Vivian Do
CSC 4320
Apr 09 2023

Spring 2023 – 4320/6320 Section 006 Operating Systems

Homework 5: Due 04/09/2023 at 11:59 PM

Your programs – if requested – must compile with gcc and execute on snowball.cs.gsu.edu! Please see https://cscit.cs.gsu.edu/sp/guide/snowball for more details. You may use whatever IDEs / text editors you like, but you must submit your responses on iCollege.

1. **(40 Points)** Let the processes P1, P2, P3, P4, and P5 be given. They arrive in the system at the same time in this order. The processes have the following service times (in time units) and priorities:

Process	**P1**	**P2**	**P3**	**P4**	**P5**
Service time (CPU burst)	10	1	3	1	2
Priority	3	1	3	4	2

For each of the scheduling methods "First-Come, First-Served", "Shortest Job First", "Round Robin", and "Priority Scheduling" specify **the execution order of the processes** and the **average execution time (average turnaround time) over all processes**. For this purpose, first determine and **state the execution times of the individual processes** and then calculate the average value.

Note: For "Shortest Job First", If the required service time (CPU time) is the same, "First-Come, First Served" applies. For "Round Robin", the time quantum q = one time unit. For "Priority Scheduling", low priority numbers stand for high priorities, with 0 being the highest possible priority. In case of the same priority, "First-Come, First-Served" applies. Also, non-preemptive scheduling should be applied in this case. In any cases, you will not have to draw Gantt charts.

First-Come, First Served:
Execution Order: P1 → P2 → P3 → P4 → P5
Average Time: 13.4

0 → 10(p1) → 1(p2) → 3(p3) → 1(p4) → 2(p5)
0 → 10 → (10+1=11) → (11+3=14) → (13+1=15) + (15+2=17)
$(\frac{10+11+14+15+17}{5}) = \frac{67}{5} = 13.4$

Shortest Job First:
Execution Order: P2 → P4 → P5 → P3 → P1
Average Time: 6.2

0 → 1(p2) → 1(p4) → 2(p5) → 3(p3) → 10(p1)
0 → 1 → (1+1=2) → (2+2=4) → (4+3=7) → (7+10=17)
$(\frac{1+2+4+7+17}{5}) = \frac{31}{5} = 6.2$

Round Robin:
Execution Order: P1 → P2 → P3 → P4 → P5 → P1 → P3 → P5 → P1 → P3 → P1 → P1 → P1 → P1 → P1 → P1 → P1
Average Time: 9

0 → 1(p1) → 1(p2) → 1(p3) → 1(p4) → 1(p5) → 1(p1) → 1(p3) → 1(p5) → 1(p1) → 1(p3) → 1(p1) → 1(p1) → 1(p1) → 1(p1) → 1(p1) → 1(p1) → 1(p1)
0 → 1 → (1+1=2) → (2+1=3) → (3+1=4) → (4+1=5) → (5+1=6) → (6+1=7) → (7+1=8) → (8+1=9) → (9+1=10) → (10+1=11) → (11+1=12) → (12+1=13) → (13+1=14) → (14+1=15) → (15+1=16) → (16+1=17)
$(\frac{1+2+3+4+5+6+7+8+9+10+11+12+13+14+15+16+17}{17}) = \frac{153}{17} = 9$

Priority Scheduling:
Execution Order: P2 → P5 → P1 → P3 → P4
Average Time: 10

0 → 1(p2) → 2(p5) → 10(p1) → 3(p3) → 1(p4)
0 → 1 → (1+2=3) → (3+10=13) → (13+3=16) → (16+1=17)
$(\frac{1+3+13+16+17}{5}) = \frac{50}{5} = 10$

2. (5 Points) In order not to let any process wait longer than 500 ms, a system developer programs the "Round Robin" procedure with a dynamic time quantum size. With n processes ready, the time quantum Q is set to 500 ms/n. What do you think of this scheduling strategy? (In which situations is this strategy practicable, and when is it not? Are there edge cases?) Explain your reasoning.

Round Robin scheduling has both advantages and disadvantages. This has a preemptive scheduling where processes are switched from a running or waiting state into a ready state, and this makes Round Robin reliable.
A practical way of using Round Robin scheduling is when the processes are all of equal importance and need to be compiled in a specific time frame. It is not practical to use Round Robin when the time quantum is either too large or too small; the processes will not be able to finish within that given time frame. There are edge cases where processes are not able to finish due to the time quantum being too small.

3. **(5 Points)** Real-time scheduling: A system consisting of two processes P1 and P2 is given. Suppose that process P1 has a period p_1 of 50, an execution time t_1 of 25, and a deadline that matches its period (50). Further suppose that P2 has a period p_2 of 75, an execution time t_2 of 30, and a deadline that matches its period (75). Is this real-time system schedulable under rate-monotonic scheduling?

Explain your answer using the concept of the overall CPU utilization.

Yes, the real time system is under the rate-monotonic scheduling.

$$\left(\frac{t1+t1}{p1+p2}\right) = \left(\frac{25+30}{50+75}\right) = \left(\frac{55}{125}\right) = 0.44$$

The overall CPU utilization of the system is 44%.
For a rate-monotonic scheduling algorithm, its scheduling condition is that the overall CPU utilization must be less than or equal to 100%. Since the overall CPU utilization is 44%, the system is schedulable under rate-monotonic scheduling.