pthread_mutex_*()

SYNOPSIS:
#include <pthread.h>

```
    int pthread_mutex_lock(pthread_mutex_t *mutex);
    int pthread_mutex_trylock(pthread_mutex_t *mutex);
    int pthread_mutex_unlock(pthread_mutex_t *mutex);
```

DESCRIPTION
If the mutex is already locked by another thread, the calling thread shall block until the mutex becomes available.
If a thread attempts to relock a mutex that it has already locked, pthread_mutex_lock() shall behave as described in the Relock column of the following table.
If a thread attempts to unlock a mutex that it has not locked or a mutex which is unlocked pthread_mutex_unlock() shall behave as described in the "Unlock When Not Owner" column of the following table.

Mutex Type	Robustness	Relock	Unlock When Not Owner
NORMAL	non-robust	deadlock	undefined behavior
NORMAL	robust	deadlock	error returned
ERRORCHECK	either	error returned	error returned
RECURSIVE	either	recursive (see below)	error returned

DEFAULT	non-robust	undefined behavior†	undefined behavior†
DEFAULT	robust	undefined behavior†	error returned

The mutex shall maintain the concept of a lock count.
When a thread successfully acquires a mutex for the first time, the lock count shall be set to one.
Every time a thread relocks this mutex, the lock count shall be incremented by one.
Each time the thread unlocks the mutex, the lock count shall be decremented by one.
When the lock count reaches zero, the mutex shall become available for other threads to acquire.

RETURN VALUE
If successful, return zero.
Otherwise, an error number shall be returned to indicate the error.

pthread_cond_*()

```
#include <pthread.h>

int pthread_cond_destroy(pthread_cond_t *cond);
int pthread_cond_init(pthread_cond_t *restrict cond, const pthread_condattr_t *restrict attr);
pthread_cond_t cond = PTHREAD_COND_INITIALIZER;
```

Description
The pthread_cond_destroy() function shall destroy the given condition variable specified by cond; the object becomes, in effect, uninitialized.
An implementation may cause pthread_cond_destroy() to set the object referenced by cond to an invalid value.
A destroyed condition variable object can be reinitialized using pthread_cond_init();
the results of otherwise referencing the object after it has been destroyed are undefined.

It shall be safe to destroy an initialized condition variable upon which no threads are currently blocked.
Attempting to destroy a condition variable upon which other threads are currently blocked results in undefined behavior.

The pthread_cond_init() function shall initialize the condition variable referenced by cond with attributes referenced by attr.
If attr is NULL, the default condition variable attributes shall be used; the effect is the same as passing the address of a default condition variable attributes object.

The result of referring to copies of cond in calls to pthread_cond_wait(), pthread_cond_timedwait(), pthread_cond_signal(), pthread_cond_broadcast(), and pthread_cond_destroy() is undefined.

Attempting to initialize an already initialized condition variable results in undefined behavior.

In cases where default condition variable attributes are appropriate, the macro PTHREAD_COND_INITIALIZER can be used to initialize condition variables that are statically allocated.
The effect shall be equivalent to dynamic initialization by a call to pthread_cond_init() with parameter attr specified as NULL, except that no error checks are performed.

Return Value
If successful, the pthread_cond_destroy() and pthread_cond_init() functions shall return zero; otherwise, an error number shall be returned to indicate the error.