Actual User Threads

Assignment 1

iLab Machine: ls.cs.rutgers.edu

Team Members:

Matthew Mann (mam1010)
Nandan Thakkar (nt284)
Joseph Moussa (jam791)

The Queues:

We implemented a multi-level priority queue with 4 levels of prioritization. We created an array of these queues for easy access. As you will see in the header file, it has a size of 6, indices 0-3 are the priorities, 4 is for waiting tasks and 5 is for completed tasks. The newest threads are created and placed in the highest priority queue (index 0) and run for 25ms after which they are dequeued and decreased in priority then enqueued into that matching priority's queue. If a thread has yielded (say a mutex was found to be locked) then it is dequeued and enqueued back into the queue without decreasing its priority/queue.

The Scheduler:

Our scheduler does most of the queueing. It is called at the end of thread create, join and yield. The scheduler checks the status of the thread, it checks if the thread status is WAITING, YIELDED, EXITING or ran for the fulltime quanta. When it came to timing, each thread ran relatively fast. As the life span of a thread increased. The amount of time it spent in a certain priority increased as its priority decreased.

Tests:

When we tested our functions, we found that our mutex and thread functions were operational. However, the scheduler had issues with making proper context switches. Specifically, the scheduler can switch between threads properly, but it loses the context for the main thread. As a result, it will end up in a "limbo state" even though the other threads, thread functions, and mutex functions ran properly. However, the scheduler seems to be able to find main and complete properly when the number of threads is greater than 9 and equal to one. Otherwise it will have trouble finishing because it lost the context for the main thread. This only occurs in parallelCal.c. When a different benchmark is run with these files. They run correctly.
When it came to timing, each thread ran relatively fast. As the life span of a thread increased. The amount of time it spent in a certain priority increased as its priority decreased.

NOTE: We left print statements inside my_pthread.c for debugging purposes and so you know what's happening.