CA1 specification

Dr Peadar Grant

December 5, 2023

Due date: as displayed on Moodle.

Contribution to Module Mark: as on module descriptor.

1 Grading

Grading will be applied using the following categories for each criterion in the specification:

No grade (N=0) — **Unsatisfactory** (U=20) – **Poor** (P=40) – **Good** (G=60) – **Excellent** (E=80) – **Exceptional** (X=100)

2 Specification

You are to build a simulation tool that demonstrates a number of different job scheduling algorithms that we have discussed in class.

2.1 Scheduling simulator (55%)

Your program must obey the following calling convention:

`./simulate.py jobs.json`

where `jobs.json` is the input JSON file.

2.1.1 Input

Input is to be a JSON file in the format:

```
[
  {
    "name": "J0",
    "arrival": 76,
    "duration": 5
  },
  {
    "name": "J1",
    "arrival": 59,
    "duration": 0
  },
  {
    "name": "J2",
    "arrival": 44,
    "duration": 1
  }
]
```

You can assume that the runtime is known absolutely and that no I/O or other blocking operations take place.

2.1.2 Output

It must show at each time step the process being run using FIFO, SJF, STCF, RR scheduling (with two different time slice values) as follows:

```
T  FIFO SJF  RR1 RR2
1  JOB1 JOB2 UNKNOWN UNKNOWN
2  JOB1 JOB1 JOB1 JOB 2
```

when any of the following happen, it should print the necessary notifications on a line of their own before the relevant time step:

```
* ARRIVED: MY_JOB
* COMPLETE: JOB1
```

The scheduler should continue simulating until no jobs run, printing:

```
= SIMULATION COMPLETE, NO JOBS LEFT
```

The mechanism you use is entirely up to you. You can either pre-compute the values for each scheduling algorithm or you can run each at each time step - there are different trade-offs inherent in each approach.

2.2 Per-job statistics (15%)

Once simulation is complete, your program must output per-job statistics for turnaround time and response time in the following format:

```
= INDIVIDUAL PER-JOB STATISTICS
# JOB FIFO SJF STCF RR1 RR2
T JOB1 10.1 5.2 21.2 4.0 5.0
T JOB2 10.4 3.3 12.3 13.4 12.0
```

You must then print the response times in a similar fashion with `R` replacing `T`.

2.3 Aggregate statistics (15%)

Your program must then output the average turnaround and response times for each scheduling algorithm:

```
= AGGREGATE STATS COMPLETE
# SCHEDULER AVG_TURNAROUND AVG_RESPONSE
@ SJF 10.53 30.2
@ FIFO 32.2 19.1
... (put in other schedulers)
@ RR2 40.0 10.0
```

2.4 Demo scripts (15%)

Supply a demo script to run your program 3 times. Each demo must be able to be invoked as `./demo.sh`

Each of the three demo cases should generate files with different overall arrival time and runtime ranges using the `ca1_jobs.py` script. Run the script with no arguments to see the required command-line parameters. There must be at least 10 jobs with a max runtime of at least 5 in all 3 demo runs.

Your program should print the program output and save it also to `demo_output1,2,3.txt` etc as appropriate.

2.5 Source control (10%)

You are required to maintain your work in your `os_labs` git repository under a folder `ca1`. It is required that:

- commits are atomic
- clear development timeline evident from history,
- meaningful concise commit messages,

3 Requirements for grading

Please note the following requirements carefully:

- Files must be presented in Git repo.
- All files must have the correct names including case.

If *any* of the above requirements are not satisfied, the file will be treated as not existing, and you **work will receive zero**. Depending on what file is missing, this may cause other components to fail and receive zero.

4 Academic integrity

You are reminded that the DkIT academic integrity policy will apply in full. Any suspected breaches of academic integrity will be progressed through the official processes without exception. *If you are unsure if a particular course of action is acceptable you must check prior to submission!*