[EOPSY] LAB 3

ERNEST POKROPEK

Warsaw University of Technology, 2020

1 TABLE OF CONTENTS

2 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.

The simulation was performed for 3 cases: using 2, 5, and 10 processes. For each such case, three elements will be taken into account (*taken from ftp/README.tjk*):

2.1 THE CONFIGURATION FILE

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

- the number of processes,
- the mean runtime for a process,
- the standard deviation in runtime for a process,
- for each process, how long the process runs before it blocks for input or output, and
- how long the simulation should run.

2.1.1 Configuration File Options

There are a number of options which can be specified in the configuration file. These are summarized in the table below.

2.1.2 Keyword Values Description

- **numprocess n** - the number of processes to create for the simulation
- **meandev n** - the average length of time in milliseconds that a process should execute before terminating
- **standdev n** - the number of standard deviations from the average length of time a process should execute before terminating.
- **process n** - 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 n** - the maximum amount of time the simulation should run in milliseconds.

2.2 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 following:

- **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.

2.3 The Summary-Processes 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)

Where

- **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).

3 SIMULATION

3.1 2 PROCESSES

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
```

Summary 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

Summary 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)
```

3.1.1 Comments

Using **nonpreemptive** (also called cooperative) scheduling, specifically the **First-Come First-Served** algorithm, once the CPU time is allocated to a process, the process 'holds' the CPU until it gets terminated or blocked (I/O blocked in the discussed laboratory project). As the name suggests, the process first to come has priority to be finished first. The total run time (4000 ms) was equal to sum of the two processes, which proves the previous sentences, as they were executed in sequential manner (thus CPU Blocked equal to 3 for both of them). Although the duration of simulation was specified to 10000 ms in the config, the programme finished earlier, as every process is blocked 3 times, dividing the execution into 4 parts:

$$4 \text{ parts} * 500 \text{ ms} = 2000 \text{ ms}, 2000 \text{ ms} * 2 \text{ processes} = 4000 \text{ ms}$$

3.2 5 PROCESSES

```
// # 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
```

Summary 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

Summary 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)
```

3.2.1 Comments

Continuing the discussion of the nonpreemptive mode, when a process is blocked by the next one, the simulation goes back to the previous, partially executed process. Therefore, the processes are executed in pairs: 0-1, 2-3, and the 4th process has no pair due to odd number of total processes (5). Total runtime (10 000 ms) was just enough to run all the processes, as the total simulation run time is equal to this number.

3.3 10 PROCESSES

```
// # 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
process 500
process 500
process 500
process 500
process 500

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

Summary 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

Summary 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)
```

3.3.1 Comments

The first notable thing is that the set up run time (10 000 ms) was not enough to execute all the processes (as only 0-3 (inclusive) have finished totally). The processes 4 and 5 were executed only partially (approximately 2/3 and 1/3, consequently), as processes 6-9 (inclusive) haven't even started due to lack of time. Knowing, that each process takes 2000 ms to execute, we would need at least 20 000 ms to fully execute all 10 of them.

4 FINAL CONCLUSIONS

First Come First Serve (FCFS) is the easiest and simplest CPU scheduling algorithm, which obviously has its pros and cons. On the presented simulations we can see, that due to limitations of time some of the processes might not finish, or even not start. Although this algorithm is primitive, it certainly is really intuitive and easy to understand, though its usage might not be sufficient for some of the applications. The CPU allocation is managed with a FIFO queue.