K.J COLLEGE OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

SUBJECT CODE: 310248

LAB MANUAL
Laboratory Practice-I
(System Programming & Operating System)

Semester – I, Academic Year: 2022-23

310248: Laboratory Practice-I

Teaching Scheme: Examination Scheme: Practical: 4 Hrs/Week Term work: 25 Marks Credits: 02
Practical: 25 Marks **List of Laboratory Assignments**

Sr. No.	Group A	Page No.
1	Design suitable data structures and implement pass-I of a two-pass assembler for pseudo-machine in Java using object oriented feature. Implementation should consist of a few instructions from each category and few assembler directives.	
1.1	Implement Pass-II of two pass assembler for pseudo-machine in Java using object oriented features. The output of assignment-1 (intermediate file and symbol table) should be input for this assignment.	
2	Design suitable data structures and implement pass-I of a two-pass macro processor using OOP features in Java	
2.1	Write a Java program for pass-II of a two-pass macro-processor. The output of assignment-3 (MNT, MDT and file without any macro definitions) should be input for this assignment.	
	Group B	
3	Write a program to simulate CPU scheduling algorithms: FCFS , SJF (Preemptive), Priority (Non-Preemptive) and Round Robin (Preemptive)	
4	Write a program to simulate page replacement algorithms using 1. FIFO 2. Least Recently Used (LRU) 3.Optimal algorithm	

EXPERIMENT NO.3

AIM: To write a program to simulate the CPU scheduling algorithms First Come First Serve (FCFS) , Shortest job first(SJF),Priority and Round Robin(RR).

Problem Statement:

Write a program to simulate CPU Scheduling Algorithms: FCFS, SJF (Preemptive), Priority (Non Preemptive) and Round Robin (Preemptive).

Objectives:

1. To study the process management and various scheduling policies viz. Preemptive and Non preemptive.
2. To study and analyze different scheduling algorithms.

Theory :

A). FIRST COME FIRST SERVE:

DESCRIPTION: To calculate the average waiting time using the FCFS algorithm first the waiting time of the first process is kept zero and the waiting time of the second process is the burst time of the first process and the waiting time of the third process is the sum of the burst times of the first and the second process and so on. After calculating all the waiting times the average waiting time is calculated as the average of all the waiting times. FCFS mainly says first come first serve the algorithm which came first will be served first.

ALGORITHM:

Step1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process name and the burst time

Step 4: Set the waiting of the first process as _0'and its burst time as its turnaround time

Step 5: for each process in the Ready Q calculate

a). Waiting time (n) = waiting time (n-1) + Burst time (n-1)

b). Turnaround time (n)= waiting time(n)+Burst time(n)

Step 6: Calculate

a) Average waiting time = Total waiting Time / Number of process

b) Average Turnaround time = Total Turnaround Time / Number of process

Step 7: Stop the process

B). SHORTEST JOB FIRST:

DESCRIPTION: To calculate the average waiting time in the shortest job first algorithm the sorting of the process based on their burst time in ascending order then calculate the waiting time of each process as the sum of the bursting times of all the process previous or before to that process.

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Start the Ready Q according the shortest Burst time by sorting according to lowest to highest burst time.

Step 5: Set the waiting time of the first process as ‗0' and its turnaround time as its burst time.

Step 6: Sort the processes names based on their Burt time

Step 7: For each process in the ready queue,

calculate

a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n)

Step 8:

Calculate

c) Average waiting time = Total waiting Time / Number of process

d)Average Turnaround time = Total Turnaround Time / Number of process

Step 9: Stop the process

C). ROUND ROBIN:

DESCRIPTION: To aim is to calculate the average waiting time. There will be a time slice, each process should be executed within that time-slice and if not it will go to the waiting state so first check whether the burst time is less than the time-slice. If it is less than it assign the waiting time to the sum of the total times. If it is greater than the burst-time then subtract the time slot from the actual burst time and increment it by time-slot and the loop continues until all the processes are completed.

ALGORITHM:

Step 1: Start the process

Step 2: Accept the number of processes in the ready Queue and time quantum (or) time slice

Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time

Step 4: Calculate the no. of time slices for each process where No. of time slice for process (n) = burst time process (n)/time slice

Step 5: If the burst time is less than the time slice then the no. of time slices =1.

Step 6: Consider the ready queue is a circular Q,

Calculate

a) Waiting time for process (n) = waiting time of process(n-1)+ burst time of process(n-1) + the time difference in getting the CPU from process(n-1)

b) Turnaround time for process(n) = waiting time of process(n) + burst time of process(n)+ the time difference in getting CPU from process(n).

Step 7:

Calculate

c) Average waiting time = Total waiting Time / Number of process

d) Average Turnaround time = Total Turnaround Time / Number of process

Step 8: Stop the process

D). PRIORITY:

DESCRIPTION: To calculate the average waiting time in the priority algorithm, sort the burst times according to their priorities and then calculate the average waiting time of the processes. The waiting time of each process is obtained by summing up the burst times of all the previous processes.

ALGORITHM:
Step 1: Start the process
Step 2: Accept the number of processes in the ready Queue
Step 3: For each process in the ready Q, assign the process id and accept the CPU burst time
Step 4: Sort the ready queue according to the priority number.
Step 5: Set the waiting of the first process as _0' and its burst time as its turnaround time
Step 6: Arrange the processes based on process priority Step 7: For each process in the Ready Q calculate
Step 8: for each process in the Ready Q
calculate
a) Waiting time(n)= waiting time (n-1) + Burst time (n-1)

b) Turnaround time (n)= waiting time(n)+Burst time(n)
Step 9: Calculate
c) Average waiting time = Total waiting Time / Number of process
d) Average Turnaround time = Total Turnaround Time / Number of process Print the results in an order.
Step10: Stop

Conclusion:

CPU policies implemented successfully

Assignment No.: 4

Problem Statement:

Write a Java Program (using OOP features) to implement paging simulation using

1. FIFO
2. Least Recently Used (LRU)
3. Optimal algorithm

Objectives:

1. To study page replacement policies to understand memory management.
2. To understand efficient frame management using replacement policies.

Theory:

CONCEPT OF PAGE REPLACEMENT:

1. Page Fault: Absence of page when referenced in main memory during paging leads to a page fault.
2. Page Replacement: Replacement of already existing page from main memory by the required new page is called as page replacement. And the techniques used for it are called as page replacement algorithms.

NEED OF PAGE REPLACEMENT:

Page replacement is used primarily for the virtual memory management because in virtual memory paging system principal issue is replacement i.e. which page is to be removed so as to bring in the new page, thus the use of the page replacement algorithms. Demand paging is the technique used to increase system throughput. To implement demand paging page replacement is primary requirement. If a system has better page replacement technique it improves demand paging which in turn drastically yields system performance gains.

PAGE REPLACEMENT POLICIES:

1. Determine which page to be removed from main memory.

2. Find a free frame.

   1) If a frame is found use it

   2) if no free frame found, use page replacement algorithm to select a victim frame.

3) Write the victim page to the disk.

3. Read the desired page into the new free frame, change the page and frame tables. 4. Restart the user process.

PAGE REPLACEMENT ALGORITHMS:

1. FIFO

This is the simplest page replacement algorithm. In this algorithm, the operating system keeps track of all pages in the memory in a queue, the oldest page is in the front of the queue. When a page needs to be replaced page in the front of the queue is selected for removal.

2. **OPTIMAL PAGE REPLACEMENT ALGORITHM:**

Replace the page that will not be used for longest period of time as compared to the other pages in main memory. An optimal page replacement algorithm has lowest page fault rate of all algorithm. It is called as OPT or MIN.

ADVANTAGE:

1) This algorithm guarantees the lowest possible page-fault rate for a fixed no. of frames.

DISADVANTAGE:

1) The optimal page replacement algorithm is very difficult to implement, as it requires the knowledge of reference strings i.e. strings of memory references

. 3. **LEAST RECENTLY USED (LRU):**

LRU algorithm uses the time of the page's last usage. It uses the recent past as an approximation of the near future, then we can replace the page that has not been used for the longest period of the time i.e. the page having larger idle time is replaced.

ADVANTAGE:

1) The LRU policy is often used for page replacement and is considered to be good.

DISADVANTAGES:

1) It is very difficult to implement.

2) Requires substantial hardware assistance.

3) The problematic determination of the order for the frames defined by the time of last usage

Algorithm/Flowchart:

1. FIFO :

1. Start the process

2. Read number of pages n

3. Read number of pages no

4. Read page numbers into an array a[i]

5. Initialize avail[i]=0 .to check page hit

6. Replace the page with circular queue, while re-placing check page availability in the frame Place avail[i]=1 if page is placed in the frame Count page faults

7. Print the results.

8. Stop the process.

2. **LEAST RECENTLY USED**

1.Start the process

2. Declare the size

3. Get the number of pages to be inserted

4. Get the value

5. Declare counter and stack

6. Select the least recently used page by counter value

7. Stack them according the selection.

8. Display the values

9. Stop the process

3. **OPTIMAL ALGORTHIM:**

1. Start Program

2. Read Number Of Pages And Frames

3.Read Each Page Value

4. Search For Page In The Frames

5. If Not Available Allocate Free Frame

6. If No Frames Is Free Replace The Page With The Page That Is Least Used

7.Print Page Number Of Page Faults

8.Stop process.

Input:

1. Number of frames
2. Number of pages
3. Page sequence

Output:

1. Sequence of allocation of pages in frames (for each algorithm)
2. Cache hit and cache miss ratio.

Test Cases:

1. Test the page hit and miss ratio for different size of page frames.
2. Test the page hit and miss ratio for both algorithms with different page sequences.

Software Requirement:

1. Fedora
2. Eclipse
3. JDK

Hardware Requirement:

Conclusion: Successfully implemented all page replacement policies.