EOPSY
Laboratory 3

Bartłomiej Mastej
album number: 300172

May 4, 2021

For all simulations the following parameters were set:

```
Scheduling Type: Batch (Nonpreemptive)
Scheduling Name: First-Come First-Served
Simulation Run Time: 10000
Mean: 2000
Standard Deviation: 0
Process I/O blocking: 500 for each process
```

1 Part 1

Gain output:

1. 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)
```

```
         Summary-Processes (END)
```

2. Summary-Results

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 Part 2

Gain output:

1. 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)
```

2. Summary-Results

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

3 Part 3

Gain output:

1. Summary-Processes:

```
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: 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)
```

2. Summary-Results

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

4   Observations

All important observations are written below:
Due to the fact that the meantime for 2 processes is 2000 (ms), hence they finish after 4000 (ms) − > it will not finish as expected after 10000 (ms). Moreover, The processes are being executed in the following manner:

1. Process #0 is registered

2. Process #0 is blocked

3. Process #1 is registered

4. Process #1 is blocked

This is being repeated 3 times.

Further, for 5 processes the meantime is still 500 (ms), however, this time the runtime 10000 (ms) will be finished. The execution of 5 processes is very similar to the execution of just 2 − > firstly, the same algorithm as for the 2 processes is being executed for processes #0 and #1, after it is being done the same is being performed for processes #2 and #3. The process #4 is being registered and blocked 3 times in the row. This is clearly the First-come first-served algorithm as the processes are being executed in the order they come - after one process is block, then executes the one which was blocked earlier or if there is no previous process to execute, then the next process is executed. This can be observed on the figure 1 when the process is blocked, then to keep the CPU busy next process is being executed. When the block is finished, then the process is in the ready state, where it waits for the currently running process to be blocked/finished and if that happens the process changes from the ready state to the run state (the loop repeats).

When it comes to 10 processes the first 4 of them are being executed as all previous − > they are running for 2000 (ms) and are blocked 3 times.

Nevertheless, when it comes to next to processes they are completed after 1000 (ms) and the process #4 is blocked 2 times and process #5 is executed just one time. Furhter process are not being executed. The only difference between 5 process and 10 process execution is that the algorithm of execution is being performed 3 times − > for processes pairs: #0&#1, #2&#3, #4&#5. For the last pair the algorithm is executed only once and further only for #4.

5 Conclusions

On the output of the Summary-Processes it can be clearly seen that the First-Come First-Served scheduling system was applied - the processes are being executed in order they are being created. After one process is being blocked, the previous one is being executed next. If no previous process is pending, then the next one is being executed. One can see that the CPU was busy for the all time of the execution (10000 ms). What is more, having 10 processes to be executed (part 3) only 6 of them are being served, hence this method might not always be the most optimal for all applications, nevertheless, the most of them are being executed few times (most of them 3). On the basis of the average time and the runtime the number of processes that will be executed might vary. One can see that there is Nonpreemptive scheduling type used as there is a clear *block*, and while being blocked the process can not be further executed until another process is stoped/blocked. This can be seen on the figure 1 - the process when is being block it is kept in the state *blocked/wait* and after the certain time of block is passed, then the process is being proceed to the *ready* state which clearly indicates that this is non-preemptive scheduling type; for the preemptive scheduling type the process goes directly from the *run* state to the *ready* state.
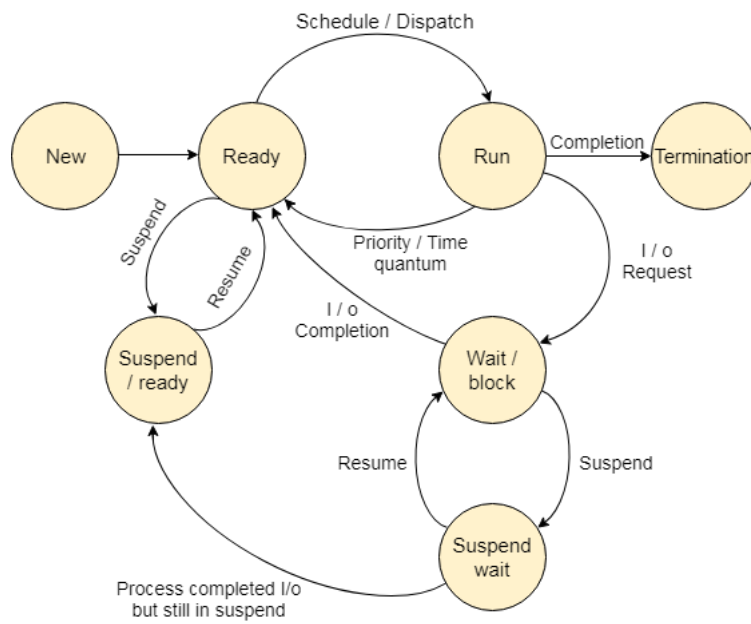
Figure 1: Graph representing process states