NSConditionLock

Inherits From:	NSObject
Conforms To:	NSLocking NSObject (NSObject)
Declared In:	foundation/NSLock.h

Class Description

The NSConditionLock class defines objects whose locks can be associated with specific, user-defined conditions. For example, using an NSConditionLock object, a thread can request a lock only if a certain condition is met. Once it has acquired the lock and executed the critical section of code, the thread could relinquish the lock and set the associated condition to something new. The conditions themselves are arbitrary: You define them as needed for your application.

Typically, you use this class when threads in your application need to execute in a particular order, such as when one thread produces data that another consumes. While the producer is executing, the consumer sleeps waiting to acquire a lock that's conditional upon the producer's completion of its operation. An application can have multiple NSConditionLock objects, each protecting different sections of code. However, these objects must be created before the application becomes multithreaded.

The locking and unlocking methods that NSConditionLock objects respond to can be used in any combination. For example, a **lock** message can be paired with **unlockWithCondition:**, or a **lockWhenCondition:** message can be paired with **unlock**.

The following example shows how the producer-consumer problem might be handled using condition locks. Imagine that an application contains a queue of data. A producer thread adds data to the queue, and consumer threads extract data from the queue.

The producer need not wait for a condition, but must wait for the lock to be made available so it can safely add data to the queue. For example, a producer could use a lock this way:

```
id condLock = [[NSConditionLock alloc] initWithCondition:NO_DATA];

[condLock lock];
/* Add data to the queue */
[condLock unlockWithCondition:HAS_DATA];
```

A consumer thread can then wait until there's data available and all other threads are out of locked critical sections. In the following code, the consumer sleeps until the producer invokes **unlockWithCondition:** with the parameter HAS_DATA:

```
[condLock lockWhenCondition:HAS_DATA];
/* Remove data from the queue */
[condLock unlockWithCondition:(isEmpty ? NO_DATA : HAS_DATA)];
```

The NSConditionLock, NSLock, and NSRecursiveLock classes all implement the NSLocking protocol with various features and performance characteristics; see the other class descriptions for more information.

Adopted Protocols

NSLocking	– lock – unlock

Method Types

Initializing an NSConditionLock	– initWithCondition:
Returning the Condition	– condition
Acquiring and Releasing a Lock	– lockBeforeDate: – lockWhenCondition: – lockWhenCondition:beforeDate: – unlockWithCondition:

Instance Methods

condition

– (int)**condition**

Returns the condition that's associated with the receiver. If no condition has been set, returns 0.

See also: – **initWithCondition:**

initWithCondition:

– (id)**initWithCondition:**(int)*condition*

Initializes a newly allocated NSConditionLock and sets its condition to *condition*. The value of the *condition* argument is user-defined; see the class description for more information. Returns **self**.

See also: – **condition:**

lockBeforeDate:

– (BOOL)**lockBeforeDate:**(NSDate *)*limit*

Attempts to acquire a lock before the date represented by *limit*. The condition associated with the receiver isn't taken into account in this operation. Returns YES if the lock is acquired within the time limit. Returns NO if the time limit expires before a lock can be acquired.

See also: – **lockWhenCondition:beforeDate:**

lockWhenCondition:

– (void)**lockWhenCondition:**(int)*condition*

Attempts to acquire a lock. The receiver's condition must be equal to *condition* before the locking operation will succeed. This method blocks the thread's execution until the lock can be acquired.

See also: – **lockWhenCondition:beforeDate:**, – **unlockWithCondition:**

lockWhenCondition:beforeDate:

– (BOOL)**lockWhenCondition:**(int)*condition* **beforeDate:**(NSDate *)*limit*

Attempts to acquire a lock before the date represented by *limit*. The receiver's condition must be equal to *condition* before the locking operation will succeed. Returns YES if the lock is acquired within this time limit. Returns NO if the time limit expires before a lock can be acquired.

See also: – **lockBeforeDate:**, – **lockWhenCondition:**

unlockWithCondition:

– (void)**unlockWithCondition:**(int)*condition*

Relinquishes the lock and sets the receiver's condition to *condition*.

See also: – **lockWhenCondition:**