Cal Poly Pomona

CS431 Project 1 Report

Anthony Lackey
Professor Dominick Atanasio
CS431
19 October 2017

In this project I was tasked with implementing four different CPU scheduling algorithms using C++, Java, or C# and I chose C#. I was able to implement all of these scheduling algorithms, which include First-Come-First-Serve, Shortest-Job-First, Round-Robin with time quantum 30, Round-Robin with time quantum 60, and Lottery. I was given four test data files which had different patterns of values. Each of these files has a different amount of processes, number of small burst times, and number of high burst times. I will compare these differences with the different algorithms.

I will analyze the results from the each of the test data sets and then go over an overview. The first data set has six processes with burst times less than 300 and five that have more than 900 burst time which gives it a wide range. For first-come-first-serve, the average turnaround time was 6006 which was average when comparing it to the other test data. Shortest-job-first was actually much faster than the first algorithm when comparing the average turnaround time which was 3230. This shows that this algorithm works pretty fast, but if it is a large array of values which you have to sorts, it could slow down the overall operation. Round-robin with time quantum 30 had an average turnaround time of 6440 which the CPU time being relative to the average turnaround time. For round-robin with time quantum 60, the average turnaround time is 6305. This was just slightly less than when time quantum 30 was used, and I think this is due to us removing twice as much from each burst time. The difference is not that much and this is probably due to the difference in time quantum's being small compared to most of the burst times in the scheduler. With lottery, the average turnaround time is 7087. This is actually much lower than the rest of the lottery's using the other test data, and this is due to randomness and most of the large values being at the beginning of the queue/scheduler. The rest of the scheduling algorithms had the same pattern, and the only algorithm that is really random and dependent on

order of data is the lottery scheduling algorithm. Lottery with test data 3 had the largest average turnaround time at 12053 and the other 2 were about 10000. These values change every run so these are just from one of the times I ran the program. Looking at the data, test data 3 had the largest average turnaround time due to it having more processes than the others and the larger numbers being in the last half of the list.

All in all, in each run there was a clear best average time and a clear worst. If you do not take into consideration the time it takes to sort the array, the shortest-job-first scheduling algorithm is definitely the fastest with lottery being the worst because it is random and just the way it works. The other algorithms are around the same turnaround time and are not a bad choice, especially if shortest-job-first is not optimal due to the requirement of sorting the array.