Project 2 (P2): MLFQ Scheduler

Instructor: Dr. Shengquan Wang

Due Time: 10PM, 10/13/2018

This project is on `xv6` scheduler. You are going to extend the `xv6` system you built in P1. There are three objectives to this assignment:

- To familiarize yourself with a real scheduler like `xv6` scheduler.
- To change that scheduler to a new algorithm.
- To display the scheduling information.

1 MLFQ Design

The existing scheduler in `xv6` is a Round-Robin (RR) scheduler. Upon each timer interrupt, the interrupt handler switches to the kernel scheduler, which then selects the next available process to run. This scheduler, while simple, is too primitive to do anything interesting. In this project, you will be putting a new scheduler – the Multi-Level Feedback Queue (MLFQ) – into `xv6`. It has six queues: Q_i, $i = 1, 2, \ldots, 6$ with time quantum size $i * 100$ ms for Q_i. The detailed policy is described as follows:

- Any new process will be inserted to the end of Queue Q_1;
- The scheduler will always choose the front process in the highest-priority non-empty queue in the order of Q_1, Q_2, ..., and Q_6;
- Any chosen process will be scheduled for the queue-corresponding quantum size or the remaining burst size if its remaining burst size is large enough;
- If the process has nothing remaining, the process will be removed from the queue after its execution;
- If the process is not done within this round, it shall be moved to the next lower-priority queue except for any process in Q_6;
- Q_6 runs a Round-Robin with time quantum size as 600 ms. A pointer is needed to maintain the position of the running process. If a process is not done within the assigned quantum size, it remains in Q_6 competing for the next round.
- The scheduler adopts priority boosting. Every 1200 ms, all jobs will be pushed to Q_1.

2 Testing

For the testing, in the xv6 shell please run as follows:

```
$ spin 100000 &; spin 200000 &; spin 300000 &;
```

These three spins are executed in a single command line and run concurrently in the background. Please use `cprintf()` in the kernel to print on console the information for running processes as follows:

```
Process spin 12 has consumed 100 ms in Q1
Process spin 13 has consumed 100 ms in Q1
Process spin 14 has consumed 100 ms in Q1
Process spin 12 has consumed 100 ms in Q2
...
```

Basically, at each timer interrupt, you print out the running process information. The information of the process ID and the process name, which are stored in the `proc` struct. Focus only the `spin` processes.

3 Tips

Most of the code for the scheduler is quite localized and can be found in `proc.c`; the associated header file, `proc.h` is also quite useful to examine. To change the scheduler, study its control flow and then try some small changes. For debugging the kernel functions, you could use `cprintf` to print to the console. You can find some examples in `proc.c`.

Since for every newly-created process `allocproc()` will be called, queue initialization should be done there. Figure out where you need to clear up process when it is done. You can use a counter of timer interrupt to maintain the time quantum size of a process in a queue. For example, if the process is in Q_2, then its initial counter will be 2. After each timer interrupt, it should be decreased by 1.

Resource The following two booklets are very helpful:

- Source code: `http://pdos.csail.mit.edu/6.828/2014/xv6/xv6-rev8.pdf`
  It is a printed booklet with line numbers that keep all `xv6` source codes together.
- Commentary: `http://pdos.csail.mit.edu/6.828/2014/xv6/book-rev8.pdf`
  It is written by the same team that ported `xv6` to x86. You can find in Chapter 5 in this booklet how the scheduler is designed.

4 Submission

You are going to report the following things:

(1) Describe in details how you implemented the new scheduling policy in the xv6. A digram might be helpful;

(2) Highlight all codes you have added or revised;

(3) Write a detailed report with the screenshots of the testing results. Each screenshot should include your username and the current time, which show that you did it by yourself.

(4) Specify the contribution made by each member if you work as a group.

It should be written in a ".docx", ".doc", or ".pdf" format, and submitted to Project P2 on Canvas. Any compression file format such as `.zip` is not permitted.