Hari Kiran Jana
HXJ190018
4348.501
12/09/2022

Project #3 Summary Report

The main goal of the scheduling algorithm project was to help us understand the way that each algorithm behaves and works. Algorithms that were explored in the project include: Round Robin (RR), Shortest Time Remaining (SRT), and Feedback (FB). Although, my theoretical understanding of the algorithms was solid implementation proved to be a challenging and time consuming. During, the project even though I understood the concept in my head I was unable to properly implement in practice. I had to go through many trials and errors in which some of them did not come to fruition. I first implemented a very inefficient method that required a lot of hard coding and then thought of ways to implement modularity. My first version was all in one class and each of the algorithms were functions that were implement using a job class that initialized all the inputs given to us in the jobs text file. I then started splitting the classes and created a Main, RR, FB, and SRT. The main class runs the other classes, and each class has an implementation of the algorithms. I used araylists, queues and Boolean arrays to store jobs, move jobs, and to print the results of the algorithm. There also a Job class that is inside the main to initialize each job to the specific inputs given. I have getter and setter methods to help me call information to other classes. The FB class uses three queues, and I also used an array list to hold all the Job objects that were initialized to each of the jobs, I first send the arrived jobs to ready queue and then run the first job, if another job arrives then I move the already running job from the high queue to mid to low. And since the quanta is one preemption only happens when there is a new job that arrives. For RR, my approach was similar but instead of three queues I only have on ready queue and when job arrives the current job in queue gets timed out by a quantum of 1. And when the service time is over than the job is removed from the queue completely and is gone from the array list. For SRT, the approach I took was the same as RR, but instead of preempting at time out, before a job runs, I check the shortest service time, and run the job that has the shortest and do this in repeat. Once the service time runs out, I remove the job from the jobs array list. The way I printed my algorithm, using the vertical way and the way I formatted is X means the Job was ran and "." means it didn't. This was used to mainly so that I can see the difference when two jobs arrive at the same time, we can properly see when there is a gap between jobs arriving. These two criteria was met and properly implemented.

The learning curve for me in this project, was to work with queues, and array along with coming up with proper algorithms that match the description of the project. A problem that I incurred was that I did not realize that running ALL would change the values of my Jobs and that I would need a way to reset them back to normal. I did not realize this early as I only individually tested the three algorithms and did not test ALL multiple times. This was great learning experience as I had to think out of the box on how I would solve this issue. I came up with a reset method that would change the service times back to their original position. Something I found interesting in the project was that I really enjoyed coding this in java, as my

Hari Kiran Jana
HXJ190018
4348.501
12/09/2022

initial approach was in C++. There were too many problems that came along with dealing with vectors. When I wanted to pass a vector through a header it posed a lot of problems as apparently, we cannot pass them and would need a much newer version.