks");
for(i=2;i<n+2;i++)
scanf("%d",&t[i]);
for(i=0;i<n+2;i++)
{
for(j=0;j<(n+2)-i-1;j++)
{
if(t[j]>t[j+1])
```

```
{
temp=t[j];
t[j]=t[j+1];
t[j+1]=temp;

}
}
}

for(i=0;i<n+2;i++)
if(t[i]==h)
{
   j=i;
k=i;
p=0;
}
while(t[j]!=0)
{
atr[p]=t[j];
j--;
p++;
}
atr[p]=t[j];
for(p=k+1;p<n+2;p++,k++)
atr[p]=t[k+1];
printf("seek sequence is:");
for(j=0;j<n+1;j++)
{
if(atr[j]>atr[j+1])
   d[j]=atr[j]-atr[j+1];
else
   d[j]=atr[j+1]-atr[j];
sum+=d[j];
printf("\n%d", atr[j]);
}
printf("\nTotal head movements:%d",sum);
}
```

OUTPUT:

```
guest-uivabk@student-OptiPlex-3040:~/Desktop$ ./a.out
enter the no of tracks to be traveresed8
enter the position of head50
enter the tracks176
79
34
60
92
11
41
114
seek sequence is:
50
41
```

34
11
0
60
79
92
114
Total head movements:226

VIVA – VOCE QUESTIONS

1. What is an operating system?
2. What are short, long and medium-term scheduling?
3. Explain the concept of the batched operating systems?
4. What is purpose of different operating systems?
5. What is virtual memory?
6. What is Throughput, Turnaround time, waiting time and Response time?
7. What are the various components of a computer system?
8. What is a Real-Time System?
9. Explain the concept of the Distributed systems?
10. What are the different operating systems?
11. What is busy waiting?
12. What are system calls?
13. What are various scheduling queues?
14. What are the states of a process?
15. What is a job queue?
16. What is a ready queue?
17. What are turnaround time and response time?
18. What are the operating system components?
19. Why thread is called as a lightweight process?
20. What are operating system services?
21. What is a process?
22. What are the different job scheduling in operating systems?
23. What is dual-mode operation?
24. What is a device queue?
25. What are the different types of Real-Time Scheduling?
26. What is starvation?
27. What is a long term scheduler & short term schedulers?
28. What is fragmentation?
29. What is the state of the processor, when a process is waiting for some event to occur?
30. What is the difference between Primary storage and secondary storage?
31. What is the difference between Complier and Interpreter?
32. What are the different functions of Scheduler
33. What are local and global page replacements?
34. What are the different operating systems?

35. What are the basic functions of an operating system?

36. What is kernel?

37. What is a process?

38. What are the states of a process?

40. What is starvation and aging?

41. What is context switching?

42. What is a thread?

43. What is process synchronization?

44. What is virtual memory?

45. What is fragmentation? Tell about different types of fragmentation?

46. What is logical and physical addresses space?

47. What is Memory-Management Unit (MMU)?

48. What is a Real-Time System?

49. What is process migration?

50. Difference between Primary storage and secondary storage?

51. Define compactions.

52. What are the disadvantages of context switching?

53. What is Dispatcher?

54. What is the Translation Lookaside Buffer (TLB)?