EOPSY - Task 3 **Abbaszade Anvar**

Scheduling Simulator.

I/O and CPU time engage as a typical process in multi-programming systems while one process use CPU and another waits for I/O. This is permissible only with process scheduling.

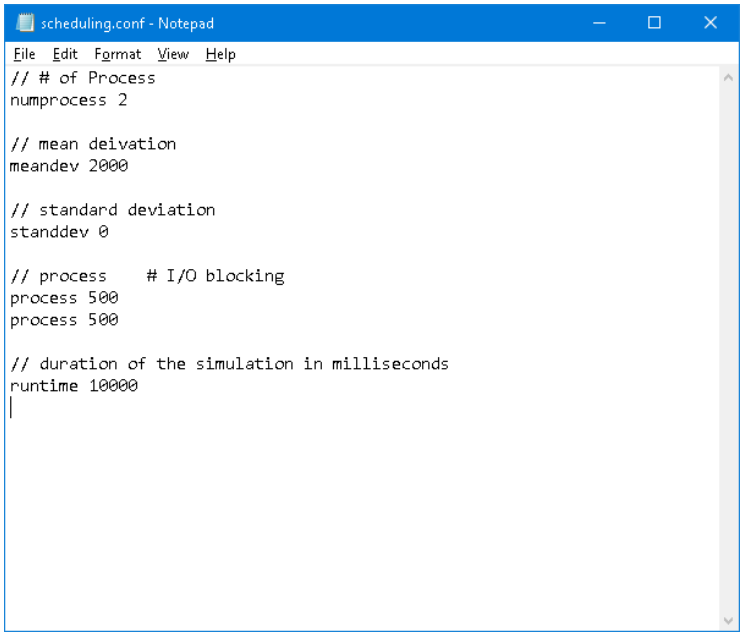
CPU scheduling is a process of determining which process will own CPU for execution while another process is on hold. There are 2 kinds of Scheduling methods: **Preemptive** – based on their priorities. Frankly speaking, when we are running a task with the higher priority before another lower priority task, the lower priority will be still running. In our task, I will be using the simplest, **non-preemptive** scheduling algorithm which is a *First Come First Serve.* This algorithm is based on the FIFO (first-in-first-out) queue. So, the 1st process asks for the CPU which is allocated the CPU first. The next processes based on the time of requests that are placed in a ready queue with the last processes to ask.

Task description.

Create a configuration file in which all processes run an average of 2000 milliseconds with a standard deviation of zero, and which are blocked for input or output every 500 milliseconds. Run the simulation for 10000 milliseconds with 2 processes. Examine the two output files. Try again for 5 processes. Try again for 10 processes. Explain what's happening.

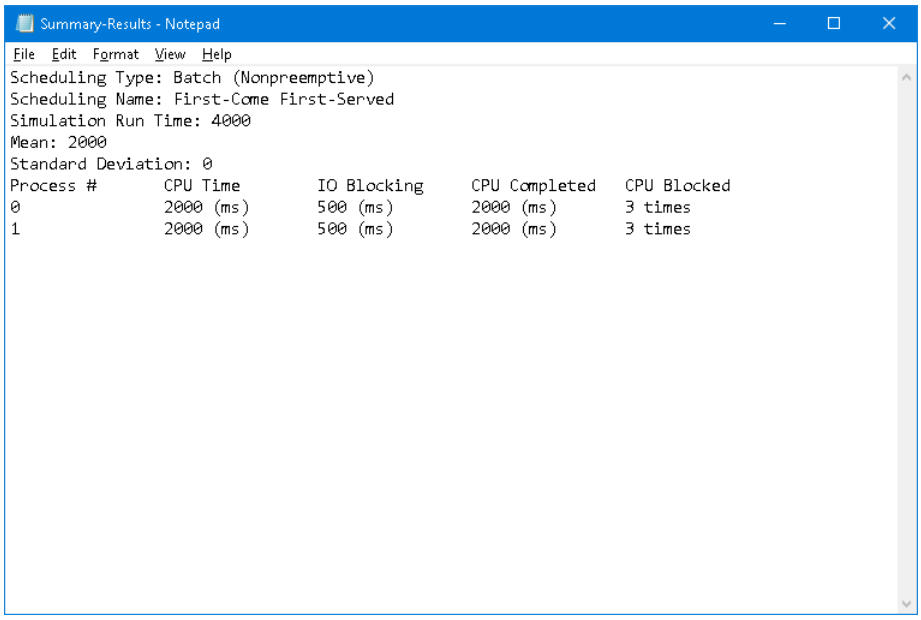
Observations and explanations.

1) **Simulation for 2 processes:**
   - Configuration:



   - Observation:

Summary results:



Here:

Mean – is the average runtime for the processes;

Process # - is the number of processes;

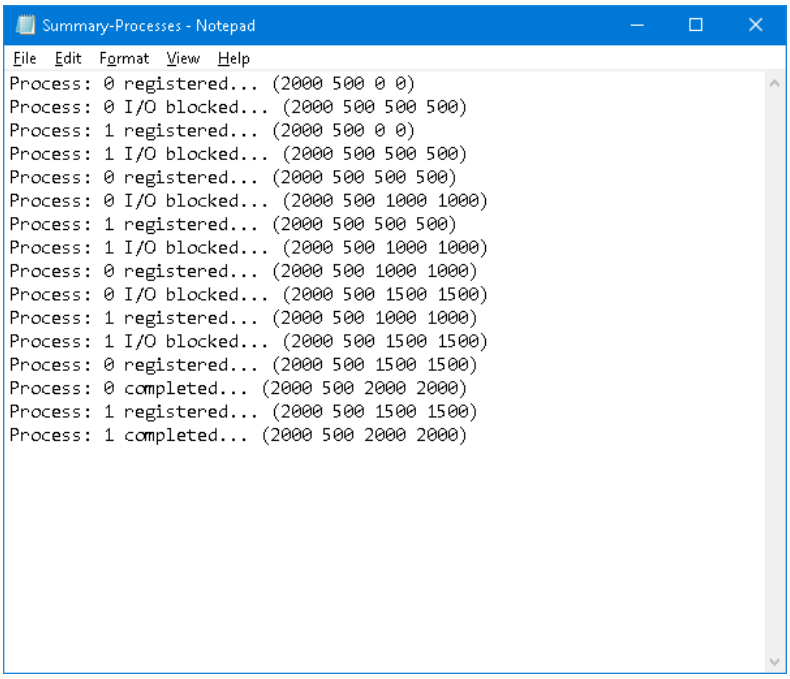
CPU Time – time that distributed to work process;

IO Blocking – the amount of time process working and where after it the process is blocked;

CPU Completed – gives information about how much time spent on work process;

CPU Blocked – is the number of time process that was blocked for both, input and output.

Summary processes:

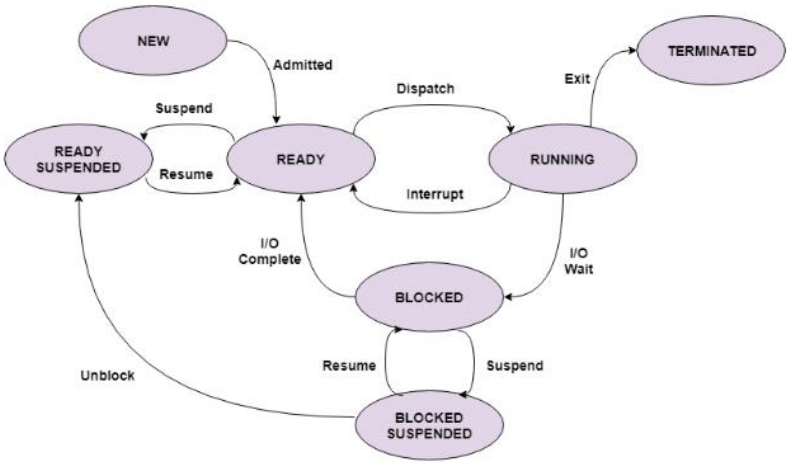


The 1st entry in the bracket here is the total amount of runtime which is allowed for the process.
The 2nd – is the block time where the amount of time to execute before blocking process.
The 3rd one is the accumulated time when the total amount of time process has been executed.
The last one is same as 3rd entry, thus we ignore it.

- Explanation:

Here, we observed that when one process is being executed, the other process is blocked. Every 500ms there is a change and the active process is being blocked and the process that previously was blocked carries on running. Parts of the process were executed sequentially, so the context switching mechanism is managing the processor's time. The simulation was run for 4000ms because each process, in total, was running for 2000ms. The processes were managed according to non-preemptive scheduling, namely a particular process is executed only after I/O blockade of the previously run process. In this case, the blockade lasts 500ms.

A process passes through 7 states, as shown in Pic. 1.



Pic. 1

Source: https://www.tutorialspoint.com/what-are-the-different-states-of-a-process

Let's start with the New state. **New state** – is the state when the process in the stage of being created.

Ready – in this state, the process is waiting for the CPU resources.

Ready Suspended – the processes in ready suspended are in secondary memory.

Running – when the I/O blocked, and the CPU is working on this process's instructions.

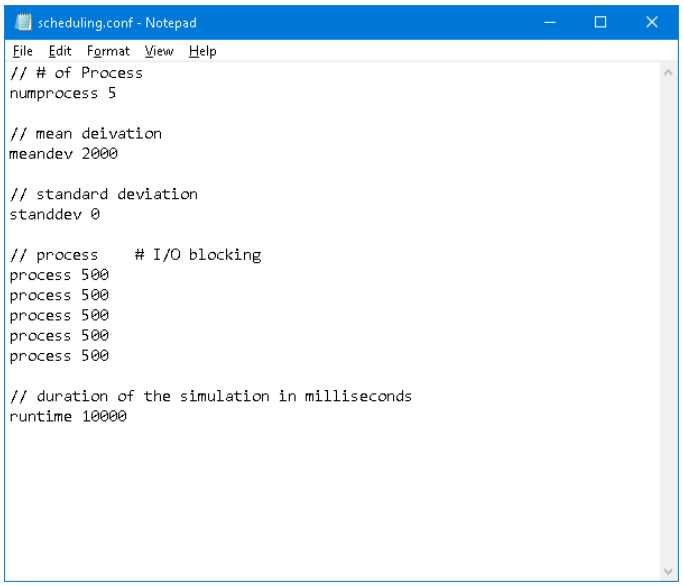
Blocked – the process is being in this state if is waiting for some event to happen. The I/O are can be executed in the main memory and don't require the CPU.

Blocked Suspended – looks like the Ready Suspended. The processes are in secondary memory, because they are originally in the Blocked state in main memory, waiting for some event, however lack of memory forces them to be suspended and gets place in the secondary memory.

And finally, **Terminated** state – in this state the process is completed.

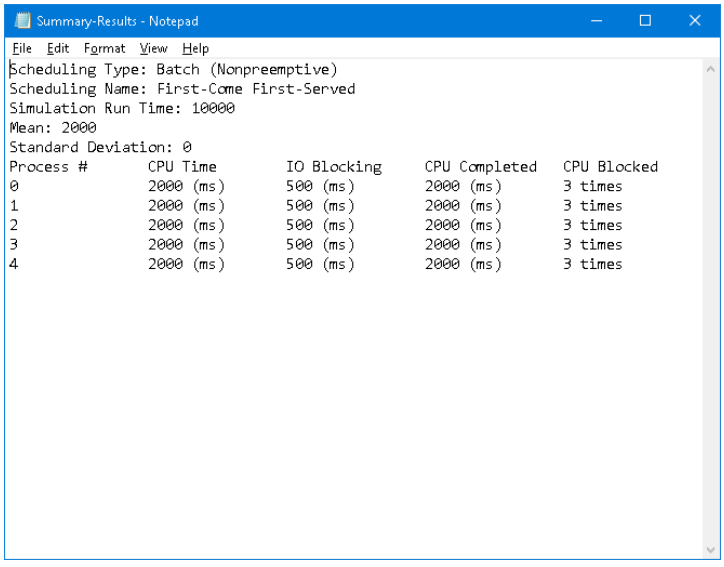
2) Simulation for 5 processes:

- Configuration:

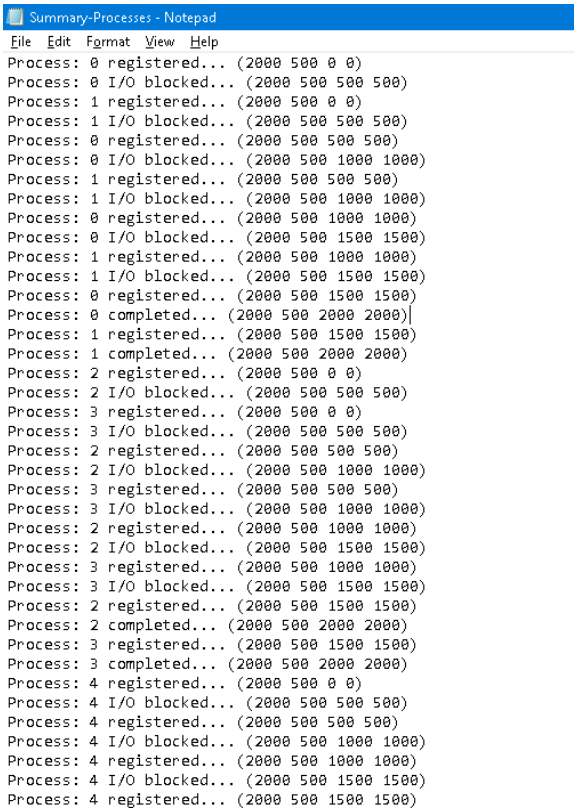


- Observation:

Summary Results:



Simulation Process:



- Explanation:

In this case, the execution of all processes took whole time of the simulation:

5 processes * 2000ms of each process = 10000ms of the simulation.

It is also noticed that the processes were executed and switched in pairs, namely after the completion of the first pair of the processes, the next pair started its execution and so on. In a pair, the processes were managed exactly as in the 1st case: parts of the process were executed sequentially every 500ms, every process was blocked 3 times. It is also the same, batch non-preemptive scheduling. The 5th process (number 4) is the only one that isn't executed in a pair because no more processes exist.

The matter of fact that since we are considering odd number of processes, the last process at the end, is switching between **Waiting** and **Running** state alone.

3) **Simulation for 10 processes:**
   - Configuration:

scheduling.conf - Notepad

File Edit Format View Help

```
// # of Process
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
```

- Observation:

Summary results:

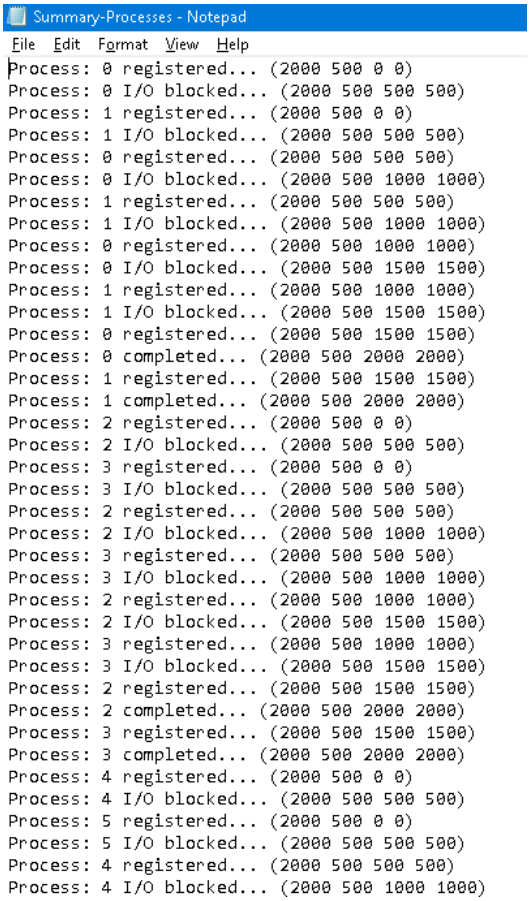
Summary-Results - Notepad

File Edit Format View Help

```
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
```

Here, the process 5 has been blocked just once, and process 4 twice. This is because the simulation time finished after 10 sec which is not sufficient for all processes to execute. The processes 6-9 have not been blocked even once.

Summary processes:



- Explanation:

Finally, in the 3rd simulation time was not enough to execute all 10 configured processed because their total execution time is twice longer than the simulation time. The simulation was very similar to the previous cases: processes were switched in pairs and were managed by according to the batch non-preemptive scheduling. Only first 4 processes completed their execution, 5th and 6th were in progress, and the remaining ones did not even start.