NXConditionLock

Inherits From: Object

Conforms To: NXLock

Declared In: machkit/NXLock.h

Class Description

NXConditionLock is a type of lock with which a state can be used. The user of the lock can request that the lock be acquired when it enters a particular state, and can reset the state when releasing the lock. The meaning of this state is defined by the user of the lock. NXConditionLock is well suited to synchronizing different modules, such as a producer-consumer problem where a producer and consumer must share data but the consumer should sleep until a condition is met (such as, until data is available).

NXConditionLock class provides two ways of locking its objects (**lock** and **lockWhen:**) and two ways of unlocking (**unlock** and **unlockWith:**). Any combination of locking method and unlocking method is legal. Following is an example of how the producer-consumer problem might be handled using condition locks. The producer need not wait for a condition, but must wait for the lock to be made available so it can safely create shared data. Example producer code follows:

```
id condLock;       // uses currentState to guard access to data

/* create the lock only once, and set initial state */
condLock = [[NXConditionLock alloc] initWith:NO_DATA];

while (/*stuff to process*/) {
    [condLock lock];
    /* Manipulate global data, change state if needed. */
    [condLock unlockWith:DATA_AVAILABLE];
}
```

A consumer can then lock until the producer has data available and the producer is out of locked critical sections:

```
for(;;) {
    [condLock lockWhen:DATA_AVAILABLE];
    /* Manipulate global data... */
    [condLock unlockWith:NO_DATA];
}
```

An NXConditionLock doesn't busy-wait, so it can be used to lock time-consuming operations without degrading system performance.

The NXConditionLock, NXLock, NXRecursiveLock, and NXSpinLock classes all implement the NXLock protocol with various features and performance characteristics; see the other class descriptions for more information.

Instance Variables

None declared in this class.

Method Types

Initializing an instance	– init – initWith:
Get the condition of the lock	– condition
Acquire or release the lock	– lock – lockWhen: – unlock – unlockWith:

Instance Methods

condition

– (int)**condition**

Returns the lock's current condition. This condition can be set with the **initWith:** or **unlockWith:** methods.

init

– **init**

Initializes a newly allocated NXConditionLock instance and sets its condition to 0.

initWith:

– **initWith:**(int)*condition*

Initializes a newly allocated NXConditionLock instance and sets its condition to *condition*. This message should not be sent to an instance that has already been initialized.

lock

– **lock**

Waits until the lock isn't in use, then grabs the lock. The lock can subsequently be released with either **unlock** or **unlockWith:**.

lockWhen:

– **lockWhen:**(int)*condition*

Waits until the lock isn't in use and the lock's condition matches *condition*, then grabs the lock. The lock's condition can be set by **initWith:** or **unlockWith:**. The lock can subsequently be released with either **unlock** or **unlockWith:**.

unlock

– **unlock**

Releases the lock but doesn't change its condition.

See also: – **unlockWith:**

unlockWith:

– **unlockWith:**(int)*condition*

Sets the lock's condition to *condition* and releases the lock.

See also: – **unlock**