Sebastian Grzelak, EOPSY, Lab 3, 05.05.2020

For 2 processes:

Summary-Results:

```
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
```

Atributes explanation:

Process # => this is the number of process

CPU Time => time which was distriute to process work

IO Blocking => this is the time of process working and after it the process is blocked (500ms in this case)

CPU Completed: this attriutes give information how much time was spend on process work

For every process the 'CPU Blocked' is equal 3, because the lifesycle of the process was like that:

1) process registered, then after 500ms I/O blocked - this is the first 'Cpu Blocked' => 500ms work on that process

2) after some time when the CPU backed to this process we had once again:

process registered, then after 500ms I/O blocked - this is the second 'Cpu Blocked' => 1000ms work on that process

3) the third point is the same as the second one, but we have the third 'Cpu Blocked' => 1500ms work on that process

4) the last point is that CPU worked on the process already 1500ms, so only 500ms left. In that fourth point after that last 500ms the process is completed and whole time work on that process is 2000ms.

In this example we created only two processes and the total amout of time was 4000ms. So this is the reason why 'Simulation Run Time' is not 10000ms, so the program was terminated earlier after the second process was finished.

Summary-Processes: Explanation =>

Process: 0 registered... (2000 500 0 0) **=> process 0 is in use, STATE: registered**

Process: 0 I/O blocked... (2000 500 500 500) **=> process 0 after 500ms is blocked, STATE: I/O blocked, totalTime: 500ms**

Process: 1 registered... (2000 500 0 0) **=> process 1 is in use right now, because process 0 is blocked, STATE: registered**

Process: 1 I/O blocked... (2000 500 500 500) **=> process 1 after 500ms is blocked, STATE: I/O blocked, totalTime: 500ms**

Process: 0 registered... (2000 500 500 500) **=> process 0 is in use right now, because process 1 is blocked, STATE: registered**

Process: 0 I/O blocked... (2000 500 1000 1000) **=> process 0 after 500ms is blocked, STATE: I/O blocked, totalTime: 1000ms**

Process: 1 registered... (2000 500 500 500) **=> process 1 is in use right now, because process 0 is blocked, STATE: registered**

Process: 1 I/O blocked... (2000 500 1000 1000) **=> process 1 after 500ms is blocked, STATE: I/O blocked, totalTime: 1000ms**

Process: 0 registered... (2000 500 1000 1000) **=> process 0 is in use right now, because process 1 is blocked, STATE: registered**

Process: 0 I/O blocked... (2000 500 1500 1500) **=> process 0 after 500ms is blocked, STATE: I/O blocked, totalTime: 1500ms**

Process: 1 registered... (2000 500 1000 1000) **=> process 1 is in use right now, because process 0 is blocked, STATE: registered**

Process: 1 I/O blocked... (2000 500 1500 1500) **=> process 1 after 500ms is blocked, STATE: I/O blocked, totalTime: 1500ms**

Process: 0 registered... (2000 500 1500 1500) **=> process 0 is in use right now, because process 1 is blocked, STATE: registered**

Process: 0 completed... (2000 500 2000 2000) **=> process 0 after 500ms is completed, because totalTime: 2000ms is equal CPU time**

Process: 1 registered... (2000 500 1500 1500) **=> process 1 is in use right now, because process 0 is blocked, STATE: registered**

Process: 1 completed... (2000 500 2000 2000) **=> process 1 after 500ms is completed, because totalTime: 2000ms is equal CPU time**

For 5 processes:

Summary-Results:

```
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
```

In this case we have the same lifesycle for all the 5 processes like in the previouse example for 2 processes.

The execution time: 10000ms was perfectly distributed to all processes, 2000ms per each.

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 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 case we can observe almost the same thins as in previous example for 2 processes. Here we have 5 processes which had the same period of time (2000ms) to work on it.

Some my conclusions:

1) Why when the process 1 is blocked we backed to the process 0 instead of create another process 2.

2)Why process 4 has no state completed?

Explanation:

1) We can also notify that the processes are executed somehow in pairs. But the reason is the algorithm: 'First-Come First-Served', it means that the priority has the proccess with the lower index / first created. So if the process 1 is blocked then the priority has process 0 if still has some avaliable time for execution.

2) The reason is that there was some small waste of time during the switching between processess. And that time 500ms on process work is rounded. So when the execution program has 10000ms we have lost some time for the process switching for instance. 50ms (this is just my random value). Because of this we do not have enought time left for whole process 4 execution. It means that program finished before the process 4 was completed.

For 10 processes:

```
Summary-Results:
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
```

We can observe the similar things like in example for 5 processes. There are small differents in case of the processes: 4 and 5.

The time of the 'CPU Completed' for processes: 4 and 5 is the same equal 1000ms. The question is why 1000ms but not 2000ms?

The reason is that after the execution of the process 3 left only 2000ms avaliable in whole program time execution. That left time: 2000ms was distribute for those two processes: 4 and 5 in equal way (1000ms). The 'CPU Blocked' for processes: 1-3 is the same as in the previous example and this is quite obvious. Why there is difference in 'CPU Blocked' for processes: 4 and 5, lets look into the Summary-Processes file:

Below is the part which is responsible for processes: 4 and 5:

Summary-Processes:

```
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)
```

We can see that after the process: 3 was finished the process: 4 was created. Process: 4 has been two times in I/O blocked state thats the reason why we have value: 2 times in 'CPU Blocked' field. As a results of this two visits in that states the 'CPU Completed' time is 1000ms.

The same is for the process 5 but the value of 'CPU Blocked' field is 1 times. Because first was the timeout of the whole program execution before the process get into STATE: I/O blocked.

The reason might be the same as in the previous example for 5 processess.

For the processes: 6-9 there was no execution of them because the timeout of the program appears first. So it is normal scenarion that the 'CPU Completed' and 'CPU Blocked' field in those cases is 0.

There is no sense to explain the rest of the Summary-Processes file, because rest part of that file is almost identical that in previous example.