Online on xv6 - Scheduler

Section: B2 Time: 30 minutes

Your task is to implement a simple priority based scheduler with an aging mechanism. You are given a user program testloop.c, which simulates a long running job by iterating a loop a given number of times. This program takes two arguments: iteration count and priority. You will also need to implement the system call to set and get priority for each process.

The process with the highest priority will keep running till completion, unless another process with higher priority arrives. In that case, the higher priority process will run till completion. **All processes have a default priority of 1000**. Processes with the same priority will run in a RR fashion. If a process remains unscheduled for 30 ticks, its priority should be increased by 10. The waiting time for the process should be reset whenever it is scheduled or its priority is adjusted.

Sample I/O:

$ testloop 200 10 &; testloop 150 5 &;
See: output1.log
Notice how the two processes create an interleaving pattern while increasing their priorities. When one process runs, the other keeps waiting and eventually gets its priority increased. Then it gets the chance to run, and the other keeps waiting, gets its priority increased again and runs. This continues until both of the processes terminate.

$ testloop 200 7 &; testloop 100 5 &; testloop 50 3 &;
See: output2.log

Note:

Set CPUS := 1 in the Makefile. You can provide the input to the shell as given in the sample I/O for testing purposes. Don't worry about interleaving prints.

You should also print a line from the kernel whenever the priority of a process is increased. Be sure to reset the waiting time whenever a process gets scheduled, or the priority for that process is increased. Your output ordering is expected to create a similar pattern like the given ones (may not match exactly, but the interleaving pattern should be present).

Submission:

```
git add --all
git diff HEAD > ../2005010.patch
```