Comparision

Converting the dining philsopher's solution to OpenMP had both difficulties and benefits. In OpenMP we lose some control over some of the explicit behvaior and controls that we had while using pthreads at the benefit of having an easier time spawning threads. In this case, threads were much easier to spawn (didn't need to jump through hoops with static functions to set up and launch tasks for to create and launch threads, didn't need to explicitly create barriers, etc). OpenMP also made some funcitonalities easier. Because of the explicit pragmas that affect all of the OpenMP calls, we could synchronize threads without passing objects through to threads. This meant we didn't need to create explicit barriers and locks acround certain regions since we could just use `omp barrier` and `omp critical`. Furthermore, we could change the number of threads that would be launched using a simple `omp_set_num_threads()`. The difficult part about using OpenMP is that all threads now need to run the same task.

Performance

- OMP: 9002 ms, 5 philosophers, 2 eats per
- Pthread: 9003 ms, 5 philosophers, 2 eats per

I did not see any noticeable performance difference when using OMP vs pthreads. My OpenMP run ran in 9002 ms while the pthread instance ran slightly longer at 9003 ms. These measurments include the creation and destruction of the threads. This may be due to my implementation using very minimal creation and destruction of threads (single creation point, single destruction point). The work done in both implementations are very similar, in some cases, swapping out a `pthread_mutex_t` for an `omp_lock_t`. This shows that for accessing simple parallelism, OpenMP can give large returns for little less devleopment cost than pthreads.