Report

Project: MLFQ Scheduler

Composed by Yu Zhang (001259692)

- **Description**

  In project, I implemented a simplified MLFQ scheduler in xv6. This MLFQ scheduler has 4 priority queue, where queue 0 has the highest priority while queue 3 has the lowest priority. When a process exhausts its quantum, it will be downgrade to a lower priority queue.

- **Implementation**

  This scheduler follows these rules below:

  - Maintain 4 queues where Q(0) > Q(1) > Q(2) > Q(3) as discussed at class.
  - Every process (including init, sh) will be created at the tail of Queue 0. A RUNNABLE process at the head of a higher priority level will always runs first. When a process is downgraded, it will be put at the end of the lower priority queue.
  - The time-slice for a lower priority level is shorter than that for a higher one. Here we have 1 time ticks for Queue 0, 2 for Queue 1, 4 for Queue 2.
  - When a process is running at Queue 3, it performs the default round robin policy and it will never go back to Queue 0.
  - When a time tick occurs, the process is assumed to have used up this time tick.

  Newly-add system calls and test files:

  - getprio() syscall to get the priority level of current process.
  - getticks() syscall to get the time (in default time tick) the current process has used in its current priority level.
  - Command “sanity” is used to test this scheduler by fork() 5 child process. Each child process will print its info for 50 times in the following format. i.e.

    Pid = 1, Priority = 2, Time in this queue = 2

  - “sanity4” is the same test case as sanity but it only prints out the result for the 1st child process.

2 Issues ignored here:

- The process running at priority 3 (last priority) may suffer from starvation.
- A process may game the scheduler by relinquishing CPU before it exhausts its time slice.

- **Execution example**

  Notice: the result below can be printed by running command "sanity4".

  Here is part of the execution result for command "sanity". It gives you the 1st child process as example to show that this MLFQ scheduler runs correct.

```
cpu0: starting
sb: size 1000 nblocks 941 ninodes 200 nlog 30 logstart 2 i
init: starting sh
$ sanity
Pid=4, Priority=0, Time=1
Pid=4, Priority=0, Time=1
Pid=4, Priority=1, Time=1
Pid=4, Priority=1, Time=1
Pid=4, Priority=1, Time=1
Pid=4, Priority=1, Time=1
Pid=4, Priority=1, Time=1
```

  At the beginning, process 4 ran in Queue 0, when it used up its time slice in Queue 0 (1 timer tick). It downgraded to Queue1.

```
Pid=4, Priority=1, Time=2
Pid=4, Priority=1, Time=2
Pid=4, Priority=1, Time=2
Pid=4, Priority=1, Time=2
Pid=4, Priority=2, Time=1
Pid=4, Priority=2, Time=1
Pid=4, Priority=2, Time=1
Pid=4, Priority=2, Time=1
```

```
Pid=4, Priority=2, Time=1
Pid=4, Priority=2, Time=2
Pid=4, Priority=2, Time=2
Pid=4, Priority=2, Time=3
Pid=4, Priority=2, Time=3
Pid=4, Priority=2, Time=3
Pid=4, Priority=2, Time=3
Pid=4, Priority=2, Time=4
Pid=4, Priority=2, Time=4
Pid=4, Priority=2, Time=4
Pid=4, Priority=2, Time=4
```

  When process 4 used up its time slice in Queue 1 (2 timer ticks) and Queue 2 (4 timer ticks), the same thing happened. And finally process 4 runs in Queue 3 in round robin.