Kunal Pednekar | netid: ksp101
Abdulellah Abualshour | netid: aha59
Chris Orthmann | netid: co178

Assignment 1: Scheduler

In this assignment, we implemented a Linux pthread library which also includes the implementation of a scheduler that handles the created threads and gives each thread a time slice to run within the time quantum specified (50ms).

The library includes all the functions required by the assignment instructions, in addition to a Queue library implementation and the Scheduler library implementation.

Details:

The user should implement his/her own program using the thread library instead of the usual pthread library in order to test the implementation. He/she can use any function in the library as long as it complies to the pthread library rules. When my_pthread_create is called, the scheduler gets initialized and a new thread is created and allocated after saving main's context in the thread table as well. After that, the scheduler handler gets called and handles the scheduling process accordingly.

- **Structs and enums:**
  - Scheduler struct
    - For the scheduler.
  - Queue struct
    - For each queue created.
  - qNode struct
    - For each queue node created.
  - Mutex struct
    - For each mutex created by the user.
  - Thread struct
    - For each thread created by the user.
  - Status enum
    - For state tracking within the scheduler.

- **Functions:**
  - my_pthread_create(…): creates a thread and allocated appropriate space for the stack.
  - my_pthread_yield(…): changes the status to yield and calls the scheduler for appropriate handling.
  - my_pthread_exit(…): sets the thread's state to TERMINATED.
  - my_pthread_join(…): joins the thread and frees the stack.
  - my_pthread_mutex_init(…): initializes a mutex object.
  - my_pthread_mutex_lock(…): lock a mutex OR adds the calling thread to the waiting queue.
  - my_pthread_mutex_unlock(…): unlocks a mutex.
  - my_pthread_mutex_destroy(…): destroys a mutex and frees the space.
  - run_thread(…): associates the thread passed to the function pointer and arguments.
  - sched_init(…): initializes the scheduler the first time a thread is requested to be created.
  - sched_add(…): adds a thread to the scheduler.
  - sched_choose(…): chooses a thread to be run.
  - queue_init(…): initializes a queue.
  - enqueue(…): adds a queue node to a queue.

- dequeue(…): returns the head of the queue.
- peek(…): for testing the queues (seeing the head).
- queueisEmpty(…): for testing the queues.
- timer_init(…): initializes the alarm and time quantum.
- sighandler(…): signal handling process.

- **Time Tests (Maintenance Cycle):**

For this test cycle, we are using a counter function provided in the sakai resources that represents each thread's context and testing for its run time.

# of Threads	Sum of Run Times of Threads*	Average for Each Thread*
5	64.18ms	12.236ms
10	110.05ms	11.005ms
15	168.19ms	11.212ms
20	235.33ms	11.766ms

# of Threads	Sum of Run Times of Scheduler*	Average for Each Scheduler Call*
5	40.54ms	8.108ms
10	78.90ms	7.890ms
15	120.65ms	8.043ms
20	158.12ms	7.906ms

*All values were rounded to three decimal places.