10/23/2016

Assignment 3

FIT2070 – Operating Systems

Patrick Shaw

User Documentation

Installation and running Python tasks

1. Open the 26898187_A3 folder.
2. Open a terminal in the folder, or change into the folder from a previously created terminal.
3. Run the command *"python3* ***<task_x.py>*** *./processes.txt"* without quotations, where "**x**" is the desired assignment task to be run.
4. The specified task should now run and begin outputting the information for the specific program.

It is important that you run the program with Python 3 and not Python 2.

Extra options

There are two options available that may be added onto the end of the previously stated command:

- **-i** immediately outputs process information to the console.
- **-d** prints debugging information output to the console

Example Execution

```
C:\Users\eastd\University\FIT2070\26898187_A3>"C:\Users\eastd\AppData\Local\Programs\Python\Python
35\python.exe" task_1.py processes.txt -i
PID:                                    P1
Turnaround time:                        3 seconds
Waiting time:                           0 seconds

PID:                                    P2
Turnaround time:                        8 seconds
Waiting time:                           2 seconds

PID:                                    P3
Turnaround time:                        9 seconds
Waiting time:                           5 seconds

PID:                                    P4
Turnaround time:                        9 seconds
Waiting time:                           7 seconds

Average turnaround time:        7.25 seconds
Average waiting time:           3.5 seconds
Throughput:                             0.26666666666666666 processes per second

C:\Users\eastd\University\FIT2070\26898187_A3>pause
Press any key to continue . . .
```

The information of each process is printed out in the order that they finish and finally the average waiting time, turnaround time and throughput are printed at the end.

Example Execution 2

```
C:\Users\eastd\University\FIT2070\26898187_A3>"C:\Users\eastd\AppData\Local\Programs\Python\Python
35\python.exe" task_1.py processes.txt -i -d
P1 was enqueued
P1 is executed for: 1
P2 was enqueued
P1 is executed for: 2
P1 finished
PID:                                    P1
Turnaround time:                        3 seconds
Waiting time:                           0 seconds

P2 is executed for: 1
P3 was enqueued
P2 is executed for: 2
P4 was enqueued
P2 is executed for: 3
P2 finished
PID:                                    P2
Turnaround time:                        8 seconds
Waiting time:                           2 seconds

P3 is executed for: 4
P3 finished
PID:                                    P3
Turnaround time:                        9 seconds
Waiting time:                           5 seconds

P4 is executed for: 2
P4 finished
PID:                                    P4
Turnaround time:                        9 seconds
Waiting time:                           7 seconds

Average turnaround time:        7.25 seconds
Average waiting time:           3.5 seconds
Throughput:                             0.26666666666666666 processes per second
```

Note that the inclusion of **-d** provides extra information during the execution of the simulation. This may help with checking the correctness of each scheduler algorithm.

Summary of source code structure and style

Important files

- *task_1.py*, *task_2.py* and *task_3.py* contain the scripts to be run to output the process information as specified in the Assignment 3 specification.
- The ./scheduler directory contains all data pertaining to the scheduler and simulation logic.
    - *fcfs_scheduler.py*: Specifies the First Come First Served scheduler algorithm.
    - *round_robin*.scheduler.py: Specifies the Round Robin scheduler algorithm.
    - *srt_scheduler.py*: Specifies the Shortest Remaining Time scheduler algorithm.

Coding Style and architecture

- The code base makes use of Sphinx Python doc strings; the feature can be utilised to its fullest with many popular IDEs such as PyCharm.
- The observer pattern, supplemented with functional programming (namely, lambda functions), is implemented in the scheduler and process in order to easily modularise what is outputted to the console.
- All schedulers make use of inheritance from a base abstract Scheduler class for easy reuse of code.

Task 4

Results

First Come First Serve

PID	Turnaround Time (seconds)	Waiting Time (seconds)
P1	3	0
P2	8	2
P3	9	5
P4	9	7

Average turnaround time: 7.25 seconds

Average waiting time: 3.5 seconds

Throughput: 0.266667 seconds

Round Robin

PID	Turnaround Time (seconds)	Waiting Time (seconds)
P1	5	2
P2	14	8
P3	9	5
P4	5	3

Average turnaround time: 8.25 seconds

Average waiting time: 4.5 seconds

Throughput: 0.266667 seconds

Shortest Remaining Time

PID	Turnaround Time (seconds)	Waiting Time (seconds)
P1	3	0
P2	14	8
P3	4	0
P4	4	2

Average turnaround time: 6.25 seconds

Average waiting time: 2.5 seconds

Throughput: 0.266667 seconds

Discussion

It should be noted that each simulation of each algorithm did not include more complex tasks and features of processors such as handling I/O, interrupts and process priorities. This should be considered when observing the results of each simulation. It should also be noted that each algorithm had the same throughput as the simulated processor was never idle.

In order of best performer to worst performer in terms of fastest average turnaround and waiting time, the scheduler algorithms were Shortest Remaining Time (SRT), First Come First Served, Round Robin (RR).

The SRT algorithm was capable of achieving its significantly turnaround and short waiting time of 6.25 seconds and 2.5 seconds respectively because the manner in which it chooses processes to execute. The algorithm will always pre-empt the currently executing process if another process is estimated to finish executing earlier. That is to say, the algorithm tries to complete as many processes as it can as fast as possible, thus, minimizing the number of processes that are waiting in the queue. Hence, SRT performed the best in terms of turnaround time and waiting time. It should be noted, however, that SRT will have difficulty executing long processes if large quantities of short processes are ready to be processed, as SRT will attempt to complete the short processes before the longer one if the longer process still has a large estimated execution time.

In contrast, the simplistic FCFS algorithm does not perform any pre-emption and instead scheduling the execution sequentially and completing each process in a FIFO manner, earning it a turnaround and waiting time of 7.25 seconds and 3.5 seconds. The FCFS algorithm, unlike SRT, favours SRT in that it will execute long processes in their entirety, however, this means that very short processes do not execute until the long processes finish which is typically undesirable.

The RR algorithm attempts to sacrifice overall performance in order to provide a fairer allocation of time to processes, giving it a turnover and waiting time of 8.25 and 4.5 seconds. This algorithm is the worst performer in terms of turnover and waiting time because it only allows a small time slice for each process to perform execution in, so every process creeps toward competition together and thus it takes a longer time for processes to actually complete. However, if it is only necessary that the process performs a small amount of work over longer intervals of time, then RR becomes an ideal candidate because of its fairness.

Each algorithm has its own benefits; SRT maximizes the number of processes that complete quickly, FCFS allows simplicity while RR provides fairness to all processes and allows a processes to complete sufficient amounts of work longer periods of time.