**OPERATING SYSTEM (CSE 202)**

**LAB FILE**

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

Logo

Description automatically generated

Submitted to:

Ms. Seema Sharma

Submitted by:

Umang Sehgal

A2305219288

4CSE-4Y

AMITY SCHOOL OF ENGINEERING AND TECHNOLOGY

AMITY UNIVERSITY UTTAR PRADESH

NOIDA – 201301

**Index**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **S.No.** | **Assignment Category** | **Code** | **Experiment No.** | **Name of Experiment** | **Date of Allotment of Experiment** | **Date of Evaluation** | **Max. Marks** | **Marks Obtained** | **Signature of Faculty** |
| **1.** | **Mandatory**  **Experiment** | **LR (10)** | **1** | To explore basic Linux commands | 17/12/20 |  | 1 |  |  |
| **2.** | **2** | To explore basic Linux commands | 24/12/20 |  | 1 |  |  |
| **3.** | **3** | To explore Linux file commands | 07/01/21 |  | 1 |  |  |
| **4.** | **4** | To explore Linux file commands | 14//01/21 |  | 1 |  |  |
| **5.** | **5** | Programs based on Shell Scripting | 21/01/21 |  | 1 |  |  |
| **6.** | **6** | Programs based on Shell Scripting | 04/02/21 |  | 1 |  |  |
| **7.** | **7** | Programs based on Shell Scripting | 11/02/21 |  | 1 |  |  |
| **8.** | **8** | Program to simulate First Come First Serve CPU scheduling algorithm | 18/02/21 |  | 1 |  |  |
| **9.** | **9** | Program to simulate Shortest Job First CPU scheduling algorithm | 04/03/21 |  | 1 |  |  |
| **10.** | **10** | Program to simulate round robin scheduling algorithm | 18/03/21 |  | 1 |  |  |
| **11.** | **11** | Program to simulate banker’s algorithm | 25/03/21 |  | 1 |  |  |
| **12.** | **12** | Program to simulate SSTF disk scheduling | 01/04/21 |  | 1 |  |  |
|  | Viva |  | **Viva** |  |  |  |  |  |  |

**Experiment – 1**

**Objective:** To explore basic Linux commands.

**Software used:** zsh terminal

**Theory:**

1. man: It is used to display the user manual of any command

Syntax: man <command>



Text

Description automatically generated

1. echo: It is used to display line of text/string that are passed into the

argument.

Syntax: echo <string to be displayed>



1. help: it is used to display information about the command from the shell’s documents.

Syntax: run-help <command> or <command> --help



Text

Description automatically generated

1. clear: it is used to clear the terminal screen

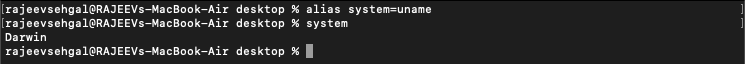
Syntax: clear

Text

Description automatically generated

1. alias: it is used to create custom shortcuts to represent commands.

Syntax: alias <aliasName>=<command>



1. uname: it is used to display the basic information about the operating system name

and system hardware.

Syntax: uname <options>



1. who: it is used to display a list of users, who are currently logged into the system.

Syntax: who



1. whoami: it is used to display the username of the current user when the command is

invoked.

Syntax: whoami



1. pwd: it is used to display the current working directory.

Syntax: pwd



1. history: it is used to display a list of commands that have been recently used.

Syntax: history

Text

Description automatically generated

1. cat: it is used to create single or multiple files, view content of file/s, concatenate files

and redirect output in terminal or files.

Syntax: cat > filename (To create a new file)

cat filename (To open a file)

cat >> filename (To append the content of a file)

cat file1>file2 (To copy content of file 1 into file 2)

Text

Description automatically generated

**Result:** Successfully executed basic Linux commands.

**Experiment – 2**

**Objective:** To explore Linux file commands.

**Software used:** zsh terminal

**Theory:**

1. cd: It is used to change the current working directory

Syntax: cd <new\_directory>

Graphical user interface, text

Description automatically generated

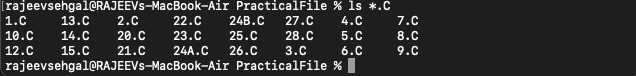
1. ls: It is used to list the files in the current directory

Syntax: ls <options>

Text

Description automatically generated with medium confidence

ls \* - it is used as a wildcard, to search files with the mentioned extension



ls -l – used to display a long list of information

Graphical user interface

Description automatically generated with medium confidence

ls -R – used to display all the fills in the current directory and the sub

directories

Text

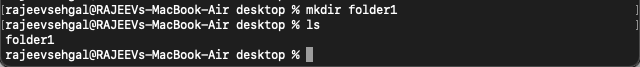
Description automatically generated with medium confidence

Text

Description automatically generated ls -i – used to display the inode number of the files.

1. mkdir: it is used to create a new directory/folder

Syntax: mkdir <directory\_name>



mkdir -p <directory1/directory2> - used to create hierarchy of directories

Text

Description automatically generated with medium confidence

rmdir – used to remove empty directory



1. mv: it is used to move or rename a file/directory

Syntax: mv <file\_name> <new\_directory>

Text

Description automatically generated

To rename a file

Text

Description automatically generated with medium confidence

To move directory

Text

Description automatically generated

mv -i – to avoid overwriting pre-existing file/directory

Text

Description automatically generated

1. cp: it is used to copy file or directory.

Syntax: cp <source\_file> <dest\_file>

Text

Description automatically generated

cp -r – used to copy directory

Text

Description automatically generated with medium confidence

cp -i – used to avoid overwriting pre-existing file/directory

Text

Description automatically generated

1. ln: it is used to make links (soft and hard)

Syntax: ln <file1> <file2> to create hard link

ln -s <file1> <file2> to create soft link

Graphical user interface, text

Description automatically generated

1. rm: it is used to remove a file or a directory

Syntax: rm <filename>

Text

Description automatically generated

rm -r – used to remove the directory even if it is not empty

A picture containing graphical user interface

Description automatically generated

**Result:** Successfully executed Linux file commands.

**Experiment – 3**

**Objective:** To explore Linux file commands.

**Software used:** zsh terminal

**Theory:**

1. cmp: It is used to compare the files byte by byte.

Syntax: cmp <option> <file1> <file2>

cmp -b: it also displays the character where the first mismatch occurred

cmp -i <num>: bytes to be skipped in both the files

cmp -n <num>: number of bytes to be compared

Text

Description automatically generated

1. sort: It is used to sort the file in alphabetical order only for display.

Syntax: sort <options> <filename>

sort -r: to sort in reverse alphabetical order

sort -n: to sort from lowest to highest number

sort -u: to remove duplicates and sort

sort -m: to sort on the basis of months

sort -f: to ignore case sensitivity and sort

Text

Description automatically generated

Text

Description automatically generated

Graphical user interface, text, application

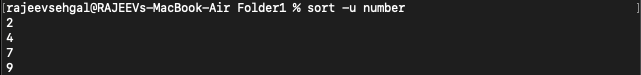
Description automatically generated

Text

Description automatically generated

Text

Description automatically generated



To sort and store: sort <src\_filename> > <dest\_filename>

Text

Description automatically generated

1. comm: it is used to find the unique and common words in different files

Syntax: comm <options> <file1> <file2>

comm -12 <file1> <file2>: will only show common words in both

files and remove the columns 1 and 2.

Text

Description automatically generated

1. wc: it is used to show the line, word and character count in the file.

Syntax: wc <options> <filename>

wc -l: to show only the line count

wc -w: to show only the word count

wc -m: to show only the character count

Text

Description automatically generated

1. vi: it is used to open the Visual editor. It is the default editor that comes with the

UNIX operating system. It has **three** modes:

* + 1. Command mode, where commands which take action on the file are written. Opens by default as we enter the vi.
    2. Insert mode, where the entered text is inserted into the file. Opens by using commands such as ‘i’, ‘a’, ‘A’, ‘o’, etc.
    3. Escape mode, from where the file can be saved and editor can be quitted. Opens by using the command ‘:’.

Syntax: vi <filename>



Shape

Description automatically generated with medium confidence

**Result:** Successfully executed Linux file commands.

**Experiment – 4**

**Objective:** To explore Linux file commands.

**Software used:** zsh terminal

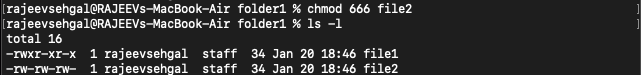
**Theory:**

1. chmod: It is used to change the access permission for the files or directories.

Syntax: chmod <class/es> <option> <permission/s> <file\_name/dir\_name>

chmod <numeric\_value> <file\_name/dir\_name>

Text

Description automatically generated

Text

Description automatically generated

1. ping: It is used to check network connectivity between host and server/host.

Syntax: ping <option> <IP/URL>

Text

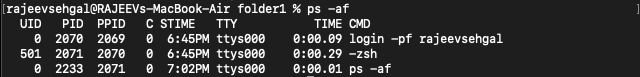
Description automatically generated

1. ps: it is used to display information related to the processes running in the system.

Syntax: ps <options>

Text

Description automatically generated



1. tty: it is used to display the file name of the terminal connected to standard input.

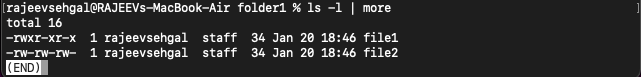
Syntax: wc <options>



1. Pipe: it is used to combine two or more commands, the output of one command acts

as the input to another command.

Syntax: <command> | <command> | <command>







1. grep – it is used to search a file for a particular pattern of characters and displays all

the lines that contain that pattern.

Syntax – grep <option> <keyword/pattern> <filename>

grep -i: ignores case sensitivity

grep -n: displays pattern along with line number

grep -v: displays lines not matching

grep -c: displays matching lines

grep -e: used to search for multiple patterns

Text

Description automatically generated



**Result:** Successfully executed Linux file commands.

**Experiment 5**

**Objective:** To write scripts for the given questions

**Software used:** zsh terminal

Q1. Write a script to swap 2 numbers.

**Code:**

Text

Description automatically generated

**Output:**

Text

Description automatically generated

Q2. Write a script to delete file.

**Code:**

Text

Description automatically generated

**Output:**

Text

Description automatically generated

Q3. Write script to add text to already existing file.

**Code:**

Text

Description automatically generated

**Output:**

Text

Description automatically generated

Q4. Write a script to demonstrate the use of arithmetic operators.

**Code:**

Text

Description automatically generated

**Output:**

Text

Description automatically generated

**Experiment 6**

**Objective:** To write scripts for the given questions

**Software used:** zsh terminal

Q1. Write a script to check whether number is positive or negative.

**Code:**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

Q2. Write a script to find greatest number among three numbers.

**Code:**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

Q3. Write a script to enter the marks of a student. If the marks are greater than 70 display grade A, if the grade is greater than 60 and less than 70 display grade B, else display “Fail”.

**Code:**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

Q4. Write a script to calculate factorial of a number.

**Code:**

**Text

Description automatically generated**

**Output:**

****

Q5. Write a script to display whether a user is valid or not.

**Code:**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Experiment 7**

**Objective:** To write scripts for the given questions

**Software used:** zsh terminal

Q1. Write a script to reverse number passed using positional parameter.

**Code:**

**Text

Description automatically generated**

**Output:**

****

Q2. Write a script to list all files in a directory using for loop.

**Code:**

**Text

Description automatically generated with medium confidence**

**Output:**

**Text

Description automatically generated**

Q3. Write a script to find largest value passed using command line

**Code:**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

Q4. Write a script to search particular file and rename it.

**Code:**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

Q5. Write a script to print Fibonacci series.

**Code:**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

Q6. Write a script to create and delete directory using case.

**Code:**

**Text

Description automatically generated**

**Output:**

**Text

Description automatically generated**

**Experiment 8**

**Objective:** To simulate FCFS CPU Scheduling algorithm.

**Code:**

#include <iostream>

using namespace std;

void storeInfo(int num, int burst[], int arrival[], int process[]){

cout<<"Enter the following information\n";

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

cout<<"Process "<<i+1<<":\n";

process[i]=i;

cout<<"Arrival time: ";

cin>>arrival[i];

cout<<"Burst time: ";

cin>>burst[i];

cout<<"-------------------------------------------"<<endl;

}

}

void dispInfo(int num, int burst[], int arrival[], int process[],int wait[]=NULL, int turnaround[]=NULL){

cout<<"Process\t\t|Arrival Time\t\t|Burst Time\t\t";

if(wait==NULL)

cout<<endl;

else

cout<<"|Wait Time\t\t|Turnaround Time\t\t\n";

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

cout<<process[i]+1<<"\t\t\t|"<<arrival[i]<<"\t\t\t\t\t|"<<burst[i]<<"\t\t\t\t";

if(wait==NULL){

cout<<endl;

continue;

}

else{

cout<<"|"<<wait[i]<<"\t\t\t\t|"<<turnaround[i]<<"\t\t\t\n";

}

}

cout<<"\n\n";

}

void waitTime(int num, int wait[], int burst[], int arrival[]){

int respTime[num];

respTime[0] = 0;

wait[0] = 0;

for (int i = 1; i < num ; i++)

{

respTime[i] = respTime[i-1] + burst[i-1];

if(respTime[i-1]<arrival[i-1])

respTime[i]+=(arrival[i-1]-respTime[i-1]);

wait[i] = respTime[i] - arrival[i];

if (wait[i] < 0)

wait[i] = 0;

}

}

void turnaroundTime(int num, int turnaround[], int wait[], int burst[]){

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

turnaround[i]=wait[i]+burst[i];

}

float avgTurnaround(int num, int turnaround[]){

float avg=0;

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

avg+=turnaround[i];

return avg/num;

}

float avgWait(int num, int wait[]){

float avg=0;

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

avg+=wait[i];

return avg/num;

}

void interchange(int arr[], int num1, int num2){

int temp;

temp=arr[num1];

arr[num1]=arr[num2];

arr[num2]=temp;

}

void sortInfo(int num, int process[], int arrival[], int burst[]){

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

for(int j=0; j<num-(i+1); j++){

if(arrival[j+1]<arrival[j]){

interchange(arrival,j+1,j);

interchange(process,j+1,j);

interchange(burst,j+1,j);

}

}

}

int main(){

int input;

cout<<"Enter the number of processes: ";

cin>>input;

int process[input], wait[input], burst[input], arrival[input], turnaround[input];

storeInfo(input, burst, arrival, process);

cout<<"--------------------------------------------------------------------------------------\n";

cout<<"Question\n";

cout<<"--------------------------------------------------------------------------------------\n\n";

dispInfo(input, burst, arrival, process);

sortInfo(input, process, arrival, burst);

waitTime(input, wait, burst, arrival);

turnaroundTime(input, turnaround, wait, burst);

cout<<"--------------------------------------------------------------------------------------\n";

cout<<"Answer\n";

cout<<"--------------------------------------------------------------------------------------\n\n";

dispInfo(input, burst, arrival, process, wait, turnaround);

cout<<"Average wait time = "<<avgWait(input, wait)<<endl;

cout<<"Average turnaround time = "<<avgTurnaround(input, turnaround)<<endl;

return 0;

}

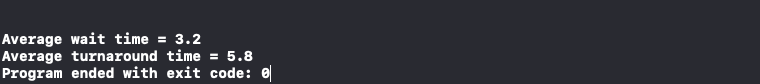
**Output:**

**Text

Description automatically generated**

**A picture containing text

Description automatically generated**

****

**Experiment 9**

**Objective:** To simulate preemptive and non-preemptive SJF CPU Scheduling algorithm.

**Code:**

#include <iostream>

using namespace std;

struct Process{

int wait, burst, turnaround, arrival;

void inputInfo(int i){

cout<<"Process "<<i<<":\n";

cout<<"Arrival time: ";

cin>>arrival;

cout<<"Burst time: ";

cin>>burst;

}

void outputInfo(int i){

cout<<i<<"\t\t\t|"<<arrival<<"\t\t\t\t\t|"<<burst<<"\t\t\t\t|"<<wait<<"\t\t\t\t|"<<turnaround<<"\t\t\t\n";

}

};

void waitTimeSJF(Process p[], int num){

int respTime[num];

respTime[0] = 0;

p[0].wait = 0;

for (int i = 1; i < num ; i++)

{

respTime[i] = respTime[i-1] + p[i-1].burst;

if(respTime[i-1]<p[i-1].arrival)

respTime[i]+=(p[i-1].arrival-respTime[i-1]);

p[i].wait = respTime[i] - p[i].arrival;

if (p[i].wait < 0)

p[i].wait = 0;

}

}

void waitTimeSRTF(Process p[], int num){

int remTime[num];

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

remTime[i] = p[i].burst;

int complete = 0, t = 0, minTime = INT\_MAX;

int indexShort = 0, finish\_time;

bool check = false;

while (complete != num) {

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

if ((p[i].arrival <= t) &&

(remTime[i] < minTime) && remTime[i] > 0) {

minTime = remTime[i];

indexShort = i;

check = true;

}

}

if (check == false) {

t++;

continue;

}

remTime[indexShort]--;

minTime = remTime[indexShort];

if (minTime == 0)

minTime = INT\_MAX;

if (remTime[indexShort] == 0) {

complete++;

check = false;

finish\_time = t + 1;

p[indexShort].wait = finish\_time -

p[indexShort].burst -

p[indexShort].arrival;

if (p[indexShort].wait < 0)

p[indexShort].wait = 0;

}

t++;

}

}

void turnaroundTime(Process p[], int num){

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

p[i].turnaround=p[i].wait+p[i].burst;

}

}

float avgTurnaround(Process p[], int num){

float avg=0;

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

avg+=p[i].turnaround;

return avg/num;

}

float avgWait(Process p[], int num){

float avg=0;

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

avg+=p[i].wait;

return avg/num;

}

void sortInfo(Process p[], int num){

Process temp;

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

for(int j=0; j<num-(i+1); j++){

if(p[j+1].arrival<p[j].arrival){

temp = p[j+1];

p[j+1]=p[j];

p[j]=temp;

}

else if(p[j+1].burst<p[j].burst){

temp = p[j+1];

p[j+1]=p[j];

p[j]=temp;

}

}

}

void calculateSJF(Process p[], int num){

waitTimeSJF(p, num);

turnaroundTime(p, num);

}

void calculateSRTF(Process p[], int num){

waitTimeSRTF(p, num);

turnaroundTime(p, num);

}

void calcAvg(Process p[], int num){

cout<<"Average wait time = "<<avgWait(p,num)<<endl;

cout<<"Average turnaround time = "<<avgTurnaround(p,num)<<endl;

}

int main() {

int num,sel;

cout<<"Enter the number of processes: ";

cin>>num;

Process process[num];

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

process[i].inputInfo(i+1);

sortInfo(process, num);

cout<<"Shortest Job First Scheduling\n";

cout<<"1. Non-preemptive\n";

cout<<"2. Preemptive\n";

cin>>sel;

switch(sel){

case 1:

calculateSJF(process, num);

cout<<"Process\t\t|Arrival Time\t\t|Burst Time\t\t|Wait Time\t\t|Turnaround Time\t\t\n";

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

process[i].outputInfo(i+1);

calcAvg(process, num);

break;

case 2:

calculateSRTF(process, num);

cout<<"Process\t\t|Arrival Time\t\t|Burst Time\t\t|Wait Time\t\t|Turnaround Time\t\t\n";

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

process[i].outputInfo(i+1);

calcAvg(process, num);

break;

default: cout<<"Enter a valid option.\n";

}

return 0;

}

**Output:**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Text

Description automatically generated**

**Experiment 10**

**Objective:** To simulate Round Robin CPU Scheduling algorithm.

**Code:**

#include <iostream>

#include <algorithm>

#include <queue>

using namespace std;

struct Process

{

int PID;

int AT;

int BT;

int FT;

int TAT;

int WT;

int ST;

int TR;

bool inWaitQueue;

bool completed;

bool hasStarted;

};

bool compareAT(Process a, Process b)

{

return a.AT < b.AT;

}

class RoundRobin

{

private:

struct Process \*q;

bool wasIdle;

int n;

int processCompleted;

int currentTime;

int timeSlice;

queue<Process \*> waitQueue;

public:

RoundRobin(int n)

{

q = new Process[n];

wasIdle = true;

this->n = n;

}

~RoundRobin()

{

delete[] q;

}

void addProcessData();

void display();

void calculateAverageTimes();

void updateWaitQueue();

void schedule();

};

void RoundRobin::addProcessData()

{

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

{

cout << "Process " << i + 1 << ": " << endl;

q[i].PID = i + 1;

cout << "Enter arrival time: ";

cin >> q[i].AT;

cout << "Enter the burst time: ";

cin >> q[i].BT;

q[i].inWaitQueue = false;

q[i].completed = false;

q[i].hasStarted = false;

q[i].TR = q[i].BT;

}

cout << "Enter time slice: ";

cin >> timeSlice;

sort(q, q + n, compareAT);

this->currentTime = q[0].AT;

processCompleted = 0;

}

void RoundRobin::display()

{

cout << "PID\t\t AT\t\t BT\t\t FT\t\t TAT\t WT\t\t\n";

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

{

cout << q[i].PID << "\t\t " << q[i].AT << "\t\t " << q[i].BT << "\t\t " << q[i].FT << "\t\t " << q[i].TAT << "\t\t " << q[i].WT << "\t\t ";

cout << endl;

}

}

void RoundRobin::calculateAverageTimes()

{

float averageWT = 0, averageTAT = 0;

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

{

averageWT += q[i].WT;

averageTAT += q[i].TAT;

}

cout << "Average Turn around Time: " << (averageTAT / n) << endl;

cout << "Average Wait Time: " << (averageWT / n) << endl;

}

void RoundRobin::updateWaitQueue()

{

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

{

if ((!q[i].inWaitQueue) && (!q[i].completed) && (q[i].AT <= currentTime))

{

waitQueue.push(&q[i]);

q[i].inWaitQueue = true;

}

}

}

void RoundRobin::schedule()

{

updateWaitQueue();

while (processCompleted < n)

{

wasIdle = true;

if (!waitQueue.empty())

{

wasIdle = false;

Process \*temp = waitQueue.front();

waitQueue.pop();

if (currentTime >= temp->AT)

{

if (temp->TR > timeSlice)

{

if (!(temp->hasStarted))

{

temp->ST = currentTime;

temp->hasStarted = true;

}

temp->TR = temp->TR - timeSlice;

currentTime += timeSlice;

}

else

{

if (!(temp->hasStarted))

{

temp->ST = currentTime;

temp->hasStarted = true;

}

currentTime += temp->TR;

temp->TR = 0;

temp->completed = true;

temp->FT = currentTime;

temp->TAT = temp->FT - temp->AT; // Turn around time

temp->WT = temp->TAT - temp->BT; // Wait time

processCompleted++;

}

}

updateWaitQueue();

if (!(temp->completed))

{

waitQueue.push(temp);

}

}

else if (wasIdle)

{

currentTime = q[processCompleted].AT;

updateWaitQueue();

}

}

}

int main()

{

int N;

cout << "Enter the number of processes: ";

cin >> N;

RoundRobin event(N);

event.addProcessData();

event.schedule();

event.display();

event.calculateAverageTimes();

return 0;

}

Output:

Text

Description automatically generated

**Experiment 11**

**Objective:** To simulate Banker’s algorithm.

**Code:**

#include <iostream>

#include <stdlib.h>

using namespace std;

void print(int x[][10], int n, int m)

{

int i, j;

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

{

cout << "\n";

for (j = 0; j < m; j++)

{

cout << x[i][j] << "\t";

}

}

}

void res\_request(int A[10][10], int N[10][10], int AV[10][10], int pid, int m)

{

int reqmat[1][10];

int i;

cout << "\n Enter additional request: \n";

for (i = 0; i < m; i++)

{

cout << "Request for resource " << i + 1 << ": ";

cin >> reqmat[0][i];

}

for (i = 0; i < m; i++)

if (reqmat[0][i] > N[pid][i])

{

cout << "\n Error encountered.\n";

exit(0);

}

for (i = 0; i < m; i++)

if (reqmat[0][i] > AV[0][i])

{

cout << "\n Resources unavailable.\n";

exit(0);

}

for (i = 0; i < m; i++)

{

AV[0][i] -= reqmat[0][i];

A[pid][i] += reqmat[0][i];

N[pid][i] -= reqmat[0][i];

}

}

int safety(int A[][10], int N[][10], int AV[1][10], int n, int m, int a[])

{

int i, j, k, x = 0;

int F[10], W[1][10];

int pflag = 0, flag = 0;

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

F[i] = 0;

for (i = 0; i < m; i++)

W[0][i] = AV[0][i];

for (k = 0; k < n; k++)

{

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

{

if (F[i] == 0)

{

flag = 0;

for (j = 0; j < m; j++)

{

if (N[i][j] > W[0][j])

flag = 1;

}

if (flag == 0 && F[i] == 0)

{

for (j = 0; j < m; j++)

W[0][j] += A[i][j];

F[i] = 1;

pflag++;

a[x++] = i;

}

}

}

if (pflag == n)

return 1;

}

return 0;

}

void accept(int A[][10], int N[][10], int M[10][10], int W[1][10], int \*n, int \*m)

{

int i, j;

cout << "\n Enter total no. of processes: ";

cin >> \*n;

cout << "\n Enter total no. of resources: ";

cin >> \*m;

for (i = 0; i < \*n; i++)

{

cout << "\n Process " << i + 1 << " :\n";

for (j = 0; j < \*m; j++)

{

cout << " Allocation for resource " << j + 1 << ": ";

cin >> A[i][j];

cout << " Maximum for resource " << j + 1 << ": ";

cin >> M[i][j];

}

}

cout << "\n Available resources: \n";

for (i = 0; i < \*m; i++)

{

cout << " Resource " << i + 1 << ": ";

cin >> W[0][i];

}

for (i = 0; i < \*n; i++)

for (j = 0; j < \*m; j++)

N[i][j] = M[i][j] - A[i][j];

cout << "\n Allocation Matrix";

print(A, \*n, \*m);

cout << "\n Maximum Requirement Matrix";

print(M, \*n, \*m);

cout << "\n Need Matrix";

print(N, \*n, \*m);

}

int banker(int A[][10], int N[][10], int W[1][10], int n, int m)

{

int j, i, a[10];

j = safety(A, N, W, n, m, a);

if (j != 0)

{

cout << "\n\n";

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

cout << "P " << a[i];

cout << "\nA safety sequence has been detected.\n";

return 1;

}

else

{

cout << "\nDeadlock has occured.\n";

return 0;

}

}

int main()

{

int ret;

int A[10][10];

int M[10][10];

int N[10][10];

int W[1][10];

int n, m, pid, ch;

cout << "\n DEADLOCK AVOIDANCE USING BANKER'S ALGORITHM\n";

accept(A, N, M, W, &n, &m);

ret = banker(A, N, W, n, m);

if (ret != 0)

{

cout << "\n Do you want make an additional request ? (1=Yes|0=No)";

cin >> ch;

if (ch == 1)

{

cout << "\n Enter process no. : ";

cin >> pid;

res\_request(A, N, W, pid - 1, m);

ret = banker(A, N, W, n, m);

if (ret == 0)

exit(0);

}

}

else

exit(0);

return 0;

}

**Output:**

**Graphical user interface

Description automatically generated with medium confidence**

**Text

Description automatically generated**

**Experiment 12**

**Objective:** To simulate Shortest Serve Time First disk scheduling algorithm.

**Code:**

#include <iostream>

#include <stdlib.h>

using namespace std;

class sstf\_disk

{

int ref[100];

int pos, size;

int find\_short(int num)

{

int min = INT\_MAX, ind = 0, temp;

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

{

if (ref[i] != -1)

{

temp = abs(num - ref[i]);

if (min > temp)

{

min = temp;

ind = i;

}

}

}

return ind;

};

public:

void getdata()

{

cout << "Enter the current position of head: ";

cin >> pos;

cout << "Enter the size of queue: ";

cin >> size;

cout << "Enter the request for tracks:\n";

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

cin >> ref[i];

};

void total\_move()

{

int num = pos, move = 0, ind;

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

{

ind = find\_short(num);

move += abs(num - ref[ind]);

num = ref[ind];

ref[ind] = -1;

}

cout << "Total head movements: " << move << "\n";

};

};

int main()

{

sstf\_disk sstf;

sstf.getdata();

sstf.total\_move();

return 0;

}

#include <iostream>

#include <stdlib.h>

using namespace std;

class sstf\_disk

{

int ref[100];

int pos, size;

int find\_short(int num)

{

int min = INT\_MAX, ind = 0, temp;

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

{

if (ref[i] != -1)

{

temp = abs(num - ref[i]);

if (min > temp)

{

min = temp;

ind = i;

}

}

}

return ind;

};

public:

void getdata()

{

cout << "Enter the current position of head: ";

cin >> pos;

cout << "Enter the size of queue: ";

cin >> size;

cout << "Enter the request for tracks:\n";

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

cin >> ref[i];

};

void total\_move()

{

int num = pos, move = 0, ind;

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

{

ind = find\_short(num);

move += abs(num - ref[ind]);

num = ref[ind];

ref[ind] = -1;

}

cout << "Total head movements: " << move << "\n";

};

};

int main()

{

sstf\_disk sstf;

sstf.getdata();

sstf.total\_move();

return 0;

}

**Output:**

**Text

Description automatically generated**