POSIX THREADS PROGRAMMING – 2

Emmanuel S. Pilli

Mutex Variables

- Mutex is an abbreviation for "mutual exclusion".
- Mutex variables are one of the primary means of implementing thread synchronization and for protecting shared data when multiple writes occur.
- A mutex variable acts like a "lock" protecting access to a shared data resource.

Mutex Variables

- The basic concept of a mutex as used in Pthreads is that only one thread can lock (or own) a mutex variable at any given time.
- Even if several threads try to lock a mutex only one thread will be successful.
- No other thread can own that mutex until the owning thread unlocks that mutex.
- Threads must "take turns" accessing protected data.

Race Conditions

- Mutexes can be used to prevent "race" conditions.

Thread 1	Thread 2	Balance
Read balance: $1000		$1000
	Read balance: $1000	$1000
	Deposit $200	$1000
Deposit $200		$1000
Update balance $1000+$200		$1200
	Update balance $1000+$200	$1200

Mutex

- Create and initialize a mutex variable
- Several threads attempt to lock the mutex
- Only one succeeds and that thread owns the mutex
- The owner thread performs some set of actions
- The owner unlocks the mutex
- Another thread acquires the mutex and repeats the process
- Finally the mutex is destroyed

Create and Destroy Mutex

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

Create and Destroy Mutex

- Mutex variables must be declared with type **pthread_mutex_t**, and must be initialized before they can be used.
- There are two ways to initialize a mutex variable:
  - Statically, when it is declared
    pthread_mutex_t mymutex =
    PTHREAD_MUTEX_INITIALIZER;
  - Dynamically, with the **pthread_mutex_init()** routine. This method permits setting mutex object attributes, *attr*
- The mutex is initially unlocked.

Create and Destroy Mutex

- The *attr* object is used to establish properties for the mutex object, and must be of type **pthread_mutexattr_t** if used (may be specified as NULL to accept defaults).
- The **pthread_mutexattr_init()** and **pthread_mutexattr_destroy()** routines are used to create and destroy mutex attribute objects respectively.
- **pthread_mutex_destroy()** should be used to free a mutex object which is no longer needed.

Lock and Unlock Mutex

- pthread_mutex_lock (mutex)
-
- pthread_mutex_trylock (mutex)

- pthread_mutex_unlock (mutex)

Lock and Unlock Mutex

- The **pthread_mutex_lock()**routine is used by a thread to acquire a lock on the *mutex* variable.
- If the mutex is already locked by another thread, this call will block the calling thread until the mutex is unlocked.
- **pthread_mutex_trylock()** will lock a mutex.
- However, if the mutex is already locked, the routine will return immediately with a "busy" error code.
- This routine may be useful in preventing deadlock conditions, as in a priority-inversion situation.

Lock and Unlock Mutex

- **pthread_mutex_unlock()** will unlock a mutex if called by the owning thread.
- Calling this routine is required after a thread has completed its use of protected data and other threads need access.
- An error will be returned if:
  - If the mutex was already unlocked
  - If the mutex is owned by another thread
- Mutexes are akin to a "gentlemen's agreement"
- It is up to the code writer to insure that the necessary threads all make the mutex lock and unlock calls correctly