UNIVERSITY OF THE WITWATERSRAND, JOHANNESBURG

SCHOOL OF COMPUTER SCIENCE & APPLIED MATHEMATICS

COMS3010A
Operating Systems

Schedular Project

Instructors

COMS3010A Lecturer:

Branden Ingram branden.ingram@wits.ac.za

Acknowledgements

Thank you to Damion Harvey and Gavin Chen for the time and effort into putting together this project.

1 Introduction

Your aim for this assignment is to create an operating system scheduler that is able to perform well in a variety of metrics, such as **turnaround time**, **response time**, **burst time** and **switching time**. You need to create a scheduler that is able to perform well in a variety of situations, and also needs to take into account I/O Interrupts. How you handle the processing is up to you.

A template file (titled *template.py*) has been provided to make setup easier.

2 Marking

Marks will be based on a competitive aspect and a baseline aspect.

There are currently 3 baselines implemented. Surpassing each baseline will guarantee you a minimum mark corresponding to that baseline. This mark will be scaled by how well you do relative to your peers at that level. For example, if three students beat First Come First Serve but are worse than Shortest to Completion First; the top student will get 75%, the middle student will get 62.5% and the weakest student at that level will get 50%

- **MLFQ** : 90.00%
- **Shortest to Completion First** : 75.00%
- **First Come First Serve** : 50.00%

3 Input and Output Handling

Your program firstly needs to take in the name of a file as an argument. We have created multiple datasets which represent different scenarios which your scheduler could run into and can be found in **Process_List**. This file contains a list of the programs and their respective details. This file is formatted as follows :

3.1 Input

The input is a number *n*, indicating the number of processes in the file, followed by *n* processes, each on a new line. Each process line is in the format:
Process Name, Process Runtime, Process Arrival Time, IO Frequency

3.2 Output

Your program will need to save its output to a text file with the name **schedulername_out_n.txt** where n is the number of processes processed (should match the input file). Inside the text file you should have 1 line representing the order of the processes as they were scheduled, separated by a space. If IO occurs for a given process, then it needs to be indicated by a **!** followed by the process name without a space. In this case the fequency of the IO is dependant on the "IO Frequency" parameter. It is also important to note that in this case **NO** other process can be scheduled concurrently with an IO request. For more clarity see the examples below.

3.3 Example Input

7
C,8,0,1
E,5,3,0
F,30,3,10
G,5,3,3
B,3,10,2
A,14,13,2
D,27,14,4

3.4 Example Output for FCFS Scheduler

C !C C !C C !C C !C C !C C !C C !C C E E E E E F F F F F F F F F F !F F F F F F F F F F F !F F F F F F F F F F F G G G !G G G B B !B B A A !A A A !A A A !A A A !A A A !A A A !A A A D D D D !D D D D D !D D D D D !D D D D D !D D D D D !D D D D D !D D D D

Here you can see that in the case of "Process C" every-time it is scheduled for 1 time unit it needs to be scheduled for 1 time unit of IO except for the last time it is processed. Hence "C" only has 7 IO requests. Likewise "F" should be scheduled for IO after every 10 normal time units except for the last time, therefore, resulting in 2 total IO requests.

Once again I recommend looking inside "template.py" to see where you need to implement your own code.

3.5 Testing

You can test your scheduler locally by running:

```
python controller.py
```

This will run your code on the dataset indicated within "config.json". It will then compare your generated output with that of the baselines listed in the following code snippet:

```
# Schedulers to be tested
schedulers = ["template", "mlfq", "stcf","fcfs"]
```

This comparison is achieved by calculating a cumulative weighted score with respect to 4 different metrics. These metrics and there weightings can be found in “config.json”. These will be the same weightings used to test your code after your final submission.

```
    "turnaround" : 0.2,
    "response" : 0.6,
    "burst" : 0.1,
    "switch" : 0.1
```

In general the lower the score the better.

3.6 Potential Bugs

- /bin/sh: 1: python: not found - Either you need to install python or possibly you are using python3 instead. If so change the following lines to use “python3” instead of “python”.
  - os.system(”python Schedulers/template.py ” + file)
  - out = scheduler + ’-’ + file + ’-’ + os.popen(”python marker.py ” + file + ” ” + scheduler).read().strip()

Academic Integrity

There is a zero-tolerance policy regarding plagiarism in the School. Refer to the General Undergraduate Course Outline for Computer Science for more information. Failure to adhere to this policy will have severe repercussions.

During assessments:

- You may not use any materials that aren’t explicitly allowed, including the Internet and your own/other people’s source code.
- You may not access anyone else’s Sakai, Moodle or MSL account.
- You may not use any device other than the lab machines.
- You may not edit your submissions using any other device either inside or outside the designated venue.

Offenders will receive 0 for that component, may receive FCM for the course, and/or may be taken to the legal office.