1 Overview

DLXOS has been changed to fit the requirements for this lab. You will have to copy this new version of DLXOS (`~/Public/cs314/project2.tgz`) to see the changes.

1.1 Setup Instructions

As before:

```
mkdir ~/cs314/project2
cp ~/Public/cs314/project2.tgz ~/cs314/project2/
cd ~/cs314/project2/
tar xvfz project2.tgz
```

Make sure you have `/home/cs314/dlxtools/bin/` in your PATH.
Compile as before : type `make` in `~/cs314/project2/src` directory.
Run in `~/cs314/project3/execs` directory

1.2 DLXOS Modifications

DLXOS has been modified as follows:

- The prototype of create_process has been changed to facilitate passing in parameters p_nice and p_info. The new prototype is:

  ```
  void process_create(int p_nice, int p_info, char * args...);
  ```

  See the included user program for examples.

- The function to return the time elapsed in seconds since the simulator start is my_timer_get().

  ```
  uint32 my_timer_get(){
     uint32 temp = total_num_quanta * 100 + total_num_quanta * 1;
     return temp;
  }
  ```

  You can implement it as `TimerGet()` which is partially there in some of the scaffolding code.

1.3 Lottery Scheduling

The basic DLXOS scheduler is round-robin with preemption. This project asks you to implement a lottery scheduler. Lottery scheduling is probabilistic, meaning that we can only affect the probability with which a particular process is selected to run. Processes are each assigned some number of "tickets" and the scheduler draws a random ticket to select the next proces, therefore granting more tickets to a process provides it with a higher chance of being chosen to run. Having at least one ticket means that a process will not starve, as it has a non-zero probability of being chosen to run each time the scheduler's selection is made.

The DLXOS functions `srandom()` and `random()` produce the same result as the `srand()` and `rand()` from the standard library.

2 Assignment

- (20 points) Describe in detail how DLXOS schedules processes using the round-robin preemptive scheduler and the preliminary design of your lottery scheduler. Do this before you begin coding. Update the document with a description of your actual implementation, explaining how it is different from what you had envisioned at the start.
- (50 points) Implement the basic lottery scheduling in DLXOS by modifying process.c and process.h, and any other files necessary. At the very least, test the implementation with the userprog provided inside project3.tgz. You'll likely need to modify at least these functions: ProcessSchedule(), ProcessSuspend(), ProcessWakeup(), and ProcessFork(), in addition to whatever utility functions are necessary.

  Let every process start with a number of tickets proportional to p_nice and the number of tickets doesn't change throughout that process' execution(hence "basic").
- (30 points) Compare round-robin and lottery scheduling by showing the effect that priority setting in lottery scheduling has on the process' time on the CPU over time. Do this by profiling both the round-robin and lottery schedulers; keep track of which process is chosen for execution and compute the time taken to complete the execution of a process (from the time the process is created until it exits).

  Consider using p_info to control the collection of CPU timings (it's not used for anything, so you can use it to pass data to the forked process).

3 What to Turn In

Turn in the following as a group:

- A .tgz of your project directory, as before. (Any additional information that would be useful for grading: testing, configuration, compiling, etc should go into a README file.) The following should be included in your tgz, in addition to your code:
  - group.txt containing information about your group.
  - design.txt containing your detailed description of existing round-robin scheduling and preliminary and final design of your lottery scheduler.
  - profiles.txt containing a description and results of your timing experiments.

Individually:

- Each group member should individually turn in a text file containing a brief account of your group activity. Keeping in mind your group's dynamic, answer at least the following questions: What worked? What did not work? What were you responsible for in the project? What could be done differently next time?