#include <pthread.h>

pthread_mutex_t mutex;

/* create the mutex lock */
pthread_mutex_init(&mutex,NULL);

/* acquire the mutex lock */
pthread_mutex_lock(&mutex);

/*** critical section ***/

/* release the mutex lock */
pthread_mutex_unlock(&mutex);

**Figure 6.30** Code sample.

Pthreads Mutex Locks

The code sample depicted in Figure 6.30 illustrates how mutex locks available in the Pthread API can be used to protect a critical section.

Pthreads uses the pthread_mutex_t data type for mutex locks. A mutex is created with the pthread_mutex_init(&mutex,NULL) function, with the first parameter being a pointer to the mutex. By passing NULL as a second parameter, we initialize the mutex to its default attributes. The mutex is acquired and released with the pthread_mutex_lock() and pthread_mutex_unlock() functions. If the mutex lock is unavailable when pthread_mutex_lock() is invoked, the calling thread is blocked until the owner invokes pthread_mutex_unlock(). All mutex functions return a value of 0 with correct operation; if an error occurs, these functions return a nonzero error code.

Pthreads Semaphores

Pthreads provides two types of semaphores—named and unnamed. For this project, we use unnamed semaphores. The code below illustrates how a semaphore is created:

```
#include <semaphore.h>
sem_t sem;

/* Create the semaphore and initialize it to 5 */
sem_init(&sem, 0, 5);
```

The sem_init() creates and initializes a semaphore. This function is passed three parameters:

a. A pointer to the semaphore
b. A flag indicating the level of sharing
c. The semaphore's initial value