EOPSY 21L LAB-3

Process Scheduling

MAY 11, 2021

KESHAV DANDEVA
Student ID: 302333

Introduction

Process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process based on a particular strategy.

The Operating System maintains the following important process scheduling queues:

- **Job queue** − This queue keeps all the processes in the system.
- **Ready queue** − This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- **Device queues** − The processes which are blocked due to unavailability of an I/O device constitute this queue.

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

A **Process Scheduler** schedules different processes to be assigned to the CPU based on scheduling algorithms. These algorithms are either non-preemptive or preemptive. **Non-preemptive algorithms** are designed so that once a process enters the running state, it cannot be preempted until it completes its allotted time, whereas the **preemptive scheduling** is based on priority where a scheduler may preempt a low priority running process anytime when a high priority process enters a ready state.

The process scheduling algorithm we are going to use for this task is **First-Come, First-Served (FCFS) Scheduling.** As the name suggests, processes are executed on first come, first serve basis. The type of scheduling is non-preemptive. Its implementation is based on First In First Out (FIFO) queue.

Task

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 (a) 2 processes. Examine the two output files. Try again for (b) 5 processes. Try again for (c) 10 processes. Explain what is happening.

THE CONFIGURATION FILE

The configuration file (scheduling.conf) is used to specify various parameters for the simulation, including:

→ **numprocess**: The number of processes to create for the simulation.
→ **meandev**: The average length of time in milliseconds that a process should execute before terminating.
→ **standdev**: The number of standard deviations from the average length of time a process should execute before terminating.
→ **process**: The amount of time in milliseconds that the process should execute before blocking for input or output. There should be a separate process directive for each process specified by the numprocess directive.
→ **runtime**: The maximum amount of time the simulation should run in milliseconds.

The Summary-Results File

The Summary-Results file contains a summary report describing the simulation and includes one line of summary information for each process. The fields and columns in the report are described as follows:

→ **Scheduling Type**: The type of the scheduling algorithm used. The value displayed is "hard coded" in the SchedulingAlgorithm.java file.
→ **Scheduling Name**: The name of the scheduling algorithm used. The value displayed is "hard coded" in the SchedulingAlgorithm.java file.
→ **Simulation Run Time**: The number of milliseconds that the simulation ran. This may be less than or equal to the total amount of time specified by the "runtime" configuration parameter.
→ **Mean**: The average amount of runtime for the processes as specified by the "meandev" configuration parameter.
→ **Standard Deviation**: The standard deviation from the average amount of runtime for the processes as specified by the "standdev" configuration parameter.
→ **Process #**: The process number assigned to the process by the simulator. The process number is between 0 and n-1, where n is the number specified by the "numprocess" configuration parameter.
→ **CPU Time:** The randomly generated total runtime for the process in milliseconds. This is determined by the "meandev" and "standdev" parameters in the configuration file.
→ **IO Blocking:** The amount of time the process runs before it blocks for input or output. This is specified for each process by a "process" directive in the configuration file.
→ **CPU Completed:** The amount of runtime in milliseconds completed for the process. Note that this may be less than the CPU Time for the process if the simulator runs out of time as specified by the "runtime" configuration parameter.
→ **CPU Blocked:** The number of times the process blocked for input or output during the simulation.

The Summary-Process File

The Summary-Processes file contains a log of the actions taken by the scheduling algorithm as it considers each process in the scheduling queue. Each line in the log file is of the following form:

Process: <process-number> <process-status> ... (<cpu-time> <block-time> <accumulated-time> <accumulated-time>)

The fields in the line are described below:

→ **process-number:** The process number assigned to the process by the simulator. This is a number between 0 and n-1, where n is the value specified for the "numprocess" configuration parameter.
→ **process-status:** The status of the process now. If "registered" then the process is under consideration by the scheduling algorithm. If "I/O blocked", then the scheduling algorithm has noticed that the process is blocked for input or output. If "completed", then the scheduling algorithm has noticed that the process has met or exceeded its allocated execution time.

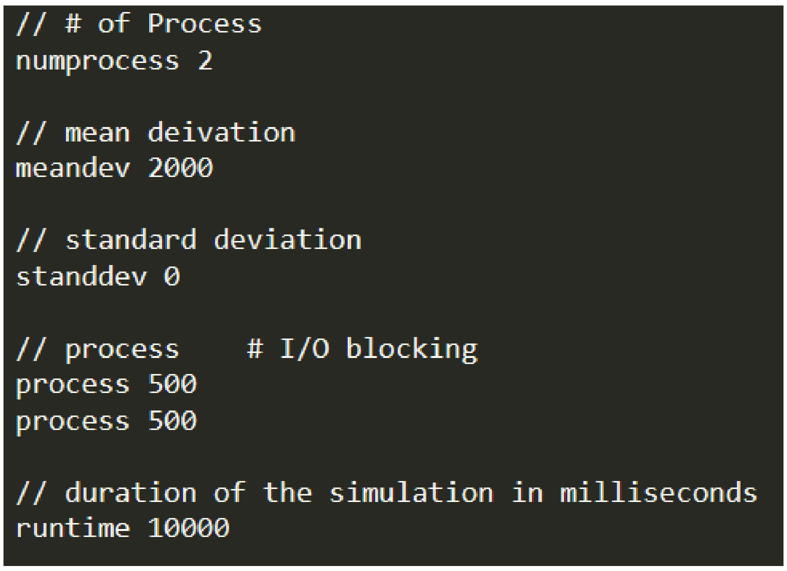
→ **cpu-time:** The total amount of run time allowed for this process. This number is randomly generated for the process based on the "meandev" and "standdev" values specified in the configuration file.

→ **block-time:** The amount of time in milliseconds to execute before blocking process. This number is specified for the process by the "process" directive in the configuration file.

→ **accumulated time:** The total amount of time process has executed in milliseconds. (This number appears twice in the log file).

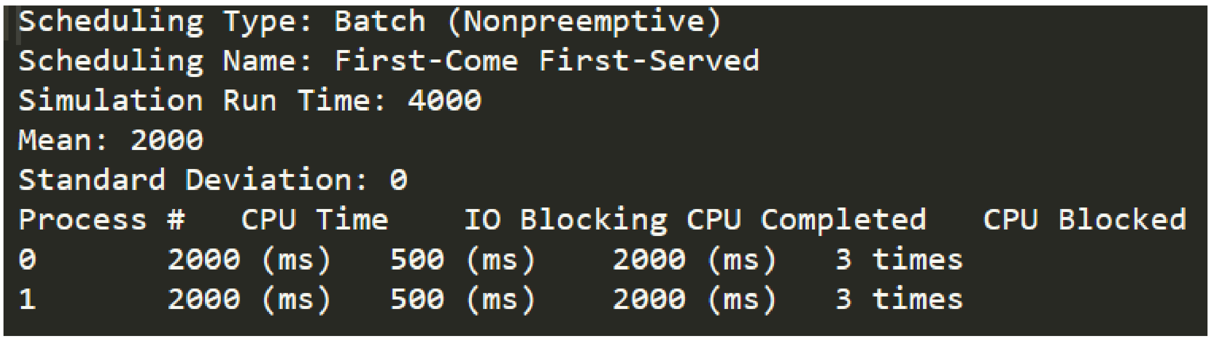
Part A: 2 Processes

Now, we will modify the configuration file “scheduling.conf” as per the given criteria and run the simulation with 2 processes. The contents of the file are shown as follows:



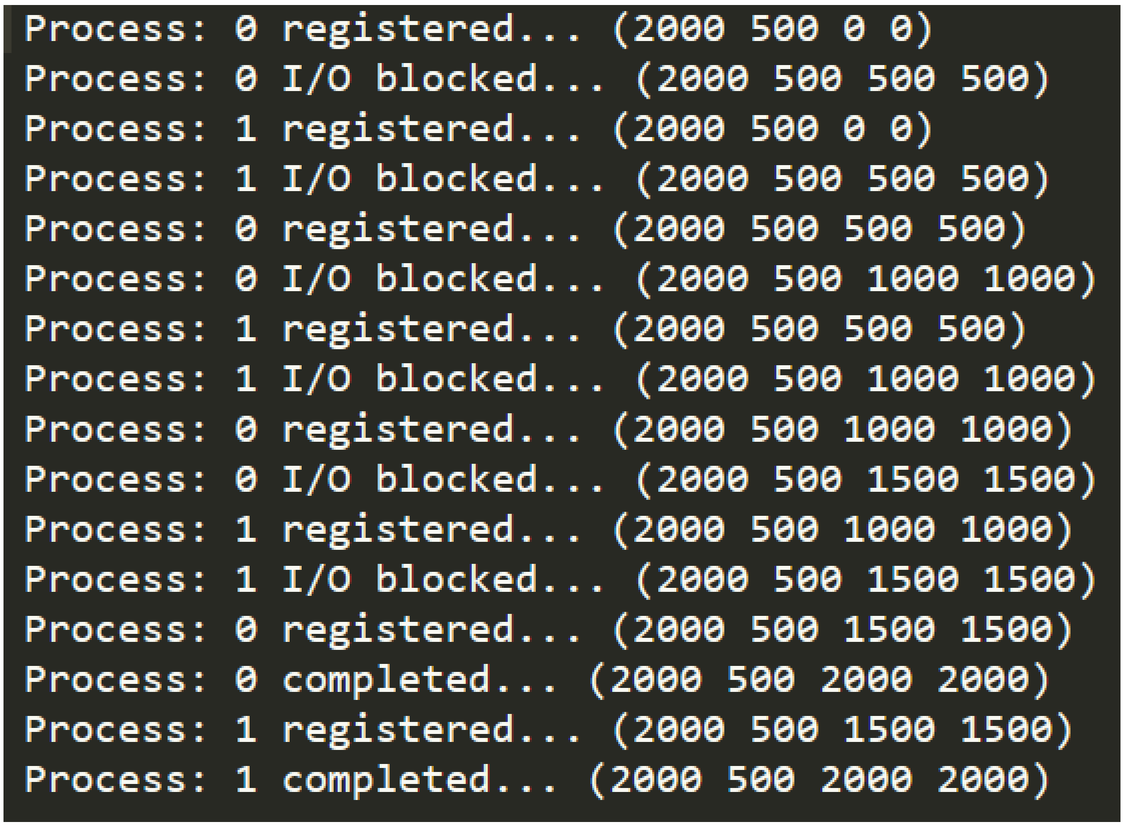
```
// # of Process
numprocess 2

// mean deivation
meandev 2000

// standard deviation
standdev 0

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

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

The **Summary-Results** file contents are shown as follows:



```
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
```

The **Summary-Process** file contents are shown as follows:



```
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)
```

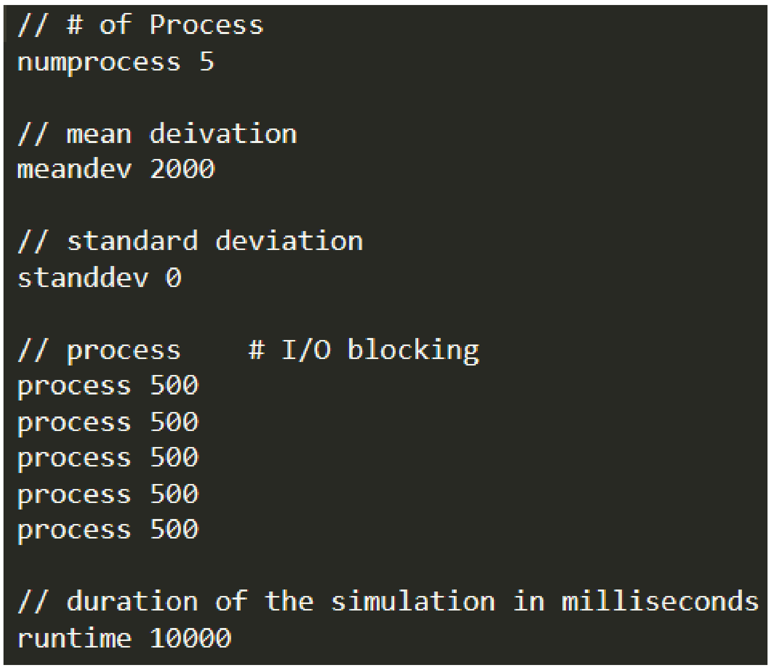
Observations

As we can see from the summary-results file, the total simulation runtime is 4000 ms. The average time taken by CPU for completion of each process is 2000 ms. There are two processes and thus the total time taken is 4000 ms which is less than the total runtime set in the configuration file i.e., 10000 ms.

We can also see that the processes were blocked 3 times by the CPU and were blocked every 500 ms. This is because it uses the first come first serve protocol, where the CPU blocks the process and goes to another process and then blocks it and reinstitute the previous one. This is better visible from the summary-process file where we can see that first the process is registered and then after 500 ms I/O blocked and other process is registered, and I/O blocked after 500 ms and then the CPU goes back to the previous process again for 500 ms. This cycle is executed till the processes are completed or the runtime is finished.

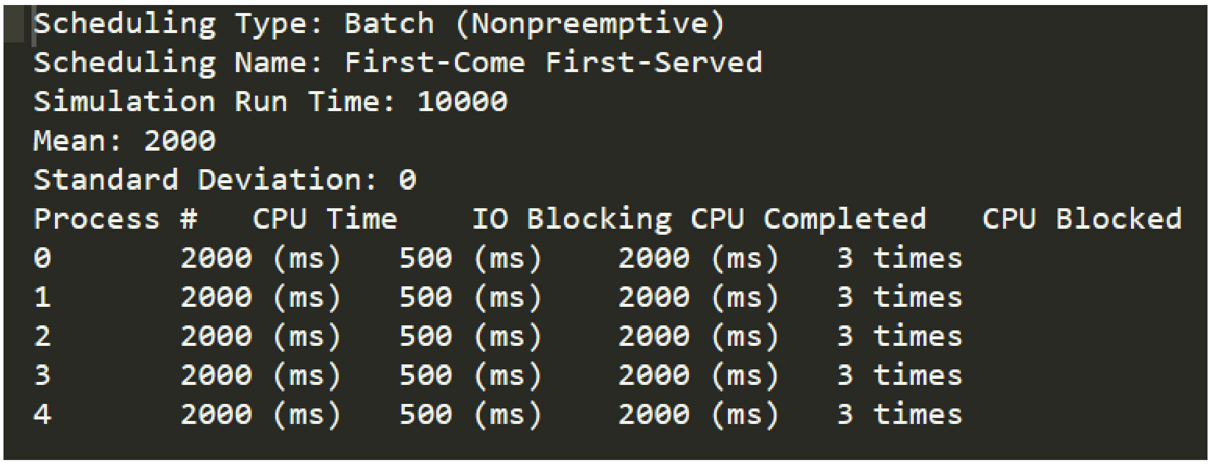
Part B: 5 Processes

Now, we will modify the configuration file “scheduling.conf” as per the given criteria and run the simulation with 2 processes. The contents of the file are shown as follows:



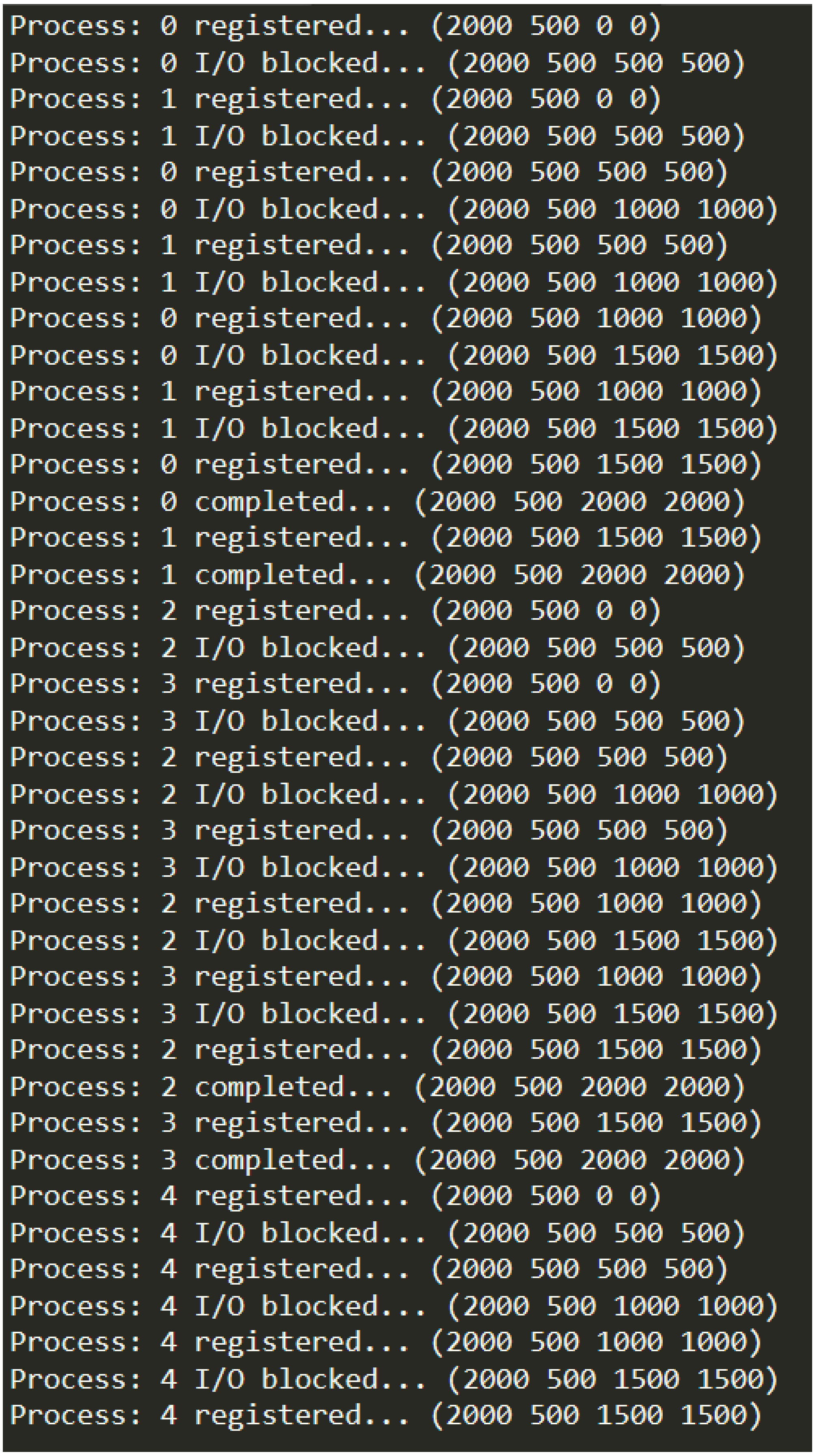
```
// # of Process
numprocess 5

// 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
```

The **Summary-Results** file contents are shown as follows:



```
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
```

The **Summary-Process** file contents are shown as follows:



```
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)
```

Observations

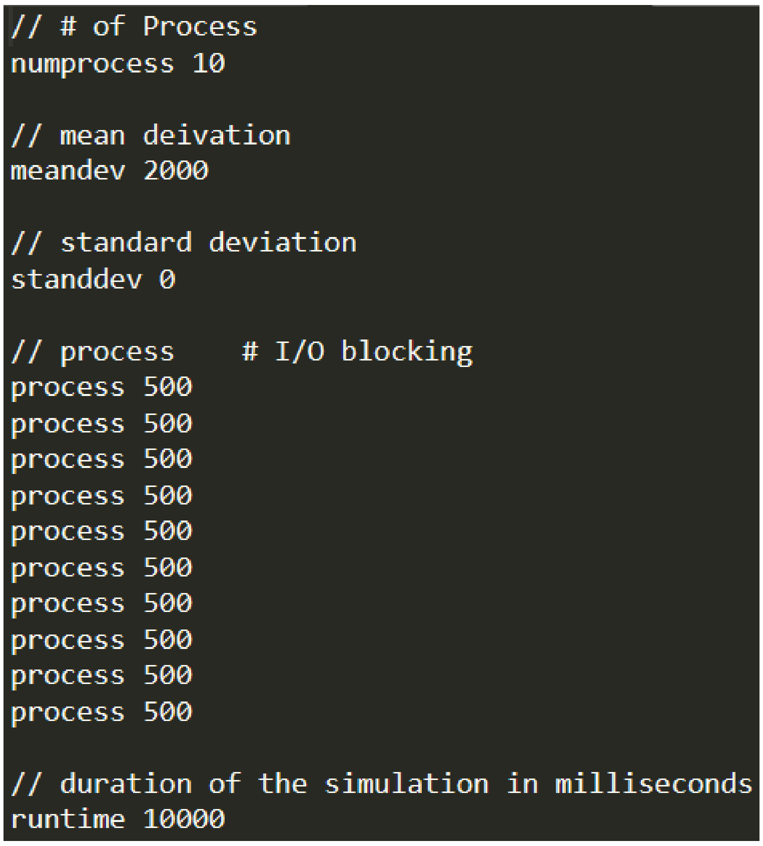
This is similar to the previous task except for the fact that we have 5 processes this time. As we can see from the summary-result file, the total simulation run time is 10000 ms which is equal to the run time we set in the configuration file. Each process takes 2000 ms to be completed and so all the 5 processes should be completed. We can also see that all the 5 processes were blocked 3 times by the CPU.

In the summary-process file, we can see that the processes are executed in pairs i.e., first the CPU completed the processes 0 & 1 and then went to the processes 2 & 3 and then finally to process 4. This is again because of the FCFS algorithm as it favours the first process that requested the processor.

Another noteworthy observation is that we cannot see the process 4 completion message in the summary-process file.

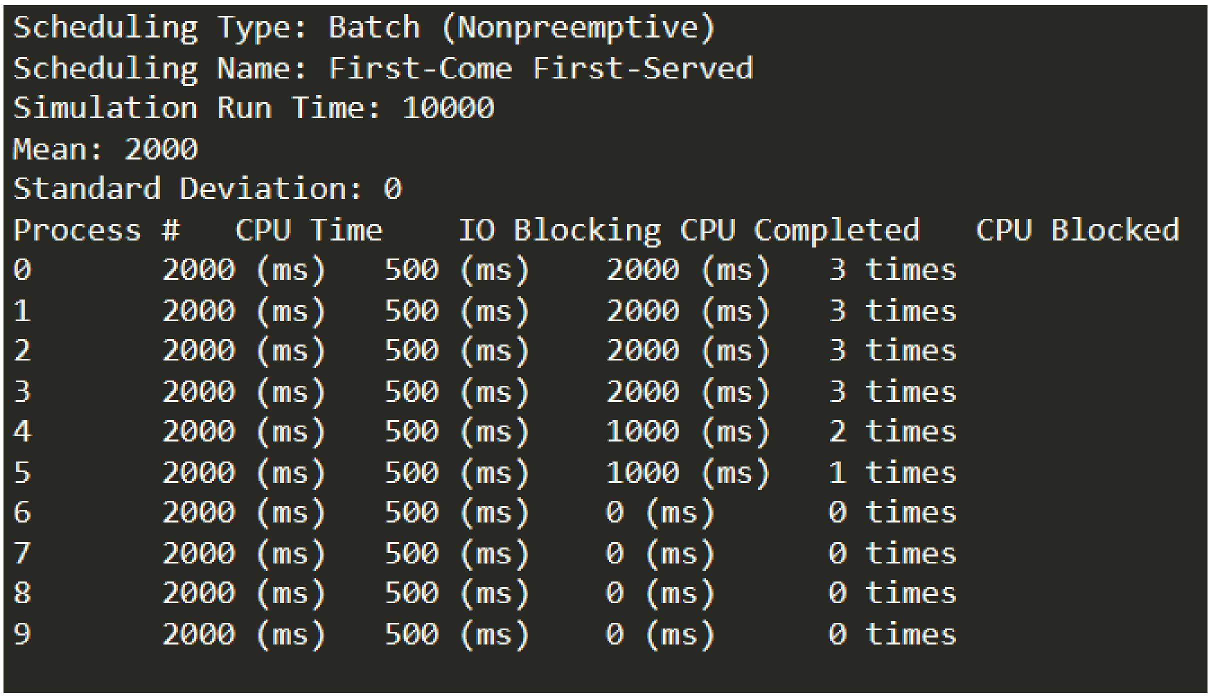
Part C: 10 Processes

Now, we will modify the configuration file “scheduling.conf” as per the given criteria and run the simulation with 2 processes. The contents of the file are shown as follows:



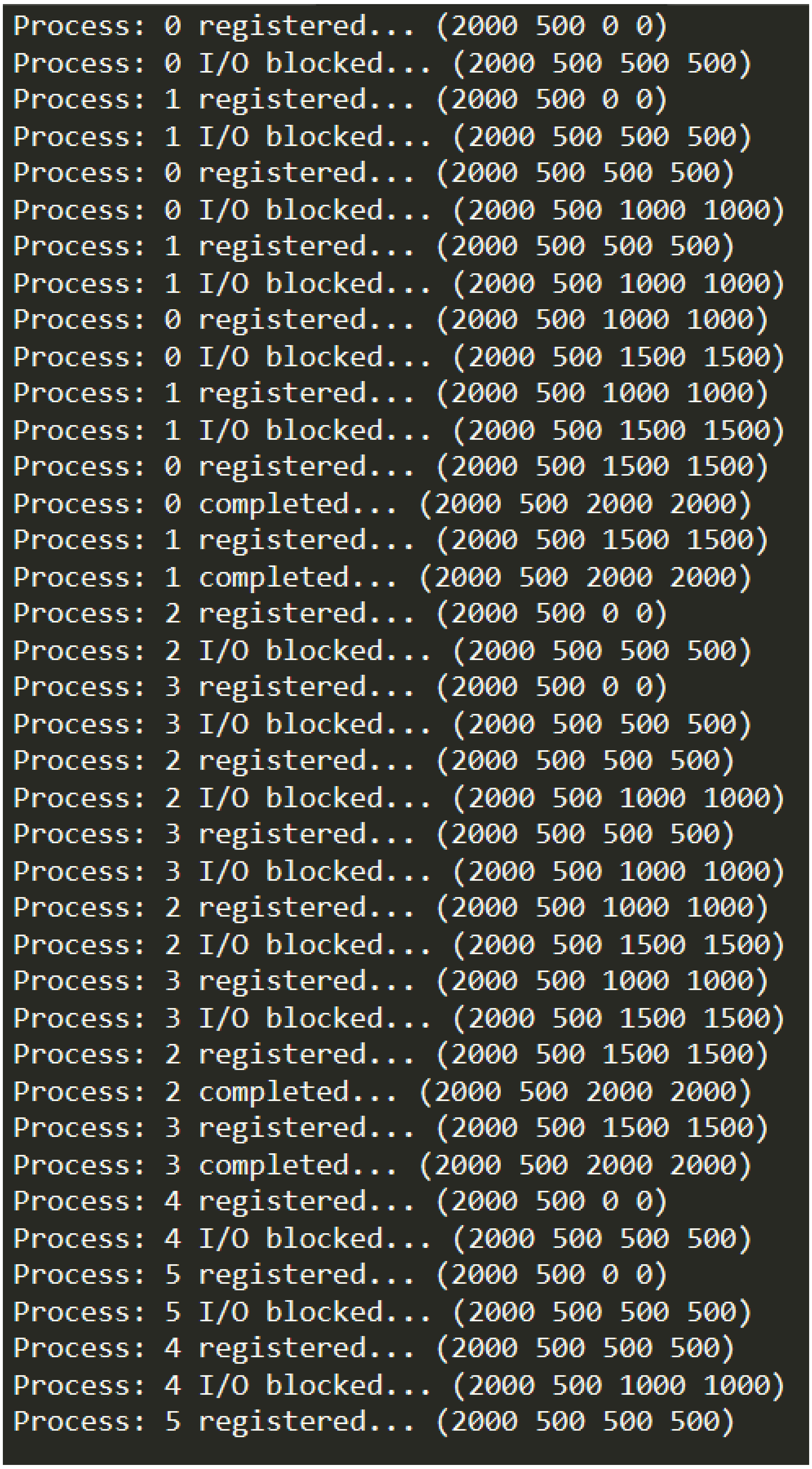
```
// # of Process
numprocess 10

// 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
```

The **Summary-Results** file contents are shown as follows:



```
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
```

The **Summary-Process** file contents are shown as follows:



```
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)
```

Observations

This time the CPU was given 10 processes to execute. The total runtime for the simulation was the same as for previous tasks i.e., 10000 ms which is not enough to execute all the 10 processes as one process takes an average of 2000 ms and 10 processes thus would take 20000 ms.

In the summary-results file, we can see that only first 4 processes are completed by the CPU. Process 4 and 5 are initiated and given 1000 ms each but not completed. This is because the FCFS algorithm executes processes in pairs and thus even though the CPU had time to execute at least 5 processes it could not do so as it takes the processes in pairs and process 4 and 5 both were left incomplete. The last 4 processes (6 to 9) were not even registered by the CPU.

Through the summary-process file we can see that the CPU registers and completes processes in pairs and in the end the process 4 and 5 were registered and blocked but because of insufficient runtime allotted through the configuration file, CPU could not complete the process and process 4 and 5 were given 1000 ms each. The process 4 was blocked 2 times and process 5 was blocked only 1 time as process 4 received the last I/O block before runtime finished.