

**CS5250 – Advanced Operating Systems**  
**AY2017/2018 Semester 2**

## **Assignment 4**

**Deadline: Friday, 27 April 2018 • 11.59pm**  
**This is the last assignment for this course.**

### **1. Objectives**

1. Get a deeper feel of process scheduling.

### **2. Rules**

1. It is fine to ask for “reasonable” amount of help from others, but ensure that you do all the tasks on your own and write the report on your own. The University’s policy on plagiarism applies here and any breaches will be dealt with severely.
2. For this assignment, you are asked to finish three tasks, and write a report (check assignment section for more details).
3. Generate your report as a pdf file, name it as “*Name (Student Number) Assignment4.pdf*” and upload your report in the IVLE folder Assignment-4 of CS5250.
4. The deadline is 27 April 2018. Late assignments lose 4 marks per day.
5. For any question, contact the teaching assistant Mr Ho Nhut Minh, [minhho@u.nus.edu](mailto:minhho@u.nus.edu).

### **3. Assignment**

In this assignment, we will study the effect of process scheduling scheme on system’s performance. In particular, the average waiting time on each process. First we will build a simulation for scheduling tasks, then we explore/compare different scheduling schemes to draw conclusion on how effective they are.

#### **Task 1: 12 Marks:**

Complete the simulation for

1. Round Robin (RR) (4 Marks)

2. Shortest remaining time first (SRTF) with given cpu burst time (4 Marks)
3. **Shortest job first (SJF)** with future prediction (4 Marks)

## Task 2: 8 Marks:

You are to create your own simulator and experiment with different input/configuration and answer the questions.

## Details:

### Task 1:

A sample simulator with FCFS schemes was completed for your reference (see link below). You can choose to work on this file, or write your own simulator in your favorite language. Please make sure input and output format are still the same as in the sample program and as described in this document.

For the simulation, we will assume that all tasks are CPU bound. And the actual cpu time required to complete the tasks are given in the input.txt file.

The `input.txt` file provides the list of processes which will be run on our simulation.

The file contains several lines, each line represent 1 process with 3 integer values:  
*process\_id arriving\_time burst\_time*

For example:

1 0 2

means: the process with id = 1 arrives at the scheduler at time  $t = 0$ , will complete after 2 time units (ms)

Time  $t = 0$  is when the simulation starts, there will always be a process at time  $t=0$ ;

The file `input.txt` will list all process in the increasing order of their arriving time. Assume only one process arrive at a time slice (if process 0 arrives at time 0, process 1 must arrive at time  $> 0$  (any number  $> 1$ )).

**Output:**

Our scheduler will decide when to switch from the current process to another process, depends on which scheme we choose. To record the result of the simulation, we will output the time stamp at each process switching event (only when the new process is different from the current process). Assuming context switching overhead is 0, and the time start at 0ms.

Using FCFS for this input:

0 0 4
1 2 3

Will produce this output:

(0, 0)
(4, 1)
Average waiting time 1.0

In the above example, the input means: process id = 0 arrives at time  $t = 0$ , will take 4 time units to complete. Process id = 1 arrives at time  $t=2$ , will take 3 time units to complete.

Using FCFS, the scheduler will switch to process 0 at time  $t=0 \Rightarrow$  line 1 (0,0) in the output. After process 0 complete then process 1 will be switched in by the scheduler at time  $t=4 \Rightarrow$  line 2: (4,1).

The average waiting time is 1.0 because process 0 didn't have to wait, process 1 have to wait 2 time units (from the time it arrives  $t=2$  to the time it starts  $t=4$ ). On average, the waiting time is  $2.0/2 = 1.0$

The given code has the sample for FCFS, your task is to complete the other 3 scheduling schemes:

1. Round robin with time quantum  $Q$ .  $Q$  is the parameter for your function,
2. SRTF with the given CPU burst time
3. SJF (non-preemptive) with the future-predicting formula in Lecture 7, slide 49. Using  $\alpha = 0.5$ , initial guess = 5 for all processes.

For simplicity, we will do SJF non-preemptive with future prediction. If the current process is scheduled, we wait until it finishes before scheduling another process. We need to keep the record of previous burst time for each process, then predict its current burst time when it arrives at the scheduler. The prediction is per-process and based on their execution history with the initial guess = 5 time units.

Commit the code for the above schemes on your Github and submit the link to the report.

## Task 2:

1. Given the below input:

```
0 0 9
1 1 8
2 2 2
3 5 2
3 30 5
1 31 2
2 32 6
0 38 8
2 60 7
0 62 2
1 65 3
3 66 8
1 90 10
0 95 10
2 98 9
3 99 8
```

Test with 2 of your implemented scheduling schemes (in Task 1.1 and Task 1.3) and compare them. Which one give the least average\_waiting\_time? Try to adjust the time quantum Q for RR and  $\alpha$  value for SJF, give the optimal value of Q and  $\alpha$  to minimize average waiting\_time for the above particular input. Include the output of your tests in the report. (4 marks)

2. What is the optimal scheduling scheme (gives minimum average waiting time) for a system with (4 marks):

- a) All short processes
- b) Very short and very long processes interleave each other with unpredictable pattern.

3. Assume your system has N CPU cores, and each process only requires burst time on 1 core. Will it make the scheduler more complicated? Suggest how to extend the current scheduler to multi-processor system. (bonus marks)

### Sample Code for Task 1 & 2:

Sample code is provided here:

<https://gist.github.com/minhnhn2910/ccfe237b32661ee330e3150d57398154>

Compatible with python 2 & 3. To run the simulation, change the input.txt file, then simply type

```
python simulator.py
```

- End of Assignment -