EOPSY LAB3

Seniv Volodymyr 309358

Theory:

The main goal of this task is to perform and study the scheduling operation in the operating system with the help of MOSS scheduling simulator.

Copied from the html file:

The scheduling simulator illustrates the behavior of scheduling algorithms against a simulated mix of process loads. The user can specify the number of processes, the mean and standard deviation for compute time and I/O blocking time for each process, and the duration of the simulation. At the end of the simulation a statistical summary is presented.

CPU Schedular: Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is caried out by the short-term scheduler (or CPU scheduler). The scheduler selects a process from the processes in memory that are ready to be executed and allocates the CPU to that process.

Dispatcher: The dispatcher is the module that gives control of the CPU of the process selected by the short-term scheduler. The time is taken for the dispatcher to stop one process and start another running is known as the dispatch latency.

CPU-scheduling decisions may take place under the following four circumstances:

1. When a process switches from the **running state** to the **waiting state**
2. When a process switches from the running state to the ready state (for example, when an interrupt occurs).

3. When a process switches from the waiting state to the ready state (for example, at completion of I/O)
4. When a process terminates

For situations 1 and 4, there is no choice in terms of scheduling. A new process (if one exists in the ready queue) must be selected for execution. However, there is a choice for situations 2 and 3.

When scheduling takes place only under circumstances 1 and 4, we say that the scheduling scheme is non preemptive or cooperative; otherwise, it is preemptive.

Simulator:

So, after the theory we can verify, what the simulator does.

The simulator runs concrete number of processes provided by the user in serial, one at a time. As it is described in the meaning of non-preemptive scheduling the simulator works in this way that one process is running, and the others are waiting. The CPU resources are given to this process until it terminates or goes to the waiting state. So, this simulator follows the ide that the process cannot be interrupted or ‘robbed’ with the CPU resources until it terminates of switches to the waiting state.

All the simulations were performed with the following parameters:

Mean deviation – 2000 milliseconds

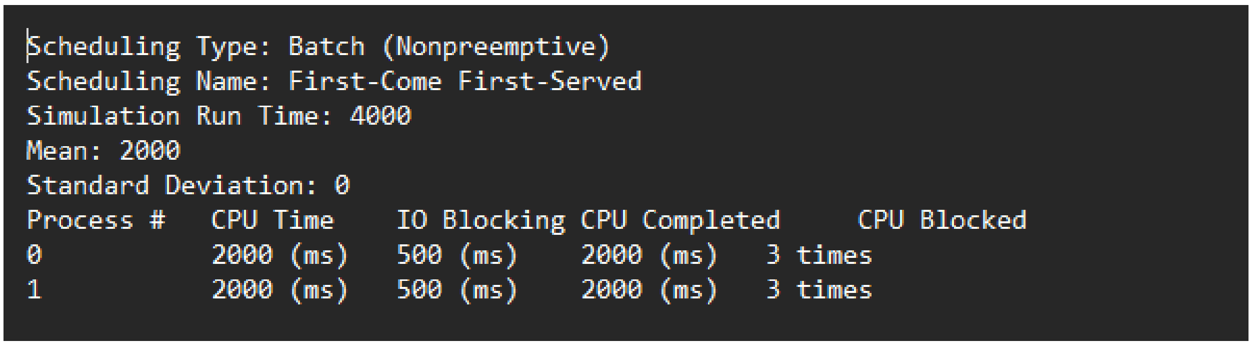
Standard deviation – 0

I/O block every 500 milliseconds

Simulation time: 10000 milliseconds

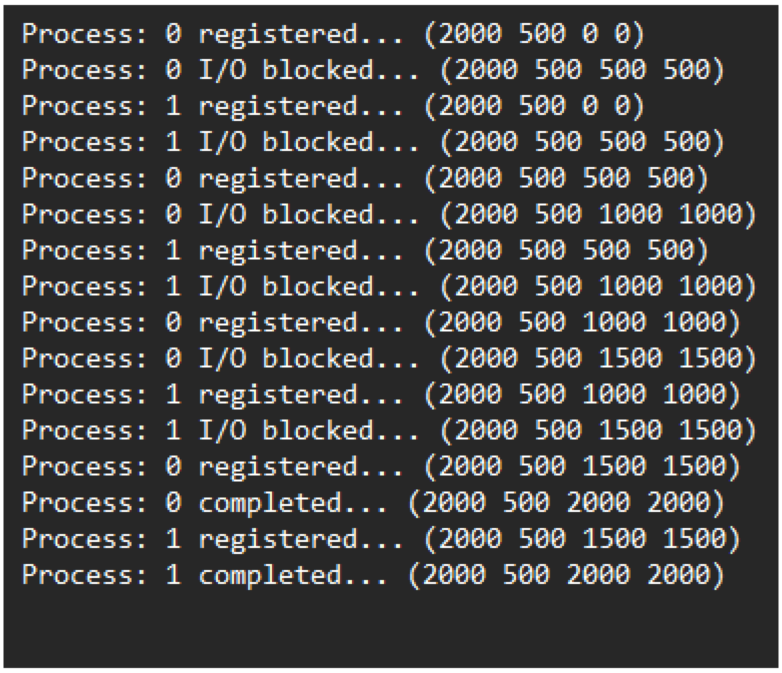
Two processes:

Summary-Result:



```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 4000
Mean: 2000
Standard Deviation: 0
Process #   CPU Time    IO Blocking CPU Completed     CPU Blocked
0           2000 (ms)   500 (ms)    2000 (ms)   3 times
1           2000 (ms)   500 (ms)    2000 (ms)   3 times
```

Summary-processes:



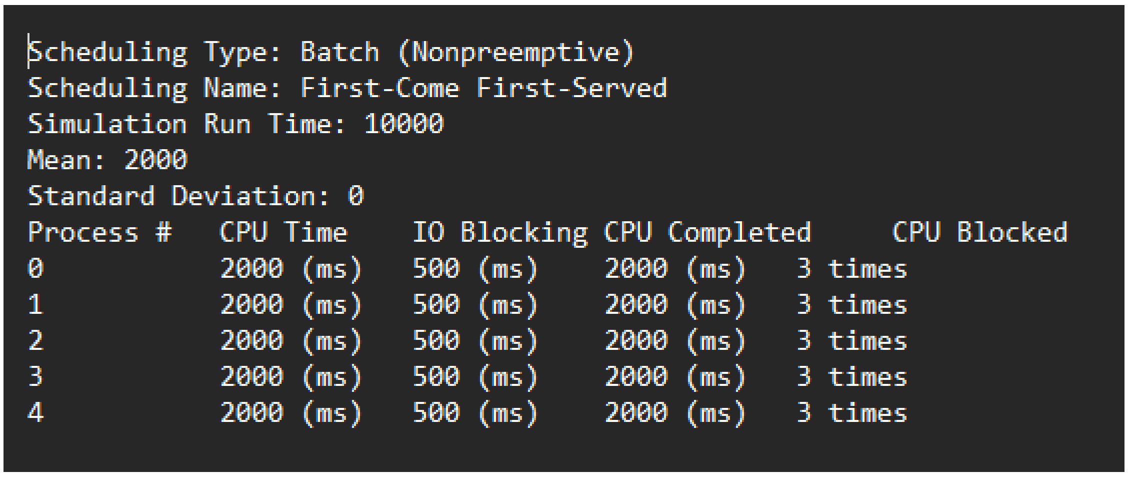
```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
```

In this task we had to do the simulation with 2 processes. Ass we can see the process 0 goes first, it gets the resources needed, runs and after the I/O block happens it goes to the waiting state. After the process 0 moved to the waiting state the resources should be given to the other process (by the operation of non-preemptive scheduling scheme), so the process 1 is assigned in the CPU by the OS. While the process does its business, meanwhile the process 0 changes his state to the ready state and prepares to its execution. After 500 ms process 1 has I/O interruption, switches to the waiting state and the now OS gives process 0 the needed resources. The main idea of this is that there was no possibility for process 0 to be handled, while the process 1 was executing. That is because of the non-preemptive schedule schema. The process 0 must wait for process one to change the state to waiting state.

The screenshots show that the processes were blocked for 3 times and the task of each process takes 2000 ms and because the simulation runtime is 10000 ms the processes must be executed till the time is over.

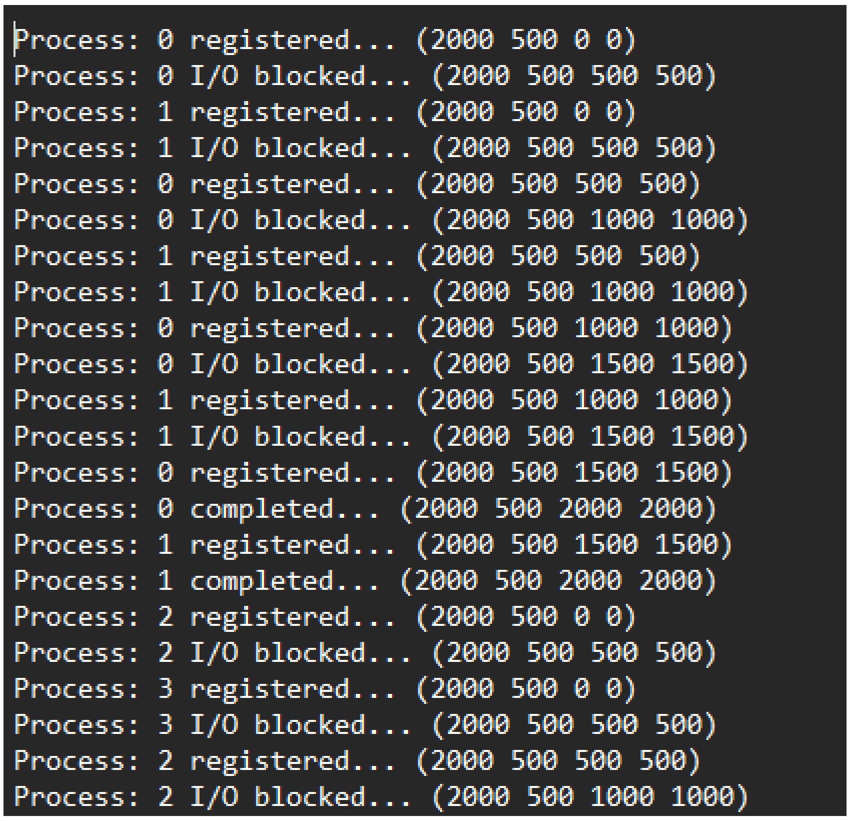
Five processes:

Summary-Result:

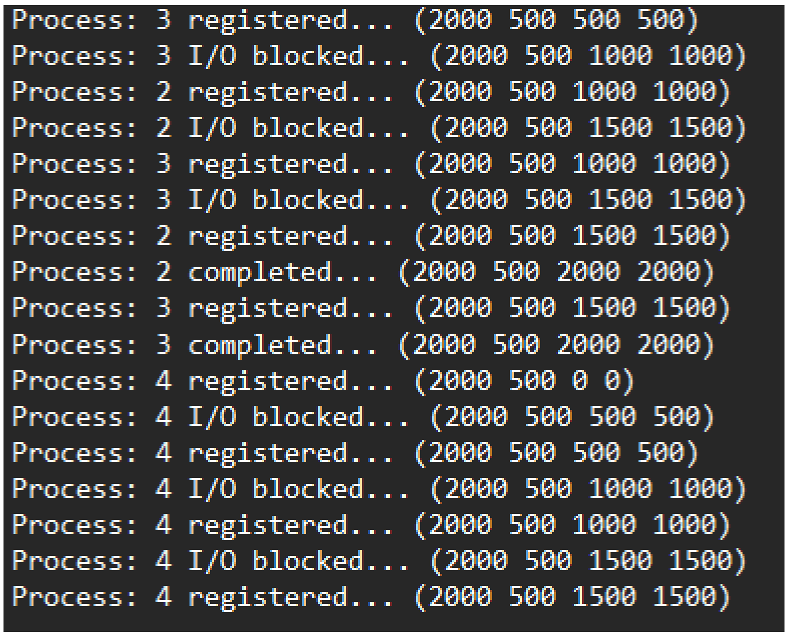


```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process #   CPU Time    IO Blocking CPU Completed     CPU Blocked
0           2000 (ms)   500 (ms)    2000 (ms)   3 times
1           2000 (ms)   500 (ms)    2000 (ms)   3 times
2           2000 (ms)   500 (ms)    2000 (ms)   3 times
3           2000 (ms)   500 (ms)    2000 (ms)   3 times
4           2000 (ms)   500 (ms)    2000 (ms)   3 times
```

Summary-processes:



```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
Process: 2 registered... (2000 500 0 0)
Process: 2 I/O blocked... (2000 500 500 500)
Process: 3 registered... (2000 500 0 0)
Process: 3 I/O blocked... (2000 500 500 500)
Process: 2 registered... (2000 500 500 500)
Process: 2 I/O blocked... (2000 500 1000 1000)
```



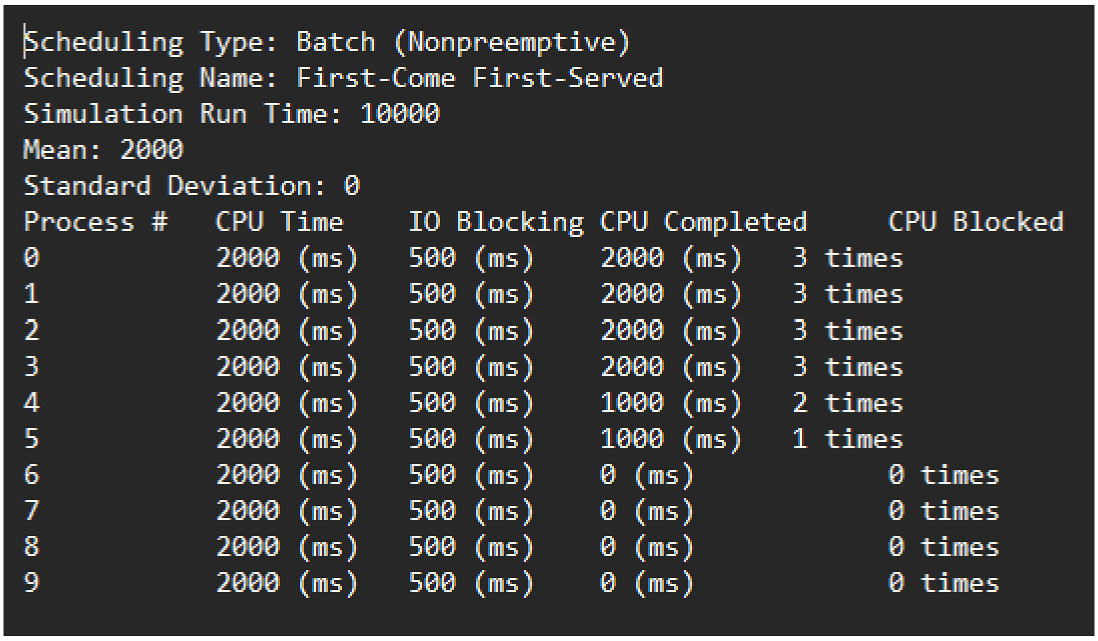
```
Process: 3 registered... (2000 500 500 500)
Process: 3 I/O blocked... (2000 500 1000 1000)
Process: 2 registered... (2000 500 1000 1000)
Process: 2 I/O blocked... (2000 500 1500 1500)
Process: 3 registered... (2000 500 1000 1000)
Process: 3 I/O blocked... (2000 500 1500 1500)
Process: 2 registered... (2000 500 1500 1500)
Process: 2 completed... (2000 500 2000 2000)
Process: 3 registered... (2000 500 1500 1500)
Process: 3 completed... (2000 500 2000 2000)
Process: 4 registered... (2000 500 0 0)
Process: 4 I/O blocked... (2000 500 500 500)
Process: 4 registered... (2000 500 500 500)
Process: 4 I/O blocked... (2000 500 1000 1000)
Process: 4 registered... (2000 500 1000 1000)
Process: 4 I/O blocked... (2000 500 1500 1500)
Process: 4 registered... (2000 500 1500 1500)
```

In this task we had to perform a simulation with 5 processes. Since the time for one process to be executed is 2000 ms we have a situation that the simulation run time and the time for all tasks to be done are equal.

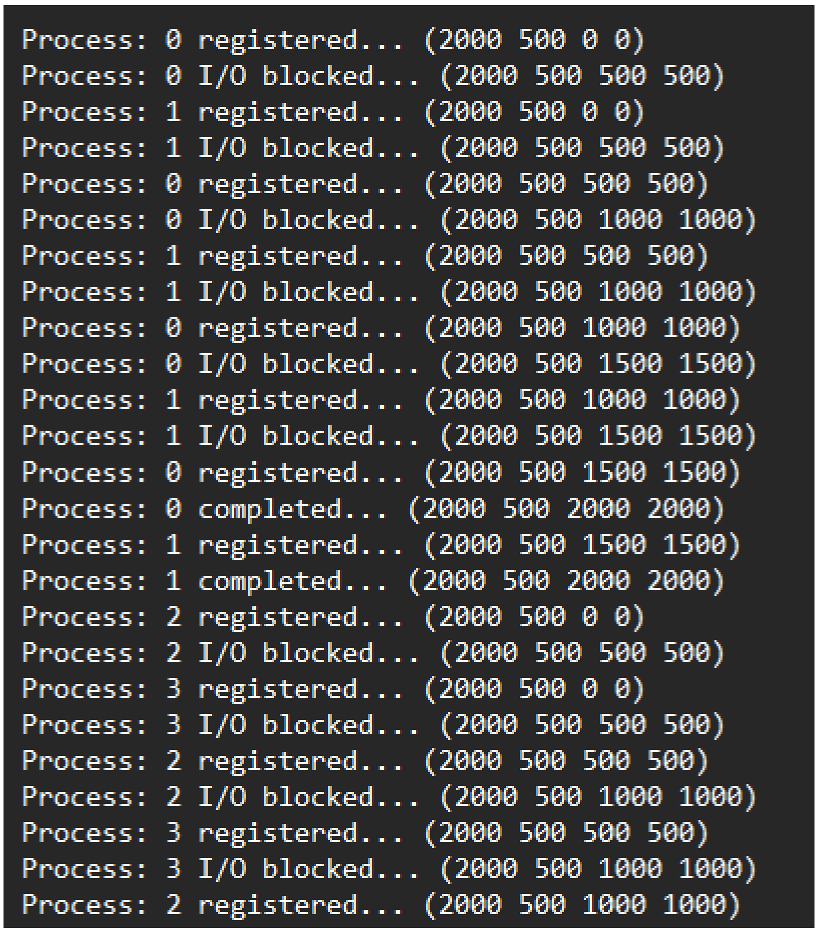
From the screenshots we can see that the processes are working in pairs. It means that when we are executing the process 0 and then it goes to the waiting state, only process 1 is executing and none of the processes are place in between. Also, it is visible that processes are going in pairs, but the process 4 is alone. So, process 0 and 1, 2 and 3 will change each other in the running state, and the process 4 will go but itself in a cycle, because there is no other process to give CPU by the OS.

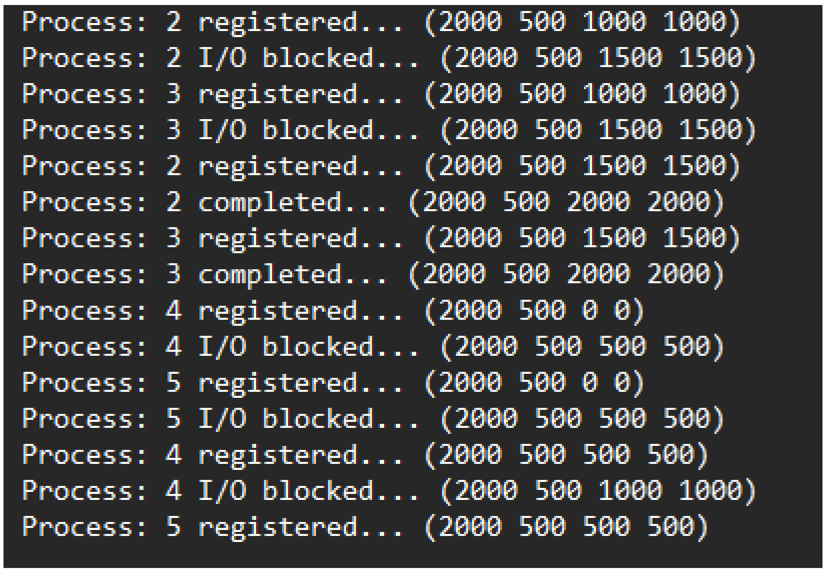
Ten processes:

Summary-Result:


```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process #   CPU Time    IO Blocking CPU Completed     CPU Blocked
0           2000 (ms)   500 (ms)    2000 (ms)   3 times
1           2000 (ms)   500 (ms)    2000 (ms)   3 times
2           2000 (ms)   500 (ms)    2000 (ms)   3 times
3           2000 (ms)   500 (ms)    2000 (ms)   3 times
4           2000 (ms)   500 (ms)    1000 (ms)   2 times
5           2000 (ms)   500 (ms)    1000 (ms)   1 times
6           2000 (ms)   500 (ms)    0 (ms)            0 times
7           2000 (ms)   500 (ms)    0 (ms)            0 times
8           2000 (ms)   500 (ms)    0 (ms)            0 times
9           2000 (ms)   500 (ms)    0 (ms)            0 times
```

Summary-processes:


```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500 500)
Process: 1 I/O blocked... (2000 500 1000 1000)
Process: 0 registered... (2000 500 1000 1000)
Process: 0 I/O blocked... (2000 500 1500 1500)
Process: 1 registered... (2000 500 1000 1000)
Process: 1 I/O blocked... (2000 500 1500 1500)
Process: 0 registered... (2000 500 1500 1500)
Process: 0 completed... (2000 500 2000 2000)
Process: 1 registered... (2000 500 1500 1500)
Process: 1 completed... (2000 500 2000 2000)
Process: 2 registered... (2000 500 0 0)
Process: 2 I/O blocked... (2000 500 500 500)
Process: 3 registered... (2000 500 0 0)
Process: 3 I/O blocked... (2000 500 500 500)
Process: 2 registered... (2000 500 500 500)
Process: 2 I/O blocked... (2000 500 1000 1000)
Process: 3 registered... (2000 500 500 500)
Process: 3 I/O blocked... (2000 500 1000 1000)
Process: 2 registered... (2000 500 1000 1000)
```



```
Process: 2 registered... (2000 500 1000 1000)
Process: 2 I/O blocked... (2000 500 1500 1500)
Process: 3 registered... (2000 500 1000 1000)
Process: 3 I/O blocked... (2000 500 1500 1500)
Process: 2 registered... (2000 500 1500 1500)
Process: 2 completed... (2000 500 2000 2000)
Process: 3 registered... (2000 500 1500 1500)
Process: 3 completed... (2000 500 2000 2000)
Process: 4 registered... (2000 500 0 0)
Process: 4 I/O blocked... (2000 500 500 500)
Process: 5 registered... (2000 500 0 0)
Process: 5 I/O blocked... (2000 500 500 500)
Process: 4 registered... (2000 500 500 500)
Process: 4 I/O blocked... (2000 500 1000 1000)
Process: 5 registered... (2000 500 500 500)
```

In this case we are dealing with 10 processes. Since the time for one process to be executed is 2000 ms we exceed the runtime limit for the simulation by 2 (10000 ms). So, as we can see from the screenshots half of the processes are not going to be executed. In this simulation process 4 got his pair (process 5) and they were blocked 2 and 1 time respectively. Also, we can see that the process 4 and 5 were not completed and they were just terminated.