Department of Computer Science and Engineering
The Chinese University of Hong Kong

CSCI 3150: Introduction to Operating Systems

Lab Five: MLFQ Scheduling

Objectives:
Understand and execute MLFQ scheduling.

1. MLFQ Scheduling

The rules of MLFQ (Multi-Level Feedback Queue) can be described like the followings:

- Rule 1: If Priority(A) > Priority(B), A runs (B doesn't).
- Rule 2: If Priority(A) = Priority(B), A & B run in round-robin fashion using the time slice (quantum length) of the given queue.
- Rule 3: When a job enters the system, it is placed at the highest priority (the topmost queue). For the jobs arriving at the same time, schedule the job with smallest pid first.
- Rule 4: Once a job uses up its time allotment at a given level (regardless of how many times it has given up the CPU), its priority is reduced (i.e., it moves down one queue and will be at the tail of the target queue).
- Rule 5: After some time period S, move all the jobs in the system to the topmost queue, and sort all the jobs by pid. Job with the smallest pid will be scheduled first.

Notes: Sorting will happen every time it arrives the Period S.

We need to define the **process information** and also the **queue information** that will be used in MLFQ scheduling.

At first, we define the format of the **process information**.

```
ProcessNum  N
pid:X1, arrival_time:X11, execution_time:X12
pid:X2, arrival_time:X21, execution_time:X22
…
pid:XN, arrival_time:XN1, execution_time:XN2
```

Here, the first line denotes there are N processes to be scheduled, and from the second line to the (N+1)th line, each shows the pid (process id), arrival time and execution time of a process with the format like "pid:X1, arrival_time:X11, execution_time:X12". The time unit here is millisecond (ms). An example is given as below.

```
ProcessNum 5
pidnum:123, arrival_time:60, execution_time:90
pidnum:13, arrival_time:70, execution_time:100
pidnum:1023, arrival_time:10, execution_time:160
pidnum:12, arrival_time:80, execution_time:28
pidnum:36, arrival_time:10, execution_time:10
```

Then, define the format of the **queue information**.

```
QueueNum n
Period_S S
Time_Slice_QN Xn Allotmenttime_QN Yn
...
Time_Slice_Q2 X2 Allotmenttime_Q2 Y2
Time_Slice_Q1 X1 Allotmenttime_Q1 Y1
```

Here, the first line denotes the number of queues we will use in the scheduler, the second line denotes the period S to move up all jobs to the topmost queue (Rule 5), the third line denotes the time slice for Queue n that is the topmost queue with the highest priority (the smallest time slice), and the subsequent lines define other queues with descending order in terms of priority. The time unit here is millisecond (ms). One example is shown below.

```
QueueNum 3
Period_S 300
Time_Slice_Q3 10 Allotmenttime_Q3 30
Time_Slice_Q2 40 Allotmenttime_Q2 80
Time_Slice_Q1 60 Allotmenttime_Q1 120
```

2. MLFQ Scheduling

To execute MLFQ scheduling, we need to figure out the processes in different queues at the same time, and also the order of processes. **You can find the whole procedure in the slides of Lab 6.**

Then, schedule the process with higher priority first. And there are **4 situations we need to write down one piece of result**:

1) Just use up the time of current time slice and no process finishes, write down current process's information;
2) Finish one process and use up the time of current time slice at the same time, write down the finished process's information.
3) Finish one process without using up the time of current time slice, we need to calculate how much time are used. Then just write down finished process's information, the time slot should be slot_beginning_time - process_finish_time.
4) Every time it arrives the Period S, the current running process will stop temporally and move to the topmost queue. Write down the process's information which is running when it arrives the Period S.

Assume the scheduling beginning at time 0, and write down the **result** in the following format:

	Time-slot	**Process ID**	**Arrival Time**	**Remaining Time**
(i)	x - y	pidnum_1	arrival_time_1	remaining_time_1
(i + 1)	y - z	pidnum_2	arrival_time_2	remaining_time_2

1. The first time slot is beginning with the earliest arrival_time. x-y denotes the time interval starting at time x and end at time y (Commonly the time interval is the time slice of current queue. If one process finishes without using the whole time slice, the time interval will be the time it used, and the smallest time interval is 1 ms.)
2. Arrival Time is the arrival_time for the process in the given information.
3. Remaining Time is the length of time of which the process still needs to finish.

The following table is just the FIFO scheduling result of the above given processes.

	Time-slot	**Process ID**	**Arrival Time**	**Remaining Time**
(1)	10 - 20	36	10	0
(2)	20 - 30	1023	10	150
(3)	30 - 40	1023	10	140
(4)	40 - 50	1023	10	130
(5)	50 - 90	1023	10	90
(6)	90 - 100	123	60	80
(7)	100 - 110	13	70	90
(8)	110 - 120	12	80	18
(9)	120 - 130	123	60	70
(10)	130 - 140	13	70	80
(11)	140 - 150	12	80	8
(12)	150 - 160	123	60	60
(13)	160 - 170	13	70	70
(14)	170 - 178	12	80	0
(15)	178 - 218	1023	10	50
(16)	218 - 258	123	60	20
(17)	258 - 298	13	70	30
(18)	298 - 300	123	60	18
(19)	300 - 310	13	70	20
(20)	310 - 320	123	60	8
(21)	320 - 330	1023	10	40
(22)	330 - 340	13	70	10

(23)	340 - 348	123	60	0
(24)	348 - 358	1023	10	30
(25)	358 - 368	13	70	0
(26)	368 - 378	1023	10	20
(27)	378 - 398	1023	10	0