CSCI 4061 Discussion 9

3/26/18

Overview

- Critical sections
- Mutual exclusion
  - Mutex locks
  - Condition variables
- Exercise

Critical Sections

- Code which must be synchronized to ensure correctness.
- Only occurs in multithreaded applications with shared resources.

Mutual Exclusion

- Ensures that access to a shared resource is protected.
- Reduces the amount of parallelism, thus inhibiting performance.

Mutex Locks

- Enforces mutual exclusion with a lock.
- Only one thread can possess the lock.
- Any thread which attempts to grab the lock will block. Blocked threads are placed in a queue to wait.
- When unlocked, first thread on the queue gains possession of the lock and runs.

Helpful Functions

```
// Create a mutex
pthread_mutex_t mutex;

// Initialize the mutex. MUST BE DONE BEFORE LOCKING!
pthread_mutex_init(pthread_mutex_t* mutex, pthread_mutexattr_t* atts);

// Grabs lock, if already locked, the calling thread will block until unlocked.
pthread_mutex_lock(pthread_mutex_t* mutex);
pthread_mutex_unlock(pthread_mutex_t* mutex);
```

Exercise

- The code provided for you today has a series of race conditions.
- Fix these **without adding excessive synchronization**.
- The account amounts printed should match the solution.
- The execution time shouldn’t exceed twice the solution’s.