Operating Systems 0512.4402

Homework 4: Adding priorities to xv6 scheduler

xv6 includes a round-robin scheduler which gives all processes equal priority. In this exercise, we'll modify it so that it takes into account user defined process priorities.

We'll modify xv6 such that each process is assigned a 3-bit exponential priority `P`. Priority 0 is the lowest, and priority 7 is the highest.

By default, all processes start with priority 0, but the priority can be changed using new system calls. Add system calls to xv6 to set/get process priorities. When a process calls `setprio(P)`, the priority of the process should be set to the specified value. Also add the system call `getprio()` to read back the priority set, in order to verify that it has worked.

Change the xv6 scheduler to take the priority `P` into account. The priority indicates the base-2 logarithm of the maximum number of consecutive timer ticks the process may get during the round-robin scanning of the processes. As long as the process is runnable (not sleeping), it should be scheduled 2^P times in sequence before considering the next process. If before the process finished its 2^P ticks, it needs to sleep waiting on some event and is not runnable, it should not be scheduled and the scheduler should continue in the round robin scanning of the next processes.

For example, if `P == 0`, it should run for one timer tick like before. If `P == 7`, it should run for 128 timer ticks before selecting the next one (as long as it is runnable after each of those ticks and does not need to sleep), etc.

Add a user-space program, `demosched.c`, which will demonstrate the new scheduler priority. It should fork multiple children that run CPU bound code, set different priorities, and demonstrate that processes finish computation in accordance to the set priorities.

Submission guidelines

- The solution should be submitted in moodle in a gzipped tar file called hw4_id1_id2.tgz, where id1 and id2 are the "tehudat zehut" of the two students (or hw4_id.tgz if submitting alone).
- The tgz file should contain a subdirectory hw4_id1_id2, and in the subdirectory include all the files that you modified in xv6, and any new files that you added, so that when we copy those files over the original public xv6 directory, the kernel will compile using make, and when running it under QEMU with make qemu-nox, the new `demosched` user space command will be available in the filesystem.
- Submit an external documentation pdf in a file called hw4_id1_id2.pdf, summarizing your scheduler solution (both your kernel changes, and your user-space test program demosched.c).