User Thread Library

Names and Net IDs of the members of the group:

* Shikha/ sv
* Praveen/pp
* Omkar/osd14

iLab M/c: ilab1

1. Contains the following files:
   1. main.c
   2. bench.c
   3. my\_pthread.c
   4. my\_pthread\_t.h
   5. list.c
   6. list.h
   7. queue.c
   8. queue.h
2. Working of the Scheduler:
   1. The scheduler has a multilevel priority queue for scheduling.
   2. There are 3 levels of the priority i.e. 0, 1, 2. The priority of the listed levels are in ascending order with a time slice of 50, 100, 200 milliseconds respectively.
   3. Initially each thread is added to the highest priority in the queue. If a thread finishes an assigned time quantum, the thread’s priority is decreased and pushed to a lower priority queue.
   4. Signal Handler: Picks up the next thread in the queue post the expiration of quanta or completion of execution.
   5. Maintenance Cycle: After a period of time of every 1000msecs, based on the following checks few solutions have been implemented
3. Maintenance Cycle uses:
   1. Problem - Starvation / Solution – Ageing
      1. During every maintenance cycle, if a thread is sitting in the queue for more than 1000msecs, the thread is moved from lowest priority to highest priority.
      2. If they do not complete running in their quanta, they are moved to the lowest priority, the cycle goes on till completion.
      3. This avoids starvation
   2. Problem - Priority Inversion / Solution – Priority Inheritance
      1. When a thread tries to lock a mutex that is already being locked by another thread, then that id is noted.
      2. During the maintenance cycle, priority is checked with respect to the mutex. If that thread exists, then the thread having the mutex inherits the property of the thread waiting for the mutex.
      3. When such an occurrence happens, the transfer of mutex’s holding from lower to higher priority occurs, and helps them run to completion.
   3. Problem - Multiple Mutexes leading to deadlock / Solution - Conditional Locking
      1. When two threads lock two mutexes respectively, a deadlock occurs when the thread try to lock each other’s mutex.
      2. To avoid this problem, mutexes are unlocked in the same order of locking.
4. Comparison with benchmark:
   1. The comparison is done with a standard pthread library and shown below as a graph.