# Exercise 1 – Operating System

## Guidelines:

### Submissions:

* The assignment must be submitted by 27.3.22 at 23:59.
* The solution will be submitted through the model in a PDF file only.
* The solution file should be written in a word processor(e.g., Word or ) or in a clear handwritten.
* The submission is in pairs only, the ID number of each one must be stated in the solution.
* The file name will be of the form EX1\_ID1\_ID2.pdf.
* Only one group member is required to upload the solution.
* Any deviation from these principals, without a written approval of the course staff, will resolve in reducing points from the assignment.

### Questions and solutions:

* Read the questions carefully and answer exactly what you were asked to do.
* The style of the solutions should be in line with what is learned in the course.
* Do not copy solutions! Copying solutions will resolve of a zero grade for the task.
* Administrative emails on work-related questions should be sent to Shlomit.harush@post.idc.ac.il (no technical questions).
* A forum for technical question is available in the Model, please check if there is already the same question you want to ask.
* For clarifications, intentions or any other assistance, reception hours are also available by appointment.

### Checking:

* Only predetermine questions are graded. Empty answers will resolve in reducing points from the assignment.

Question 1:

In a batch system, there are the following process:

|  |  |  |
| --- | --- | --- |
| CPU time | Arrival time | Process |
| 5 | 0 | P1 |
| 4 | 0 | P2 |
| 6 | 0 | P3 |
| 1 | 2 | P4 |
| 1 | 4 | P5 |

The system uses a shortest job first which is non-preemptive.

Compute average turnaround.

Question 2:

From the following optimization metrics: response time, fairness, throughput, turnaround. What is the best one for modern laptop? Please provide examples.

Question 3:

1. public class Semaphore{
2. private Mutex m;
3. private int counter;
4. private List<Thread> sleeping;
5. public Semaphore(int n){
6. m = new Mutex();
7. counter = n;
8. sleeping = new List<Thread>();
9. }
10. public void Down(){
11. m.lock();
12. if(counter == 0){
13. sleeping.Add(Thread.CurrentThread);
14. Thread.Sleep();
15. }
16. counter--;
17. m.unlock();
18. }
19. public void Up(){
20. m.lock();
21. counter++;
22. if(sleeping.Count > 0)
23. sleeping.RemoveFirst().Wakeup();
24. m.unlock();
25. }
26. }

Python implementation:

1. Class Semaphore:
2. m = None # Mutex pointer
3. counter = 0
4. sleeping = None # List of Threads
5. def Semaphore(self, n: int)-> None:
6. m = Mutex()
7. counter = n
8. sleeping = []
9. def Down(self) -> None:
10. m.acquire() # m.lock()
11. if counter == 0:
12. sleeping.append(Thread.CurrentThread)
13. Thread.Sleep()
15. counter--
16. m.release # m.unlock()
18. def Up(self) -> None:
19. m.acquire() # m.lock()
20. counter += 1
21. if len(sleeping) > 0:
22. sleeping.pop().Wakeup()
23. m.release # m.unlock()

Find one synchronizing problem in the above code, please provide the initial state, actions, and non-legitimate final state.

Question 4:

In time t=0 there are three processes, when it is known that:

Process 1, Priority 1, 50% busy in writing to a printer.

Process 2, Priority 2, 25% busy in writing to disk C.

Process 3, Priority 3, 25% busy in reading from disk D.

Process 1 is ending after 5 CPU seconds.

Process 2 is ending after 12 CPU seconds.

Process 3 is ending after 13 CPU seconds.

In addition, when there are three processes the OS is scheduling the processes via priority queue (highest priority is selected, i.e., 1 is better than3),but when there are two processes the OS is scheduling the processes via random selection (i.e., selects randomly a process)

Please provide a table of CPU usage probabilities between the processes.

|  |  |  |  |
| --- | --- | --- | --- |
| Process \ Time | Provide here the different timelines and probabilities | … | … |
| P1 |  |  |  |
| P2 |  |  |  |
| P3 |  |  |  |

Question 5:

Given a list of numbers, and assume we want to merge them into one list that is not contain duplicate numbers, in a parallel way. For this we have the following code:

01 void MergeInto(List<int> lSource, List<int> lTarget)

02 {

03 int i,j;

04 for (i = 0; i < lSource.Count; i++)

05 {

06 bool bExists = false;

07 int number;

08 number= lSource[i];

09 for (j = 0; j < lTarget.Count && !bExists; j++)

10 {

11 if (number == lTarget[j])

12 bExists = true;

13 }

14 if (!bExists)

15 {

16 lTarget.Add(number);//adds to the end of the list

17 }

18 }

19 }

Pythonic implementation:

01 def MergeInto(lSource: list, lTarget: list) -> None:

02 for i in range(len(lSource)):

03 bExists = False

04 number= lSource[i]

05 for j in range(len(lTarget.Count)):

06 if bExists:

07 break

08 if number == lTarget[j]:

09 bExists = True

10 if not bExists:

11 lTarget.append(number) # adds to the end of the list

While number of Threads are running the code, all the threads get the same destination list (ITarget) and every Thread is getting a different source list (ISource). In this question, you can assume that the Class List is defend any insertion well, meaning there is no race inside the code of insertion (the Add function)

1. Find one synchronizing problem in the above code, please provide the initial state, actions, and non-legitimate final state.
2. Which lines are the critical section for your stated race?

Question 6:

In a system there are the given processes:

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Arrival time | CPU time | Blocked probability |
| A | 0 | 0.75 | 50% |
| B | 0 | 1.5 | 50% |
| C | 2 | 6 | 50% |
| D | 4 | 3 | 40% |
| E | 4 | 4 | 30% |

Write the CPU utilization in every time point: t1 = 1sec , t2 = 3sec, t3 = 5sec.

Question 7:

Given a batch system with five processes that all arrive in the same time t=0.

Every process is a CPU-bound with running time of 4,8,16,32,64.

Compute the average turn-around and the maximal starvation

for each of the following scheduling algorithms:

1. Round robin with quantum length of 3.
2. Shortest job first (non-preemptive).

Question 8:

Given the following solution to protect a critical section, assume only two threads are in the system.

|  |  |
| --- | --- |
| T0:   1. a[0] = true 2. While(a[1]){ 3. <CS> 4. } 5. a[0] = false; | T1:   1. While(a[0]){ 2. a[1] = true 3. <CS> 4. } 5. a[1] = false; |

Find one synchronizing problem in the above code, please provide the initial state, actions, and non-legitimate final state.

Question 9:

Given an interactive system which is preemptive, running the scheduling algorithm Round Robin(RR). Write all the possible connections between the process states in the given system. For every connection, write a concrete example.