Project #2

1. Before starting

In this project, you don't need to care Multi-Core/CPU environment. So, please fix your `Makefile` at line `85` from `CPUS := 2` to `CPUS := 1`.

2. Process Scheduler in xv6

In this project, you'll be adding a new scheduler to xv6, called a simple priority-based scheduler. The basic idea is simple: assign each running process a priority, which is an integer number. In this project, we have only two priorities, 1 (low priority) and 2 (high priority). At any given instance, the scheduler should run processes that have the high priority (2). If there are two or more processes that have the same high priority, the scheduler should round-robin between them. A low-priority (1) process does NOT run as long as there are high-priority jobs available to run.

2.1. Details

Here are detailed explanations about what you should implement.

- `int setpri(int num)`

  This call set the priority of the calling process. By default, all process should get a priority of 1; calling this routine makes it such that a process can raise or lower its priority.

- `int getpinfo(struct pstat *info)`

  This call updates some basic information about each running process (=variables in the `pstat` structure), including how long it has run at each priority (measured in clock ticks) and its process ID. 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.

If either of these calls are passed bad parameters, return -1 to indicate a failure. Otherwise, return 0 upon success.

2.2. Hints

You need to modify/add codes mostly in the following files:

- `include/syscall.h`
- `kernel/defs.h`
- `kernel/proc.c`
- `kernel/proc.h`
- `kernel/syscall.c`
- `kernel/sysfunc.h`
- `kernel/sysproc.c`
- `user/user.h`
- `user/usys.S`

Following file will be provided:

- `include/pstat.h`

You can download it from the KLMS. Please move this file to your "xv6/`include`" directory.

3. Deliverable

Create following two system calls in xv6 kernel.

- `setpri()`
- `getpinfo()`

Modify current process scheduler to priority-based scheduler.

4. Grading Criteria

We will use 10 scripts for grading. You can check the descriptions below.

awake_high_priority (10): Test whether setpri works well: We will count the number of high-priority processes

change_priority (10): Test whether setpri works well: We will repeatedly change priority of a process

check_hticks (10): Test whether it manages lticks and hticks correctly

default_priority (10): Default priority should be low

getpinfo (10): getinfo should return -1 when it gets bad pointer

multiple_high_priority (10): Check whether low priority process does not run while high priority processes are running

processesinuse (10): Test whether it correctly counts the number of processes in use

run_high_priority (10): Check the updated pstat variables while one high priority process is running

setpri (10): Check the return value of the setpri. setpri needs to return -1 for the wrong input

sleep_high_priority (10): Low priority process should be able to run while high priority process is sleeping

After finishing your implementation, compress your xv6 source codes using the commands below. Then, submit the compressed file on KLMS.

```
#First, move to a directory containing the xv6 directory
mv <DIRECTORY_CONTAINING_THE_XV6_DIRECTORY>

#Then, compress your xv6 directory
tar  -czvf  <Student ID>_proj2.tar.gz  xv6
```