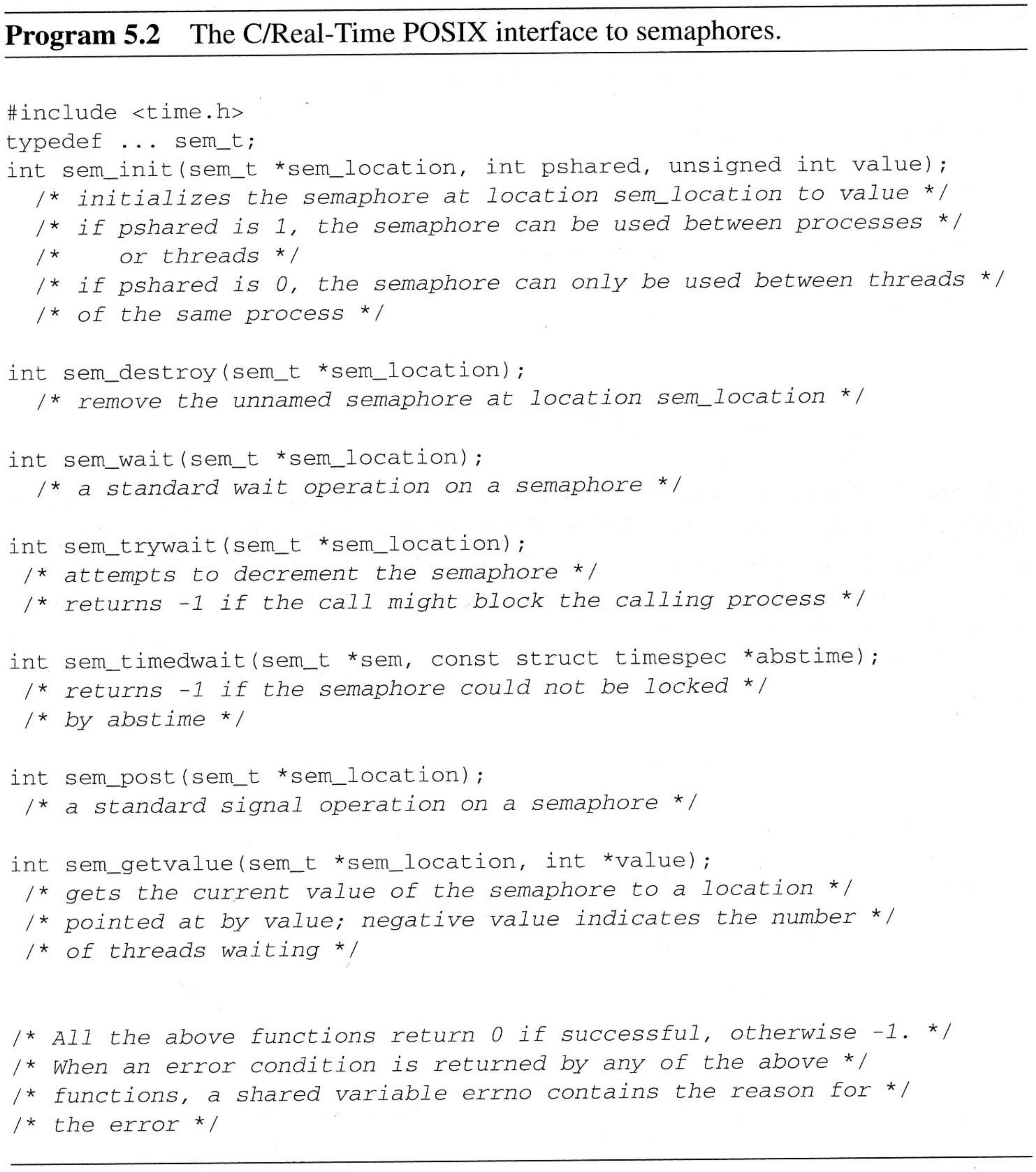
Beispiele zu Kapitel 3: Kommunikation und Synchronisation – Gemeinsame Daten

Aus: Alan Burns, Andy Wellings: *Real-Time Systems and Programming Languages. Ada, Real-Time Java and C/Real-Time POSIX*. Addison Wesley, 2009. (Kapitel 5)

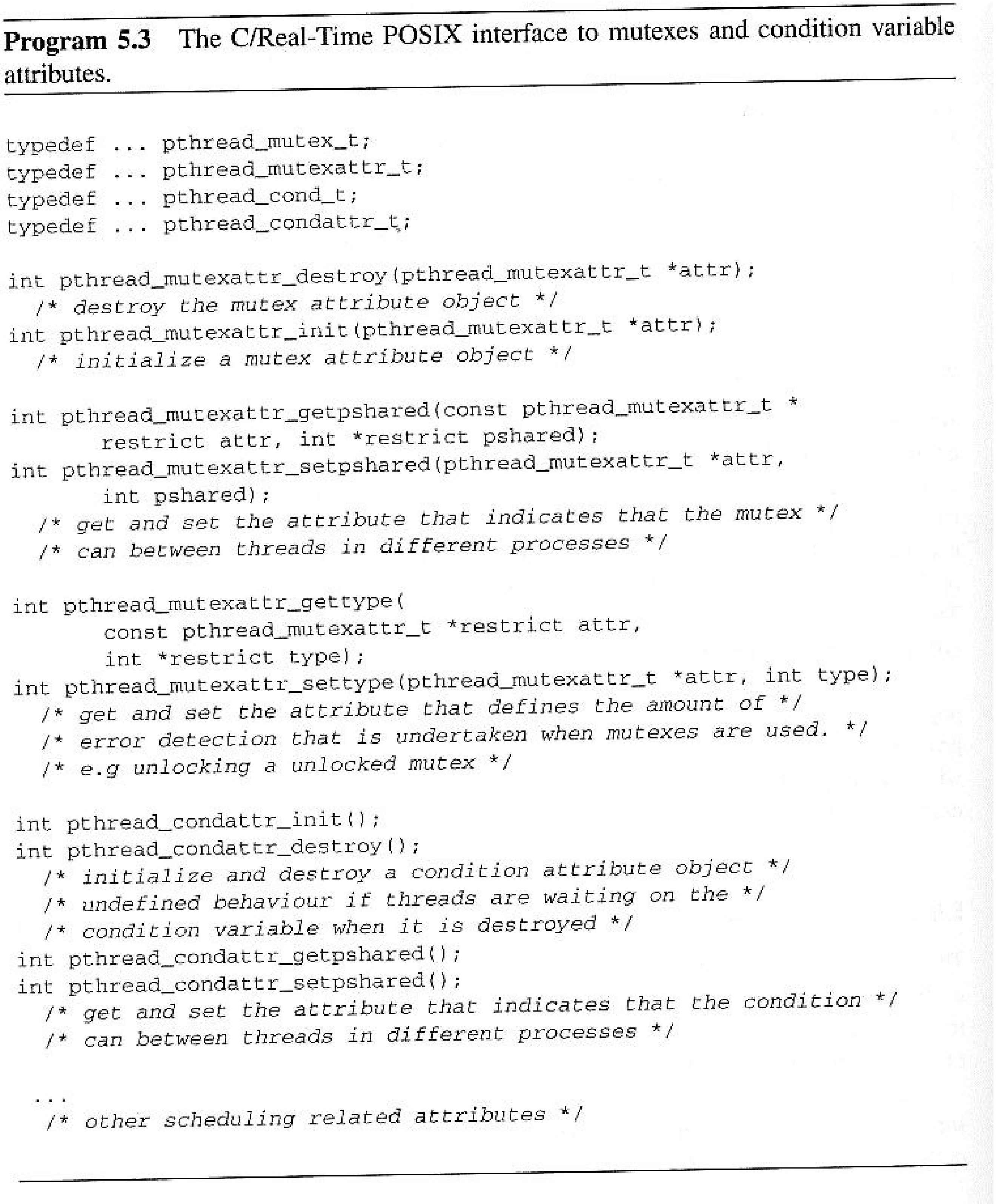
Beispiel 3-1: C/Real-Time POSIX-Schnittstelle für Semaphore



Program 5.2 The C/Real-Time POSIX interface to semaphores.

```
#include <time.h>
typedef ... sem_t;
int sem_init(sem_t *sem_location, int pshared, unsigned int value);
  /* initializes the semaphore at location sem_location to value */
  /* if pshared is 1, the semaphore can be used between processes */
  /*    or threads */
  /* if pshared is 0, the semaphore can only be used between threads */
  /* of the same process */

int sem_destroy(sem_t *sem_location);
  /* remove the unnamed semaphore at location sem_location */

int sem_wait(sem_t *sem_location);
  /* a standard wait operation on a semaphore */

int sem_trywait(sem_t *sem_location);
 /* attempts to decrement the semaphore */
 /* returns -1 if the call might block the calling process */

int sem_timedwait(sem_t *sem, const struct timespec *abstime);
 /* returns -1 if the semaphore could not be locked */
 /* by abstime */

int sem_post(sem_t *sem_location);
 /* a standard signal operation on a semaphore */

int sem_getvalue(sem_t *sem_location, int *value);
 /* gets the current value of the semaphore to a location */
 /* pointed at by value; negative value indicates the number */
 /* of threads waiting */


/* All the above functions return 0 if successful, otherwise -1. */
/* When an error condition is returned by any of the above */
/* functions, a shared variable errno contains the reason for */
/* the error */
```

<u>Beispiel 3-2:</u> C/Real-Time POSIX-Schnittstelle für Mutex- und Bedingungsvariablen-Attribute



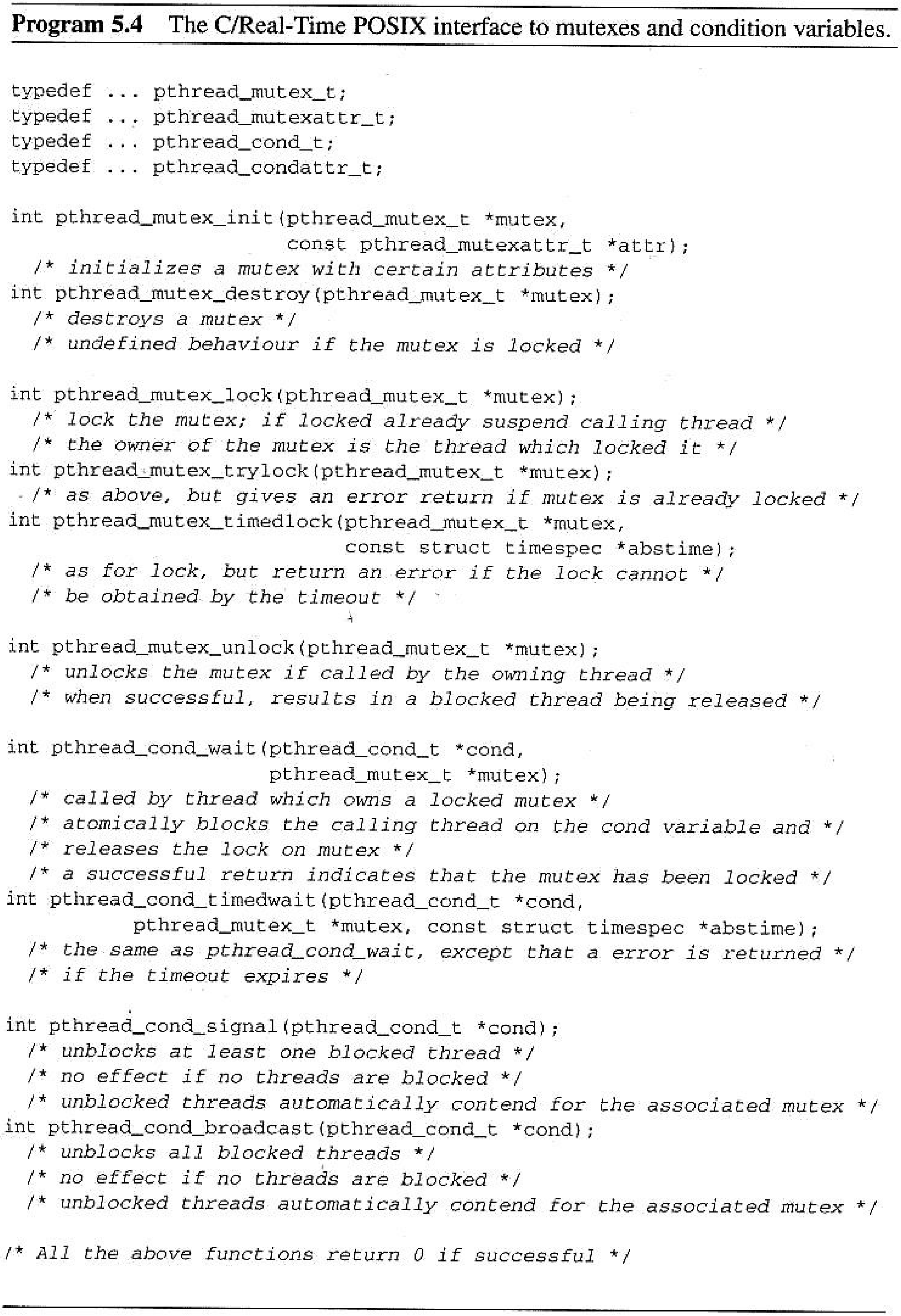
Program 5.3 The C/Real-Time POSIX interface to mutexes and condition variable attributes.

```
typedef ... pthread_mutex_t;
typedef ... pthread_mutexattr_t;
typedef ... pthread_cond_t;
typedef ... pthread_condattr_t;

int pthread_mutexattr_destroy(pthread_mutexattr_t *attr);
  /* destroy the mutex attribute object */
int pthread_mutexattr_init(pthread_mutexattr_t *attr);
  /* initialize a mutex attribute object */

int pthread_mutexattr_getpshared(const pthread_mutexattr_t *
        restrict attr, int *restrict pshared);
int pthread_mutexattr_setpshared(pthread_mutexattr_t *attr,
        int pshared);
  /* get and set the attribute that indicates that the mutex */
  /* can between threads in different processes */

int pthread_mutexattr_gettype(
        const pthread_mutexattr_t *restrict attr,
        int *restrict type);
int pthread_mutexattr_settype(pthread_mutexattr_t *attr, int type);
  /* get and set the attribute that defines the amount of */
  /* error detection that is undertaken when mutexes are used. */
  /* e.g unlocking a unlocked mutex */

int pthread_condattr_init();
int pthread_condattr_destroy();
  /* initialize and destroy a condition attribute object */
  /* undefined behaviour if threads are waiting on the */
  /* condition variable when it is destroyed */
int pthread_condattr_getpshared();
int pthread_condattr_setpshared();
  /* get and set the attribute that indicates that the condition */
  /* can between threads in different processes */

 ...
  /* other scheduling related attributes */
```

<u>Achtung:</u> bei den letzten 4 Funktionen fehlt jeweils der Parameter für das Bedingungsattribut: Die richtige Signatur lautet:

```
        int pthread_condattr_...(pthread_condattr_t * attr);
```

Beispiel 3-3: C/Real-Time POSIX-Schnittstelle für Mutexe und Bedingungsvariablen



Program 5.4 The C/Real-Time POSIX interface to mutexes and condition variables.

```
typedef ... pthread_mutex_t;
typedef ... pthread_mutexattr_t;
typedef ... pthread_cond_t;
typedef ... pthread_condattr_t;

int pthread_mutex_init(pthread_mutex_t *mutex,
                       const pthread_mutexattr_t *attr);
  /* initializes a mutex with certain attributes */
int pthread_mutex_destroy(pthread_mutex_t *mutex);
  /* destroys a mutex */
  /* undefined behaviour if the mutex is locked */

int pthread_mutex_lock(pthread_mutex_t *mutex);
  /* lock the mutex; if locked already suspend calling thread */
  /* the owner of the mutex is the thread which locked it */
int pthread_mutex_trylock(pthread_mutex_t *mutex);
  /* as above, but gives an error return if mutex is already locked */
int pthread_mutex_timedlock(pthread_mutex_t *mutex,
                            const struct timespec *abstime);
  /* as for lock, but return an error if the lock cannot */
  /* be obtained by the timeout */

int pthread_mutex_unlock(pthread_mutex_t *mutex);
  /* unlocks the mutex if called by the owning thread */
  /* when successful, results in a blocked thread being released */

int pthread_cond_wait(pthread_cond_t *cond,
                      pthread_mutex_t *mutex);
  /* called by thread which owns a locked mutex */
  /* atomically blocks the calling thread on the cond variable and */
  /* releases the lock on mutex */
  /* a successful return indicates that the mutex has been locked */
int pthread_cond_timedwait(pthread_cond_t *cond,
          pthread_mutex_t *mutex, const struct timespec *abstime);
  /* the same as pthread_cond_wait, except that a error is returned */
  /* if the timeout expires */

int pthread_cond_signal(pthread_cond_t *cond);
  /* unblocks at least one blocked thread */
  /* no effect if no threads are blocked */
  /* unblocked threads automatically contend for the associated mutex */
int pthread_cond_broadcast(pthread_cond_t *cond);
  /* unblocks all blocked threads */
  /* no effect if no threads are blocked */
  /* unblocked threads automatically contend for the associated mutex */

/* All the above functions return 0 if successful */
```