Operating Systems

Laboratory 3

Author: Wiktor Łazarski

Field of study: Computer Science

Faculty: Electronics and Information Technology, Warsaw University of Technology

Scheduling

0. Laboratory performing script.

To perform all researches I wrote a bash script called *perform_task.sh*. There are two option to execute that script:

1. Without parameters – this call results that the script will perform simulation based on all **.conf* files specified inside *./configurations* folder. Results for all simulation can be found in newly created folder *./results*.

2. With *'clean'* parameter – this call results that the script will remove all the obtain results and MOSS Simulator environment.

Link: https://github.com/wiktorlazarski/Operating-Systems/tree/master/3.scheduling

1. Configuration files for two processes.

./configurations/two_precesses.conf

```
// # of processes
numprocess 2

// mean deviation
meandev 2000

// standard deviation
standdev 0

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

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

Keyword	Description
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. Separate for every process.
runtime	The maximum amount of time the simulation should run in milliseconds.

Remark: *meandev* and *standdev* determines the length space of time in milliseconds that a process should execute before terminating. Due to, the fact that *standdev* is zeroed, we explicitly set execution time of every process to be 2000 ms.

2. Two processes simulation summaries

After performing simulation summaries can be found in *./results* folder. The name of a file containing summaries for two processes simulation is called *two_processes-results.txt*. Content of this file is presented below:

./results/*two_processes-results.txt*

```
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)

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
```

Observations:

SUMMARY RESULTS

- `Scheduling Type` was `Batch` and what is more important it was `Nonpreemptive` – in this scheduling, process holds the CPU till it gets terminated or it reaches a waiting state, like for example I/O blocking which is set in simulation configuration file to happen every 500 ms for every process.
- `Scheduling Name` informs us about scheduling algorithm picked, which in our case is `First-Come First-Served`. This algorithm assumes that the order of process execution is specified by the order they appear in a scheduling queue.
- `Simulation Run Time` is equal to 4000 ms which makes sense because if we look once again on the configuration file, we set execution time of every process to be equal 2000 ms. Hence, multiplying this value by 2 derives 4000 ms and there are no more processes so simulation does not need to last 10000 ms as specified in *conf* file by `runtime` parameter.
- `Mean: 2000, Standard Deviation: 0` also values equal to the once we specified in a simulation configuration file.
- The table at the end of this section contains general properties regarding time of execution:
  - `CPU Time` – the amount of runtime process needs to held CPU to finish execution.
  - `IO Blocking` – the amount of runtime process held CPU before being I/0 blocked. This value reflects `process` parameter specified in a configuration file.
  - `CPU Completed` – the amount of runtime in milliseconds completed for a process.

- `CPU Blocked` – the number of times the process blocked for I/O during the simulation. We obtained 3 for every process because it had to, after every 500 ms spend on holding CPU, block for I/O. The 4th block was unobserved because it would happen after 2000th ms of execution but the process terminated after this time.

SUMMARY PROCESSES

- From this section we can follow the actions taken by scheduling algorithm as it considers each process in the scheduling queue.
- After keyword `Process` we can read number of a process for which action was taken.
- In the brackets there are 3 values of variables presented:
  - `cpu-time` – the total amount of run time allowed for this process.
  - `block-time` – the amount of time in milliseconds to execute before blocking. This is specified in configuration file by `process` parameter.
  - `accumulated-time` – the total amount of time process has executed in milliseconds(this value appear twice).
- Looking at the order of action we can observe that process 0 appeared in the queue first and starts its execution before process 1. It executes until I/O block – 500 ms – and then it is block for a while and it is a time when algorithm registers process 1 on the CPU. It also executes until I/O block – 500 ms – and then it is blocked. During when process 1 was executing, process 0 signalled that is ready for further execution and when process 1 releases the CPU, algorithm schedules process 0 to CPU. This works in loop until all processes terminates or the simulation runtime finished. In case of this simulation, all processes 0 and 1 completed what was signalled by algorithm with those lines:

```
Process: 0 completed... (2000 500 2000 2000)
Process: 1 completed... (2000 500 2000 2000)
```

3. Repeating simulation for five and ten processes.

After researching simulation summaries for two processes I repeated an experiment for five and ten processes. The configuration in terms of other parameters remains the same. The only change is that we added more processes. Configuration files for five and ten processes can be found respectively in *./configurations/five_processes.conf* and *./configurations/ten_processes.conf* files.

./configurations/five_precesses.conf

```
// # of processes
numprocess 5

// mean deviation
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 ms
runtime 10000
```

./configurations/ten_precesses.conf

```
// # of processes
numprocess 10

// mean deviation
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 ms
runtime 10000
```

After performing simulation summaries can be found in *./results* folder. The name of a files containing summaries for five processes simulation is called *five_processes-results.txt* and for ten processes simulation is called *ten_processes-results.txt*. Content of those files is presented on two following pages.

./results/*five_processes-results.txt*

```
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)

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
```

./results/ten_*processes-results.txt*

```
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)

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
```

Observations for five processes:

- It can be seen that runtime of the simulation was equal 10000 ms and every process needed 2000 ms to complete. Five times 2000 gives 10 000. Hence, we can conclude that all the processes should be able to complete. However, scheduler does not informs us about Process 4 to be completed. We cannot see at the end of `SUMMARY PROCESSES` section any information about its completion. If we take a look at what happens after 0, 1, 2, 3 processes complete, we can see that process 4 has an exclusive use of CPU. It is only blocked after every 500^{th} ms of execution and right after that it is registered on CPU again. This means that it waits for quite a while and this quite a while time must influence that algorithm does not inform about process completion.
  To make sure that it may be a case I slightly increased `runtime` value in *five_processes.conf* file to 10020 ms and I was able to spot the line that informs about this process termination.
  Moreover, it can be seen that in `SUMMARY RESULTS`, Process 4 held CPU for 2000 ms (highlighted by a red colour in table showing results above). Hence, we may assume that it does not make it to inform about its completion because terminating simulation had higher priority.

Observations for ten processes:

- It can be seen that some of the processes does not even had an opportunity to hold CPU. It is due to, the fact that the algorithm used is `First-Come First-Served` and the `runtime` is set to 10000 ms. The simulation terminates before even some of processes will be registered on CPU by algorithm because previous processes have not completed their execution yet.

Observations based on comparison of both result files:

- Those simulations depict how algorithm works for more processes and it can be easily seen that two processes interchangeably shared CPU. Due to the same fact that was describe before when analysing `SUMMARY PROCESSES` for two processes simulation. Only when one of those pair terminates another process from a scheduling queue is registered on CPU.
- There is a difference in Process 4 summary results. In five processes simulation it spent more time holding CPU than in ten processes simulation. It is due to, the fact that at the last milliseconds of five processes simulation it had excusive use of CPU, because all other processes where completed so it could be registered on CPU right after I/O blocking. In ten processes simulation it has to wait for Process 5 to be completed or I/O blocked and therefore the time of `CPU Completed` for Process 4 is different in `SUMMARY RESULTS` for five and ten processes.

Scheduling

```
#!/bin/bash

# sets up simulator for experiments
function setup_simulator()
{
    # add folder to a classpath
    CLASSPATH=.:$CLASSPATH
    export CLASSPATH

    # unpack files
    tar -xf task3.tgz
    cd task3/ftp
    ./setUp

    # jump out of folder
    cd ../..

    # display final message
    echo "Setting up simulator - DONE"
}

# cleans experiment results and simulator env
function clean()
{
    rm -r task3 results

    # display final message
    echo "Cleaning - DONE"
}

# performs set of experiments and saves in results folder
function perform_experiments()
{
    SCHEDULER_CLASSNAME="Scheduling"
    SUMMARY_PROCESSES_PATH="./task3/work/Summary-Processes"
    SUMMARY_RESULTS_PATH="./task3/work/Summary-Results"
    CONFIGS_PATH="./configurations"

    mkdir results

    for file in "$CONFIGS_PATH"/*
    do
        basename=$(basename $file)

        cd "task3/work"
        java $SCHEDULER_CLASSNAME "../../configurations/$basename"
        cd "../.."

        no_extension_basename=$(echo $basename | sed "s/\..*//")
        output_file="./results/$no_extension_basename-results.txt"
        touch $output_file

        echo "SUMMARY PROCESSES" > $output_file
        cat $SUMMARY_PROCESSES_PATH >> $output_file
        echo "" >> $output_file

        echo "SUMMARY RESULTS" >> $output_file
        cat $SUMMARY_RESULTS_PATH >> $output_file
    done

    echo "Task completed. Results are READY."
}

# lab script
if [ "$1" == "clean" ]; then
    clean
else
    setup_simulator
    perform_experiments
fi
```