Project 2: Threadpool

We wrote threadpool.c which is a fork/join thread pool that implements work stealing. Our thread pool is made up of a global task list, an integer for the number of threads, a list for the worker threads, an array for thread ID numbers, a conditional variable for waking up the workers, a mutex for accessing pool data, and a mutex for accessing worker data.

How Our Thread Pool Works:

The number of threads in the pool is specified by the user. Each thread runs in a infinite loop until the thread pool begins shutting down. In the threadpool, there is a global queue of tasks that the worker threads take work from, and the each has its own local queue as well. Our implementation utilizes work-sharing and work-stealing to improve performance.

Testing:

After extensive testing with all the various scripts and debugging methods, our thread pool passes all tests every time except one. The test is ./fib_test -n 4 38. We fail this test due to timing out and other times we pass this test. Our thread pool handles all the basic tests and performs well on all the sorting tests except the one fib test.