CS3523: Operating Systems - II

Quiz 1

# Soumi Chakraborty

ES19BTECH11017

# Question 1

| **Process** | A | B | C | D | E |
| --- | --- | --- | --- | --- | --- |
| **Running time** | 15 | 9 | 3 | 6 | 12 |
| **Priorities** | 6 | 3 | 7 | 9 | 4 |

Turnaround time of a process is the time elapsed between the time of submission to the time of completion.

## 1.a Round Robin

Round Robin is like FCFS but with pre-emption.

Hence, with a time quantum of 1 minute, the Gnatt chart will have the following sequence of processes, where each letter represents 1 minute:

ABCDE ABCDE ABCDE ABDE ABDE ABDE ABE ABE ABE AE AE AE A A A

| **Process** | **Turnaround Time (in minutes)** |
| --- | --- |
| A | 45 |
| B | 35 |
| C | 13 |
| D | 26 |
| E | 42 |
| Average | 32.2 |

## 1.b Priority Scheduling

Priority Scheduling is non-preemptive and once a process starts executing, it continues to do so until it finishes.

**Order of execution:**

BEACD

| **Process** | **Turnaround Time (in minutes)** |
| --- | --- |
| A | 9 + 12 + 15 = 36 |
| B | 9 |
| C | 9 + 12 + 15 + 3 = 39 |
| D | 9 + 12 + 15 + 3 + 6 = 45 |
| E | 9 + 12 = 21 |
| Average | 51.8 |

## 1.c FCFS

First come, first serve.

**Order of execution:**

ABCDE

| **Process** | **Turnaround Time (in minutes)** |
| --- | --- |
| A | 15 |
| B | 15 + 9 = 24 |
| C | 15 + 9 + 3 = 27 |
| D | 15 + 9 + 3 + 6 = 33 |
| E | 15 + 9 + 3 + 6 + 12 = 45 |
| Average | 28.8 |

## 1.d Shortest Job First

Runs the process with the first shortest CPU burst first.

**Order of execution:**

CDBEA

| **Process** | **Turnaround Time (in minutes)** |
| --- | --- |
| A | 3 + 6 + 9 + 12 + 15 = 45 |
| B | 3 + 6 + 9 = 18 |
| C | 3 |
| D | 3 + 6 = 9 |
| E | 3 + 6 + 9 + 12 = 30 |
| Average | 21 |

# Question 2

## 2.a

| **Process** | A | B | C |
| --- | --- | --- | --- |
| **Running time** | 100 | 120 | 60 |

**Gnatt Chart**

| **Process** | A | B | C | A | B | C | A | B | A | B | B | B |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **End Time** | 40 | 80 | 120 | 150 | 180 | 200 | 220 | 240 | 250 | 260 | 270 | 280 |
| **Time left for process** | 60 | 80 | 20 | 30 | 50 | 0 | 10 | 30 | 0 | 20 | 10 | 0 |

| **Process** | **Waiting Time (in ms)** |
| --- | --- |
| A | 80 + 50 + 20 = 150 |
| B | 40 + 70 + 40 + 20 = 170 |
| C | 80 + 60 = 140 |
| Average | 153.33 |

## 2.b

**Advantages:**

* Shorter tasks are able to finish running quicker and clear the queue for the larger processes.

**Disadvantages:**

* The constantly switching quantum takes up resources because the scheduler also needs to remember what stage of the round-robin the queue is in
* The smaller the quantum gets, the need for context switching rises because the frequency of shifting between processes rises. Thus, a significant chunk of the quantum time assigned to a process might get eaten up due to context switching whereas it might have finished running with ease in the regular RR algorithm.

# Question 3

| **Process** | A | B | C | D | E |
| --- | --- | --- | --- | --- | --- |
| **Running time** | 9 | 6 | 3 | 5 | X |

Response time for a process is the time elapsed between its submission to the first response. In the case of a non-preemptive scheduling algorithm, the response time is the same as the waiting time.

Thus, having shorter jobs at the front and longer jobs at the back would greatly minimise the response time. The best-case scenario would be to have the jobs arranged in ascending order of their run times. Hence, the jobs should be ordered accordingly:

CDBA, with E positioned in such a way that the ascending order of the run times is maintained.

# Question 4

| **Process** | **Period** | **Run-time** |
| --- | --- | --- |
| P1 | 40 | 25 |
| P2 | 75 | 30 |

## 4.a Rate-monotonic Scheduling

This type of scheduling preempts lower-priority processes if a higher-priority process becomes available to execute. In this type of scheduling, a shorter period implies higher priority and vice-versa. Hence, in this scenario, process P1 has a higher priority than process P2.

CPU utilisation (ti/pi):

**P1**: 25/40 = 62.5%

**P2**: 30/75 = 40%

This gives us a total CPU utilisation as 102.5%, whereas the worst-case CPU utilisation for scheduling two processes with RMS is bounded at 83%, as mentioned in the book. Hence, it is uncertain whether RMS can successfully schedule these processes.

**Gnatt Chart:**

| **Process** | P1 | P2 | P1 | P2 |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **End Time** | 25 | 40 | 65 | 80 |  |  |  |  |

The first deadline for P1 is 40 whereas that for P2 is 75.

* P1 completes its first burst at 25, which meets its first deadline of 40.
* P2 runs from 25 to 40 before it gets preempted by P1. Thus, it still has 15 left in its CPU burst.
* P1 runs again from 40 to 65 thereby meeting its second deadline of 80 as well.
* Now P2 resumes its first CPU burst but is unable to meet its deadline of 75.

Thus, RMS cannot schedule these two processes.

## 4.b Earliest Deadline First

This type of scheduling dynamically assigns priorities to its processes. The earlier the deadline, the higher the priority of the process.

**Gnatt Chart:**

| **Process** | P1 | P2 | P1 | P1 | P2 | P1 |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **End Time** | 25 | 55 | 80 | 105 | 135 | 160 |  |  |

The first deadline for P1 is 40 whereas that for P2 is 75.

* P1 has more priority because of its earlier deadline and thus finishes executing up to 25.
* Now, P2’s deadline is 75 whereas P1’s is 80, so P2 is allowed to finish executing up to the 55 mark and meet its first deadline.
* Now P1 takes precedence again since its deadline of 80 is before P2’s deadline of 100 and executes till the 80 mark.
* P1’s deadline: 120. P2’s deadline: 150. Thus P1 executes again in its new period up to 105.
* P1’s deadline: 160. P2’s deadline: 150. Thus P2 executes up to 105.
* At 150, the start of P2’s new period, P1’s deadline is closer at 160 hence it is allowed to finish executing without being pre-empted.

Thus, EDF is able to schedule these processes without missing deadlines.

# Question 5

| **Process** | **Period (T)** | **Processing-time (C)** | **CPU Utilisation (C/T)** |
| --- | --- | --- | --- |
| P1 | 90 | 20 | 0.22 |
| P2 | 250 | 30 | 0.12 |
| P3 | 370 | 70 | 0.189 |
| P4 | 330 | 50 | 0.152 |
| P5 | 2000 | 125 | 0.0625 |

Thus, the total CPU utilisation approximately amounts to 0.7435 or 74.35%.

Now, upon substituting n = 5 in the formula for worst-case CPU utilisation for scheduling n processes, we get approximately 0.7435. Since 0.7435 <= 0.7435, these tasks are guaranteed to be successfully scheduled using RMS.