Practical 1

Aim: To study the types of operating systems.

Operating System lies in the category of system software. It basically manages all the resources of the computer. An operating system acts as an interface between the software and different parts of the computer or the computer hardware. The operating system is designed in such a way that it can manage the overall resources and operations of the computer.

TYPES OF OPERATING SYSTEMS:

*1****. Batch Operating System***

This type of operating system does not interact with the computer directly. The operator takes similar jobs having the same requirement and groups them into batches.

**Advantages of Batch Operating System**

* Multiple users can share the batch systems.
* The idle time for the batch system is very less.
* It is easy to manage large work repeatedly in batch systems.

**Disadvantages of Batch Operating System**

* The computer operators should be well known with batch systems.
* Batch systems are hard to debug.
* It is sometimes costly.
* The other jobs will have to wait for an unknown time if any job fails.

*2.* ***Multi-Programming System***

It has more than one program that is present in the main memory and any one of them can be kept in execution. It is used for better execution of resources.

**Advantages of Multi-Programming Operating System**

* Multi Programming increases the throughput of the System.
* It helps in reducing the response time.

**Disadvantages of Multi-Programming Operating System**.

* There is not any facility for user interaction of system resources with the system.

*3****. Time-Sharing Operating System***

Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of the CPU as they use a single system.

**Advantages of Time-Sharing OS**

* Each task gets an equal opportunity.
* Fewer chances of duplication of software.
* CPU idle time can be reduced.

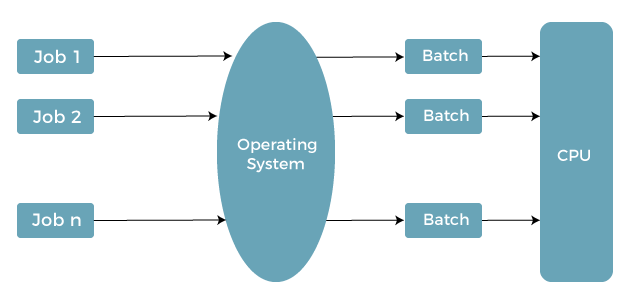


Fig: Batch Operating System

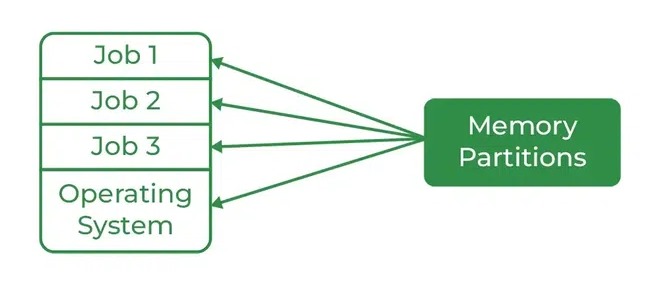


Fig: Multiprogramming Operating System

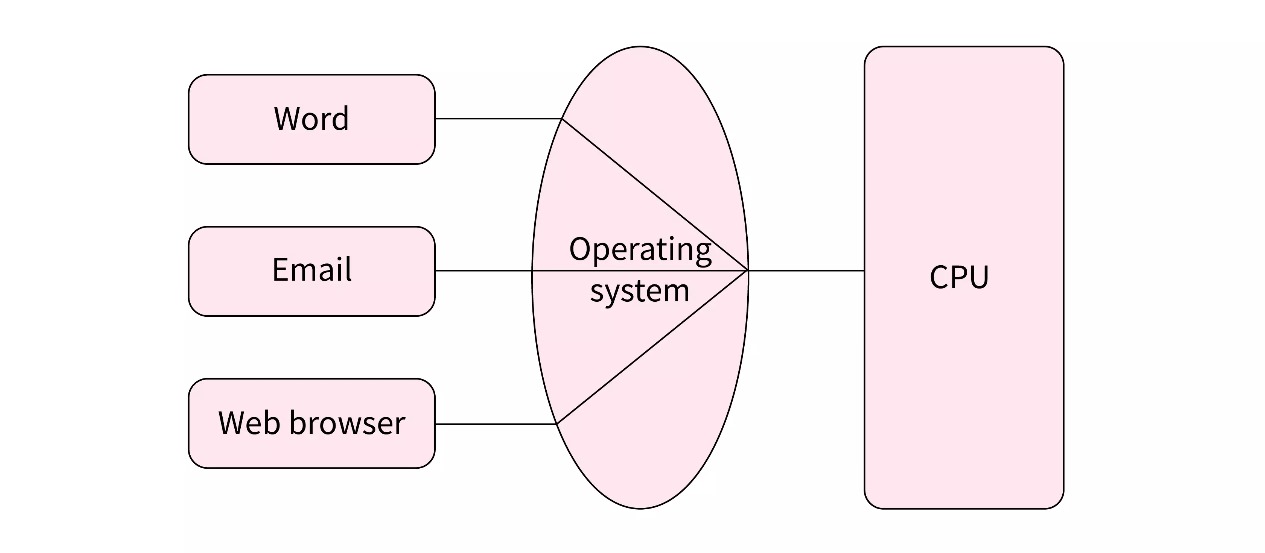


Fig: Time Sharing Operating System

Resource Sharing: Time-sharing systems allow multiple users to share hardware resources such as the CPU, memory, and peripherals, reducing the cost of hardware and increasing efficiency.

* Improved Productivity: Time-sharing allows users to work concurrently, thereby reducing the waiting time for their turn to use the computer. This increased productivity translates to more work getting done in less time.
* Improved User Experience: Time-sharing provides an interactive environment that allows users to communicate with the computer in real time, providing a better user experience than batch processing.

**Disadvantages of Time-Sharing OS**

* Reliability problem.
* One must have to take care of the security and integrity of user programs and data.
* Data communication problem.
* High Overhead: Time-sharing systems have a higher overhead than other operating systems due to the need for scheduling, context switching, and other overheads that come with supporting multiple users.
* Complexity: Time-sharing systems are complex and require advanced software to manage multiple users simultaneously. This complexity increases the chance of bugs and errors.
* Security Risks: With multiple users sharing resources, the risk of security breaches increases. Time-sharing systems require careful management of user access, authentication, and authorization to ensure the security of data and software.

*4****. Distributed Operating System***

Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU. These are referred to as [loosely coupled systems or distributed systems](https://www.geeksforgeeks.org/difference-between-loosely-coupled-and-tightly-coupled-multiprocessor-system/). The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.

**Advantages of Distributed Operating System**

* Failure of one will not affect the other network communication, as all systems are independent of each other.
* Electronic mail increases the data exchange speed.
* Since resources are being shared, computation is highly fast and durable.
* Load on host computer reduces.
* These systems are easily scalable as many systems can be easily added to the network.
* Delay in data processing reduces.

**Disadvantages of Distributed Operating System**

* Failure of the main network will stop the entire communication.
* To establish distributed systems the language is used not well-defined yet.

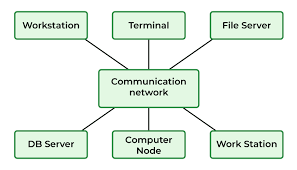


Fig: Distributed Operating System

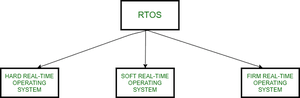


Fig: Real-time Operating System

* These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet.

*5****. Real-time Operating System***

*These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called* ***response time****.****Real-time systems*** *are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.*These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.   
**Real-time systems** are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc. These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.   
**Real-time systems** are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

**Advantages of RTOS**

* **Maximum Consumption:** Maximum utilization of devices and systems, thus more output from all the resources.
* **Task Shifting:** The time assigned for shifting tasks in these systems is very less. For example, in older systems, it takes about 10 microseconds in shifting from one task to another, and in the latest systems, it takes 3 microseconds.
* **Focus on Application:** Focus on running applications and less importance on applications that are in the queue.
* Real-time **operating system in** the **embedded system:** Since the size of programs is small, RTOS can also be used in embedded systems like in transport and others.
* **Error Free:** These types of systems are error-free.
* **Memory Allocation:** Memory allocation is best managed in these types of systems.

**Disadvantages of RTOS**

* **Limited Tasks:** Very few tasks run at the same time and their concentration is very less on a few applications to avoid errors.
* **Use heavy system resources:** Sometimes the system resources are not so good and they are expensive as well.
* **Complex Algorithms:** The algorithms are very complex and difficult for the designer to write on.
* **Device driver and interrupt signals:** It needs specific device drivers and interrupts signal to respond earliest to interrupts.
* **Thread Priority:** It is not good to set thread priority as these systems are very less prone to switching tasks.

Result: Hence, we’ve successfully studied the types of operating systems, their advantages and disadvantages.

Practical 2

Aim: Study of various unix commands and editors on unix terminal.

*date: it is used to display the system date and time. date command is also used to set date and time of the system*.

guest-hazqk6@admin6:~$ date

Wed Sep 6 11:38:06 IST 2023

*cal:* *it is a calendar command in Linux which is used to see the calendar of a specific month or a whole year.*

guest-hazqk6@admin6:~$ cal

September 2023

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

*ls: it lists directory contents of files and directories. It provides valuable information about files, directories, and their attributes.*

guest-hazqk6@admin6:~$ man ls

ls - list directory contents

*lp: arranges for the files specified by the Files parameter and their associated information (called a request) to be printed by a line printer.*

guest-hazqk6@admin6:~$ man lp

lp - print files

*who: The who command is used to get information about currently logged in user on to system.*

guest-hazqk6@admin6:~$ who

guest-hazqk6 tty7 2023-09-06 11:37 (:0)

*whoami: it displays the username of the current user when this command is invoked.*

guest-hazqk6@admin6:~$ whoami

guest-hazqk6

*uname: it is a command line tool most commonly used to determine the processor architecture, the system hostname and the version of the kernel running on the system.*

guest-hazqk6@admin6:~$ uname

Linux

*hostname: hostname command in Linux is used to obtain the DNS (Domain Name System) name and set the system’s hostname or NIS (Network Information System) domain name.*

guest-hazqk6@admin6:~$ hostname

admin6

*cat: it reads data from the file and gives its content as output. It helps us to create, view, and concatenate files. So let us see some frequently used cat commands.*

guest-hazqk6@admin6:~$ man cat

cat - concatenate files and print on the standard output

*grep: The grep filter searches a file for a particular pattern of characters, and displays all lines that contain that pattern. The pattern that is searched in the file is referred to as the regular expression (grep stands for global search for regular expression and print out).*

guest-hazqk6@admin6:~$ man grep

grep, egrep, fgrep, rgrep - print lines matching a pattern

*cp: cp stands for a copy. This command is used to copy files or groups of files or directories. It creates an exact image of a file on a disk with a different file name. cp command requires at least two filenames in its arguments.*

guest-hazqk6@admin6:~$ man cp

cp - copy files and directories

*mv: mv command is used to move existing file or directory from one location to another. It is also used to rename a file or directory.*

guest-hazqk6@admin6:~$ man mv

mv - move (rename) files

*cut: The cut command in UNIX is a command for cutting out the sections from each line of files and writing the result to standard output. It can be used to cut parts of a line by byte position, character and field.*

guest-hazqk6@admin6:~$ man cut

cut - remove sections from each line of files

*head: it is a command-line utility, which prints the first 10 lines of the specified files. If more than one file name is provided then data from each file is preceded by its file name.*

guest-hazqk6@admin6:~$ man head

head - output the first part of files

*tail: It is the complementary of head command. It prints the last N number of data of the given input. By default it prints the last 10 lines of the specified files. If more than one file name is provided then data from each file is precedes by its file name.*

guest-hazqk6@admin6:~$ man tail

tail - output the last part of files

*chmod: it is an abbreviation of change mode which states that every file and directory has a set of permissions that control the permissions like who can read, write or execute the file. In this the permissions have three categories: read, write, and execute*

guest-hazqk6@admin6:~$ man chmod

chmod - change file mode bits

*wc: It is used to find out number of lines, word count, byte and characters count in the files specified in the file arguments.*

guest-hazqk6@admin6:~$ man wc

wc - print newline, word, and byte counts for each file

*mkdir: it allows the user to create directories (also referred to as folders in some operating systems). This command can create multiple directories at once as well as set the permissions for the directories.*

guest-hazqk6@admin6:~$ man mkdir

mkdir - make directories

*editor: it is a simple, display-oriented text editor. Commands and their Control key shortcuts are displayed at the bottom of the screen. As characters are typed, they are immediately inserted into the text.*

guest-hazqk6@admin6:~$ man editor

nano - Nano's ANOther editor, an enhanced free Pico clone

*vi: it is default editor that comes with the UNIX operating system is called vi (visual editor). Using vi editor, we can edit an existing file or create a new file from scratch. we can also use this editor to just read a text file.*

guest-hazqk6@admin6:~$ man vi

vim - Vi IMproved, a programmers text editor

Result : Hence we studied about different unix CLI commands and CLI text editors

Practical 3

import java.util.Scanner;

import java.util.Arrays;

public class FCFS {

public static void calculateAverage(int[] array, int size) {

int sum = 0;

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

sum += array[i];

}

int average = sum / size;

System.out.println("Average is " + average);

}

public static void bubbleSort(int size, int[] arrivalTime, int[] burstTime) {

int temp;

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

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

if (arrivalTime[j - 1] > arrivalTime[j]) {

// Swap arrival time

temp = arrivalTime[j - 1];

arrivalTime[j - 1] = arrivalTime[j];

arrivalTime[j] = temp;

// Swap burst time

temp = burstTime[j - 1];

burstTime[j - 1] = burstTime[j];

burstTime[j] = temp;

}

}

}

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter number of processes");

int numProcesses = sc.nextInt();

int[] burstTime = new int[numProcesses];

int[] arrivalTime = new int[numProcesses];

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

System.out.println("Process no." + (i + 1));

System.out.println("Enter Burst time");

burstTime[i] = sc.nextInt();

System.out.println("Enter Arrival time");

arrivalTime[i] = sc.nextInt();

}

bubbleSort(numProcesses, arrivalTime, burstTime);

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

System.out.println("Process Arrived First " + (i + 1) + " Burst time:" + burstTime[i] +

" Arrival time:" + arrivalTime[i]);

}

int[] turnaroundTime = new int[numProcesses];

int[] waitingTime = new int[numProcesses];

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

if (i == 0) {

turnaroundTime[i] = burstTime[i] - arrivalTime[i];

} else {

turnaroundTime[i] = burstTime[i] - arrivalTime[i] + turnaroundTime[i - 1];

}

System.out.println("Turn around time of " + (i + 1) + " is " + turnaroundTime[i]);

}

calculateAverage(turnaroundTime, numProcesses);

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

if (i == 0) {

waitingTime[i] = arrivalTime[i];

} else {

waitingTime[i] = turnaroundTime[i] - burstTime[i];

}

System.out.println("Waiting time of " + (i + 1) + " is " + waitingTime[i]);

}

calculateAverage(waitingTime, numProcesses);

sc.close();

}

}

Practical 4

public class SJF {

public static void Sahil(float[] a, float[] b, int i) {

float[] CT = new float[i];

float sum = 0, avg;

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

if (roll == 0) {

CT[roll] = b[roll] + a[roll];

} else {

CT[roll] = b[roll] + CT[roll - 1];

}

}

float[] TAT = new float[i];

float[] WT = new float[i];

sum = 0;

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

TAT[roll] = CT[roll] - a[roll];

System.out.println("Process no " + (roll + 1) + " Turn around time is " + TAT[roll]);

sum += TAT[roll];

}

avg = sum / i;

System.out.println("Average Turn around time is " + avg);

avg = sum = 0;

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

WT[roll] = TAT[roll] - b[roll];

System.out.println("Process no " + (roll + 1) + " Waiting time is " + WT[roll]);

sum += WT[roll];

}

avg = sum / i;

System.out.println("Average Waiting time is " + avg);

}

public static float Shubhankar(float[] a) {

float i = a[0];

for (int roll = 0; roll < a.length; roll++) {

if (i > a[roll]) {

i = a[roll];

}

}

return i;

}

public static void main(String[] args) {

float bt[] = { 7, 3, 2, 10, 8 };

float at[] = { 1, 3, 6, 7, 8 };

float[] tbt = new float[5];

float[] tat = new float[10];

float[] wt = new float[10];

int limit = 5, roll = 0, i = 0;

float pointer = 999;

float timer = Shubhankar(at);

float micro = timer;

while (roll != limit) {

pointer = 999;

for (int dice = 0; dice < limit; dice++) {

if (at[dice] <= timer && bt[dice] > 0 && pointer > bt[dice]) {

pointer = bt[dice];

i = dice;

}

}

wt[roll] = at[i];

timer += bt[i];

tat[roll] = timer;

bt[i] = 0;

if (roll == 0) {

tbt[roll] = timer - micro;

} else {

tbt[roll] = timer - tat[roll - 1];

}

roll++;

}

System.out.println("Order of process execution\nBurst time Arrival time ");

for (roll = 0; roll < limit; roll++) {

System.out.println(tbt[roll] + " " + wt[roll]);

}

Sahil(wt, tbt, limit);

}

}

Practical 5

import java.util.LinkedList;

import java.util.Queue;

public class RoundRobin {

public static void main(String[] args) {

int[] processes = {1, 2, 3, 4};

int[] burstTime = {10, 5, 8, 12};

int timeQuantum = 2;

roundRobin(processes, burstTime, timeQuantum);

}

public static void roundRobin(int[] processes, int[] burstTime, int timeQuantum) {

int n = processes.length;

int[] remainingTime = new int[n];

int[] waitingTime = new int[n];

int[] turnaroundTime = new int[n];

int totalTime = 0;

Queue<Integer> queue = new LinkedList<>();

// Initialize remaining time for each process

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

remainingTime[i] = burstTime[i];

totalTime += burstTime[i];

queue.add(i);

}

// Process the queue in a round-robin manner

int currentTime = 0;

while (!queue.isEmpty()) {

int currentProcess = queue.poll();

if (remainingTime[currentProcess] <= timeQuantum) {

// Process completes within time quantum

currentTime += remainingTime[currentProcess];

turnaroundTime[currentProcess] = currentTime;

waitingTime[currentProcess] = currentTime - burstTime[currentProcess];

remainingTime[currentProcess] = 0;

} else {

// Process needs more time, reduce remaining time and enqueue again

currentTime += timeQuantum;

remainingTime[currentProcess] -= timeQuantum;

queue.add(currentProcess);

}

}

// Calculate and display turnaround time and waiting time

float totalWaitingTime = 0;

float totalTurnaroundTime = 0;

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

totalWaitingTime += waitingTime[i];

totalTurnaroundTime += turnaroundTime[i];

System.out.println("Process " + processes[i] +

" Turnaround Time: " + turnaroundTime[i] +

" Waiting Time: " + waitingTime[i]);

}

// Display average turnaround time and waiting time

System.out.println("Average Turnaround Time: " + totalTurnaroundTime / n);

System.out.println("Average Waiting Time: " + totalWaitingTime / n);

}

}

Practical 6

Fork System Call

Code:-

include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

int main()

{

fork();

printf("Hello world!, process\_id(pid) = %d \n",getpid());

return 0;

}

Output:-

75@laptop:~/Desktop$ gcc Fork.c

75@laptop:~/Desktop$ ./a.out

Hello world!, process\_id(pid) = 4466

Hello world!, process\_id(pid) = 4467

Kill System Call

Code:-

#include <stdio.h>

#include <sys/types.h>

#include <unistd.h>

#include <time.h>

int main()

{

fork();

fork();

printf("Process %d\n",getpid());

sleep(60);

return 0;

}

Execution:-

shubhankar75@laptop:~/Desktop$ ./a.out

Process 28834

Process 28835

Process 28837

Process 28836

List of process running (ps -ef):-

shubhan+ 28834 21046 0 22:49 pts/3 00:00:00 ./a.out

shubhan+ 28835 28834 0 22:49 pts/3 00:00:00 ./a.out

shubhan+ 28836 28834 0 22:49 pts/3 00:00:00 ./a.out

shubhan+ 28837 28835 0 22:49 pts/3 00:00:00 ./a.out

Using Kill Command :-

kill 28835

List of Process after kill command (ps -ef) :-

shubhan+ 28834 21046 0 22:49 pts/3 00:00:00 ./a.out

shubhan+ 28835 28834 0 22:49 pts/3 00:00:00 [a.out] <defunct> //Process Killed

shubhan+ 28836 28834 0 22:49 pts/3 00:00:00 ./a.out

shubhan+ 28837 1291 0 22:49 pts/3 00:00:00 ./a.out

The Second Process is kill here

Practical 7

1.Code:- First Fit

#include<stdio.h>

int main()

{ int j;

printf("Enter the number of blocks:");

scanf("%d",&j);

int Bs[j];

for(int roll=0;roll<j;roll++){

printf("Enter Size of Block %d:",roll+1);

scanf("%d",&Bs[roll]);

}

int i;

printf("Enter number of process:");

scanf("%d",&i);

int Ps[i];

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

printf("Enter Size of Process %d:",roll+1);

scanf("%d",&Ps[roll]);

}

int allocation[j],flag[j];

for(int roll=0;roll<j;roll++){

allocation[roll]=-1;

flag[roll]=0;

}

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

for(int spin=0;spin<j;spin++){

if(flag[spin]==0&&Bs[spin]>=Ps[roll]){

flag[spin]=1;

allocation[spin]=roll;

break;

}

}

}

printf("\nBlock no.\tsize\t\tprocess no.\t\tsize");

for(int roll=0;roll<j;roll++){

printf("\n%d\t\t%d\t\t", roll+1, Bs[roll]);

if(flag[roll] == 1)

printf("%d\t\t\t%d",allocation[roll]+1,Ps[allocation[roll]]);

else

printf("Not allocated");

}

}

Output:-

Enter the number of blocks:3

Enter Size of Block 1:12

Enter Size of Block 2:7

Enter Size of Block 3:4

Enter number of process:3

Enter Size of Process 1:7

Enter Size of Process 2:4

Enter Size of Process 3:9

Block no. size process no. size

1 12 1 7

2 7 2 4

3 4 Not allocated

2.Code:- Best Fit

#include <stdio.h>

int main()

{

int j;

printf("Enter the number of blocks:");

scanf("%d",&j);

int Bs[j];

for(int roll=0;roll<j;roll++){

printf("Enter Size of Block %d:",roll+1);

scanf("%d",&Bs[roll]);

}

int i;

printf("Enter number of process:");

scanf("%d",&i);

int Ps[i];

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

printf("Enter Size of Process %d:",roll+1);

scanf("%d",&Ps[roll]);

}

int allocation[j],Fragment[i],flag[j],temp,mark;

for(int roll=0;roll<j;roll++){

allocation[roll]=-1;

flag[roll]=0;

}

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

temp=9999;

for(int spin=0;spin<j;spin++){

if(flag[spin]==0&&Bs[spin]>=Ps[roll]){

Fragment[roll]=Bs[spin]-Ps[roll];

}else{

continue;

}

if(Fragment[roll]<temp){

temp=Fragment[roll];

allocation[spin]=roll;

mark=spin;

}

}

flag[mark]=1;

}

printf("No. Block\_Size Process Process\_Size\n");

for(int roll=0;roll<j;roll++){

printf("%d %d %d %d ",roll+1,Bs[roll],allocation[roll]+1,Ps[allocation[roll]]);

printf("\n");

}

return 0;

}

Output:-

Enter the number of blocks:3

Enter Size of Block 1:12

Enter Size of Block 2:7

Enter Size of Block 3:4

Enter number of process:3

Enter Size of Process 1:7

Enter Size of Process 2:4

Enter Size of Process 3:9

No. Block\_Size Process Process\_Size

1 12 3 9

2 7 1 7

3 4 2 4

3.Code:- Worst Fit

#include <stdio.h>

int main()

{

int j;

printf("Enter the number of blocks:");

scanf("%d",&j);

int Bs[j];

for(int roll=0;roll<j;roll++){

printf("Enter Size of Block %d:",roll+1);

scanf("%d",&Bs[roll]);

}

int i;

printf("Enter number of process:");

scanf("%d",&i);

int Ps[i];

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

printf("Enter Size of Process %d:",roll+1);

scanf("%d",&Ps[roll]);

}

int allocation[i],flag[i];

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

allocation[roll]=-1;

flag[roll]=0;

}

int Frag[j],temp,mark;

for(int roll=0;roll<j;roll++){

temp=-1;

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

if(flag[spin]==0&&Bs[roll]>=Ps[spin]){

Frag[roll]=Bs[roll]-Ps[spin];

}else{ continue; }

if(Frag[roll]>temp){

temp=Frag[roll];

allocation[roll]=spin;

mark=spin; }

}

flag[mark]=1;

}

mark=(j>=i)?j:i;

printf("No. Block Size Process Process size\n");

int spin=0;

for(int roll=0;roll<j||spin<i;roll++){

if(allocation[spin]==-1){

printf("%d %d Not Allocated",roll+1,Bs[roll]);

}else

printf("%d %d %d %d \n",roll+1,Bs[roll],allocation[spin]+1,Ps[allocation[spin]]);

spin++;

} return 0; }

Output:-

Enter the number of blocks:3

Enter Size of Block 1:12

Enter Size of Block 2:7

Enter Size of Block 3:4

Enter number of process:3

Enter Size of Process 1:7

Enter Size of Process 2:4

Enter Size of Process 3:9

No. Block Size Process Process size

1 12 2 4

2 7 1 7

3 4 Not Allocated

Practical 8

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class FIFOCache {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int frames, pointer = 0, hit = 0, fault = 0, ref\_len;

int buffer[];

int reference[];

int mem\_layout[][];

System.out.println("Please enter the number of Frames: ");

frames = Integer.parseInt(br.readLine());

System.out.println("Please enter the length of the Reference string: ");

ref\_len = Integer.parseInt(br.readLine());

reference = new int[ref\_len];

mem\_layout = new int[ref\_len][frames];

buffer = new int[frames];

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

buffer[j] = -1;

System.out.println("Please enter the reference string: ");

for (int i = 0; i < ref\_len; i++) {

reference[i] = Integer.parseInt(br.readLine());

}

System.out.println();

for (int i = 0; i < ref\_len; i++) {

int search = -1;

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

if (buffer[j] == reference[i]) {

search = j;

hit++;

break;

}

}

if (search == -1) {

buffer[pointer] = reference[i];

fault++;

pointer++;

if (pointer == frames)

pointer = 0;

}

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

mem\_layout[i][j] = buffer[j];

}

}

System.out.println("Order of Page Execution:");

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

for (int j = 0; j < ref\_len; j++) {

System.out.printf("%3d ", mem\_layout[j][i]);

}

System.out.println();

}

System.out.println("The number of Hits: " + hit);

System.out.println("Hit Ratio: " + (float) ((float) hit / ref\_len));

System.out.println("The number of Faults: " + fault);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.util.ArrayList;

public class LRUCache {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int frames, pointer = 0, hit = 0, fault = 0, ref\_len;

boolean isFull = false;

int buffer[];

ArrayList<Integer> stack = new ArrayList<Integer>();

int reference[];

int mem\_layout[][];

System.out.println("Please enter the number of Frames: ");

frames = Integer.parseInt(br.readLine());

System.out.println("Please enter the length of the Reference string: ");

ref\_len = Integer.parseInt(br.readLine());

reference = new int[ref\_len];

mem\_layout = new int[ref\_len][frames];

buffer = new int[frames];

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

buffer[j] = -1;

System.out.println("Please enter the reference string: ");

for (int i = 0; i < ref\_len; i++) {

reference[i] = Integer.parseInt(br.readLine());

}

System.out.println();

for (int i = 0; i < ref\_len; i++) {

if (stack.contains(reference[i])) {

stack.remove(stack.indexOf(reference[i]));

}

stack.add(reference[i]);

int search = -1;

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

if (buffer[j] == reference[i]) {

search = j;

hit++;

break;

}

}

if (search == -1) {

if (isFull) {

int min\_loc = ref\_len;

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

if (stack.contains(buffer[j])) {

int temp = stack.indexOf(buffer[j]);

if (temp < min\_loc) {

min\_loc = temp;

pointer = j;

}

}

}

}

buffer[pointer] = reference[i];

fault++;

if (!isFull) {

pointer++;

if (pointer == frames) {

pointer = 0;

isFull = true;

}

}

}

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

mem\_layout[i][j] = buffer[j];

}

}

System.out.println("Order of Page Execution:");

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

for (int j = 0; j < ref\_len; j++) {

System.out.printf("%3d ", mem\_layout[j][i]);

}

System.out.println();

}

System.out.println("The number of Hits: " + hit);

System.out.println("Hit Ratio: " + (float) ((float) hit / ref\_len));

System.out.println("The number of Faults: " + fault);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

public class OptimalCache {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int frames, pointer = 0, hit = 0, fault = 0, ref\_len;

boolean isFull = false;

int buffer[];

int reference[];

int mem\_layout[][];

System.out.println("Please enter the number of Frames: ");

frames = Integer.parseInt(br.readLine());

System.out.println("Please enter the length of the Reference string: ");

ref\_len = Integer.parseInt(br.readLine());

reference = new int[ref\_len];

mem\_layout = new int[ref\_len][frames];

buffer = new int[frames];

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

buffer[j] = -1;

System.out.println("Please enter the reference string: ");

for (int i = 0; i < ref\_len; i++) {

reference[i] = Integer.parseInt(br.readLine());

}

System.out.println();

for (int i = 0; i < ref\_len; i++) {

int search = -1;

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

if (buffer[j] == reference[i]) {

search = j;

hit++;

break;

}

}

if (search == -1) {

if (isFull) {

int index[] = new int[frames];

boolean index\_flag[] = new boolean[frames];

for (int j = i + 1; j < ref\_len; j++) {

for (int k = 0; k < frames; k++) {

if ((reference[j] == buffer[k]) && (index\_flag[k] == false)) {

index[k] = j;

index\_flag[k] = true;

break;

}

}

}

int max = index[0];

pointer = 0;

if (max == 0)

max = 200;

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

if (index[j] == 0)

index[j] = 200;

if (index[j] > max) {

max = index[j];

pointer = j;

}

}

}

buffer[pointer] = reference[i];

fault++;

if (!isFull) {

pointer++;

if (pointer == frames) {

pointer = 0;

isFull = true;

}

}

}

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

mem\_layout[i][j] = buffer[j];

}

}

System.out.println("Order of Page Execution:");

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

for (int j = 0; j < ref\_len; j++) {

System.out.printf("%3d ", mem\_layout[j][i]);

}

System.out.println();

}

System.out.println("The number of Hits: " + hit);

System.out.println("Hit Ratio: " + (float) ((float) hit / ref\_len));

System.out.println("The number of Faults: " + fault);

}

}

Practical 9

class SCAN {

public static void main(String[] args) {

int no\_of\_tracks = 200;

int request\_queues[] = {82, 170, 43, 140, 24, 16, 190};

int curr\_position = 50;

// For Forward Direction

boolean goBackword = false;

int lastTrackBackword = curr\_position;

int lastTrackForward = curr\_position;

for(int track: request\_queues){

if(track < curr\_position){

goBackword = true;

lastTrackForward = no\_of\_tracks - 1;

if(track < lastTrackBackword){

lastTrackBackword=track;

}

}

if(!goBackword){

if(lastTrackForward < track)

lastTrackForward = track;

}

}

int no\_of\_track\_movement = lastTrackForward - curr\_position;

if(goBackword)

no\_of\_track\_movement += lastTrackForward - lastTrackBackword;

System.out.println("No of Track Movement: "+no\_of\_track\_movement);

}

}

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

class CSCAN {

public static void main(String[] args) {

int no\_of\_tracks = 200;

int request\_queues[] = {82, 170, 43, 140, 24, 16, 190};

int curr\_position = 50;

// For Forward Direction

boolean goBackword = false;

int lastTrackBackword = 0;

int lastTrackForward = curr\_position;

for(int track: request\_queues){

if(track < curr\_position){

goBackword = true;

lastTrackForward = no\_of\_tracks - 1;

if(lastTrackBackword < track){

lastTrackBackword=track;

}

}

if(!goBackword){

if(lastTrackForward < track)

lastTrackForward = track;

}

}

int no\_of\_track\_movement = lastTrackForward - curr\_position;

if(goBackword)

no\_of\_track\_movement += lastTrackForward + lastTrackBackword;

System.out.println("No of Track Movement: "+no\_of\_track\_movement);

}

}

­­­­­­­­­­­­­­­

Practical 10

Code:-

import java.util.\*;

class Banker

{

static int P = 5;

static int R = 3;

static void calculateNeed(int need[][], int maxm[][],int allot[][]){

System.out.println("The Need Matrix is");

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

for (int j = 0 ; j < R ; j++){

need[i][j] = maxm[i][j] - allot[i][j];

System.out.print(need[i][j]+" ");

}

System.out.println();

}

}

static boolean isSafe(int processes[], int avail[], int maxm[][],int allot[][]){

int [][]need = new int[P][R];

calculateNeed(need, maxm, allot);

boolean []finish = new boolean[P];

int []safeSeq = new int[P];

int []work = new int[R];

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

work[i] = avail[i];

int count = 0;

while (count < P){

boolean found = false;

for (int p = 0; p < P; p++){

if (finish[p] == false){

int j;

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

if (need[p][j] > work[j]) break;

if (j == R){

for (int k = 0 ; k < R ; k++)

work[k] += allot[p][k];

safeSeq[count++] = p;

finish[p] = true;

found = true;

}

}

}

if (found == false){

System.out.print("System is not in safe state");

return false;

}

}

System.out.print("System is in safe state.\nSafe"+" sequence is: ");

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

System.out.print(safeSeq[i] + " ");

System.out.println();

return true;

}

public static void main(String[] args) {

int processes[] = {0, 1, 2, 3, 4};

int avail[] = {3, 3, 2};

int maxm[][] = {{7, 5, 3},

{3, 2, 2},

{9, 0, 2},

{2, 2, 2},

{4, 3, 3}};

int allot[][] = {{0, 1, 0},

{2, 0, 0},

{3, 0, 2},

{2, 1, 1},

{0, 0, 2}};

isSafe(processes, avail, maxm, allot);

}

}

Output:-

The Need Matrix is

7 4 3

1 2 2

6 0 0

0 1 1

4 3 1

System is in safe state.

Safe sequence is: 1 3 4 0 2