Pthreads

By: Steven Gray

The goal of the assignment was to create a multithreaded application where the program uses x number of threads to count up by y each, determined by user inputs for the x and y. After fixing some issues with the example program, all of my programs were built off of that baseline, changing the static memory allocation.

Each program would print out whenever a thread was created, as well as whenever the counter was incremented. The programs were then timed with linux's time command, giving 3 scores: a real time measurement, a user time measurement, and kernel time measurement. The average results of several tests are shown in the table below.

	No Sync	Locks	Test and Set
real	0.385s	0.310s	0.712s
user	0.044s	0.028s	0.276s
sys	0.108s	0.068s	0.080s

For a performance comparison, the program with no synchronization and the program with pthread locks performed fairly similarly in real time, only differing between roughly 0.065s on avg. A significant difference, but there was enough variance in the results that a bad lock run could do worse than a good no sync run. Locks were just generally faster. Test and Set lagged far behind, doing on average 2x worse than locks and even sometimes no sync.

Locks also sees a better outcome in user time, getting nearly half the no sync average. Though at such low times it is hard to say it's massive. Test and Set once again comes out last, being an order of magnitude slower than the locks time.

In kernel time, much of these looked near identical. They'd all vary between 0.060s and 0.120s. The end results have different averages, but it could have really gone any way here.