NSThread

Inherits From:	NSObject
Conforms To:	NSObject (NSObject)
Declared In:	Foundation/NSThread.h

lass Description

An NSThread object controls a thread of execution. Use NSThread when you want to have an Objective-C message run in its own thread of execution or if you need to terminate or delay the current thread.

A *thread* is an executable unit. A *task* is made up of one or more threads. Each thread has its own execution stack and is capable of independent input/output. All threads share the virtual memory address space and communication rights of their task. When a thread is started, it is *detached* from its initiating thread. The new thread runs independently. That is, the initiating thread does not know the new thread's state.

To have an Objective-C message run in its own thread of execution, send the message **detachNewThreadSelector:toTarget:withObject:** to the NSThread class object. This method detaches a new thread from the current thread, and the specified target executes the specified method in that thread. When the target has finished executing the method, the thread exits.

When you use **detachNewThreadSelector:toTarget:withObject:**, your application becomes multithreaded. At any time you can send **isMultiThreaded** to find out if the application is multithreaded, that is, if a thread was ever detached from the main thread. **isMultiThreaded** returns YES even if the detached thread has completed execution.

Note: Do not interchange the use of the **cthreads** functions and NSThread objects within an application. In particular, do not use **cthread_fork()** to create a thread that executes an Objective-C message. **isMultiThreaded** returns YES only if **detachNewThreadSelector:toTarget:withObject:** was used to create the thread.

If you need to terminate the current thread, send the **exit** message to the NSThread class object. Similarly, you send the **sleepUntilDate:** message to the NSThread class object to block the current thread for a period of time.

lethod Types

Querying an NSThread	+ isMultiThreaded
	+ currentThread
	– threadDictionary

Detaching a thread	+ detachNewThreadSelector:toTarget:withObject:
Stopping the Current Thread	+ sleepUntilDate: + exit

lass Methods

currentThread

+ (NSThread *)**currentThread**

Returns an object representing the current thread of execution.

See also: + **detachNewThreadSelector:toTarget:withObject:**

detachNewThreadSelector:toTarget:withObject:

+ (void)**detachNewThreadSelector:**(SEL)*aSelector*
toTarget:(id)*aTarget*
withObject:(id)*anArgument*

Detaches a new thread for the message **[***aTarget aSelector***:***anArgument***]**. The method *aSelector* must take only one argument and must not have a return value. The objects *aTarget* and *anArgument* are retained during the execution of the detached thread, then released. The detached thread is exited (using the **exit** class method) as soon as *aTarget* has completed executing the *aSelector* method.

Most OpenStep kits are not capable of being used by several threads simultaneously. In particular, the Application Kit cannot be used by more than one thread at a time.

If this is the first thread detached from the current thread, this method posts the NSWillBecomeMultiThreadedNotification with **nil** to the default notification center.

See also: + **currentThread**, + **isMultiThreaded**

exit

+ (void)**exit**

Terminates the current thread, using the **currentThread** class method to access that thread. Before exiting the thread, this method posts the NSThreadWillExitNotification with the thread being exited to the default notification center.

See also: + **currentThread**, + **sleepUntilDate:**

isMultiThreaded

+ (BOOL)**isMultiThreaded**

Returns YES if the application is multithreaded. An application is considered to be multithreaded if a thread was ever detached from the main thread using **detachNewThreadSelector:toTarget:withObject:**. If you detach a thread using the **cthread_fork()** function, this method returns NO. The detached thread does not have to be running for an application to be considered multithreaded; it may have already exited.

See also: + **detachNewThreadSelector:toTarget:withObject:**

sleepUntilDate:

+ (void)**sleepUntilDate:**(NSDate *)*aDate*

Blocks the current thread until the time specified by *aDate*. No run loop processing occurs while the thread is blocked.

See also: + **currentThread**, + **exit**

nstance Methods

threadDictionary

– (NSMutableDictionary *)**threadDictionary**

Returns the NSThread's dictionary, to which you can add data specific to the receiving NSThread. The thread dictionary is not used during any manipulations of the NSThread; it is simply a place where you can store any interesting attributes of a thread. For example, Foundation uses it to store the thread's NSRunLoop instance. You may define your own keys for the dictionary. Accessing the thread dictionary may be slower than it usually is to access an NSMutableDictionary.

otifications

NSWillBecomeMultiThreadedNotification

Notification Object	None
Userinfo	None

Posted when the first thread is detached from the current thread. NSThread posts this notification at most once—the first time a thread is detached using **detachNewThreadSelector:toTarget:withObject:**. Any subsequent invocations of **detachNewThreadSelector:toTarget:withObject:** do not post this notification.

NSThreadWillExitNotification

Notification Object	The exiting NSThread
Userinfo	None

Posted when a thread exits. NSThread posts this notification whenever it receives the **exit** message. The notification is posted to the current thread's default notification center. (The current thread is the thread being exited.)