Diego Jared Avina

Dr. Moore

CS4375-13948 Fall 2023

HW 3: Priority-based Scheduler for xv6

Task 1: Modify the ps Command:

In task 1, we were told to modified the **ps** command to print the priority of each process. This mainly involved changing the **ps.c** file located inside the **user** directory. In order to retrieve the priority for each process, we created two new methods **getpriority** and **setpriority**.

Results of ps command task 1

```
init: starting sh
$ pexec 10 ps
pid     state   size    ppid    name    priority
1       sleeping        12288   0       init    0
2       sleeping        16384   1       sh      0
3       sleeping        12288   2       pexec   10
4       running         12288   3       ps      0
```

Task 2: Add a readytime field and Age Calculation

In task 2, we needed to add a new field named **readytime** to the **struct proc** and modified the **ps** command located inside the **user** directory to print a process's age. In order to calculate a process's age we updated the **readytime** field whenever a process became **RUNNABLE**.

The age calculation is done as follows:

Age = current time (uptime()) – process.readytime;

Results:

```
init: starting sh
pe$ xec 10 ps
pid     state   size    ppid    name    priority        cputime age
1       sleeping        12288   0       init    0       0
2       sleeping        16384   1       sh      0       0
3       sleeping        12288   2       pexec   10      0
4       running         12288   3       ps      0       0       -84214978
$ QEMU: Terminated
```

In this case, for my results, I only get one have and I get a negative number. Im not sure what failed in my code because every time I ran the readytime by itself it returned that same number but positive.

Task 3: Implement Priority-Based Scheduler

In task 3 we implement a priority-based scheduler. The primary changes were made inside the **proc.c** file, more specifically the **scheduler** function; there were a few changes in other files like **param.h**. After running a few tests in the programs using the priority scheduler, we observed that processes with higher priorities indeed run first.

What I learned:

Implementing a priority-based scheduler helps in prioritizing tasks based on their importance. This task has shown us the significance of process priorities in managing system resources efficiently.

Difficulties:

I encountered multiple challenges going making the scheduler “work”. I had to carefully debug and understand the xv6 codebase.

Task 4: Adding Aging to the Priority-Based Scheduler

In Task 4, we added aging to our priority-based scheduler. Our aging policy includes periodically increasing the priority of all processes in the system to prevent lower-priority processes from being starved.

Difficulties:

Implementing the aging policy required a lot of dedication and coordination with the current scheduler code I had. Debugging and thorough testing were necessary to ensure that some of the code worked correctly.

```
xv6 kernel is booting

init: starting sh
$ pexec 5 matmul &; matmul 10 &
$ Time: 14 ticks
Time: 66 ticks
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       sleeping        12288   2       pexec   10      0
9       running         12288   8       ps      0       0       33751
$
```