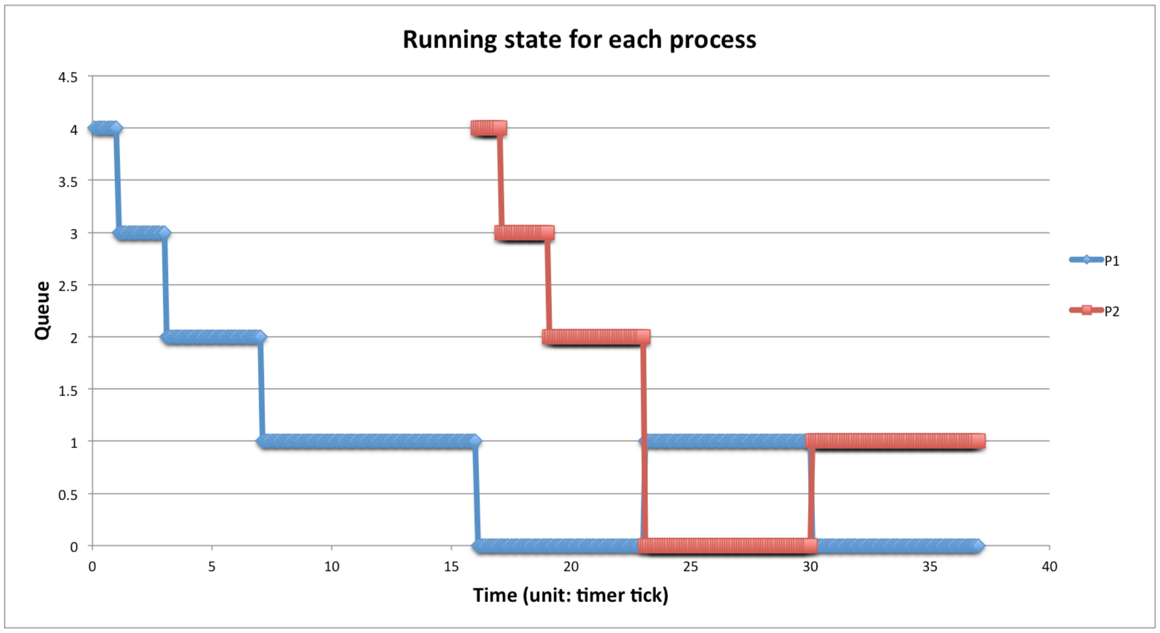
Author: Jin Ruan   cs login: jin   wiscmail: jruan3@wisc.edu

I have made up a very simple workload using a user program I scripted in xv6. In the program, I use a fork() to create a child process. And the parent process will do a counting from 0 to 1100 while the child process will do a counting from 0 to 1000.

It was quite weird that when the main() method just started, the parent process was already in the queue of priority 3, the lowest level. Perhaps it is because it takes time to do some initialization or preprocessing. And when the parent process spent 9 timer ticks in the lowest queue, the child process was created and was added into the queue of priority 0. The child process preempted the parent process and kept running until it was added into the lowest queue. After this happened, the parent resumes running and the two processes were scheduled in the manner of Round-Robin. The following is the timeline graph.



Please note that P1 (Blue) represents the parent process and P2 (Red) represents the child process. The child process is created at 16 timer ticks after the fork(). The vertical axis of the step chart means the process is running in a certain queue (i.e. 4: running in the top priority, 3: running in the second priority, 2: running in the third priority, 1: running in the last priority, 0: not in running state), the number notation of which is different from the program.