Task 3 Michał Kurczak

Scheduling Simulator

A typical process involves both I/O time and CPU time. In multi programming systems, one process can use CPU while another is waiting for I/O. This is possible only with process scheduling.

CPU Scheduling is a process of determining which process will own CPU for execution while other processes are on hold. This helps to keep the processor busy and utilize it maximally. There are two types of CPU scheduling: **pre-emptive** and **non-pre-emptive.** The pre-emptive scheduling is bases on the priorities of the processes. In simple words, the higher the priority, the more CPU time assigned to the process and the more likely to be picked. In our case, we'll be using the simplest, non-pre-emptive scheduling algorithm which is the **First Come First Serve.**
This algorithm is based on the FIFO queue. The first process that asks for the CPU is allocated the CPU first. Next processes, based on the time of requests are placed in the ready queue, with the last process to ask, at the tail.

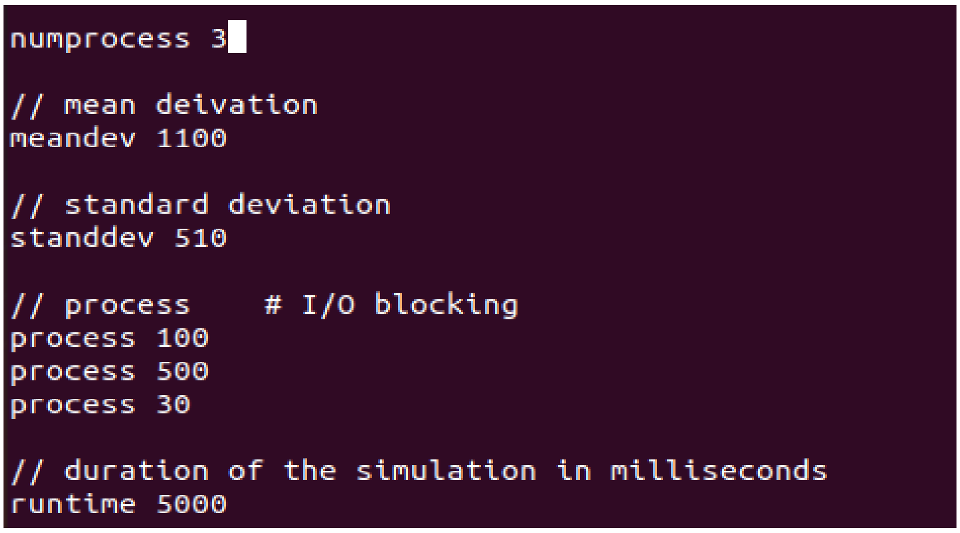
Task Description

In our task we were asked to create a configuration file with the following parameters:

- Run the simulation respectively for 2, 5 and 10 processes
- All processes run for an average of 2000 milliseconds
- Standard deviation set for 0
- Every process is blocked for I/O for 500 milliseconds
- Simulation should be run for 10000 milliseconds

The configuration file scheduling.conf consists of the following parameters:

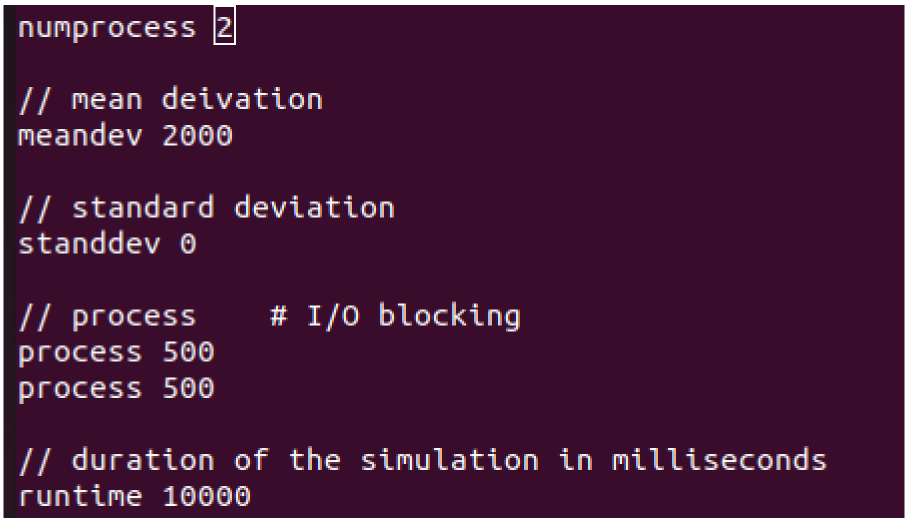
- number of processes
- the mean runtime for a process
- the standard deviation in runtime for a process
- I/O blocking time for each process separately
- Duration time of the simulation



```
numprocess 3

// mean deivation
meandev 1100

// standard deviation
standdev 510

// process    # I/O blocking
process 100
process 500
process 30

// duration of the simulation in milliseconds
runtime 5000
```

Figure 1. The scheduling.conf configuration file structure

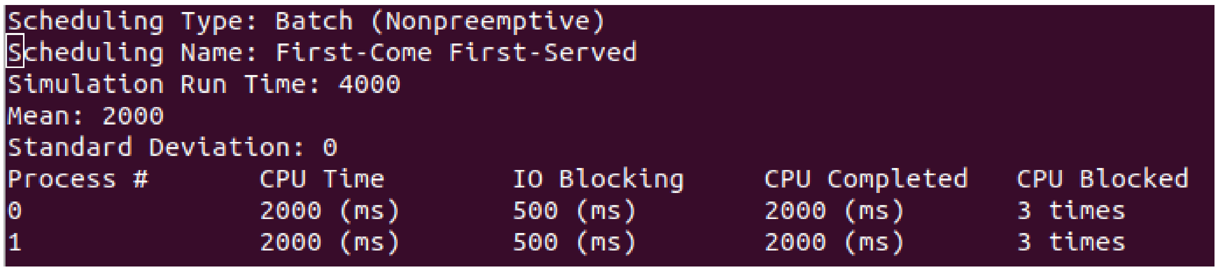
Simulation configured for 2 processes



```
numprocess 2

// mean deivation
meandev 2000

// standard deviation
standdev 0

// process    # I/O blocking
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
```

Figure 2. Configuration for 2 processes

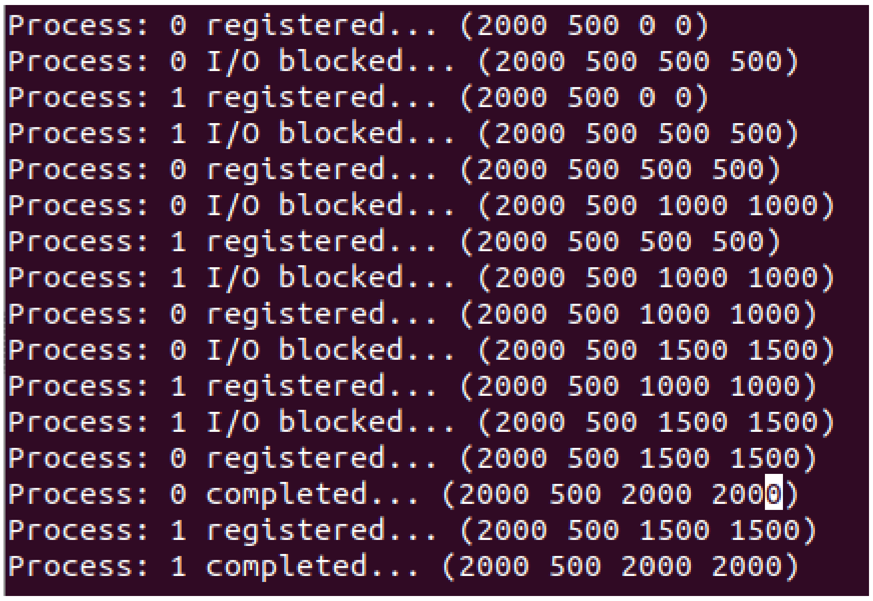
2 processes, 2000ms runtime each, standard deviation set to 0, each process is equal in terms of CPU resources (non-pre-emptive) so each one is blocked for I/O after 500ms, total runtime of the simulation is 10000ms.



```
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
```

Figure 3. output file "Summary-Results"

As we can see, the simulation has been run for 4000ms, which is smaller than the set 10000ms.
Mean is the average runtime for the processes.
CPU Time is the total runtime for the process.
IO Blocking is the amount of time process runs before it blocks for input and output.
CPU Completed is the amount of time it took to complete a process.
CPU Blocked is the number of times process has been blocked for input or output.
The simulation took only 4000ms because all of the processes have been completed after this time.



```
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)
```

Figure 4. Output file "Summary Process"

The first entry in the bracket is the total amount of runtime allowed for the process.
The second one is the block-time – the amount of time to execute before blocking process.
The third is the accumulated time – the total amount of time process has executed. The last one is the same as the third argument, hence we ignore it.

Process may be in one of 5 states, as shown in Figure 5 below.

- **New** – The process is in the stage of being created
- **Ready** – The process is waiting for the CPU resources
- **Waiting** – The process is waiting for the I/O
- **Running** – The I/O is blocked and the CPU is working on this process's instructions
- **Terminated** – The process has completed

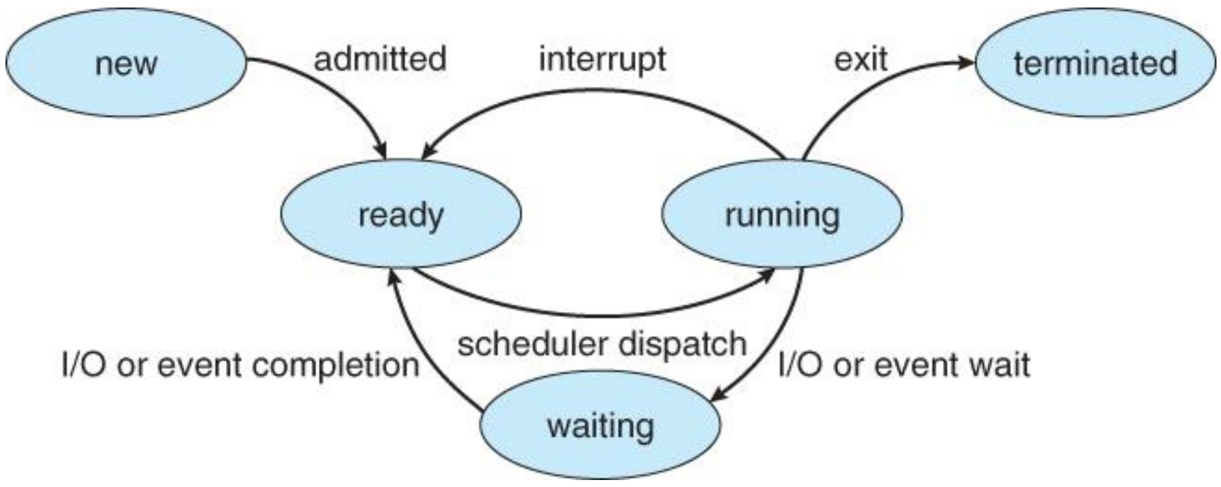
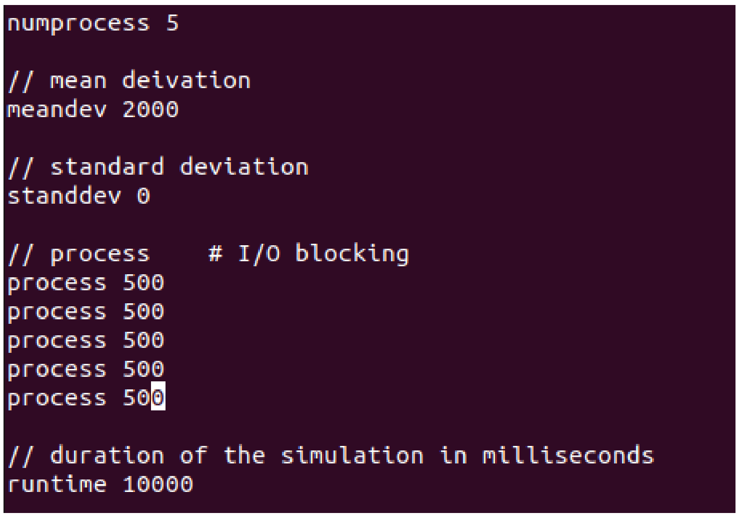


Figure 5. Diagram of process state
Source: https://www2.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/3_Processes.html

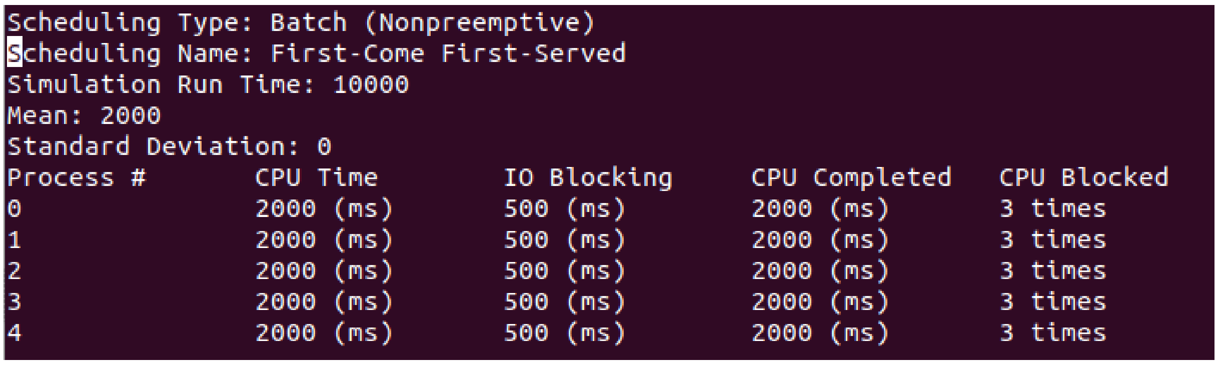
The first process to ask for CPU time is process 0. It's being admitted and placed in the **ready** state (registered). Next, since the CPU is free it's dispatched by the scheduler to the **running** state for the CPU to work on it. After the set 500ms process 0 is I/O blocked and set aside by the kernel to the **waiting** state. In the meantime, since the **ready** state is free new process (process 1) Is being registered and immediately given the CPU resources, as the process 0 waits for I/O. Then, after the 500ms process 1 is being I/O blocked, put to the **waiting** state and the process 0 is being registered as ready and given the CPU time in order for it not to be idle. The process repeats until all of the processes are completed, since we have more than enough simulation time for them to finish.

Simulation configured for 5 processes



```
numprocess 5

// mean deivation
meandev 2000

// standard deviation
standdev 0

// process    # I/O blocking
process 500
process 500
process 500
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
```

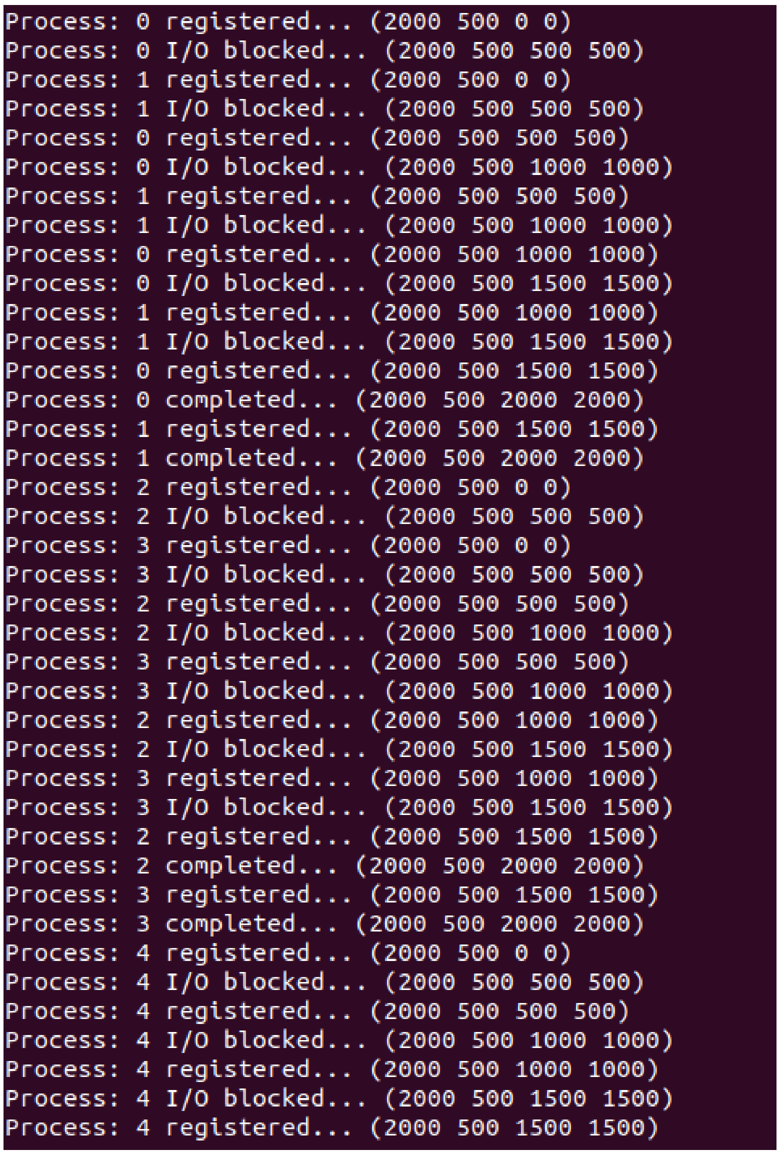
Figure 6 Configuration file



```
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
```

Figure 7. Output file "Summary Results"

Since 10000ms is equal to the summed CPU Time of 5 processes, all of the processes have utilized the assigned CPU Time.

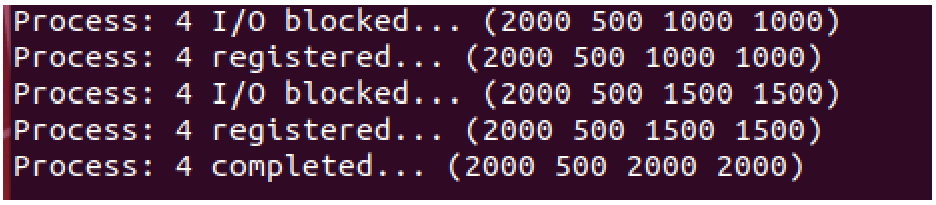


```
Process: 0 registered... (2000 500 0 0)
Process: 0 I/O blocked... (2000 500 500 500)
Process: 1 registered... (2000 500 0 0)
Process: 1 I/O blocked... (2000 500 500 500)
Process: 0 registered... (2000 500 500 500)
Process: 0 I/O blocked... (2000 500 1000 1000)
Process: 1 registered... (2000 500 500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

Figure 8. Output file "Summary-Processes"

Similarly to the case with 2 processes, we observe the alternating pattern of dispensing the CPU resources. The next process is registered only if the ready state is free, so the next processes are registered right after the first two are completed. We can notice that the 4th process does not seem to be completed. In fact, it also managed to complete, yet since the simulation time is exactly equal to the time summed time of execution of all processes the program did not notice it. If we increase the simulation time by even a single millisecond process 4 is also noted as completed.

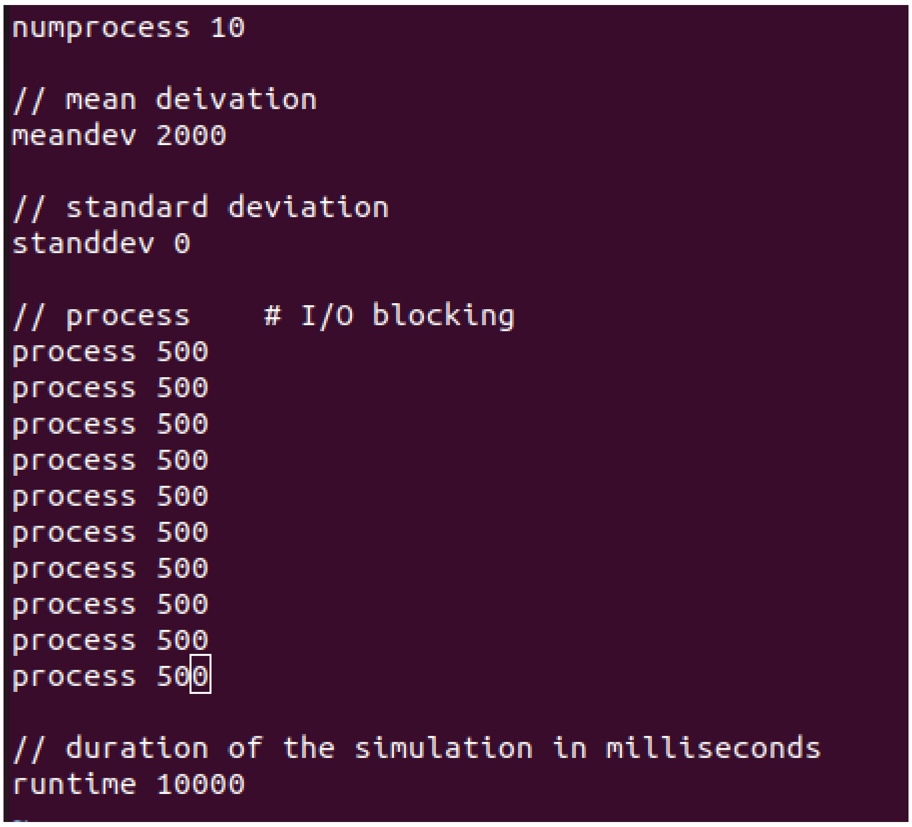
What is worth to notice, is the fact that since we are considering odd number of processes, the last process at the end is switching between **waiting** and **running** state alone.



```
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)
Process: 4 completed... (2000 500 2000 2000)
```

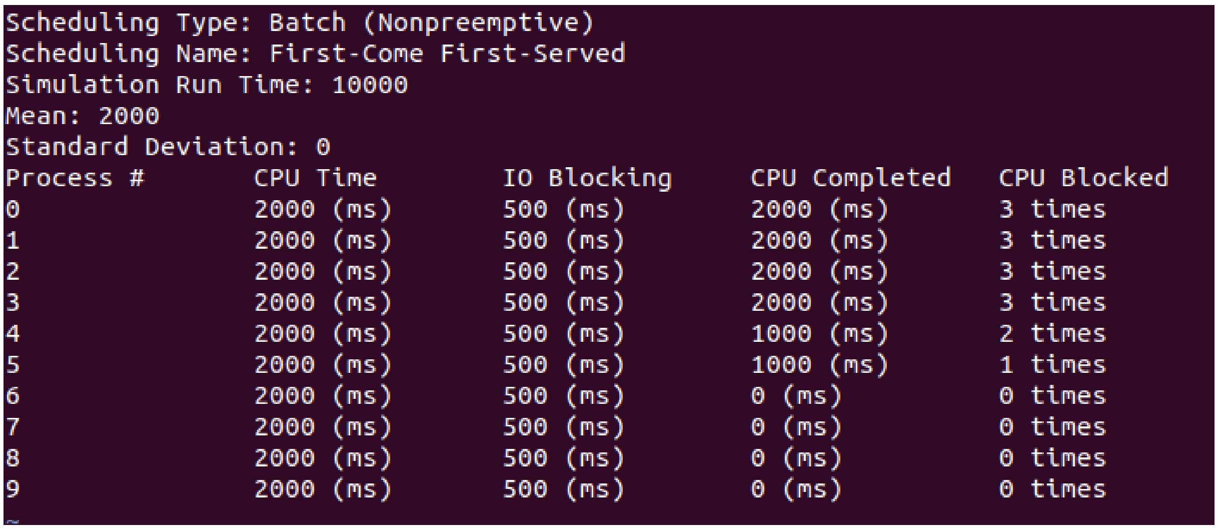
Figure 9. Snippet of the "Summary Results" file for the simulation time set for 10001 milliseconds

Configuration for 10 processes



```
numprocess 10

// mean deivation
meandev 2000

// standard deviation
standdev 0

// process    # I/O blocking
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500
process 500

// duration of the simulation in milliseconds
runtime 10000
```

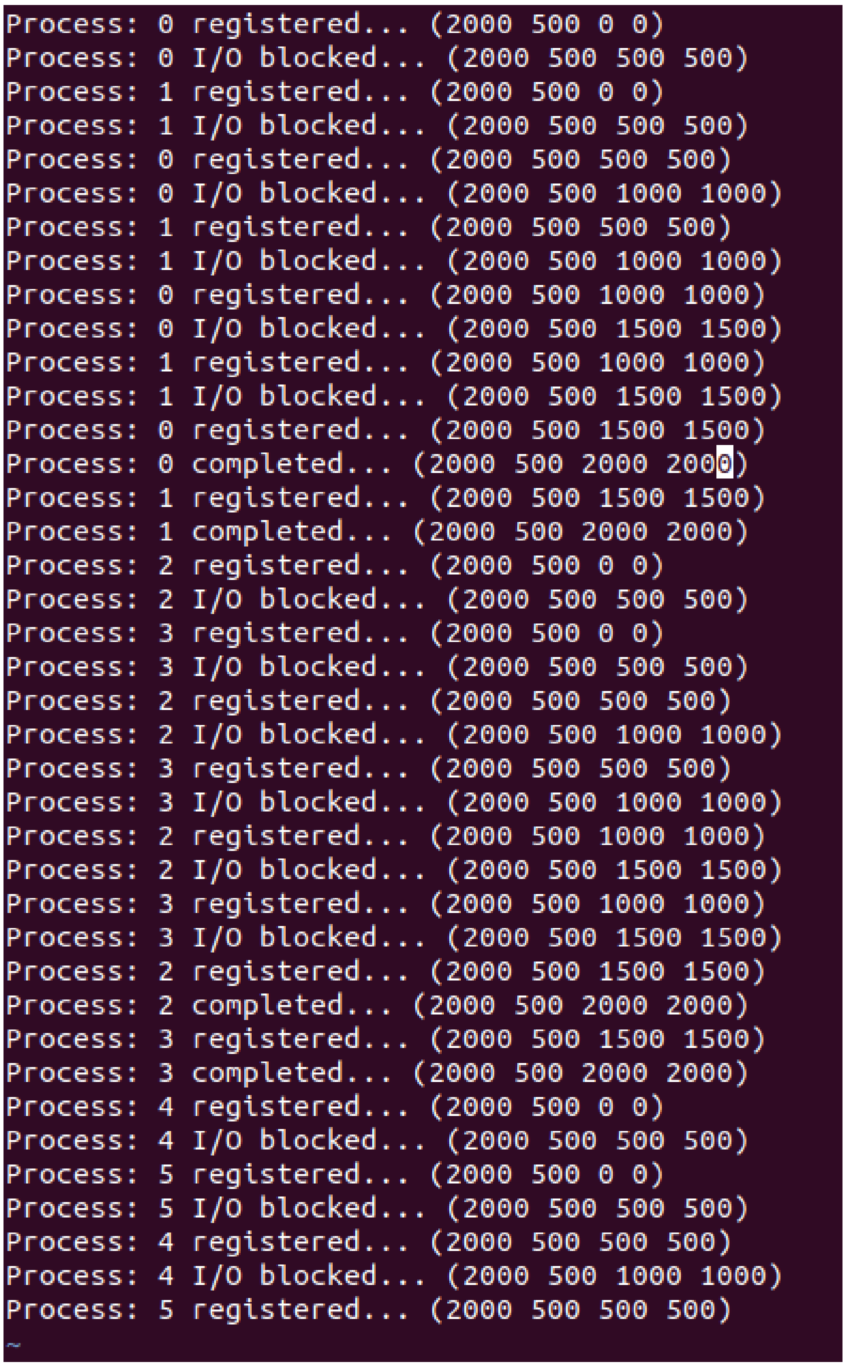
Figure 10. Configuration file



```
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
```

Figure 11. Output file "Summary Results"

AS we can notice, processes 6 – 9 have not been blocked even single time. Process 5 has been blocked once and process 4 twice. This is due to the fact that the simulation time ended after 10000ms which is not sufficient for all the processes to execute.



```
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: 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)
~
```

Figure 12. Output file "Summary Processes"

As we could notice from the previous output, processes 0-3 executed, process 4 used 1000ms from its CPU time and the process 5 have used 1000ms, but the I/O block was not noted by the program since the simulation time was not sufficient to register that change.

Resources

Geeks for geek: https://www.geeksforgeeks.org/cpu-scheduling-in-operating-systems/
Dr Tomasz Kruk's EOPSY lecture slides: http://www.ia.pw.edu.pl/~tkruk/edu/eopsy/
Dr John Bell's Operating System lecture:
https://www2.cs.uic.edu/~jbell/CourseNotes/OperatingSystems/3_Processes.html