Implement Scheduler

Connor Dubois, Jim Harrell, Daniel Hoog, Eric Pereira
CSE4001

December 4$^{\text{th}}$, 2018

Contents

1 Design

The design of the new scheduler is a simple first in first out (FIFO), which replaces the basic round robin system already implemented inside the operating system os161. First in first out acts a lot like a queue data structure, where a process is given to the CPU and the process will not stop until it is finished. A round robin will switch between the processes continuously, executing multiple processes at once and switching between them with a specified time. Here is an example:

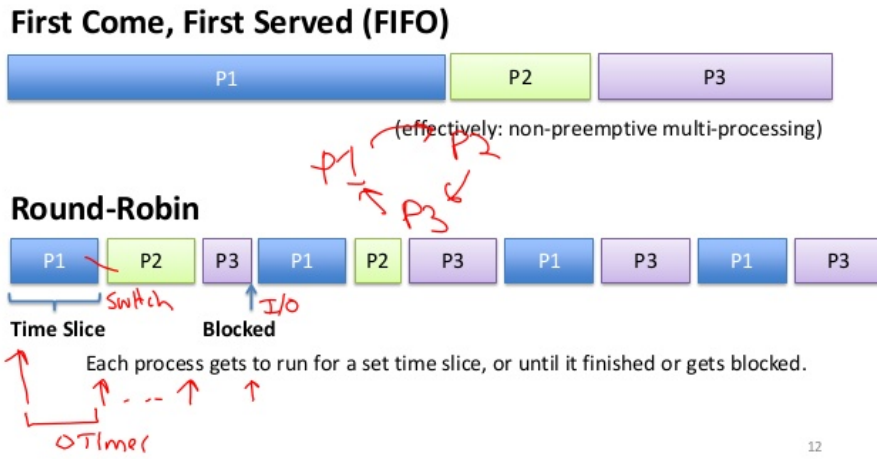

Figure 1: FIFO vs Round Robin

2 Implementation

The implementation of the new scheduler was actually really simple. Instead of creating something new in the scheduler function all that was needed to do was go the the clock.c and change the hardclock function so that it prevents the current scheduler from setting a time and switching processes. What this does is prevents it from changing This makes it a first in first out, where the scheduler will not change unless the process it is running finishes.

3 Benchmark

3.1 Round Robin Tests:

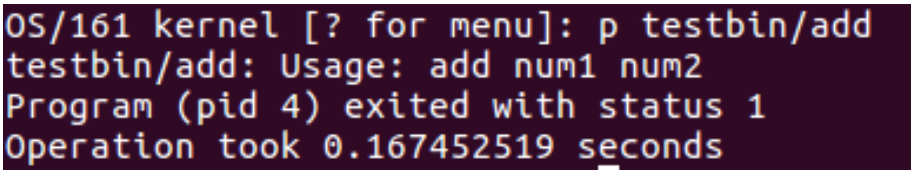

Figure 2: Round Robin Add

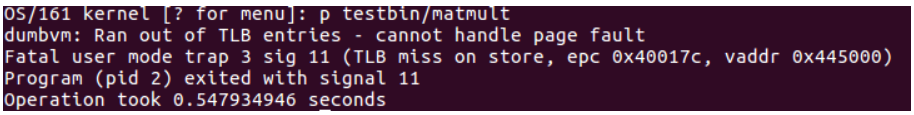

Figure 3: Round Robin Matmult

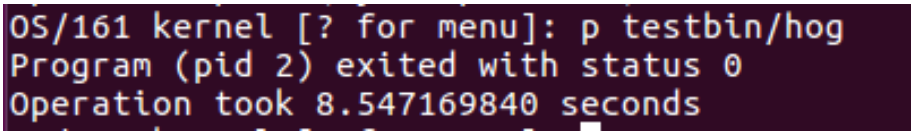

Figure 4: Round Robin Hog

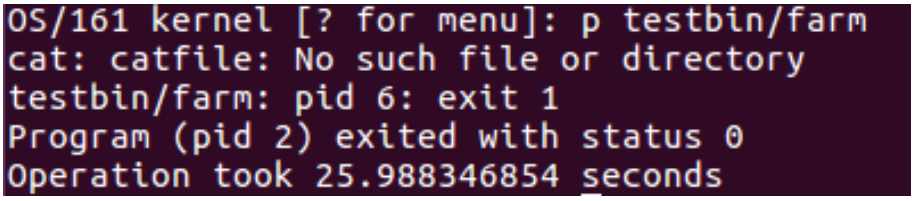

Figure 5: Round Robin Farm

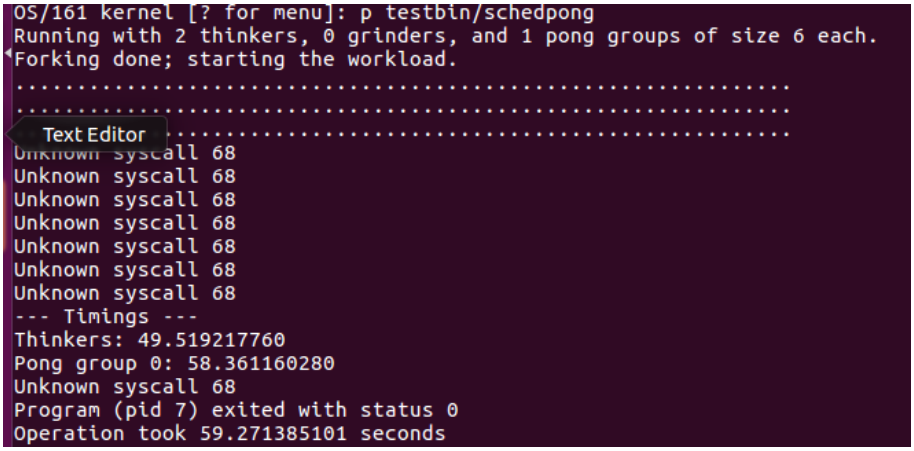

Figure 6: Round Robin schedpong

3.2 FIFO Tests:

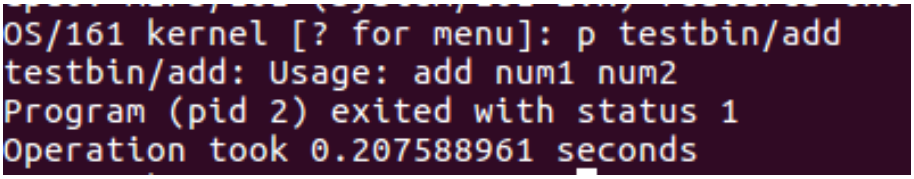

Figure 7: FIFO add

Figure 8: FIFO add

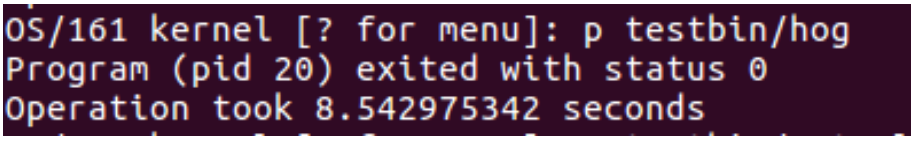

Figure 9: FIFO hog

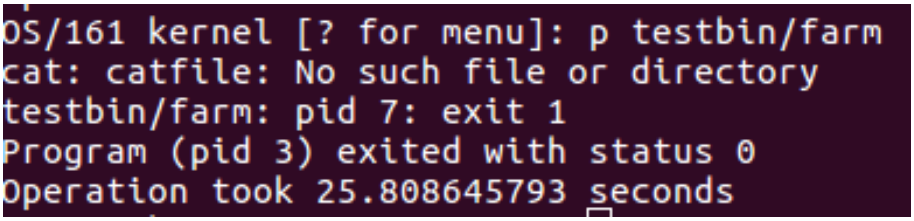

Figure 10: FIFO farm

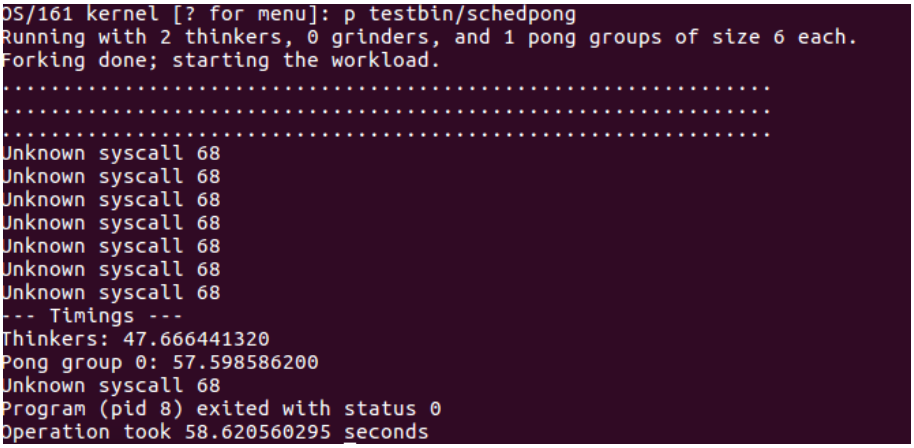

Figure 11: FIFO schedpong

3.3 Discussion

There were some strange errors that we got on some of the operations, using the clone github and recompiling and setting up os161 with the source given we recieved a few errors. First being that `p testbin/schedpong` would have strange `unknown syscall 68` errors. Upon further review it looked like syscall 68 was the `sys_remove` call. When grep searching through the source we were not able to find another `sys_remove` specified anywhere else other than the syscall.h file. There was also an issue with the `p testbin/matmult` test, it seems that os161 could not handle the page fault. We gave os161 about 16K of memory. We started at 4K, but went up to see if memory was the problem, it was not. We included the photos in our benchmark tests, but they should not give any insight as to the scheduler's time. We also had a strange error on the farm, there seemed to be no `catfile`, one of the files that pertains to the farm, the tests still show some time difference.

Overall the results show that adding is significantly faster on round robin, at about .04 seconds faster. The matmult test, although maybe an inaccurate test, shows that FIFO is faster by .08 seconds. FIFO also ends up being faster in the hog test, at .05 seconds faster the FIFO farm test is faster than round robin at .19 seconds, and FIFO is faster than round robin at the schedpong test with about 0.65 second difference.

The tests actually show that typically it seems that FIFO is overall quicker than round robin, and according to figure 1 in the inital design portion this makes sense. Due to the scheduler not wasting time on switching it will overall be quicker, however this does not give these tests do not show the whole picture. Round-robin is used in order to allow for multiple processes to run at once, so that certain things can finish, or occur earlier than usual, and it prevents long programs from spending a large bulk of time performing a task, preventing others from doing a task.