LOVELY PROFESSIONAL UNIVERSITY

Academic Task-3(Operating System)

Name of Faculty member-Ashu ma'am
Course Code-CSE 316

Student Name: Km Suruchi Srivastava

Student ID: 11805235

E-mail: suruchisrivastava39@gmail.com

GitHub Username: suruchi1180

GitHub Link: https://github.com/suruchi1180/osproject.github.io.git

Description:

For problem 24:

This is a scheduling program to implement a Queue with two levels:

Level 1 : Fixed priority preemptive Scheduling

Level 2 : Round Robin Scheduling

For a Fixed priority pre-emptive scheduling if one process P1 is scheduled and running and another process P2 with higher priority comes. The New process with high priority process P2 preempts currently running process P1 and process P1 will go to second level queue. Time for which process will strictly execute must be considered in the multiples of 2.

All the processes in second level queue will complete their execution according to round robin scheduling.

In this program Queue 2 will be processed after Queue 1 becomes empty and Priority of Queue 2 has lower priority than in Queue 1.

Algorithm:

In this program algorithm for round robin scheduling and multilevel queue scheduling is used.

Algorithm For Multilevel Queue:

When a process starts executing then it first enters queue 1.

In queue 1 process executes for 4 unit and if it completes in this 4 unit or it gives CPU for I/O operation in this 4 unit than the priority of this process does not change and if it again comes in the ready queue than it again starts its execution in Queue 1.

1. If a process in queue 1 does not complete in 4 unit then its priority gets reduced and it shifted to queue 2.
2. Above points 2 and 3 are also true for queue 2 processes but the time quantum is 8 unit.In a general case if a process does not complete in a time quantum than it is shifted to the lower priority queue.
3. In the last queue, processes are scheduled in FCFS manner.
4. A process in lower priority queue can only execute only when higher priority queues are empty.
5. A process running in the lower priority queue is interrupted by a process arriving in the higher priority queue.

Algorithm for round robin scheduling:

```
1- Create an array rem_bt[] to keep track of remaining burst
  time of processes. This array is initially a
  copy of bt[] (burst times array)
2- Create another array wt[] to store waiting times of
  processes. Initialize this array as 0.
3- Initialize time : t = 0
4- Keep traversing the all processes while if it is all processes
  are not done. Do following for i'th process
  not done yet.
   a- If rem_bt[i] > quantum
     (i)  t = t + quantum
     (ii) bt_rem[i] -= quantum;
   c- Else // Last cycle for this process
     (i)  t = t + bt_rem[i];
     (ii) wt[i] = t - bt[i]
     (ii) bt_rem[i] = 0; // This process is over
```

Description

The given problem is based on the concept of Priority Scheduling and Round Robin Scheduling. The process are executed one at a time on the basis of arrival time but if a process whose priority is higher than the currently executing process occurs then the new arrived process pre-empts the currently executing process. All the pre-empted process are executed according to round robin scheduling in the end. Priority Scheduling. Priority scheduling is one of the most common scheduling algorithms in batch systems. Each process is assigned a priority. Process with the highest priority is to be executed first and so on. Processes with the

same priority are executed on first come first served basis. Priority can be decided based on memory requirements, time requirements or any other resource requirement.

Round Robin Scheduling- Round Robin is a CPU scheduling algorithm where each process is assigned a fixed time slot in a cyclic way.

- It is simple, easy to implement, and starvation-free as all processes get fair share of CPU.
- One of the most commonly used technique in CPU scheduling as a core.
- It is pre-emptive as processes are assigned CPU only for a fixed slice of time at most.
- The disadvantage of it is more overhead of context switching

Complexity:

For line 5 to 39 complexity is n.

For line 40 to 62 complexity is n^2.

For line70 to 166 complexity is n^2.

For line172to 266complexity is n^2.

For line 269to287complexity is n^2.

Forline299 to326 complexity is n^2.

Overall= n^2

Boundary Condition

Number of processes should not exceed 3 i.e. n<=3.
Arrival time for all processes should be different.

Test Cases

1.
Total Process = 3
: For P1
Arrival Time= 1
Burst Time = 4
Priority= 5
: For P2
Arrival Time= 2
Burst Time = 2

Priority= 7

: For P3

Arrival Time= 3

Burst Time = 3

Priority= 4

At arrival time 1 Process P1 will start to execute, at time 2 P2 will arrive but since its priority is less than that of P1(higher numerical value less priority) it will wait. At time 3 process P3 will arrive and its priority is greater than that of P1 so it will pre-empt the process P1 and continue to execute. When P3 will execute completely than P2 will again arrive and start its execution, here at this point of time P2 will not get executed before P1 because it has been pre- empted and all the pre-empted process has to be executed at the end by round robin scheduling. After P2 is executed P3 will arrive and get executed.

2.

Total Process = 3

:For P1

Arrival Time= 1

Burst Time = 4

Priority= 2

:For P2

Arrival Time= 2

Burst Time = 5

Priority= 1

:For P3

Arrival Time= 3

Burst Time = 2

Priority= 4

GitHub Link: https://github.com/suruchi1180/osproject.github.io.git