Synchronizing PThreads

A *mutex* is basically a *lock*. We can protect some critical section **C** by *locking* the mutex before entering **C** and *unlocking* the mutex upon leaving **C**.

Mutexes have type **pthread_mutex_t**.

There are five basic mutex functions:

pthread_mutex_init creates a new mutex.

pthread_mutex_destroy destroys a mutex.

pthread_mutex_lock locks a mutex.

pthread_mutex_unlock unlocks a mutex.

pthread_mutex_trylock tries to lock a mutex without blocking.

```
#include <pthread.h>

int pthread_mutex_init(pthread_mutex_t * mutex1,
               const pthread_mutex_attr * attr);
```

This routine creates a new mutex, called *mutex1*.

Its attributes are specified by `attr`. The default attributes are used if `attr` is NULL.

If the routine succeeds, it will return 0.

Otherwise, an error number will be returned.

pthread_mutex_destroy

```
#include <pthread.h>

int pthread_mutex_destroy(pthread_mutex_t * mutex1);
```

This routine destroys the mutex *mutex1*.

If the routine succeeds, it will return 0.

Otherwise, an error number will be returned.

Basic Synchronization Functions pthread_mutex_lock

```
#include <pthread.h>

int pthread_mutex_lock(pthread_mutex_t * mutex1);
```

This routine locks the mutex `mutex1`.

If the mutex is already locked, the calling thread blocks until the the mutex becomes available.

If the routine succeeds, it will return 0.

Otherwise, an error number will be returned.

```
#include <pthread.h>

int pthread_mutex_unlock(pthread_mutex_t * mutex1);
```

If the current thread is the owner of the mutex, then this routine unlocks the mutex `mutex1`.

If any threads are waiting on the mutex, then the scheduler decides which one will get the lock next.

Otherwise, the next thread that calls `pthread_mutex_lock` will get it.

If the routine succeeds, it will return 0.

Otherwise, an error number will be returned.

Example: Too-Much-Milk

Time	You	Your Roommate
3:00	Arrive Home	
3:05	Look in fridge; no milk	
3:10	Leave for grocery	
3:15		Arrive home
3:20	Arrive at grocery	Look in fridge; no milk
3:25	Buy milk	Leave for grocery
3:35	Arrive home; put milk in fridge	
3:45		Buy milk
3:50		Arrive home; oh no!

Thread 1	**Thread 2**
a = data;	b = data;
a++;	b–;
data = a;	data = b;