EOPSY
Task 3
Robert Dwornik

1 Theoretical Introduction

1.1 Type of scheduling: Non – preemptive

Non – preemptive scheduling is not letting to allocate processor time in any moment. In this scheduling, one the resources (CPU Cycles) are allocated to the process, the process hold the CPU **till it gets terminated or it reaches a waiting state.** In case of non-preemptive scheduling does not interrupt a process running CPU in middle of the execution. Instead, it waits till the process complete its CPU burst time and then it can allocate the CPU to another process.

Properties:
• throughput - maximize jobs per hour,
• turnaround time - minimize time between submission and termination
• CPU utilization - keep the CPU busy all the time.

1.2 Scheduling algorithm: First Come First Serve (FCFS)

1.2.1 Jobs are executed on first come, first serve basis.
1.2.2 It is a non-preemptive scheduling algorithm.
1.2.3 Its implementation is based on FIFO queue.
1.2.4 Poor in performance as average wait time is high.

1.3 Table with description of Summary-Result File

Field	Description
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.

1.4 Table with description of Summary-Processes File

Each line in the log file is of the following form:

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

Field	Description
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 at this point in time. 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; one should be removed).

2 Results

2.1 Result of simulation with 2 processes

2.1.1 Configure File

```
 # 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
```

2.1.2 Summary-Process File

```
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)
```

2.1.3 Summary-Result File

```
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
```

2.2 Result of simulation with 5 processes

2.2.1 Configure File

```
 # 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
```

2.2.2 Summary-Result 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
```

2.2.3 Summary-Process File

```
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)
```

2.3 Result of simulation with 10 processes

2.3.1 Configure file

```
 # 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
```

2.3.2 Summary-Result 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
```

2.3.3 Summary-Process File

```
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)
```

3 Conclusions

3.1 Since we are using *batch* scheduling with *First Come First Serve* we are not surprised that processes are completed in order in which they where created. We can observe it in Summary process files where completed processes are marked bold.

3.2 We can also observe that other process can’t start executing until current process is either completed or block. Which also agrees with theory. When first process is blocked second process started to register when second process is blocked after first is unblocked it starts to register.

3.3 As we observe from result can see that when some process is blocked other is starting to execute. Whats more as observe Summary result file each in any of case run time did not exceed mean time, which meets exception. Since according to theory in batch systems we keep CPU busy all the time and minimize time between submission and termination. In our case this difference is 0 since set standard derivation to zero.

3.4 On the other hand we observe very long waiting time for further process to start executing. Some process starting from number 4 where waiting for so long that they didn’t manage to complete or even to start executing before maximal runtime exceeds . We can observe this situation better in case of 10 process, where maximal runtime exceed in the moment when id process 5 starts to register. In summary we can see that processes with id 4,5 completed partially and with id 6,7,8,9 didn’t not complete at all. This results meets the expectation where according to our assumption waiting time for proccess which our further scheduled in queue is very long which leads to in efficiency.

3.5 We might wonder in case with 5 process completed why process with id is fully completed but in case with 10 process it partially completed, it is because process with id 5 “stole” the time when process with id 4 might executing since process has to share CPU time between each other we can observe that it might leads to not completed processes.