COMS3010A
Operating Systems

Schedular Project

Instructors

COMS3010A Lecturer:

Branden Ingram branden.ingram@wits.ac.za

COMS3023A Lecturer:

Damion Harvey damion.harvey1@students.wits.ac.za

Acknowledgements

Thank you to Damion Harvey and Gavin Chen for the time and effort spent in 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 *studentnum.py*) has been provided to make setup easier.

2 Marking

Marks will be based on baseline aspect. There are currently 3 baselines implemented. Surpassing each baseline will guarantee you a minimum mark corresponding to that baseline. For example, beating MLFQ guarantees you 100.00% for the project. Each scheduler you are able to beat guarantees a certain mark. Please see the below:

- **Beating MLFQ** : 100.00%
- **Beating Shortest to Completion First (STCF)** : 75.00%
- **Beating First Come First Serve (FCFS)** : 50.00%
- **Any other valid submission** : 40.00%

Please Note The code you submit will be marked on an unseen dataset that will be in the same format as the ones provided. Please ensure your scheduler works with all the example data given.

3 Input and Output Handling

We have created multiple datasets which represent different scenarios which your scheduler could run into. These can be found in the **Data** folder. These are *easy.txt*, *medium.txt*, *hard.txt*, *extreme.txt*. These files are formatted as follows :

3.1 Process File

Each process file follows the same format. The first line 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 store the schedulers output in the *output* variable provided in the *studentnum.py* file. You should represent this output as a single line (string) 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 frequency 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

5
A,7,0,4
B,6,5,2
E,1,7,0
D,3,10,0
C,1,13,1

3.4 Example Output for FCFS Scheduler

A A A A !A A A A A B B !B B B !B B B E D D D C

Here you can see that in the case of "Process A" every-time it is scheduled for 4 time units it needs to be scheduled for 1 time unit of IO except for the last time it is processed. Hence "A" only has 1 IO request since it required to process 7 time units. Likewise "C" should be scheduled for IO after every 1 time units except for the last time, therefore, resulting in 0 total IO requests due to its runtime only being 1. Processes only required an IO if there is more time units need to process.

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

3.5 Setup

First, download the project file from Moodle. Extract this to a project directory. There are a couple files in this folder. First is a **run.sh** script that will automatically run everything for you. Please ensure to edit this **run.sh** file to reflect your student number, as well as the dataset you want to use at the top of the file. Please also make sure to rename the **studentnum.py** file with your own student number. Once edited you can run the shell file from any Bash terminal by running the following commands:

```
chmod +x run.sh
./run.sh
```

Please ensure to have Python version 3.11 installed and working otherwise the script will give you an error. **NB** Please ensure to have packages : “numpy” and “tabulate” installed in your environment.

3.6 Testing

You can test your scheduler locally by running:

```
./run.sh
```

This will run your code on the dataset indicated at the top of “run.sh”. It will then rank your code and compare it to the other baselines and display an ordered table of the scores. If your program generates any errors, these will be indicated in the terminal output.

This score is achieved by calculating a cumulative weighted score with respect to 4 different metrics. These metrics and there weightings can be seen below. These will be the same weightings used to test your code after your final submission.

```
    Turnaround Time : 15%,
    Response Time : 60%,
    Burst Time : 15%,
    Process Switch Time : 10%
```

The lower the score the better. Being higher up on the leader-board is better. Your scheduler will automatically be added to the leaderboard once it produces the correct output.

If your code produces any errors, they will be stated clearly above printed leaderboard.

3.7 Potential Bugs

- Error: python is not installed. - Either you need to install python or it cannot be found. Please install Python by using Conda, or via the Windows Store, and then create a new environment with any Python 3.11 version and ensure you are able to access it within your bash terminal.
- Error: Python version must be 3.11.*. - Please make sure that you are using the correct environment with Python 3.11.*. Any Python 3.11 version will work.

3.8 Submission

Please submit ONLY your python file titled *studentnum.py* to the Moodle portal.

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.