CSE 310 : Operating System Lab
Lab 06

Process Scheduling Algorithms (Part-1)

Objective: Writing codes in Java or in C/C++ to simulate various Process Scheduling Algorithms

IDE: C/C++, Java

Background:

6a. First Come First Serve Process Scheduling

For FCFS scheduling algorithm, you shall get the number of processes/jobs in the system and their CPU burst times. This is a non-preemptive scheduling where a process getting the CPU will finish its works and the leave the CPU after it exhausts its required CPU time. The scheduling is performed on the basis of arrival time of the processes irrespective of their other parameters. Each process will be executed according to its arrival time. The waiting time and turnaround time of each of the processes can be calculated as well as the average system turnaround and waiting time.

Example:

Process	Arrival Time	CPU Time
P1	4	5
P2	0	7
P3	2	9

Gantt chart:

|--------P2---------|--------P3---------|--------P1---------|

0 7 16 21

	Waiting Time	Turnaround Time
P1	16-4=12	12+5=17
P2	0	0+7=7
P3	7-2=5	5+9=14

Average Waiting Time: (12+0+5)/3 = 5.66

Average Turnaround Time: (17+7+14) / 3 = 12.66

Algorithm

Input the processes along with their cputime and arrivaltime
Find waiting time (wt) for all processes.
As the first process comes in and do not need to wait; waiting time for process 0 will be 0 i.e. waitingtime [0] = 0.
Find waiting time for all other processes i.e. for
process i ;
waitingtime[i] = cputime[i-1] + waitingtime [i-1] – arrivaltime[i].
Find turnaround time[i] = waiting_time [i] + cputime[i]
for all processes.
Find average waiting time =
total_waiting_time / no_of_processes.
Similarly, find average turnaround time =
total_turn_around_time / no_of_processes.

Sample input:

Enter the number of process: 3

Enter the CPU times

5 7 9

Enter the arrival times

4 0 2

Sample Output:

Process 1: Waiting Time: 12 Turnaround Time: 17

Process 2: Waiting Time: 0 Turnaround Time: 7

Process 3: Waiting Time: 5 Turnaround Time: 14

Average Waiting time: 5.66

Average Turnaround time: 12.66

6.b Shortest Job First (Non Pre-emptive version)

For the **Shortest Job First scheduling** algorithm, you shall get the number of processes/jobs in the system and their CPU burst times. This is a non-preemptive scheduling where a process getting the CPU will finish its works and the leave the CPU after it exhausts its required CPU time. The scheduling is performed on the basis of the shortest CPU time of the processes irrespective of their other parameters. Each process will be executed according to its arrival time. The waiting time and turnaround time of each of the processes can be calculated as well as the average system turnaround and waiting time.

Example

Process	Arrival Time	CPU Time
P1	4	5
P2	0	7
P3	2	9

Gantt chart:

|--------P2---------|--------P1---------|--------P3---------|

0 7 12 21

	Waiting Time	Turnaround Time
P1	7- 4= 3	3+5=8
P2	0	0+7=7
P3	12-2=10	10+9=19

Average Waiting Time: (3+0+10)/3 = 4.33

Average Turnaround Time: (8+7+19) / 3 = 11.33

Algorithm Hint

Input the processes along with their cputime and arrivaltime

Select a process *i* having the lowest arrival time.
Keep a timer variable that is incremented as follows
Timer = timer + cputime [i]
For all the processes having their arrival time <= Timer
Select a process *i* having the lowest arrival time.

Sample input:

Enter the number of process: 3

Enter the CPU times

5 7 9

Enter the arrival times

4 0 2

Sample Output:

Process 1 : Waiting Time : 3 Turnaround Time : 8

Process 2 : Waiting Time : 0 Turnaround Time : 7

Process 3 : Waiting Time : 10 Turnaround Time : 19

Average Waiting time : 4.66

Average Turnaround time : 11.66

Lab Report

1. Implement the code for Shortest Job First (SJF) with preemption
2. Analyze all the code (including SJF preemption) that you have implemented to solve the problem.
3. Comment on your findings and write down the challanges you faced to implement those problems.

Note:

- You may use Linux OS for compiling and running your code. You may use any programming language (C / gcc / C++) for implementing the program.

- Commands to compile and run C/C++ programs in Linux environement.

  gcc -o output online.c (for C)
  g++ -o output online.cpp (for C++)
  Executing the compiled code: ./output

- You also can use code blocks or any other IDE for compile and run your code