Project #1 Report

By: Jacob Cole, Scott McKeefer, Mark Doggendorf

# Introduction

The purpose of this project is to simulate different scheduling algorithms at work. The algorithms we used were FIFO, SJF, and RR.

We created a process class which held the arrival time, wait time, pid, number of cycles and memory footprint. Then each process object was stored into a “map” C++ container and referenced by its pid.

# Algorithm: FIFO

This algorithm was implemented by looping from the first process to the last process. Since each process arrives 50 cycles after the previous one, the order of the processes is already correct for FIFO. First, the wait time of each process is determined by the current overall execution time of the system. Then, the overall time is increased by the number of cycles of that process. Next, the total wait time of the system is increased by the current process’ wait time. Last, the overall time is increased by a context switch if not currently on the last process.

# Algorithm: Round Robin

For round robin it was a bit tricky because you have to very careful about the order that things execute. Thanks to the arrival time and the quantum size having the same value of 50 the next process would arrive by the time the previous process had finished. In the event that there was not another process in the round robin the currently process will continue to be worked on in 50 set chunks. When ever there is less than 50 to process then the program will switch over to the next ready process. Every time the process is being worked on it updates it wait time. The algorithm for the wait time is waitTime= overAllTime – (arrivleTime +(quantumSize \*numOfTimesProcess)).