COEN 11 - Fall 2017 - Practice VIII

Solutions

1. Splitting the data -- write a thread function to initialize int array x so that each element receives its index in the array: x[i] = i, and each thread initializes its portion of the array. Note that i relates to the entire array. The size of the array is N, and your program will execute with nthreads (which is a global value). Assume N is a multiple of nthreads.

```
#include <stdio.h>
#include <pthread.h>
void * init(void *arg);
int main()
{
   int i;
   pthread_t thr[nthreads];
   for (i = 0; i < nthreads; i++)
          pthread_create (&thr[i], NULL, init, (void *) i);
   for (i = 0; i < nthreads; i++)
          pthread_join (thr[i], NULL);
}

void init (void *arg)
{
   int id = (int)arg;
   int size = N / nthreads;
   int my_start = id * size;
   int my end = id * size + size;

   for (i = my_start; i < my_end; i++)
          x[i] = i;

}
```

2. Splitting the data -- write a thread function to initialize int 2D array x (NxN) so that each thread initializes its portion with i+j in each slot. Note that i and j relate to the entire array. Each thread operates on a strip independently, and your program will execute with nthreads (which is a global value). Assume N is a multiple of nthreads.

```
void
init (void *arg)
{
   int id = (int)arg;
   int size = N / nthreads;
   int my_start = id * size;
   int my end = id * size + size;

   for (for (i = my_start; i < my_end; i++)
          for (j = 0; j < N; j++)
                 x[i][j] = i + j;
```

```
    }
```

3. Sum of an array

```
int array[N];
int sum = 0;

void* thread_sum(void *id)
{
        int id = (int)arg;
        int size = N / nthreads;
        int my_start = id * size;
        int my_end = id * size + size;

        int my_sum=0;

        for (i = my_start; i < my_end; i++)
        {

                pthread_mutex_lock (&mutex);
                sum = sum + array [i];
                pthread_mutex_unlock (&mutex);

        }

        return NULL;
}
```

The more efficient way:

```
void* thread_sum(void *id)
{
        int id = (int)arg;
        int size = N / nthreads;
        int my_start = id * size;
        int my_end = id * size + size;

        int my_sum=0;

        for (i = my_start; i < my_end; i++)
        {
                my_sum = my_sum + array [i];
        }

        pthread_mutex_lock (&mutex);
        sum += my_sum;
        pthread_mutex_unlock (&mutex);

        return NULL;
 }
```