Project: Scheduler
Group member: Jing Qian, Xinyi Zhang
Date: April 26 2016

Feature: 1. Implement 4-level feedback queue, mixed with lottery scheduler.
2. Implement zombie-cleaner mechanism in queue and random-number-generator in lottery.
3. Implement syscall statuler to check status of process in each queue
4. Implement test program “lazytest”

Bug unfixed: 1. priority of a process surpasses 3 at a very little probability.

Description:

1. Four-level feedback queue
   - Data structure
     Queues respectively contain process with priority 0,1,2,3, using data structure array.
   - Priority in different queue
     Any process is allocated in Q0, it is always scheduled before process in Q1 and so on.
   - Priority in same queue
     Processes in same queue will be scheduled by lottery. Winner uses CPU and run.
   - Time slice in each queue
     When a process is scheduled, it “clicks” the CPU. Process can run in a queue with its “Clicks limit” times.
   - Reset time
     Each process contributes to the total clicks. If total clicks reach 250, all unfinished process reset to Q0. When queues reset, system automatically shows as follow:.

	Priority	Clicks limit	Reset clicks	Size	First process pointer’s location
Queue 0	0	2	250	64	Q0[0]
Queue 1	1	4		64	Q1[0]
Queue 2	2	8		64	Q2[0]
Queue 3	3	No need		64	Q3[0]

Table 1. Four level feedback queue

2. Lottery mechanism
   - Initial lottery
     Each process is allotted lotteries at initialization.
   - Parameter
     1) Number of process within that queue.
     2) Totalclick (This is relative time parameter)
   - Winner
     Use “random number” to decide winner, and winner will use CPU with 1 “click”.
3. Zombie cleaner
   - Function
     Turn ZOMBIE process into UNUSED, remove the process from Queues.
     Free its stack and virtual memory for future use.

4. Syscall statuler
   - Function
     Use statuler to check the status of each process at process table.
   - Note
   - Use this function before there are more than 64 process running.
5. Test program lazytest
   - Function
     Fork several process and each will sleep for a while. This will help statuler to check the status of process.
   - Note
     Please run this in background. Namely type ”lazytest&”
     This is not a perfect test program. If a priority surpasses 3 at a very little probability, use this again when last program finish. For most time, it runs without mistake.

```
PID     P_Name          P_State P_Priority      P_Clicks        Reset   P_tickets
1       init            2       3               2               0       2000
2       sh              2       3               7               0       1500
5       lazytest        2       3               4               0       1200
4       lazytest        2       2               4               0       1250
6       lazytest        2       3               3               0       1166
7       lazytest        2       3               3               0       1142
8       lazytest        2       3               2               0       1125
9       lazytest        2       3               2               0       1111
10      lazytest        2       3               3               0       1100
11      lazytest        2       3               3               0       1090
12      statuler        4       2               1               0       1083
```

Plot 1. Scheduling process

This shows if there is any process in queue 3, no process in queue 4 can run.
Also, if there is any process in queue 1, no process in queue 2 or lower can run.

```
$ PID   P_Name          P_State P_Priority      P_Clicks        Reset   P_tickets
1       init            2       0               0               1       2000
2       sh              2       0               0               1       1500
5       lazytest        2       0               0               1       1200
4       lazytest        2       0               0               1       1250
6       lazytest        2       0               0               1       1166
7       lazytest        3       0               0               1       1142
8       lazytest        3       0               0               1       1125
9       lazytest        3       0               0               1       1111
10      lazytest        2       0               0               1       1100
11      lazytest        3       0               0               1       1090
```

Plot 2. Reset

This shows all process are re-arranged into Queue 0 when it reaches reset time. Note that it is an automatic print when the scheduler reset.

```
statuler
PID     P_Name          P_State P_Priority      P_Clicks        Reset   P_tickets
1       init            2       1               0               7       2000
2       sh              2       2               0               7       1500
38      statuler        4       1               2               7       1026
$
```

Plot 3. Finished status

This means all lazytest program have been finished.