Locking and Unlocking Mutexes

Routines:

`pthread_mutex_lock(mutex)`

`pthread_mutex_trylock(mutex)`

`pthread_mutex_unlock(mutex)`

Usage:

The `pthread_mutex_lock()` routine is used by a thread to acquire a lock on the specified mutex variable. If the mutex is already locked by another thread, this call will block the calling thread until the mutex is unlocked.

`pthread_mutex_trylock()` will attempt to lock a mutex. However, if the mutex is already locked, the routine will return immediately with a "busy" error code. This routine may be useful in preventing deadlock conditions, as in a priority-inversion situation.

`pthread_mutex_unlock()` will unlock a mutex if called by the owning thread. Calling this routine is required after a thread has completed its use of protected data if other threads are to acquire the mutex for their work with the protected data. An error will be returned if:

- If the mutex was already unlocked
- If the mutex is owned by another thread

There is nothing "magical" about mutexes…in fact they are akin to a "gentlemen's agreement" between participating threads. It is up to the programmer to ensure that all threads make lock and unlock mutexes appropriately. The following scenario demonstrates a logical error:

Thread 1	Thread 2	Thread 3
Lock	**Lock**	
A = 2	**A = A+1**	**A = A*B**
Unlock	**Unlock**	

Question: When more than one thread is waiting for a locked mutex, which thread will be granted the lock first after it is released?

▶ Click for answer.

Home Privacy & Legal Notice