CS3500: Lab 2

Prashanth L.A.

2023-08-23

Instructions

- Submission deadline is Aug. 30, 2023, 11:59pm. Submission link is available on Moodle.
- Demo to TAs due on or before Sep. 1, 2023, 5.30pm using code downloaded from Moodle.
- Demo can be shown on your laptop or on one of DCF lab machines.
- MOSS/other code similarity checking software will be used.
- Submit a C/C++ file for 1st question and a text file with your answer for 2nd question

1. Multi Level feedback queue Implementation

Aim

The objective of this lab assignment is to implement commonly used scheduling algorithms (FCFS, SJF, RR) as discussed in class, in a system that supports multi-level queues with feedback.

Requirements

The system has four queues: Q4 is highest priority and Q1 has lowest priority. The CPU schedules from Qx only if there are no processes in Q(x+1).

Q4 uses RR; Q3 uses SJF; Q2 uses SJF; Q1 uses FCFS. All algorithms will be in non-preemptive mode; A process under RR will execute under the specified timeslice runs out or it completes its burst.

After a process spends some specified threshold time in the system, the system moves to the immediate higher queue. Processes in Q1 cannot *obviously* move to a higher level.

Here is a brief specification; you can make additional assumptions as needed. All units are taken to be in milli-seconds.

The command-line arguments are:

- Q: time quantum for Robin-robin scheme (a number between 10 and 20)
- T: time threshold for moving a process to the next level (a number between 100 and 50000)
- F: input filename
- P: output filename
- any other inputs that you wish.

The input file (specified by the command line), has a set of entries, one entry per line, as shown below:

```
1 3 0 100
2 4 0 70
3 2 10 35
4 1 20 40
...
```

The specified fields are in the following order: ID, Initial Queue Level,Arrival time (ms),CPU Burst time (ms).

What to Measure: For each process, measure the time between completion time and the time of addition to the first queue it was assigned to. For the system, measure the throughput (no. of processes completed per unit time).

What to Output: For each process, upon its completion, print the following in the output file:

```
ID: xy; Orig. Level: ab; Final Level: cd; Comp. Time(ms): ;TAT (ms):
```

At the end of running all processes specified in the file, print the following information in the output file:

```
Mean Turnaround time:  abc.de (ms); Throughput: xyz.gh processes/sec
```

2. Multi Level feedback queue simulation

Go to the following link and download `mlfq.py`. Go through the `README` and understand how to use the script. Now try out the following using the script.

1. Given a system with a quantum length of 10 ms in its highest queue, how often would you have to boost jobs back to the highest priority level (with the -B flag) in order to guarantee that a single longrunning (and potentially-starving) job gets at least 5% of the CPU?

Miscellaneous

1. WARNING ABOUT ACADEMIC DISHONESTY: Do not share or discuss your work with anyone else. The work YOU submit SHOULD be the result of YOUR efforts. The academic conduct code violation policy and penalties, as discussed in the class, will be applied.
2. You can use C/C++ STL for implementing Queues, Heaps, etc.
3. You can get *gettimeofday()* function to determine when a process was added to a given queue.
4. You can use *nanosleep* function to emulate a process' execution for its specified burst duration (given in milli-seconds).
5. If your system crashes often because of this, please don't blame the instructor or TAs.
6. If you find this assignment to be not as challenging and prefer to implement this in a real kernel, please talk to the instructor.