Tony Mai, Ryan Chen, Jiayi An

CSC 332

CPU Scheduling Algorithm

Scheduling algorithms is used to show how much CPU time is used and allocated to processes and threads. There are many CPU scheduling algorithm like first come first serve (FCFS), shortest job next (SJN), round robin with a time quantum, and many more. For this project we will primarily focus on FCFS and round robin with a time quantum of 10. A CPU scheduling algorithm can be preemptive or non-preemptive. In a preemptive algorithm the process can be preempted, it allows the process to jump back and forth from ready state and running state. As for non-preemptive, the first process entering the running state must finish running before another process can enter. Preemptive scheduling algorithm allows fairness and every process gets a turn to be in the running state.

The objective of CPU scheduling algorithm is to maximize the CPU utilization that keeps the CPU busy. Creating a fair allocation of the CPU and minimize the turnaround time, waiting time, and response time. By lowering the turnaround time it allows a process to finish faster, as for waiting time it allows the time of a process in the ready queue to be lower. Lastly, a lower response time allows a lower first response from the first process. If a CPU scheduling algorithm is able to do the objectives stated above, it will create a faster and more efficient usage of the CPU. Our main goal is to keep our CPU as busy as possible and to process as many jobs as possible efficiently.

In CPU scheduling we have process scheduling and schedulers. Job queue keeps all the processes to run in the system. Ready queue keeps all the processes residing in main memory waiting to be executed, and device queue are processes that are blocked due to the unavailability of the I/O device. As for the scheduler, we have long-term, short-term, and medium-term schedulers. The long-term admits a program into the system as a process and the short-term changes the process from ready to running state. The medium-term is used to remove a process from memory.

Focusing on the CPU scheduling algorithm FCFS, it is considered the simplest algorithm that schedules according to the arrival time of the process. It is non-preemptive, so therefore it cannot be preempted. The advantage of FCFS is that the jobs are executed on a first come, first serve basis, but the disadvantage is that if there are short processes in the back and longer process in the front; you would have to wait for the longer process upfront. As for round robin, each process is processed in a fixed time called a time quantum in a cyclic way. It is a preemptive algorithm because it goes through context switching based on the time quantum. An advantage for this algorithm would be that each process gets an equal share of the CPU and is considered fair, but the disadvantage can come from the time quantum itself. If our time quantum is too large it may act like FCFS and it is too small it will cause a lot of switches between each process in the system. For Shortest job next, we find the process with the shortest service time and we run it. The advantage is that we may run the process with less waiting time and the total time required may be lower compared to the other two algorithm.

As for our code we had to create a struct called job that keeps the information of our scheduling algorithm. It keeps the name, service time, arrival time, start time, end time, status, and the turnaround time. The processes are made into an array under the instance of job and are maintained there. We have a display method primarily to display the information of our FCFS algorithm and an input data that with column 1 that is arrival time and column 2 for service time Before we get into the code for our CPU scheduling algorithm, we have to display the input arguments. We use argv[] as the input arguments to tell the program what to display. For argv[1], “-f” display our FCFS scheduling algorithm, “-r” displays the round robin scheduling algorithm, and “-d” displays both. As for argv[2], it take in “-d” to show a detailed information for the algorithm or “-x” for a non-detailed information. After we use argv[3] to input our text file into the program which contains our input data.

With First Come First Served, first we have check out first arrival job, and run the job. Next, we use for loop to detect next earliest arrival job, and add up with job’s service time into present time. We keep repeating all above procedures to finish all processes.

In Round Robin, we create a queue to handle all arrival jobs. If any jobs arrive, we push job into queue. At same time, jobs are running. When any jobs have no enough service time leave, we pop the jobs out of the queue that is what we consider “Job done”. If the job still has time leave, we create a temp to store job, and service time substract with time quantum. We pop old job, and push the temp. Finally, we use a do while loop to check the queue whether empty. If there none of jobs leave, we consider as all jobs are done. If there still have job, we repeat the loop.

For the SJN code, we first check if the process is in the jobs queue and then we find the shortest service time out of all the processes. Afterwards we will run the process and find the next shortest while updating the information for the average waiting time, turnaround time, and the total time. Afterwards we calculated the CPU utilization in the code.

As for comparing the output of our program, we created an excel file to keep all the data into a table. We observed the total time for FCFS is 646, while the total time for round robin with a time quantum of 10 to be 700. We calculated the CPU Utilization to be 97.1% for FCFS and 92.3% for Round Robin. The Average turnaround time is for FCFS is 325 and round robin is 368. The average waiting time for FCFS is 293 and for round robin it is 91. As for SJN, the total time is 679, CPU utilization is 97.2%, the average turnaround time is 356, and the average waiting time is 323. Overall, FCFS is the better scheduling algorithm here because it has the lowest total time as well as waiting time and turnaround time.