Author1: Yunge Hao Author2: Yuerui Wang

ID1: 214361760 ID2: 214540785

Username1: hyggs username2: yrwang

Difficulties

First off we built a dumb one, it does not have automatic executions, everything is just handled manually, the only point is its output that tells you which process it is. But then we thought it was too bad, it’s like cheating. So we destroyed everything and started over. This time it’s a much better and much realer cpu simulator.

It is hard to update cpu by time. We tried to use Thread, Timer and synchronized method, but all failures. We don’t have the skill and knowledge. I searched on Google and found a way to reset (re-start a method call) by every certain seconds.

A screenshot of a social media post

Description generated with very high confidence

A screenshot of a social media post

Description generated with very high confidence

Important Notes and Analysis

There are four major classes in our design, *Process*, *ProcessController*, *Scheduler*, and *Main*. ProcessController is intended to serve as process generator, to set the default states and to associate with Scheduler, in which the cpu cycle is simulated. The controller is also responsible for adding newly-generated processes to the ready queue of scheduler.

I use ArrayList for all algorithms. Because we need to add and remove elements, which are basically processes, from its readyQueue all the time. It is dynamic, and thus it is best suit for ArrayList. Array is inefficient and wastes a lot of memory. We don’t want LinkedList because we need to perform index manipulation when doing sorting, from time to time. LinkedList is inefficient for sorting.

I use

Timer.scheduleAtFixedRate(new TimerTask() {

@Override

public void run() {

……

}

}, x, y);

in Main to update the state of the simulated system. In Main class, you can see two of this method. The first one is called every 20 seconds and each call is delayed for 5 seconds. (The delay is just simulating real cpu environment where a process has to wait for a while before it gets ready after it is created.) It is for generating processes. Naturally, it calls method generate() in ProcessController. The second Timer method is to set the next process. It is called every second and delayed 0 second. Here we assume that one time unit is represented by one real second.

Every 20 time units, a new sequence of processes will be generated, of which the state will be set to NEW. It then, after another five time units, adds these new processes to the ready queue, which is the scheduler. Then the state of these processes will be set to READY.

The actual running process is totally determined by time. Time invokes methods that do things. Class scheduler is decides the algorithms corresponding to each Scheduling. Its method setNext() has runtime O(n) since in all the switch cases there is only one for loop. While its memory usage is O(n) because the processes are continued to be added or removed.

Three enum classes. Scheduling contains all 8 schedules. State contains various process states. Type is for Multi Queue Scheduling, I need to differentiate whether this series of processes is emergent(Queue1) or important(Queue2) or necessary(Queue3) or redundant(Queue4).

Method generate() in ProcessController has runtime O(n^2), it has no nested loop neither, but it invokes setNext(), and memory usage O(n).

The key difference between non-preemptive and preemptive is that preemptive scheduling does the search no matter what state the process is in, namely, it updates the whole queue every unit time, while the non-preemptive scheduling searches only when the process is terminated or is in READY, where the first process occurs.