Source: https://computing.llnl.gov/tutorials/pthreads/

PTHREADS

Routine Prefix	Functional Group
pthread_	Threads themselves and miscellaneous subroutines
pthread_attr_	Thread attributes objects
pthread_mutex_	Mutexes
pthread_mutexattr_	Mutex attributes objects.
pthread_cond_	Condition variables
pthread_condattr_	Condition attributes objects
pthread_key_	Thread-specific data keys
pthread_rwlock_	Read/write locks
pthread_barrier_	Synchronization barriers

Creating and Terminating Threads

```
pthread_create (thread,attr,start_routine,arg)

pthread_exit (status)

pthread_cancel (thread)

pthread_attr_init (attr)

pthread_attr_destroy (attr)
```

Joining and Detaching Threads

```
pthread_join (threadid,status)

pthread_detach (threadid)

pthread_attr_setdetachstate (attr,detachstate)

pthread_attr_getdetachstate (attr,detachstate)
```

Creating and Destroying Mutexes

```
pthread_mutex_init (mutex,attr)

pthread_mutex_destroy (mutex)

pthread_mutexattr_init (attr)

pthread_mutexattr_destroy (attr)
```

Locking and Unlocking Mutexes

```
pthread_mutex_lock (mutex)

pthread_mutex_trylock (mutex)

pthread_mutex_unlock (mutex)
```

Creating and Destroying Condition Variables

```
pthread_cond_init (condition,attr)

pthread_cond_destroy (condition)

pthread_condattr_init (attr)

pthread_condattr_destroy (attr)
```

Waiting and Signaling on Condition Variables

```
pthread_cond_wait (condition,mutex)

pthread_cond_signal (condition)

pthread_cond_broadcast (condition)
```

```c
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>


#define NUM_THREADS  3
#define TCOUNT 10
#define COUNT_LIMIT 12

int     count = 0;
int     thread_ids[3] = {0,1,2};
pthread_mutex_t count_mutex;
pthread_cond_t count_threshold_cv;

void *inc_count(void *t) {
   int i;
   long my_id = (long)t;

   for (i=0; i<TCOUNT; i++) {
      pthread_mutex_lock(&count_mutex);
      count++;

      /*
        Check the value of count and signal waiting thread when condition is
        reached.  Note that this occurs while mutex is locked.
      */
      if (count == COUNT_LIMIT) {
         pthread_cond_signal(&count_threshold_cv);
         printf("inc_count(): thread %ld, count = %d  Threshold reached.\n",
                my_id, count);
      }
      printf("inc_count(): thread %ld, count = %d, unlocking mutex\n",
             my_id, count);
      pthread_mutex_unlock(&count_mutex);

      /* Do some "work" so threads can alternate on mutex lock */
      sleep(1);
    }
    pthread_exit(NULL);
}

void *watch_count(void *t) {
   long my_id = (long)t;

    printf("Starting watch_count(): thread %ld\n", my_id);

    /*
       Lock mutex and wait for signal.  Note that the pthread_cond_wait
       routine will automatically and atomically unlock mutex while it waits.
       Also, note that if COUNT_LIMIT is reached before this routine is run by
       the waiting thread, the loop will be skipped to prevent pthread_cond_wait
       from never returning.
    */
    pthread_mutex_lock(&count_mutex);
    while (count<COUNT_LIMIT) {
       pthread_cond_wait(&count_threshold_cv, &count_mutex);
       printf("watch_count(): thread %ld Condition signal received.\n", my_id);
       count += 125;
       printf("watch_count(): thread %ld count now = %d.\n", my_id, count);
    }
    pthread_mutex_unlock(&count_mutex);
    pthread_exit(NULL);
}

int main (int argc, char *argv[]){
   int i, rc;
   long t1=1, t2=2, t3=3;
   pthread_t threads[3];
```

```
    pthread_attr_t attr;

    /* Initialize mutex and condition variable objects */
     pthread_mutex_init(&count_mutex, NULL);
     pthread_cond_init (&count_threshold_cv, NULL);

    /* For portability, explicitly create threads in a joinable state */
     pthread_attr_init(&attr);
     pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
     pthread_create(&threads[0], &attr, watch_count, (void *)t1);
     pthread_create(&threads[1], &attr, inc_count, (void *)t2);
     pthread_create(&threads[2], &attr, inc_count, (void *)t3);

    /* Wait for all threads to complete */
    for (i=0; i<NUM_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }
    printf ("Main(): Waited on %d  threads. Done.\n", NUM_THREADS);

    /* Clean up and exit */
     pthread_attr_destroy(&attr);
     pthread_mutex_destroy(&count_mutex);
     pthread_cond_destroy(&count_threshold_cv);
     pthread_exit(NULL);

}
```