Muslim Hussaini Operating Systems HW #2

## 1) Consider a preemptive priority scheduling algorithm based on dynamically changing priorities. Larger priority numbers imply higher priority. When a process is waiting in the ready queue, its priority increases at a rate $\alpha$. When it is running, its priority increases at a rate $\beta$. Both $\alpha$ and $\beta$ are greater than 0. All processes are given a priority of 0 when they enter the ready queue. The parameters $\alpha$ and $\beta$ can be set to design many different scheduling algorithms.

**(a) What are the implications of $\beta<\alpha$? What scheduling algorithm does this scenario mimic?**

The longer something is waiting for the higher its priority is. This implies that that the scheduler will favor processes that have been waiting for longer which is similar to FCFS scheduling, whichever is next that waited longest will be next.

**(b) What are the implications of $\alpha<\beta$? What scheduling algorithm does this scenario mimic?**

The longer something is running the higher its priority is. This implies that the scheduler will allow the current running job to continue, and not switch to another job, which is similar to Shortest Job First.

## 2) Explain how the following scheduling algorithms discriminate either in favor of or against short processes:

**(a) FCFS**
Discriminates against short processes because if a long process comes in first, all the shorter processes will have to wait behind it.

**(b) RR**
Depending on the quantum, if the quantum is small, it slightly favors short processes, but if its large, it discriminates against short jobs.

**(c) SRTF**
Favors shorter processes, always schedules process with the least remaining time and directly favors short jobs.

**(d) Multilevel feedback queues**
Jobs that finish quickly start in the higher queue and finish quickly, longer jobs sink to the lower priority queues so this one favors short processes technically.

## 3) Suppose that the following processes arrive for execution at the times indicated. Each process will run for the amount of time listed. In answering the questions, use non-preemptive scheduling, and base all decisions on the information you have at the time the decision must be made

| Processes | Arrival time | Service time |
|---|---|---|
| $P_1$ | 0.0 | 8 |
| $P_2$ | 0.4 | 4 |
| $P_3$ | 1.0 | 1 |

**(a) Compute the average turn around time for these processes with FCFS scheduling algorithm.**
(8.0 + 11.6 + 12) / 3 = average turnaround time of 10.53... units

**(b) Compute the average turnaround time for these processes with SJF scheduling algorithm.**
(8.0+8.0+12.6) / 3 = average turnaround time of 9.53... units

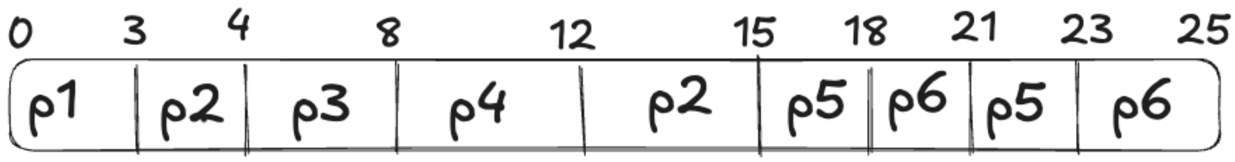
**(c) Compute what the average turnaround time will be if the CPU is left idle for the first 1 unit and then SJF scheduling is used.**
( 1 + 5.6 + 14 ) / 3 = average turnaround time of 6.87... units

## 4) The following processes are being scheduled using a preemptive, priority-based, round-robin scheduling algorithm.

| Processes | Arrival time | Service time | Priority |
|---|---|---|---|
| $P_1$ | 0 | 3 | 8 |
| $P_2$ | 0 | 4 | 3 |
| $P_3$ | 4 | 4 | 4 |
| $P_4$ | 8 | 4 | 4 |
| $P_5$ | 15 | 5 | 5 |
| $P_6$ | 18 | 5 | 5 |

**Each process is assigned a numerical priority, with a higher number indicating a higher relative priority. The scheduler will execute the highest priority process. For processes with the same priority, a round- robin scheduler will be used with a time quantum of 3 time units. If a process is preempted by a higher-priority process, the preempted process is placed at the end of the queue.**

**(a) Show the scheduling order of the processes using a Gantt chart.**



**(b) What is the turnaround time for each process?**

P1: 3
P2: 15
P3: 4
P4: 4
P5: 8
P6: 7

**(c) What is the waiting time for each process?**

P1: 0
P2: 11
P3: 0
P4: 0
P5: 3
P6: 2

## 5) Consider an interesting variation of the round-robin scheduler where it assigns each process a time quantum and a priority. The initial value of a time quantum is 5 time units. However, every time a process has been allocated the CPU and uses its entire time quantum (does not block for I/O), 1 time unit is added to its time quantum, and its priority level is boosted. Consider the time quantum for a process can be increased to a maximum of 10 time units. When a process blocks before using its entire time quantum, its time quantum is reduced by 2 time units, but its priority remains the same. What type of process (CPU-bound or I/O-bound) does this variation of round-robin scheduler favor? Explain.

This variation of round robin seems to favor CPU bound processes, because when a process does not block for I/O, its time unit is increased ( until the maximum ) and its priority keeps going higher. However for I/O bound processes that DO block for I/O its time quantum gets reduced by 2 units, meaning I/O bound processes are not favored like CPU bound processes are.

## 6) What are the implications of assigning the following values to the parameters used by SJF algorithm with exponential averaging for CPU burst period prediction?

**(a) $\alpha$=0 and $\tau_0$ =100 time units**

Tn+1 = 0 * tn + (1-0) * tn

If t0 = 100, then each process will be predicted to have a burst of 100 Time units. This feels like a FCFS algorithm because the scheduler predicts that they all have the same burst, no way of scheduling a shorter job first

**(b) $\alpha$= 0.99 and $\tau_0$ = 10 time units**

Tn+1  = .99 * tn (.1) * tn

The value of an estimated burst is not going to be the same each time, and will actually be able to predict based on Tn ( prev burst ) what the next burst might be, and this is more similar to an actual adaptive Shortest Job first ( because now based on the last job we can make an estimate of what the next jobs burst will be and schedule based on that)