Mutex Variables Overview

Mutex is an abbreviation for "mutual exclusion". Mutex variables are one of the primary means of implementing thread synchronization and for protecting shared data when multiple writes occur.

A mutex variable acts like a "lock" protecting access to a shared data resource. The basic concept of a mutex as used in Pthreads is that only one thread can lock (or own) a mutex variable at any given time. Thus, even if several threads try to lock a mutex only one thread will be successful. No other thread can own that mutex until the owning thread unlocks that mutex. Threads must "take turns" accessing protected data.

Mutexes can be used to prevent "race" conditions. An example of a race condition involving a bank transaction is shown below:

Thread 1	Thread 2	Balance
Read balance: $1000		**$1000**
	Read balance: $1000	**$1000**
	Deposit $200	**$1000**
Deposit $200		**$1000**
Update balance $1000+$200		**$1200**
	Update balance $1000+$200	**$1200**

In the above example, a mutex should be used to lock the "Balance" while a thread is using this shared data resource.

Mutex are usually used when updating global variables. This is a safe way to ensure that when several threads update the same variable, the final value is the same as what it would be if only one thread performed the update. The variables being updated belong to a *critical section*.

A typical sequence in the use of a mutex is as follows:

- Create and initialize a mutex variable
- Several threads attempt to lock the mutex
- Only one succeeds and that thread owns the mutex
- The owner thread performs some set of actions
- The owner unlocks the mutex
- Another thread acquires the mutex and repeats the process
- Finally the mutex is destroyed

When several threads compete for a mutex, the losers block at that call - a non-blocking call is available with “trylock” instead of the “lock” call.

When protecting shared data, it is the programmer’s responsibility to make sure every thread that needs to use a mutex does so. For example, if 4 threads are updating the same data, but only one uses a mutex, the data can still be corrupted.