Operating System 2: Exam 1

27th January 2022

Tanmay Garg CS20BTECH11063

**Q1. Five batch jobs, A through E, arrive at a computer center at essentially the same time. They have**

**an estimated running time of 15, 9, 3, 6, and 12 minutes, respectively. Their (externally defined)**

**priorities are 6, 3, 7, 9, and 4, respectively, with a lower value corresponding to a higher priority.**

**For each of the following scheduling algorithms, determine the turnaround time for each process**

**and the average turnaround for all jobs. Ignore process switching overhead. Explain how you**

**arrived at your answers. In the last three cases, assume only one job at a time runs until it finishes,**

**and all jobs are completely processor bound.**

**(a) round robin with a time quantum of 1 minute**

**(b) priority scheduling**

**(c) FCFS (run in order 15, 9, 3, 6, and 12)**

**(d) shortest job first**

Sol.

Turnaround time = Time of completion – Time of arrival

Since all processes arrived at t=0, time of arrival = 0

(a) Round Robin method with time quantum = 1 min

The order of occurrence of all the processes:

A, B, C, D, E, A, B, C, D, E, A,B,C,D,E,A,B,D,E,A,B,D,E,A,B,D,E,A,B,E,A,B,E,A,B,E,A,E,A,E,A,E,A,A,A

Turnaround time for each process:

A: 45 min, B: 35 min, C: 13 min, D: 26 min, E: 42 min

Average turnaround time: 32.2 min

(b) Priority Scheduling

The order of occurrence of all the processes:

B, E, A, C, D

Turnaround time for each process:

A: 36 min, B: 9 min, C: 39 min, D: 45 min, E: 21 min

Average turnaround time: 30 min

(c) FCFS

The order of occurrence of all the processes:

A,B,C,D,E

Turnaround time for each process:

A: 15 min, B: 24 min, C: 27 min, D: 33 min, E: 45 min

Average turnaround time: 28.8 min

(c) Shortest Job First

The order of occurrence of all the processes:

C,D,B,E,A

Turnaround time for each process:

A: 45 min, B: 18 min, C: 3 min, D: 9 min, E: 30 min

Average turnaround time: 21 min

**Q2. Consider a variation of round robin scheduling, say NRR scheduling. In NRR scheduling, each**

**process can have its own time quantum, q. The value of q starts out at 40 ms and decreases by 10**

**ms each time it goes through the round robin queue, until it reaches a minimum of 10 ms. Thus,**

**long jobs get decreasingly shorter time slices.**

**(a) Develop the Gantt chart for scheduling algorithm for three jobs A, B, and C that arrive in the**

**system having estimated burst times of 100 ms, 120 ms, and 60 ms respectively. Then compute the**

**waiting times and the average waiting time as well. (6 pts)**

**(b) Also identify some advantages and disadvantages that are associated with this algorithm as**

**compared to traditional round robin. (3 pts)**

Sol.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | C | A | B | C | A | B | C | A | B | C | A | B | C | A | B | A | B | B | B | B |  |
| T=0 | 40 | 70 | 90 | 100 | 110 | 120 | 130 | 140 | 150 | 160 | 170 | 180 | 190 | 200 | 210 | 220 | 230 | 240 | 250 | 260 | 270 | 280 |

Waiting Time = Completion Time – Burst Time

Waiting time for A: 140 ms, B: 160ms, C: 150ms

Average waiting time: 150ms

Comparison of NRR with RR:

* This dynamic change in time quantum can be useful in the case when processes are of large time bursts, so NRR can significantly reduce, the waiting time for individual processes
* The total number of context switches would reduce significantly in the starting few time intervals as the time quantum has increased significantly
* There can be a CPU overhead to keep track of time quantum and updating it regularly
* In the case, when a new process arrives after a certain point in time, it would not be able to take advantage of the initial high time quantum and would be forced to undergo a normal RR
* If the processes are of long CPU bursts, then over many numbers of processes, the entire scheduling would effectively become a RR scheduling method

**Q3. Five jobs are waiting to be run. Their expected run times are 9, 6, 3, 5, and X. In what order**

**should they be run to minimize average response time? (Your answer will depend on X.) (3 pts)**

Sol.

* In order to minimize the average response time, the order in which the processes should be run will depend on the range of values of X (shortest Job first)
* , then the order should be E, C, D, B, A
* , then the order should be C, E, D, B, A
* , then the order should be C, D, E, B, A
* , then the order should be C, D, B, E, A
* , then the order should be C, D, B, A, E

**Q4. Consider two processes, P1 and P2 , where p1 = 40, t1 = 25, p2 = 75, and t2 = 30.**

**(a) Can these two processes be scheduled using rate-monotonic scheduling? Illustrate your answer**

**using a Gantt chart. (4 pts)**

**(b) Illustrate the scheduling of these two processes using earliest-deadline-first (EDF) scheduling.**

**(4 pts)**

Sol.

(a)

* No, these processes cannot be scheduled using rate-monotonic scheduling
* When we schedule P1 first it will take its complete time in CPU i.e. 25 units
* Then P2 is scheduled, and it is run for only 15 units, as it will be preempted by P1 at (t=40units)
* Then P1 again takes complete time of CPU, and then switches to P2 to complete it before its deadline before t = 75 units
* But the process P2 will not be able to finish as it has 15 units of process still remaining and time available is 10 units
* Moreover, the CPU utilization for both the processes is , which is more than 100%

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| P1 | P2 | P1 | P2 | P2 |  |
| T=0 | 25 | 40 | 65 | 75 |  |

(b)

* P1 will be scheduled first as it has an earlier deadline of t = 40 units
* With EDF, P1 will run completely before switching to P2
* When it switches to P2 at t = 25, P2 will run further as it has a deadline (t=75) earlier than P1(t=80)

Gantt Chart will look something like this

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| P1 | P2 | P1 | P1 | P2 | P1 | P1 | P2 | P1 |  |  |  |  |  |
| T=0 | 25 | 55 | 80 | 105 | 135 | 160 | 185 | 215 | 240 |  |  |  |  |

* The cycle seems to repeat at every t=80 units

**Q5. A system is predominated by periodic tasks and so rate monotonic scheduling (RMS) is proposed**

**as a way to resolve multitask scheduling conflicts. Assume that in a given time span the system has**

**five tasks with parameters as listed below:**

**• Task P1: Processing Time C1 = 20; Period T1 = 90**

**• Task P2: Processing Time C2 = 30; Period T2 = 250**

**• Task P3: Processing Time C3 = 70; Period T3 = 370**

**• Task P4: Processing Time C4 = 50; Period T4 = 330**

**• Task P5: Processing Time C5 = 125; Period T5 = 2000**

**We have seen that the following equation provides an upper bound on the number of tasks that Rate**

**Monotonic Scheduling (RMS) algorithm can successfully schedule: n(2**

**(1/ n)−1) .**

**If RMS is used, analyze whether the tasks can be successfully scheduled as per this equation.**

Sol.

Calculating the value of Total CPU utilization for all the processes

CPU utilization = , where t is the time is takes to execute the process, and p is the period

For task 1:

CPU utilization:

For task 2:

CPU utilization:

For task 3:

CPU utilization:

For task 4:

CPU utilization:

For task 5:

CPU utilization:

Total CPU utilization is sum of all the above values:

Worst case CPU utilization according to RMS value:

Therefore, we cannot schedule the above tasks according to RMS as the total CPU utilization of the above tasks is more than the worst-case CPU utilization according to RMS expression