Project 2 Write Up

Lawrence Chau

October 27, 2014

What do you think the main point of this assignment is?

I think the main point of these assignments so far is to help us continue get more and comfortable with the Linux kernel source code. This includes getting familiar with the location and content of specific files, like the I/O scheduler in this case. This was not something we could have easily jumped into without reading the book and understanding how schedulers behave. The frustration of having to recompile such a large image over and over again, no matter how small the change may be, helped us develop a firm understanding behind the workings of the scheduler.

How did you personally approach the problem? Design decisions, algorithm, etc.

In order to approach this problem, we had to first understand the Noop scheduler, Shortest Seek Time First schedulers, and how they prioritize I/O requests. From a bit of reading, I learned that the Noop scheduler takes the next request from the beginning of the queue and dispatches it, and appends requests to the back of the queue. It works almost like a FIFO. Shortest Seek Time First takes the position of the head, the position of the request, and the direction of the head in order to find the closest file to dispatch

I personally first searched for and researched the concept of a "Shortest Seek Time First" scheduler. After I got a basic understanding that it is essentially a FIFO algorithm prioritizing I/O request closest to the read write head. After looking at the current schedulers installed on the computer with the command *cat /sys/block/queue/scheduler* I noticed that the Noop scheduler was installed by default. I then copied the Noop scheduler and adjusted the algorithm to add a descending prioritisation. Most of the structures used were already defined in the Noop Scheduler so transitioning was fairly simple.
My team and I devised a solution that will go through the entire queue and find which I/O request has the shortest seek time.

How did you ensure your solution was correct? Testing details, for instance.

We tested our solution by taking advantage of the *printk* function to print debugging information. With the debugging print statements in place we copied a folder with multiple files into a new location and analyzed the output via serial.

What did you learn?

I learned how to make an I/O scheduler which I thought would have been much more complicated. I did however take a look at the CFQ scheduler and notice that it is much more complex than the simple FIFO based algorithms I took advantage of. I also learned how to make a changes to source code and make the Makefile only re-compile my changes by deleting the '.o' file.