­

# HW 1 – Operating System

**Noa Ben David, ID:300950565**

**Yonatan Greenshpan, ID:204266191**

**Question 1:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Process | P2 | P4 | P5 | P1 | P3 |
| Arrival Time | 0 | 2 | 4 | 0 | 0 |
| CPU | 4 | 1 | 1 | 5 | 6 |
| Completion time | 4 | 5 | 6 | 11 | 17 |
| TA | 4 | 3 | 2 | 11 | 17 |

Therefore, the Avg turnaround = (4 + 3 + 2+11+17)/5 = 7.4

**Question 2:**

The best optimization metric out of the four given metrics; response time, fairness, throughput and  
turnaround, for a modern laptop would depend on the specific use case and requirements. For most domestic and personal usage the best optimization metric is Respond time - The time from the moment a user creates a request (for example, presses a button) until the request is executed

**Question 3:**

According to the provided Python implementation code, a possible problem is that a thread acquires the mutex and then there is no way for realising it. Consider the following example:

**Initial state:**

semaphore = Semaphore(0)

**Steps:**

1. semaphore.Down() #mutex is locked

**Final non-legitimate state:**

After one step (Down/Up) the mutex is locked and no one can get to line 25 or 18 for releasing the mutex and all the threads will hang on line 12 or 21 and therefore we get - Deadlock

**Question 4:**

**Time point 1:**

In time t=0 there are three processes so we should calculate the probability according to the priority scheduling

* Process 1:
* Process 2:
* Process 3:

Therefore, Time Point 1 ends after 10 CPU seconds when Process 1 completes.

**Time point 2:**

Process 1 finished on second 10 therfore will calculate based on the results on time point 1 and then the remain probabilities of process 2 and process 3

Remain probabilities:

* Process 2:
* Process 3:

Therefore, Time Point 2 ends after 17.6 CPU seconds when Process 2 completes.

**Time point 3:**

Process 2 finished on second 17.6 therfore will calculate based on the results on time point 2 and then the remain probability of process 3 an:

* Process 3:

Therefore, Time Point 3 ends after 5.083 CPU seconds when Process 3 completes.

|  |  |  |  |
| --- | --- | --- | --- |
|  | t1:0-10 CPU sec | t2:10-27.6 CPU sec | t3:27.6- 32.083CPU sec |
|  | 0.5 | - | - |
|  | 0.375 | 0.46875 | - |
|  | 0.09375 | 0.46875 | 0.75 |

**Question 5 :**

1. Based on the Pythonic implementation code, consider the following synchronizing problem:

**Initial state:**

L1 = [1,2,3]

L2 = [1,4,5]

Assuming both threads start exactly in the same time.

**Steps:**

1. Both threads calls MergeInto at the same time.
2. Both gets length 0 for lTarget
3. Both of the threads add the number 1 to the list

**Final non-legitimate state:**

The number 1 is added twise to the list hence the list contains duplications.

1. The critical section is from line 05 to 11 since the threads asks for the length of lTarget only once and if another thread is adding more number to the list the original thread doesn’t iterate on them.

**Question 6:**

**Time point 1:**

Process A and B arrive on second t0 = 0sec so we should check if they are still alive on t1 = 1sec.

* Process A:
* Process A finished on second 2.
* Process B:
* Process B finished on second 4

CPU Utilization:

**Time point 3:**

Process A finished on second 2 therfore will not be alive on second 3.

Process B finished on second 4 therfore will be alive on second 3

Process C arrived on second 2 but its is 6 therefore will be alive on second 3.

CPU Utilization:

**Time point 5:**

Process C arrive on second 2 but its is 6 therefore will be alive on second 5. Process D arrive on second 4 but its is 3 therefore will be alive on second 5.

Process E arrive on second 4 but its is 4 therefore will be alive on second 5.

CPU Utilization:

Let’s summarize the results:

|  |  |
| --- | --- |
| Time point | CPU Utilization |
| t1 = 1sec | 0.75 |
| t2 = 3sec | 0.75 |
| t3 = 5sec | 0.94 |

**Question 7:**

1. Round robin – quantum length of 3 and 4,8,16,32,64

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 |
| P1 | R | R | R | W | W | W | W | W | W | W | W | W | W | W | W | R |  |
| P2 | W | W | W | R | R | R | W | W | W | W | W | W | W | W | W | W | R |
| P3 | W | W | W | W | W | W | R | R | R | W | W | W | W | W | W | W | W |
| P4 | W | W | W | W | W | W | W | W | W | R | R | R | W | W | W | W | W |
| P5 | W | W | W | W | W | W | W | W | W | W | W | W | R | R | R | W | W |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 |
| P1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 | R | R | W | W | W | W | W | W | W | W | W | R | R |  |  |  |  |
| P3 | W | W | R | R | R | W | W | W | W | W | W | W | W | R | R | R | W |
| P4 | W | W | W | W | W | R | R | R | W | W | W | W | W | W | W | W | R |
| P5 | W | W | W | W | W | W | W | W | R | R | R | W | W | W | W | W | W |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
| P1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 | W | W | W | W | W | R | R | R | W | W | W | W | W | W | R | R | R |
| P4 | R | R | W | W | W | W | W | W | R | R | R | W | W | W | W | W | W |
| P5 | W | W | R | R | R | W | W | W | W | W | W | R | R | R | W | W | W |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 51 | 52 | 53 | 54 | 55 | 56 | 57 | 58 | 59 | 60 | 61 | 62 | 63 | 64 | 65 | 66 | 67 |
| P1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 | W | W | W | W | W | W | R |  |  |  |  |  |  |  |  |  |  |
| P4 | R | R | R | W | W | W | W | R | R | R | W | W | W | R | R | R | W |
| P5 | W | W | W | R | R | R | W | W | W | W | R | R | R | W | W | W | R |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 68 | 69 | 70 | 71 | 72 | 73 | 74 | 75 | 76 | 77 | 78 | 79 | 80 | 81 | 82 | 83 | 84 |
| P1 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| P4 | W | W | R | R | R | W | W | W | R | R | R | W | W | W | R | R | R |
| P5 | R | R | W | W | W | R | R | R | W | W | W | R | R | R | W | W | W |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 85 | 86 | 87 | 88 | 89 | 90-121 |
| P1 |  |  |  |  |  |  |
| P2 |  |  |  |  |  |  |
| P3 |  |  |  |  |  |  |
| P4 | W | W | W | R | R |  |
| P5 | R | R | R | W | W | R |

Let’s summarize the results:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| P1 | 0-3 | 15 |  |  |  |  |  |  |  |  |  |
| P2 | 3-5 | 16-18 | 28-29 |  |  |  |  |  |  |  |  |
| P3 | 6-8 | 19-21 | 30-32 | 39-41 | 48-50 | 57 |  |  |  |  |  |
| P4 | 9-11 | 22-24 | 33-35 | 42-44 | 51-53 | 58-60 | 64-66 | 70-72 | 76-78 | 82-84 | 88-89 |
| P5 | 12-14 | 25-27 | 36-38 | 45-47 | 54-56 | 61-63 | 67-69 | 73-75 | 79-81 | 85-87 | 90-121 |

P1 turnaround – 16

P2 turnaround – 30

P3 turnaround -58

P4 turnaround -90

P5 turnaround -122

System average turnaround – 63.2 seconds

Maximal starvetion: 12 seconds

1. Shortest job first (non-preemptive)
   1. Average turn-around: (4 + 12 + 28 + 60 + 124)/5 = 45 seconds
   2. Maximal starvation is of p5 which needs to wait for all other process to finish, hence max starvation = 60 seconds

**Question 8 :**

**Initial state:**

a[0]=False

a[1]=False

**Steps:**

1. T0 starts execute line 1 a[0]=true then preempt.
2. T1 starts and go throuth line 3 <cs> then preempt
3. Because a[1] set to true by T1, T0 continue to <CS>

**Final non-legitimate state:**

Both threads inside the critical section at the same time and non of them will ever go out from the loop, because of the race condition.

**Question 9 :**

* Start - > Ready: The process begins and ready for the scheduler to allocate the needed resources.
* Ready -> Running : The process set for running and gets resources untill it ends or blocked
* Running -> Blocked: The process is blocked by another process or by the OS and waits for release
* Running -> Ready: The process is preempted and ready for continue running
* Blocked -> Ready: The process is blocked and ready for continue running and after realese
* Running -> Done: The running process complets all his tasks