CS4375-13948 Fall 2023
David Vasquez
Davasquez7@miners.utep.edu

Homework Report
Submitted on 10/20/23

HW 3: Priority-based Scheduler for xv6

Note: The pictures were taken after I worked on all the task, I will make sure take pictures as I go on next time

Task 1. Modify the provided ps command to print the priority of each process.

When defining system calls, the first thing you have to do is add them to syscall.c, users.h, and syscall.h to define them. We then create out functions in the sysproc.c, getprioirty is the simpler function that just returns the priority from my proc. And for the setpriority we initialize a priority int, from there we see if there is space for the priority and if there is we set the priority attribute on the proc struct to the size of that address and return 0, if it does not have any space return -1. Because procinfo returns the value of all the information of the given processes, we will initialize the priority attribute in this function so we can add it to the information that it returns.

```
$ QEMU: Terminated
david321239@ubuntu1:~/Desktop/Os2$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
1 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-devi
ce,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

init: starting sh
$ ps
pid     state           size    ppid    name    priority cputime age
1       sleeping        12288   0       init    0       0
2       sleeping        16384   1       sh      0       0
3       running         12288   2       ps      0       0       20
```

Task 2. Add a readytime field to struct proc, initialize it correctly, and modify ps to print a process's age.

The process has an attribute called readytime. It will initialize when the process is changed to running then in ps.c to get the age by getting the current time and subtracting by the amount of time it was in the runnable state(readytime).

```
$ QEMU: Terminated
david321239@ubuntu1:~/Desktop/Os2$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
1 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-devi
ce,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

init: starting sh
$ ps
pid     state           size    ppid    name    priority cputime age
1       sleeping        12288   0       init    0       0
2       sleeping        16384   1       sh      0       0
3       running         12288   2       ps      0       0       20
```

Task 3. Implement a priority-based scheduler.

Open ~/Desktop/Os2/kernel **proc.c** Save

param.h | proc.c | proc.h | pstat.h | defs.h | syscall.c

```c
//  - swtch to start running that process.
//  - eventually that process transfers control
//    via swtch back to the scheduler.
void
scheduler(void)
{
  struct proc *p;
  struct cpu *c = mycpu();
  struct proc *maxproc;
  c->proc = 0;
  for(;;){
    // Avoid deadlock by ensuring that devices can interrupt.
    if(scheduletype == 0){

        intr_on();

    for(p = proc; p < &proc[NPROC]; p++) {
      acquire(&p->lock);
      if(p->state == RUNNABLE) {
        p->state = RUNNING;
        c->proc = p;
        swtch(&c->context, &p->context);
        c->proc = 0;
      }
      release(&p->lock);
    }
  }else{
            int maximum_process = 0;
            maxproc = proc;
        for(p = proc; p < &proc[NPROC]; p++){
            acquire(&p->lock);

            if(p->state == RUNNABLE){
                int age = sys_uptime() - p->readytime;
                if(p->priority + (age) > maximum_process){
                    maximum_process = p->priority + (age);
                    maxproc = p;
                }
            }
                release(&p->lock);
        }

        intr_on();
        acquire(&maxproc -> lock);
        if(maxproc->state == RUNNABLE) {
        // Switch to chosen process.  It is the process's job
        // to release its lock and then reacquire it
        // before jumping back to us.
        maxproc->state = RUNNING;
        c->proc = maxproc;
        swtch(&c->context, &maxproc->context);
        // Process is done running for now.
```

C Tab Width: 8 Ln 526, Col 17 INS

We change the schedular method in proc.c and we add the scheduling types in param.h.
In proc.c we had edited the scehdular method to include a priority based schedular. We will now have two types of schedulars priority and roundrobin. If the type is 0 it is going to be round robin otherwise it will priority-based scheduling. I have learned how to implement a scheduler

into my operating system codebase by making modifications to the proc file and adding scheduling types in the param.h header. A challenged I faced was in the actual implementation of the priority scheduler as well as some annoying debugging.

Task 4. Add aging to your priority-based scheduler.
Our policy is that that if our process plus age is greater than maximum processes, which is declared as a local variable that holds the process plus its ageing, will define if that processes will be running next. I learned how to calculate age then using the processes ages and in this case using it in a priority scheduler. Some difficulties I ran into was navigating through the xv6 directories to make sure that all of the attributes were properly passing their values.

```
david321239@ubuntu1:~/Desktop/Os2$ make qemu
qemu-system-riscv64 -machine virt -bios none -kernel kernel/kernel -m 128M -smp
1 -nographic -drive file=fs.img,if=none,format=raw,id=x0 -device virtio-blk-devi
ce,drive=x0,bus=virtio-mmio-bus.0

xv6 kernel is booting

init: starting sh
$ ps
pid     state           size    ppid    name    priority cputime age
1       sleeping        12288   0       init    0       0
2       sleeping        16384   1       sh      0       0
3       running         12288   2       ps      0       0       20
$ pexec 5 matmul 5&; matmul 10 &
$ pexec 10 ps
pid     state           size    ppid    name    priority cputime age
1       sleeping        12288   0       init    0       0
2       sleeping        16384   1       sh      0       0
8       runnable        12288   6       matmul  0       0
7       runnable        12288   1       matmul  0       0
6       sleeping        12288   1       pexec   0       0
9       sleeping        12288   2       pexec   0       0
10      running         12288   9       ps      0       0       24366
$ Time: 60 ticks
Time: 122 ticks
```