CS444 - Homework Writeup # 2

May 5, 2017

Group 10-05

Jason Klindtworth — Brandon To — Rhea Mae Edwards

Abstract

In this second homework assignment, we are to implement the LOOK I/O scheduler by using the yocto Linux kernel sources and then testing our solution on a virtual machine. We are creating our elevator algorithm based off the implementation of the FIFO (NOOP) given to us to use.

PROJECT 2: I/O ELEVATORS

0.1 Design Plan of SSTF Algorithms

The following is the design we planned to use to implement the SSTF algorithms.

- Differences between NOOP and SSTF...
  These two scheduling algorithms are very similar to one another. The main difference between the two is how they add requests to the queue, which is further explained in the next step of our plan.

- Differences between function modifications, NOOP and SSTF_ADD_REQUEST...
  Within the the two different files of the NOOP scheduler and the SSTF scheduler, the function of sstf_add_request differ. The differences between them is based on the idea on how a request can either be added after or before to following request within the queue for the implementation of the SSTF scheduler. Whereas for the NOOP scheduler, a request is just added at the tail of the queue, or at the end before.

- Why SSTF is better than NOOP...
  Benefits of have the SSTF scheduler versus that NOOP scheduler, is the flexibility given by the implementation of the SSTF scheduler. Being able to add requests from either behind or in front of another request, can help move the flow of the data within the queue quicker or smoother through the queue, and also with completing the task that has to be done.

- Other Stuff...

0.2 Version Control and Work Log

We were given two weeks to work on our assignment. During the first week, we spent a lot of our time trying to figure out what exactly we were suppose to do... It was a struggle. We either got together and tried to solve an issue or multiple issues together, or once we felt comfortable to focus to separate parts individually, we did that, splitting the work fairly and in order to complete what we had to get done, and that is what the second week mostly consisted of. Either if each of our tasks were to do further research in order to bring back to the group the next time we would meet up, or write a file that would test our code to see if our scheduler works properly on the virtual machine, or to generally make further progress with out separate concurrency assignment or implementation and further setup on and with the kernel.

Our version control log was as used as often as it should have been, due to the struggle of completing our work in an odd fashion, and also at times working on it together, where agreements with creations and changes were done collectively as a group in person. But as a group as a whole, we kept one another up-to-date when we were apart by communicating through Google Hangout, which also helped us know what should be worked on and who was working on what. For out group, this was an efficient system in order to complete our assignment overall.

0.3 Questions

Question 1: What do you think the main point of this assignment is?

We believe the main point of this assignment was to actually be exposed to using the kernel we made and learning how to implement a solution on a virtual machine. As well as understanding how to write an elevator algorithm that would be valid to use and also knowing how and being able to work with afterwards.

Question 2: How did you personally approach the problem? Design decisions, algorithm, etc.?

Personally, just simply approaching the problem required a lot of research, talking to others, and exploring the idea of what we were actually suppose to do. Just to getting started was very difficult, because we pretty much all have no idea what to do, we were lost. So by working with what we were given and by outside knowledge just to understand what we had to do, was a challenge within itself to get started.

Once we had an idea on what we had to do, we started looking around with what we know what we had to play with, and focused and worked with it. Again, with more information for the internet, notes from the TAs and instructor during lecture, we worked on developing our scheduler, algorithm, and solution for the assignment. Design wise, it was more after fully understanding the concepts, we worked on our scheduler and the elevator algorithm for the problem, and then testing it with our solution on the virtual machine.

Question 3: How did you ensure your solution was correct? Testing details, for instance?

In order to test to ensure that our solution was correct, we first wanted to check that our scheduler that we wanted to use to implement our solution, was actually being used of the virtual machine, because we were unsure when we first

started. Our default setting were correct and the file for our scheduler was created, but it did not seem like it was being used. To test this, we commented out the entire file that we were working with, and saw if our virtual machine would work properly, which it shouldn't, but by doing so, it still did, so we did some further investigating, and solved that issue.

Another method that we conducted in to ensure the correctness of our solution, is that we created a simple I/O file to test out our scheduler. A simple file that would just read and write data. And then we also wanted to see what was happening, so we used printk to print actions on the command, printing that information and data out on the screen, which also went into a file, that collected the output results, in order to be saved and to be able to looked at for later use. By doing so, we were able to monitor the behavior of what we created, testing how correct our solution was. By analyzing its behavior, we were able to decide what needed to be fixed or what was perfectly okay based off the output of these test.

Question 4: What did you learn?

Along with creating our elevator algorithm and working with it on a virtual machine, we learned how to actually develop an algorithm by using the yocto Linux kernel sources and by implementing the LOOK and CLOOK I/O schedulers in the creation of our solution. With also further understanding those tools and concepts even further, in order to fully grasp the idea of what we are doing. Such as how the algorithm and schedulers work with the system, how to edit and create functions in order to do what is suppose to do, especially when it comes to the merging characteristics of the given behavior, and then putting the programs and kernel all together, in order to later test it later on a virtual machine.