P5 Design Document
Taylor Williamson
125007948
4/16/2020

In this assignment, the task is to implement a scheduler for multiple kernel level threads. A scheduler is a third party whose purpose is to ensure that the threads are playing nice and each one is getting its fair share of CPU time which is determined by the scheduling algorithm. In this assignment we will be implementing a First In First Out (FIFO) scheduler.

The scheduler keeps and maintains a ready queue, which lists threads that are waiting to get to the CPU. When the current thread calls yield(), the scheduler gets the head of the ready list and calls the dispatcher to invoke a context switch. The resume(t1) method is called when the system decides that thread t1 should become ready to execute on the CPU again. The add(t1) function holds the instructions for making the thread t1 runnable by the scheduler. This is run after thread creation. Finally, the terminate(t1) method removes thread t1 from the scheduler in preparation for destruction and handles the case where the thread wants to terminate itself.

The threads that kernel.C is producing are non-terminating and only stop executing when they pass control to another thread. When a thread is terminated, we need to worry about releasing the CPU, releasing memory, giving control to the next thread, etc. We will test this by making our kernel threads return after DEFAULT_NB_ITERS loops.