Project 2

CPU Scheduler Simulator – A Rite of Passage (160 points total)

Overview:

This project has been a rite of passage for Operating Systems students for many many years (at least 12 by my reckoning). As we learned in class there are many different types of scheduling algorithms. In this project you will create a simulator that, for a given list of processes, you will simulate the completion of each using one of seven schedulers listed below.

You have a choice of using any programming language (though I recommend Java given the object oriented nature of this project), provided you have support of reading from a plain text text-file. Upon completion you will schedule demonstration time with the TA, at which time you will receive a new process text-file to run your simulations with. The TA will compare your output to correct expected output and grade you accordingly. This project will be a group project in which you will work in teams of 2 or 3.

Scheduler 1 – First Come First Serve (FCFS)

With the FCFS scheduler, the first process to enter the ready queue will begin executing and will not be interrupted (preempted). If any other processes arrive in the ready queue, they will wait (in the order they were received) until the processor becomes free (the current process finishes executing its current CPU burst).

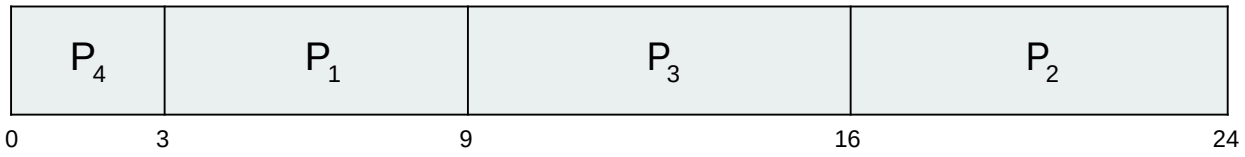
Process	Burst Time
P_1	24
P_2	3
P_3	3

Scheduler 2 – Shortest Job First (SJF)

SJF grabs whatever process in the ready queue that has the shortest CPU burst. Like FCFS, any process that is in the processor will not be preempted, and will only leave the CPU when its current CPU burst is

finished. In the below example, consider the processor just became free and the ready queue contains Processes 1-4.

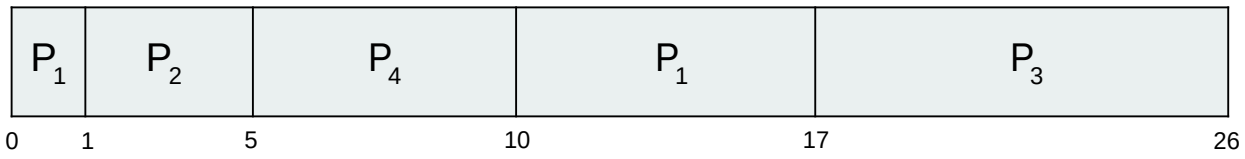
ProcessArrival	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3



Scheduler 3 – Shortest Job Remaining (SJR)

SJR is essentially the preemptive version of SJF. If a job enters the ready queue and has less of a CPU burst than the current process in the processor, the current process executing will be removed and the new process will be placed inside the processor.

Process	*Arrival* Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5



Scheduler 4 – Priority Scheduling (PS)

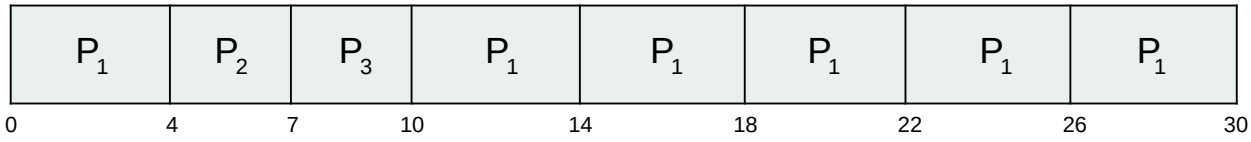
PS is a preemptive scheduler that puts the process with the highest priority (lowest integer value) into the processor. If a process in the ready queue has a higher priority than the process currently executing, that process go into the processor and the current process will be removed.

Process	Burst Time	Priority
P_1	11	3
P_2	5	1
P_3	2	4
P_4	1	5

Scheduler 5 – Round Robin (RR)

RR is a preemptive scheduler that uses a time quantum *q* that allows a process to execute for at most *q* cycles. Which process enters the queue next is served in a FIFO fashion. If the next process to execute has a CPU burst time less than the *q*, will execute the remaining burst time, then load a next process. A process leaving the process that still has CPU burst and is not ready to go to I/O will return to the ready queue.

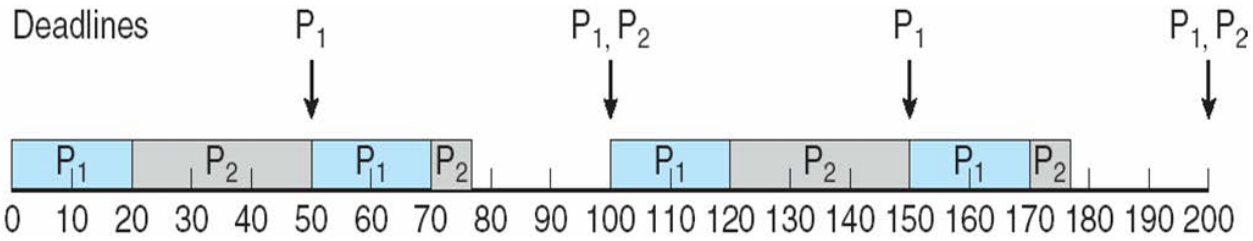
Process	Burst Time	Quantum = 4
P_1	24	
P_2	3	
P_3	3	



Scheduler 6 – Priority Rate Monotonic (PRM)

PRM is a priority based scheduler that assigns a higher priority to shorter periods and lower priorities to longer periods. Remember that a period is a constant interval in cycles that a process must be serviced and finish its execution by. A period starts as soon as the process enters the ready queue, and the process must start executing and finish before its period is over. If the process does not finish executing before its period is over, the process has missed its deadline.

In the example below P1 has a period of 50, and P2 has a period of 100.

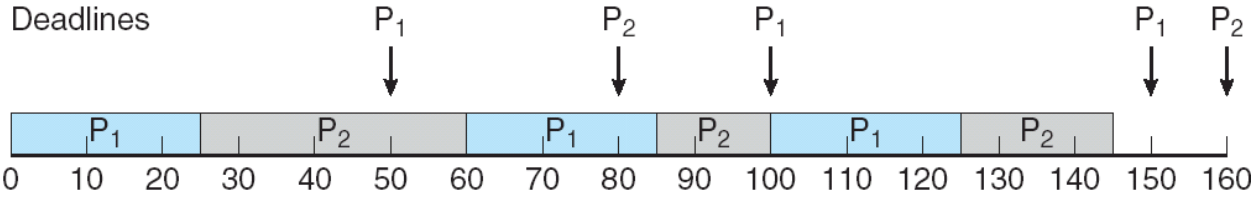


Scheduler 7 – Earliest Deadline First (EDF)

EDF is another preemptive priority based scheduler similar to PRM, except it assigns its priority based on the deadline. If a process has a deadline that is sooner than another process, it has a higher priority. To simplify, we can define deadline for any given Process X with the following formula:

$$deadline_{P_x} = period_{P_x} - CPU\ Burst_{P_x}$$

In the example below P1 has a period of 50 and arrives in the ready queue every period, while P2 has a period of 80, and arrives in the ready queue every 80 cycles.



Description of Project Details

In the system you are building, you will simulate each process in each of the above 7 algorithms. Assume each process arrives one at a time (Process 1 arrives at Cycle 0, Process 2 at Cycle 1, … Process N arrives at Cycle N-1). You may assume that the processor doesn't start processing until all the processes are loaded into the ready queue. Each process is characterized by four parameters : the process id (PID), the CPU burst, I/O burst, the priority, and the period (for the real-time algorithms).

Assume that you have only one i/o burst and if a process needs an i/o operation, it occurs exactly half way through the CPU burst (for example, process 1 has a CPU burst of 10 cycles and I/O burst of 5 seconds then I/O burst occurs after the process has run for 5 cycles).

You should be able to read the information about processes from a file. Note that FCFS and SJF are non-preemptive and the remainder are preemptive. Keep all units in cycles. Break all ties based on the Bakery Algorithm (the process with the smaller process id number will get ahead). The only exception is in the case of a preemptive tie. If there exists a preemptive tie with a process that is already executing let it keep executing (this rule supersedes the Bakery Algorithm). Assume you have infinite instances of I/O devices (i.e. no need to create an I/O queue to schedule I/O). There should be no need for user interaction.

For PRM and EDF if a Process needs to go to I/O and does so before its period is over it has successfully met its deadline. Once I/O is over however and the process has re-entered the ready queue, its period resets and restarts.

Your program should generate two reports (as text files):

- Report 1 – Final Report
  - Throughput for each scheduling algorithm
  - Average waiting time for each scheduling algorithm
  - CPU utilization for each scheduling algorithm
  - Turn-around time for each scheduling algorithm
  - Sequence of processes in the CPU for each scheduling algorithm
  - Number of Deadline Violations (PRM and EDF only)
  - A rating (1st place, 2nd place, etc) for each of the first 5 scheduling algorithm for the given input file. This rating should take into consideration throughput, wait time, CPU utilization and turn-around time.
  - A rating (1st place and 2nd place) for PRM and EDF taking into consideration the deadline violations, throughput, wait time, CPU utilization, and turn-around time.
- Report 2 – Cumulative Snap shots
  - Snapshot of the ready queue and processes in I/O every x cycles.
  - Snapshot of which process is executing every x cycles.
  - Snapshot of the remaining CPU burst time and I/O burst time every x cycles.
  - You may default x to 10, but only if the user does not provide a different value at command line (read below).

The structure of the input file will look exactly like this:

Total_Processes 10

Quantum 3

% P_ID	CPU_burst	I/O_burst	Priority	Period
0	10	4	2	17
1	8	2	1	15

2	12	0	5	20
3	2	4	4	5
4	8	3	0	14
5	6	4	2	18
6	4	0	5	12
7	16	7	5	35
8	14	0	1	32
9	2	10	1	8

The first line of the text file specifies exactly how many processes will be read in (for advanced allocation of memory in the program…like an array). The second line specifies the time quantum to use for RR. The third line you can just skip… it is only there for your visual explanation.

Finally the executable should be command line friendly. That is to say whatever the compiled programs name is (let us assume it is 'simulation') I should be able to run it in the following format by command line.

>simulation processfile.dat 10

Where processfile.dat is where the process information is located, and 10 is the number of cycles I want a snapshot taken for Report 2. Be sure to add error checking to make sure the user is provided the parameters (Example: "Warning, you are using the simulation incorrect, correct usage is: simulation processfile.dat 10)

Point distribution

1. **Comments are a necessity**. Comments at the top of the assignment with your name, date, and description of the assignment, followed by in-code comments describing logical chunks of code (for instance, describe why you have this if statement**). No comments are equivalent to no work and the project will receive a zero.** On the other hand, comments on non-functioning code may get you some partial credit. Since this is a team project, begin each comment with your initials…
   //CR: look I made a comment!
2. **You must demonstrate this project to the TA otherwise it will not be graded**.
3. **You must also submit your source code to myCourses**.
4. Simulation of FCFS (5 points)
5. Simulation of SJF (5 points)
6. Simulation of SJR (10 points)
7. Simulation of PS (10 points)
8. Simulation of RR (10 points)

9. Simulation of PRM (25 points)
10. Simulation of EDF (25 points)
11. Report 1: (44 pts total distributed as follows)
    a. Throughput for each scheduling algorithm (7 pts)
    b. Average waiting time for each scheduling algorithm (7 pts)
    c. CPU utilization for each scheduling algorithm (7 pts)
    d. Turn-around time for each scheduling algorithm (7 pts)
    e. Sequence of processes in the CPU for each scheduling algorithm (7 pts)
    f. Number of Deadline Violations (PRM and EDF only) (2 pts)
    g. A rating (1^{st} place, 2^{nd} place, etc) for each of the first 5 scheduling algorithm for the given input file. This rating should take into consideration throughput, wait time, CPU utilization and turn-around time. (5 pts)
    h. A rating (1^{st} place and 2^{nd} place) for PRM and EDF taking into consideration the deadline violations, throughput, wait time, CPU utilization, and turn-around time. (2 pts)
12. Report 2: (6 pts total distributed as follows)
    a. Snapshot of the ready queue and processes in I/O every x cycles. (2 pts)
    b. Snapshot of which process is executing every x cycles. (2 pts)
    c. Snapshot of the remaining CPU burst time and I/O burst time every x cycles. (2 pts)
13. Command line friendly with error checking (20 pts)

Submit your solution to myCourses before the due date. A 10% per day penalty will be applied to assignments that are late.