EOPSY LAB #3

Baran Korkmaz 302809

Introduction

In this exercise, three different scenarios simulated using MOSS Scheduling Simulator and conclusions drawn from analyzed results. Goal of this exercise to better our understanding of processes and scheduling.

MOSS Scheduling Simulator

The scheduling simulator illustrates the behavior of scheduling algorithms against a simulated mix of process loads. The user can specify the number of processes, the mean and standard deviation for compute time and I/O blocking time for each process, and the duration of the simulation. At the end of the simulation a statistical summary is presented. Students may also be asked to write their own scheduling algorithms to be used with process loads defined by the instructor.

Summary-Processes 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.

Summary-Result File

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).

Theoretical Information

Process

Process is an instance of a program that is being executed.

Process States

1.New

State where process has just been created. Initial state in process life-cycle.

2.Ready

State immediately after the new state. In this state, process is waiting to be assigned to a processor by the scheduler.

3.Running

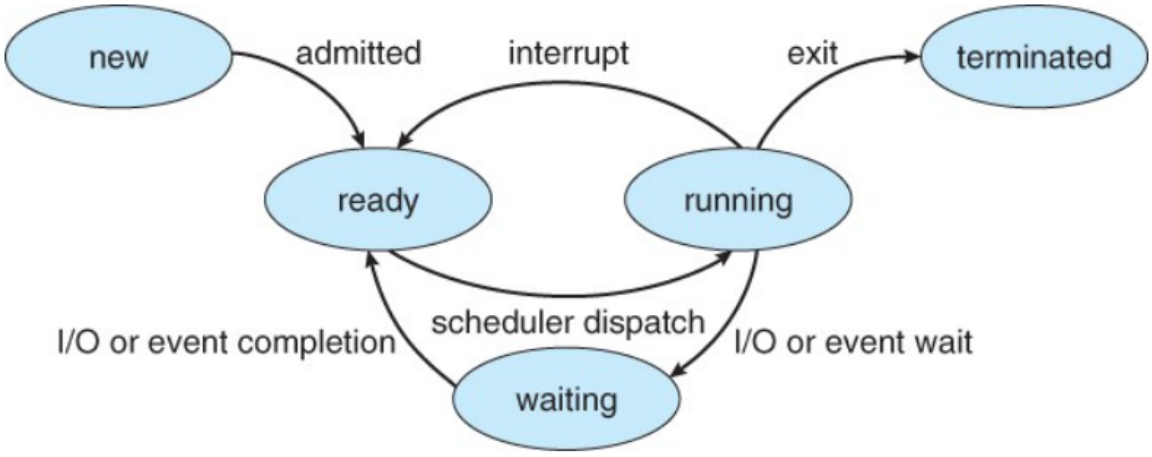
State where process instructions being executed by the processor.

4.Waiting

In this state, the process waiting for an event like I/O input or a signal so it can go back to ready state.

5.Terminated

The process is terminated once it finishes its execution. In the terminated state, the process is removed from main memory and its process control block is also deleted.



Simulations will be run using a non-preemptive batch scheduling (without human interaction) with First Come First Serve algorithm in this exercise.

Scheduling

It is the process of removing the running task from the processor and selecting another task for processing. There are two type of scheduling:

1.Preemptive Scheduling

Preemptive Scheduling is a scheduling method where the tasks are mostly assigned with their priorities. Sometimes it is important to run a task with a higher priority before another lower priority task, even if the lower priority task is still running.

2.Non-preemptive Scheduling

Non-preemptive scheduling is a method that once resources are assigned to a process, they are held until it is completed or changes to the waiting state and it may not be interrupted in the middle.

When a non-preemptive process with a high CPU burst time is running, the other process would have to wait for a long time, and that increases the process average waiting time in the ready queue. However, there is no overhead in transferring processes from the ready queue to the CPU under non-preemptive scheduling.

Scheduling Algorithms

Scheduling Algorithm is the algorithm which tells us how much CPU time we can allocate to the processes.

In this task we will be using First Come First Serve algorithm.

First Come First Serve Algorithm

It is a scheduling algorithm used by operating systems and networks to efficiently and automatically execute queued tasks, processes and requests by the order of their arrival. We will be using non-preemptive version in this simulation but it can also be used as preemptively.

Simulations

1st Simulation

2 processes executed with the given configurations below.

Configuration

```
// # 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
```

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)
```

Results

```
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
```

Analysis

- Processes are completed in the same order as they are started like expected, since simulation uses First Come First Serve algorithm.
- We can observe that all processes are completed.

2nd Simulation

5 processes executed with the given configurations below.

Configuration

```
// # 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
```

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)
```

Results

```
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
```

Analysis

- Processes are again completed in order.
- Like the 1st simulation, all processes completed in maximal time even though simulation didn't have enough time to print the 4th process' completion. However, it can still be observed from results file.

3rd Simulation

10 processes executed with the given configurations below.

Configuration

```
// # 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
```

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: 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)
```

Results

```
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
```

Analysis

- Processes are again completed in order.
- Unlike 2nd simulation, not all processes completed in maximal time as process 4 and 5 partially completed and any other processes after them not even started.

Conclusions

- We can observe that processes run until they are blocked or completed and only after that another process can start running. They are also completed with same order as they are queued. This is fitting with non-preemptive scheduling with First Come First Serve algorithm.
- The CPU times of processes are 2000ms since mean time is set to 2000ms with a standard deviation of 0.
- Processes run for 500ms at a time before getting blocked since I/O blocking times were set to 500ms, therefore processes can be run for that amount of time.
- In the 3rd simulation, 4th and 5th processes only partially completed while in the 2nd simulation, 4th process was fully completed in the same maximal time. This caused by 5th process stealing time from 4th process in 3rd simulation, while in the 2nd simulation, 4th process was blocked and executed consequently without having to share run-time.
- As expected, the waiting times may be too long in this type of scheduling. Results showed that in the 3rd simulation, 4 processes could not find the time to run even once until maximal time reached.