Mutual exclusion (mutexes)

Juan Felipe Medina Lee, Ph.D.

Computer Science and Engineering
University of Puerto Rico Mayagüez Campus

Overview

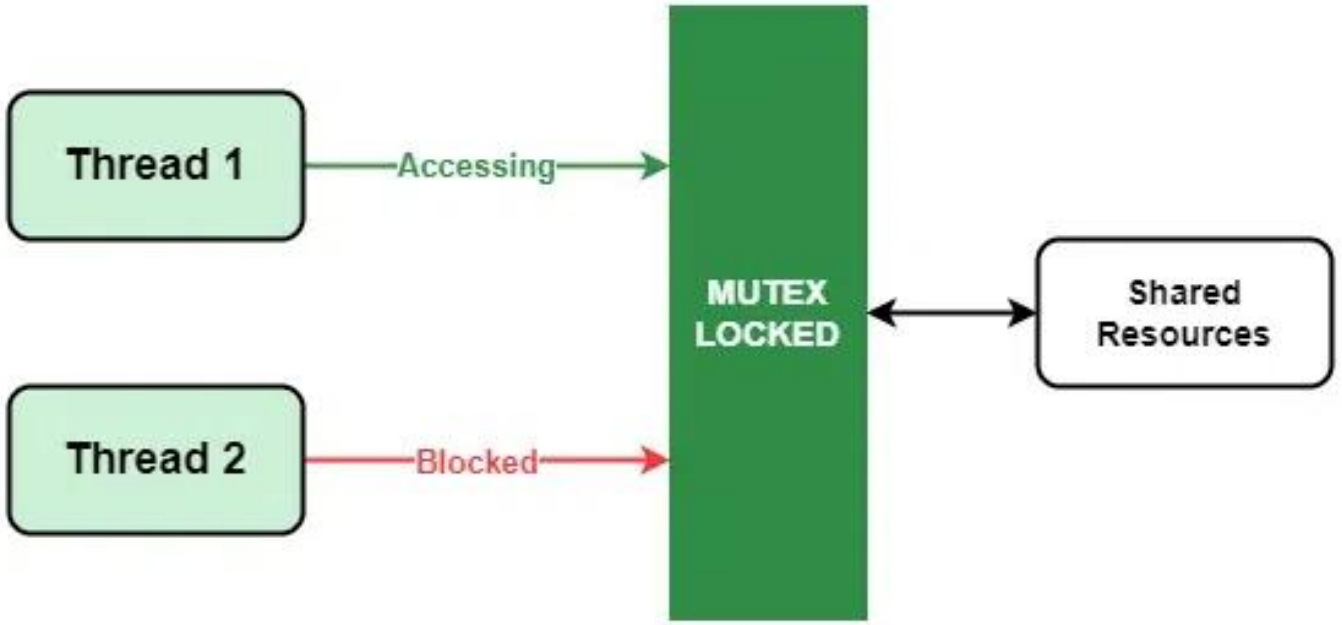
- Overview
- usage
- Implementation in pthread

What is a mutex?

- A mutex (short for *mutual exclusion*) is a synchronization primitive that prevents multiple threads from accessing a shared resource or critical section of code simultaneously.

- The purpose of a mutex is to ensure that only one thread can access the shared resource at a time, thereby
  - avoiding *race conditions*
  - ensuring *data integrity*.

How does a mutex works?



- Using a mutex, each thread can lock the access to shared resources.
- Other threads have to wait for the thread to unlock the access.

Mutex in Posix (1)

```
pthread_mutex_init (mutex,attr)
pthread_mutex_destroy (mutex)
pthread_mutexattr_init (attr)
pthread_mutexattr_destroy (attr)
```

- Mutex variables are type **`pthread_mutex_t`**
- Attr variables are type **`pthread_mutexattr_t`**
  - It allows to modify properties like recursiveness or robustness
- Mutexes always are initialized **unlocked.**

Mutexes in Posix (2)

```
pthread_mutex_lock (mutex)
pthread_mutex_trylock (mutex)
pthread_mutex_unlock (mutex)
```

- **`pthread_mutex_lock:`** initializes a mutually excluded zone.
  - If another thread has locked the mutex, the thread becomes locked.
- **`pthread_mutex_unlock (mutex):`** Find if getting into the mutually excluded zone is possible.
  - If the mutex is locked, the function returns an error, but it does not get locked.
  - It's helpful to prevent **deadlocks** and **priority inversion.**

Conditional variables

- **Conditional Variable**: A condition variable is a synchronization primitive that enables threads to wait for certain conditions to be met before continuing. It allows one or more threads to "sleep" until another thread signals that the condition has been met.

- I’m willingly giving you the lock control so you can finish your work while I sleep.

Conditional variables in posix

- How `pthread_cond_wait()` Works:
  - A thread calls `pthread_cond_wait()` when it needs to wait for a certain condition.
  - Before calling `pthread_cond_wait()`, the thread must lock a mutex.
  - `pthread_cond_wait()` does two things:
    - It **unlocks the mutex** so that **other threads can acquire the lock** and change the shared data the thread is waiting for.
    - It puts **the thread in a blocked (waiting) state** until it is signaled by another thread that the condition has changed.

Example conditional variables

```
pthread_cond_t cond_var;
pthread_mutex_t mutex;
```

Global variables

THREAD A:

```
///
pthread_mutex_lock(&mutex);
///do some work
…
///willingly go to sleep so others can work
pthread_cond_wait(&cond_var, &mutex);
pthread_mutex_unlock(&mutex);
```

THREAD B:

```
///
pthread_mutex_lock(mutex);
///do some work
…
///notify other threads that the work is done
pthread_cond_signal(&cond_var);
pthread_mutex_unlock(&mutex);
```

Thank you

Juan F. Medina

Juan.medina26@upr.edu