Navajit Baruah
S. Harish Koushik

EECS 678 - Project 2 : Scheduler

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of other process on the basis of a particular strategy. Process scheduling is an essential part of multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and loaded process shares the CPU accordingly.

In this project we have implemented a scheduler that schedules a job run based on two types of algorithms preemptive and non-preemptive. In the non-preemptive algorithm we have implemented fcfs, priority based scheduling, and sjf. In the preemptive algorithms we have used preemptive priority based algorithm, preemptive sjf and round robin with different quantum sizes.

We have made use of two files, libpiqueue.c and libscheduler.c provided to use for the implementation of the scheduler. We made use of the helper functions provided in these two files. The libpriqueue.c is basically a queue that is implemented using a linked list, where in each node holds the data and the address of the next node. We refer to the helper fucntions present in libpriqueue.c while writing libscheduler.c.

On the other hand, libscheduler.c implements the scheduling using different algorithms. This file has functions related to scheduling a new job, selecting an algorithm based on the user preference and processing the jobs accordingly. There are functions related round robin algorithm that uses quantum sizes.

Single Core Functionality:

We have implemented the scheduling algorithm for the single core architectures. We have verified our implementation of first-come-first-serve, preemptive priority based algorithm, priority based algorithm without preemption, round robin algorithm with quantum sizes 1,2,4, shortest job first and priority shortest job first with simulator.c and all our results of average waiting time, average response time and average turnaround time have been accurate and is consistent with the results that were provided along with the project description.

Multicore Functionality:

We have implemented scheduling on multi core as well and our implementation of multicore is limited to only preemptive algorithms. Our results of average waiting time, average turn around non preemptive algorithms have been consistent with results provided in the example folder along with the project description. But the results of average response time is slightly different. Apart from this limitation the other aspects of multicore implementation is working fine.