Summary

The main goal of this project was to implement three scheduling algorithms, [RR, SRT, and FB]. Since the implementation of the project was left completely open ended, I found it quite simple and almost enjoyable to complete. Blindly following a format and then learning how to use a certain way to format classes while coding is not enjoyable and at often quite tedious. I do understand this method of teaching, however, since it would be difficult for students to learn how to produce good code without learning from example and standing on the shoulders of giants.

Seeing as the implementation was open ended, I created a class named Process (implementing java.io.Serializable) containing a String name, int startTime, and int duration. The reason Process needs to implement Serializable was so that I could produce deep copies of Process[] when the input parameter is ALL. My scheduling algorithms decrement the duration remaining of the process when producing a result, requiring the creation of new instances of the Process[] array to run all three with a single input. This was the most difficult part of the project for me, requiring multiple searches through web forums and stackoverflow-like websites to find a way to deep clone an object. Most of the solutions I found were shallow copies or would deep copy primitive data types but only produce a shallow copy for objects. Once I found the method to deep copy objects, it was smooth sailing from then on out.

Back to the scheduling algorithms, they were very easy to build and I don’t really understand what the complexity about this project was supposed to be. Debugging was very simple, and no problem took more than five minutes to solve when it came to making them work initially. Seriously, how in the world did this meet the cutoff for projects, compared to the first and second projects that I struggled with this project was a complete cakewash. I understand that the second project was also considered easy, but the concept of semaphores was fairly difficult to understand and took me quite some time to understand and debug until it was working properly. Actually, I’m not too upset about the difficulty of the project, since I could finish it quickly and then move on to more pressing homework.

I’m really having difficulty coming up with more insights to write about regarding Project 3, mostly because I felt like I was not challenged during the process. This is a new feeling, instead of slowly working through numerous bugs I can get the feeling of having no serious problem whatsoever. A few null pointer exceptions proving that I messed up the loop control variable, a few infinite loops that prove I forgot to increment the counter, or some other silly mistake. The existence of the List<> class with ArrayList<> and Queue<> functionality was very helpful for this project, but even if they didn’t exist these functions aren’t particularly difficult to build by themselves.

I found JavaFX to be very confusing and not seem to perfectly match the requirements necessary to build a graph. If I wasn’t allowed to do a text-based graph I would have chosen to use Swing. As of the moment where I am writing this, I completed a text-based graph output that meets requirements and would be perfectly happy to not improve my project to include the graph using a graphics package.

Checking back in on the day of the deadline, had difficulties accessing the ECS lab yesterday (why is it closed on Sunday, darn it) but managed to get back in and test my code today. It seems like it might be ideal for me to get my own linux environment for testing on my own, and then test on the cslinux1 server after I have configured all my changes. That way I would not have to work on the project at the absolute last minute (day) and panic slightly about the work. What a pain, really. What follows is the lines from testing my code on linux.