Member 1: Rithvik Aleshetty netid: rra76
Member 2: Minhesota Geusic netid: mtg110

MyPthread User-level Thread Library and Scheduler Report

mypthread Functions:

- mypthread_create:
  - Initialize the scheduler context (if not already initialized) in the function mypthread_init and in here we create the main queue of tcb's and setup the timer. Setup a new tcb here and add it onto our main queue (master_queue) and then call makecontext
- mypthread_yield
  - Pause the timer, and set the global state variable to yielded, then swap context to the scheduler to schedule another thread
- mypthread_exit
  - stop the timer, set the global state variable to Finished, take care of the return variable, and set the context back to the scheduler
- mypthread_join
  - check if the thread passed in is done, if not, go back to scheduler. Execution will not continue until thread is exited
- mypthread_mutex_init
  - set available field of the mutex to 0, in our case meaning that the lock is available. Set the thread field to -1 as there is no thread locking it yet
- mypthread_mutex_lock
  - Use the atomic test and set function to check if the mutex can be locked. If it cannot: then set the thread field to the current thread id, and set state to Mutexed, set current mutex variable, then swap context to the scheduler. The scheduler will then call a function which puts the current thread into the blocked thread field of the mutex
- mypthread_mutex_unlock
  - Release the lock by setting the available field back to 0 and setting thread to -1. Then go through the mutex's blocked list and pop each one and put it back onto the master queue for scheduling
- mypthread_mutex_destroy
  - Does not have to do anything since no dynamic memory allocated, returns 0

STCF scheduler:

- Schedule()
  - Just pauses timer and calls the sched_stcf
- Sched_stcf()
  - Depending on the type of interrupt (timer, yielded, mutex) calls another function to find next thread to run: either sched_stcf_by_interrupt() or sched_stcf_by_mutex()
  - Record time of the start of next thread, Start timer and run next thread
  - Deallocates any memory if exited

Member 1: Rithvik Aleshetty netid: rra76
Member 2: Minhesota Geusic netid: mtg110

- Sched_stcf_by_mutex()
    - Add the current context onto the list of blocked threads on the mutex after adjusting priority of current context
- Sched_stcf_by_interrupt()
    - Adjust priority and the time that the current thread has ran for so far and then add it back onto our master_queue
    - Sorts the current thread based on its priority in the master_queue (shortest first) unless the current thread is the only thread it added it back to the queue to continue running

Benchmarks:

Tester file	# of threads	Our mypthread using stcf	Actual pthread library
parallel_cal	10	3094 microseconds	1092 microseconds
parallel_cal	70	3116 microseconds	1010 microseconds
vector_multipy	10	45 microseconds	222 microseconds
vector_multipy	70	99 microseconds	315 microseconds
external_cal	10	7488 microseconds	4333 microseconds
external_cal	70	7507 microseconds	4315 microseconds

Most Challenging Part:

The scheduling was the most challenging part especially since we have not planned it out properly from the start. Planning out the scheduling would have also helped being on the same page within our group regarding making all the functions work together.

Where Could we have done better?

Our implementation could have been better in the way we optimized our data structures. Our times in the parallel_cal test and external_cal test could have been better since as of now, they are a lot higher than the pthread library. For example, having a min heap or optimization. We also could have a more formatted data structure for the context.