Report for XV6 Scheduler

Amey Choudhary
2021113017

A. Implementation of various scheduling algorithms:

a. FCFS (First Come First Serve)

In FCFS, we run the process which arrived first. To implement this, we would have to maintain the arrived time for each process and then select the process with the earliest time.

In our code, we do this by:

1. The process struct already contained a variable, “ctime” which was initialised on creation and set to ticks. This allowed us to maintain the order of arrival.
2. We then run a loop (in proc.c) to check all processes and select the one with the lowest ctime. This process is then run in the scheduler.

To disable preemption, we can not give back “yield()” in FCFS (when timer interrupts in trap.c). To do this, we disabled it for FCFS.

b. MLFQ (Multi-Level Feedback Queues)

We had to implement a preemptive MLFQ. In this, all processes are first queued into a high priority queue and run. In case any process takes more time than the specified time for the queue, it is interrupted and sent to lower priority queues. Also, to prevent starvation (hogging of CPU by higher priority queues), we do ageing i.e. we check if a process has stayed in a queue for more than the specified wait time. If yes, it will be boosted to a higher order queue.

In our code, we do this:

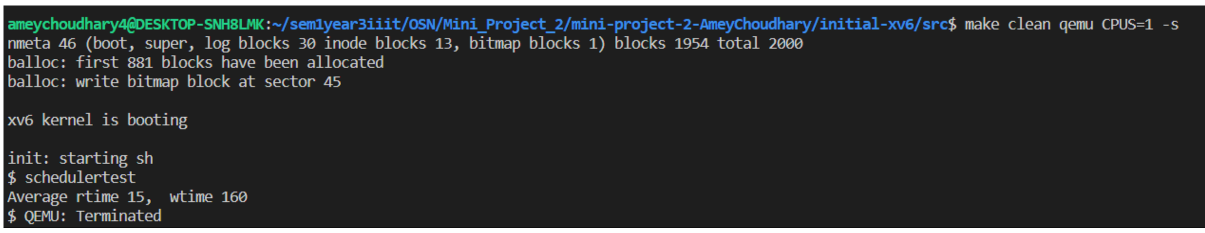
1. Maintain new variables in process, which store the current queue number, entry time, last run time in queue and run time in particular queue (in proc.h)
2. Initialise these in allocproc in proc.c when the process is created. The process will be assigned the highest order queue on initialisation.
3. For the scheduler in proc.c, we iterate through the runnable processes to find the one which is either higher priority than current or entered the queue before the current.
4. We also made changes to wakeup and kill for when the process relinquishes CPU for I/O or is killed by usertrap to leave the system.

5. In trap.c, we ensure that any process, if has waited more than the threshold time (30 ticks in code), is boosted to a higher priority queue. Also, we check if the current process has utilised the time splice of the queue it is in or if any previously boosted process has level higher than current. In either case, if true, we make the current process yield().

B. Performance comparison

We run for 1 CPU and different schedulers

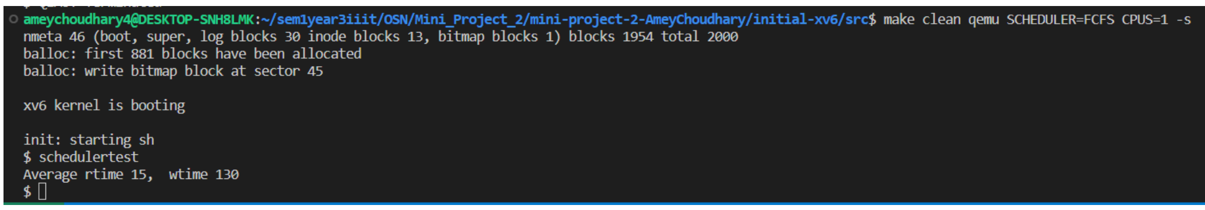
1. Default (Round Robin)



```
ameychoudhary4@DESKTOP-SNH8LMK:~/sem1year3iiit/OSN/Mini_Project_2/mini-project-2-AmeyChoudhary/initial-xv6/src$ make clean qemu CPUS=1 -s
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 881 blocks have been allocated
balloc: write bitmap block at sector 45

xv6 kernel is booting

init: starting sh
$ schedulertest
Average rtime 15,  wtime 160
$ QEMU: Terminated
```

Average rtime: 15, wtime 160

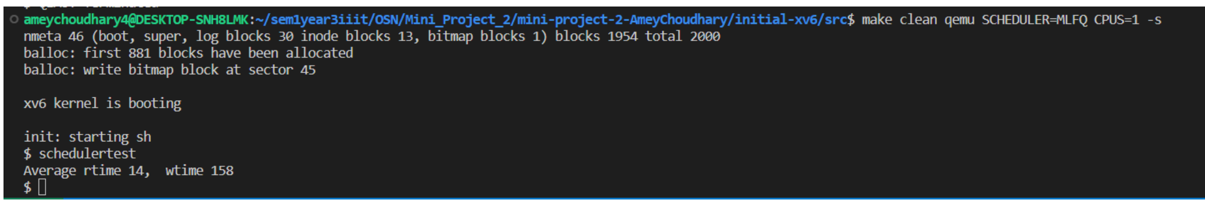
2. FCFS



```
ameychoudhary4@DESKTOP-SNH8LMK:~/sem1year3iiit/OSN/Mini_Project_2/mini-project-2-AmeyChoudhary/initial-xv6/src$ make clean qemu SCHEDULER=FCFS CPUS=1 -s
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 881 blocks have been allocated
balloc: write bitmap block at sector 45

xv6 kernel is booting

init: starting sh
$ schedulertest
Average rtime 15,  wtime 130
$ 
```

Average rtime: 15, wtime 130

3. MLFQ



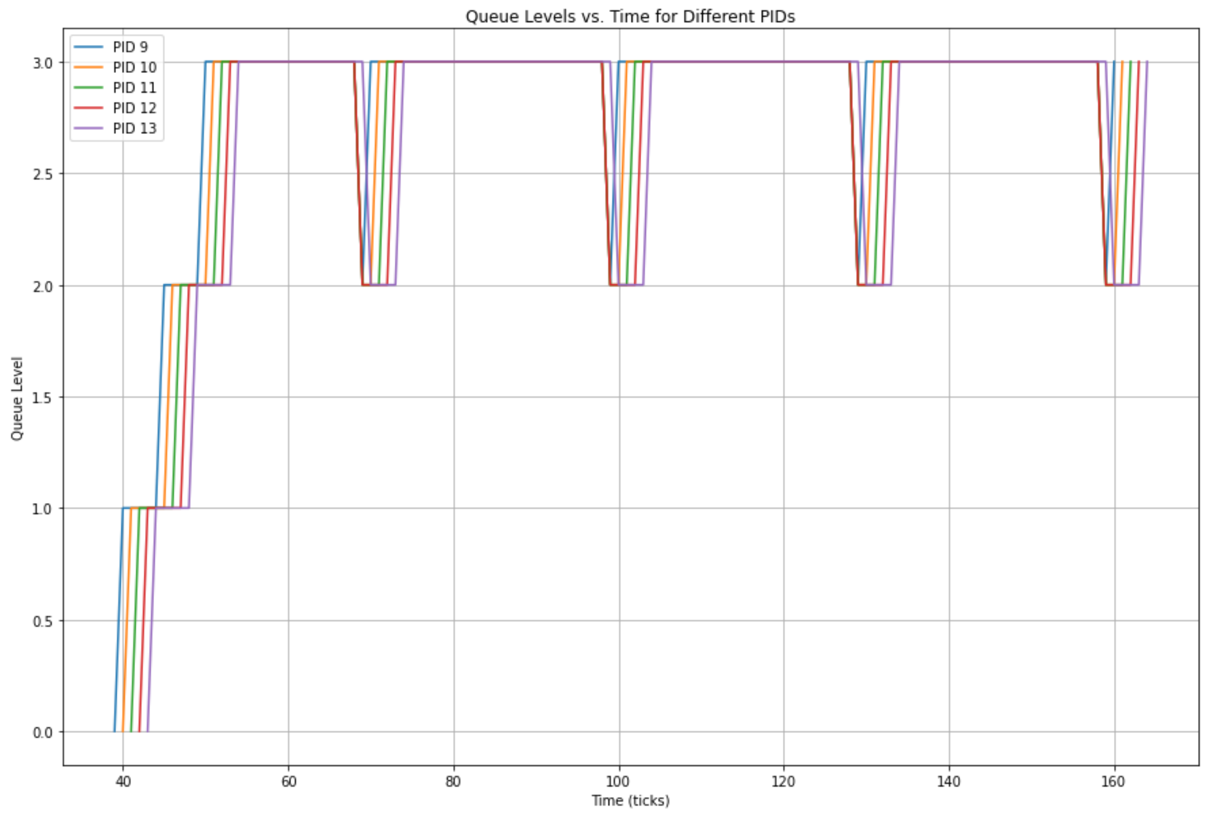
```
ameychoudhary4@DESKTOP-SNH8LMK:~/sem1year3iiit/OSN/Mini_Project_2/mini-project-2-AmeyChoudhary/initial-xv6/src$ make clean qemu SCHEDULER=MLFQ CPUS=1 -s
nmeta 46 (boot, super, log blocks 30 inode blocks 13, bitmap blocks 1) blocks 1954 total 2000
balloc: first 881 blocks have been allocated
balloc: write bitmap block at sector 45

xv6 kernel is booting

init: starting sh
$ schedulertest
Average rtime 14,  wtime 158
$ 
```

Average rtime 14, wtime 158

As we can see, FCFS has the lowest wtime. Also, the rtime is nearly the same.

C. MLFQ Analysis

I created the time variate plot for different pids. The following graph contains the pids for 1 CPU and 30 ticks for aging. It shows the transitions among queues for different pids. The time slices are as mentioned in the document.



The Jupyter Python Notebook as well as the input for it are in the Reports folder.