In this project, you'll be putting a new scheduler into xv6. It is a mix of two schedulers: the **multi-level feedback queue (MLFQ)** and the **lottery scheduler**.

The basic idea is simple: Build a simple two-level scheduler which first places jobs into the high-priority queue. When a job uses its time slice on the first queue, move it to the lower-priority queue; jobs on the lower-priority queue should run for two time slices before relinquishing the CPU.

When there is more than one job on a queue, each should run in proportion to the number of tickets it has; the more tickets a process has, the more it runs. Each time slice, a randomized lottery determines the winner of the lottery; that winning process is the one that runs for that time slice.

Details

You'll need a couple of new system calls to implement this scheduler. The first is **int settickets(int num)** , which sets the number of tickets of the calling process. By default, each process should get one ticket; calling this routine makes it such that a process can raise the number of tickets it receives, and thus receive a higher proportion of CPU cycles. This routine should return 0 if successful, and -1 otherwise (if, for example, the user passes in a number less than one).

The second is **int getpinfo(struct pstat *)** . This routine returns some basic information about each running process, including how many times it has been chosen to run and its process ID, and which queue it is on (high or low). You can use this system call to build a variant of the command line program **ps** , which can then be called to see what is going on.

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, not much needs to be done; study its control flow and then try some small changes.

You'll need to figure out how to generate random numbers in the kernel; some searching should lead you to a simple pseudo-random number generator, which you can then include in the kernel and use as appropriate.

You'll need to understand how to fill in the structure **pstat** in the kernel and pass the results to user space.

Read Chapter 5 of xv6 book to have understanding of scheduling.

What To Turn In

Beyond the usual code, you'll have to **make a graph** for this assignment. The graph should show some timelines of processes running in each scheduler, including which queue they are on, and how much CPU they received when assigned some particular number of tickets. Use the graphs to prove that your scheduler is working as desired.