
NSThread Class Reference

[Cocoa > Process Management](#)



2007-12-11



Apple Inc.
© 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Cocoa, Mac, Mac OS, and Objective-C are trademarks of Apple Inc., registered in the United States and other countries.

iPhone is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSThread Class Reference 5

Overview	5
Subclassing Notes	5
Tasks	6
Initializing an NSThread Object	6
Starting a Thread	6
Stopping a Thread	6
Determining the Thread's Execution State	6
Working with the Main Thread	7
Querying the Environment	7
Working with Thread Properties	7
Working with Thread Priorities	7
Class Methods	8
callStackReturnAddresses	8
currentThread	8
detachNewThreadSelector:toTarget:withObject:	8
exit	9
isMainThread	10
isMultiThreaded	10
mainThread	10
setThreadPriority:	11
sleepForTimeInterval:	11
sleepUntilDate:	12
threadPriority	12
Instance Methods	13
cancel	13
init	13
initWithTarget:selector:object:	14
isCancelled	15
isExecuting	15
isFinished	15
isMainThread	16
main	16
name	16
setName:	17
setStackSize:	17
stackSize	18
start	18
threadDictionary	18
Notifications	19
NSDidBecomeSingleThreadedNotification	19

NSThreadWillExitNotification 19
NSWillBecomeMultiThreadedNotification 19

Document Revision History 21

Index 23

NSThread Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Threading Programming Guide
Declared in	NSThread.h
Related sample code	QTAudioExtractionPanel QTEExtractAndConvertToAIFF QTEExtractAndConvertToMovieFile SimpleThreads Vertex Optimization

Overview

An `NSThread` object controls a thread of execution. Use this class when you want to have an Objective-C method run in its own thread of execution. Threads are especially useful when you need to perform a lengthy task, but don't want it to block the execution of the rest of the application. In particular, you can use threads to avoid blocking the main thread of the application, which handles user interface and event-related actions. Threads can also be used to divide a large job into several smaller jobs, which can lead to performance increases on multi-core computers.

Prior to Mac OS X v10.5, the only way to start a new thread is to use the [detachNewThreadSelector:toTarget:withObject:](#) (page 8) method. In Mac OS X v10.5 and later, you can create instances of `NSThread` and start them at a later time using the [start](#) (page 18) method.

In Mac OS X v10.5, the `NSThread` class supports semantics similar to those of `NSOperation` for monitoring the runtime condition of a thread. You can use these semantics to cancel the execution of a thread or determine if the thread is still executing or has finished its task. Canceling a thread requires support from your thread code; see the description for [cancel](#) (page 13) for more information.

Subclassing Notes

In Mac OS X v10.5 and later, you can subclass `NSThread` and override the `main` method to implement your thread's main entry point. If you override `main`, you do not need to invoke the inherited behavior by calling `super`.

Tasks

Initializing an NSThread Object

- `init` (page 13)
Returns an initialized NSThread object.
- `initWithTarget:selector:object:` (page 14)
Returns an NSThread object initialized with the given arguments.

Starting a Thread

- + `detachNewThreadSelector:toTarget:withObject:` (page 8)
Detaches a new thread and uses the specified selector as the thread entry point.
- `start` (page 18)
Starts the receiver.
- `main` (page 16)
The main entry point routine for the thread.

Stopping a Thread

- + `sleepUntilDate:` (page 12)
Blocks the current thread until the time specified.
- + `sleepForTimeInterval:` (page 11)
Sleeps the thread for a given time interval.
- + `exit` (page 9)
Terminates the current thread.
- `cancel` (page 13)
Changes the cancelled state of the receiver to indicate that it should exit.

Determining the Thread's Execution State

- `isExecuting` (page 15)
Returns a Boolean value that indicates whether the receiver is executing.
- `isFinished` (page 15)
Returns a Boolean value that indicates whether the receiver has finished execution.
- `isCancelled` (page 15)
Returns a Boolean value that indicates whether the receiver is cancelled.

Working with the Main Thread

- + [isMainThread](#) (page 10)
Returns a Boolean value that indicates whether the current thread is the main thread.
- [isMainThread](#) (page 16)
Returns a Boolean value that indicates whether the receiver is the main thread.
- + [mainThread](#) (page 10)
Returns the `NSThread` object representing the main thread.

Querying the Environment

- + [isMultiThreaded](#) (page 10)
Returns whether the application is multithreaded.
- + [currentThread](#) (page 8)
Returns the thread object representing the current thread of execution.
- + [callStackReturnAddresses](#) (page 8)
Returns an array containing the call stack return addresses.

Working with Thread Properties

- [threadDictionary](#) (page 18)
Returns the thread object's dictionary.
- [name](#) (page 16)
Returns the name of the receiver.
- [setName:](#) (page 17)
Sets the name of the receiver.
- [stackSize](#) (page 18)
Returns the stack size of the receiver.
- [setStackSize:](#) (page 17)
Sets the stack size of the receiver.

Working with Thread Priorities

- + [threadPriority](#) (page 12)
Returns the current thread's priority.
- + [setThreadPriority:](#) (page 11)
Sets the current thread's priority.

Class Methods

callStackReturnAddresses

Returns an array containing the call stack return addresses.

```
+ (NSArray *)callStackReturnAddresses
```

Return Value

An array containing the call stack return addresses. This value is `nil` by default.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSThread.h

currentThread

Returns the thread object representing the current thread of execution.

```
+ (NSThread *)currentThread
```

Return Value

A thread object representing the current thread of execution.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [detachNewThreadSelector:toTarget:withObject:](#) (page 8)

Declared In

NSThread.h

detachNewThreadSelector:toTarget:withObject:

Detaches a new thread and uses the specified selector as the thread entry point.

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

Parameters

aSelector

The selector for the message to send to the target. This selector must take only one argument and must not have a return value.

aTarget

The object that will receive the message *aSelector* on the new thread.

anArgument

The single argument passed to the target. May be `nil`.

Discussion

For non garbage-collected applications, the method *aSelector* is responsible for setting up an autorelease pool for the newly detached thread and freeing that pool before it exits. Garbage-collected applications do not need to create an autorelease pool.

The objects *aTarget* and *anArgument* are retained during the execution of the detached thread, then released. The detached thread is exited (using the `exit` (page 9) class method) as soon as *aTarget* has completed executing the *aSelector* method.

If this thread is the first thread detached in the application, this method posts the [NSWillBecomeMultiThreadedNotification](#) (page 19) with object `nil` to the default notification center.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [currentThread](#) (page 8)
- + [isMultiThreaded](#) (page 10)
- [start](#) (page 18)

Related Sample Code

OpenGLCaptureToMovie
 QTAudioExtractionPanel
 QTExtractAndConvertToAIFF
 SharedMemory
 SimpleThreads

Declared In

NSThread.h

exit

Terminates the current thread.

```
+ (void)exit
```

Discussion

This method uses the [currentThread](#) (page 8) class method to access the current thread. Before exiting the thread, this method posts the [NSThreadWillExitNotification](#) (page 19) with the thread being exited to the default notification center. Because notifications are delivered synchronously, all observers of [NSThreadWillExitNotification](#) (page 19) are guaranteed to receive the notification before the thread exits.

Invoking this method should be avoided as it does not give your thread a chance to clean up any resources it allocated during its execution.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [currentThread](#) (page 8)
- + [sleepUntilDate:](#) (page 12)

Related Sample Code

SimpleThreads

Vertex Optimization

Declared In

NSThread.h

isMainThread

Returns a Boolean value that indicates whether the current thread is the main thread.

+ (BOOL)isMainThread

Return Value

YES if the current thread is the main thread, otherwise NO.

Availability

Available in Mac OS X v10.5 and later.

See Also+ [mainThread](#) (page 10)**Declared In**

NSThread.h

isMultiThreaded

Returns whether the application is multithreaded.

+ (BOOL)isMultiThreaded

Return Value

YES if the application is multithreaded, NO otherwise.

Discussion

An application is considered multithreaded if a thread was ever detached from the main thread using either [detachNewThreadSelector:toTarget:withObject:](#) (page 8) or [start](#) (page 18). If you detached a thread in your application using a non-Cocoa API, such as the POSIX or Multiprocessing Services APIs, this method could still return NO. The detached thread does not have to be currently running for the application to be considered multithreaded—this method only indicates whether a single thread has been spawned.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSThread.h

mainThread

Returns the NSThread object representing the main thread.

+ (NSThread *)mainThread

Return Value

The NSThread object representing the main thread.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [isMainThread](#) (page 16)

Declared In

NSThread.h

setThreadPriority:

Sets the current thread's priority.

```
+ (BOOL)setThreadPriority:(double)priority
```

Parameters

priority

The new priority, specified with a floating point number from 0.0 to 1.0, where 1.0 is highest priority.

Return Value

YES if the priority assignment succeeded, NO otherwise.

Discussion

The priorities in this range are mapped to the operating system's priority values.

Availability

Available in Mac OS X v10.2 and later.

See Also

+ [threadPriority](#) (page 12)

Related Sample Code

ExtractMovieAudioToAIFF

QTAudioExtractionPanel

QTEExtractAndConvertToAIFF

QTEExtractAndConvertToMovieFile

Vertex Optimization

Declared In

NSThread.h

sleepForTimeInterval:

Sleeps the thread for a given time interval.

```
+ (void)sleepForTimeInterval:(NSTimeInterval)ti
```

Parameters*ti*

The duration of the sleep.

Discussion

No run loop processing occurs while the thread is blocked.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSThread.h

sleepUntilDate:

Blocks the current thread until the time specified.

+ (void)sleepUntilDate:(NSDate *)aDate

Parameters*aDate*

The time at which to resume processing.

Discussion

No run loop processing occurs while the thread is blocked.

Availability

Available in Mac OS X v10.0 and later.

See Also+ [currentThread](#) (page 8)+ [exit](#) (page 9)**Related Sample Code**

Core Data HTML Store

SharedMemory

SimpleThreads

TrivialThreads

Declared In

NSThread.h

threadPriority

Returns the current thread's priority.

+ (double)threadPriority

Return Value

The current thread's priority, which is specified by a floating point number from 0.0 to 1.0, where 1.0 is highest priority.

Discussion

The priorities in this range are mapped to the operating system's priority values. A “typical” thread priority might be 0.5, but because the priority is determined by the kernel, there is no guarantee what this value actually will be.

Availability

Available in Mac OS X v10.2 and later.

See Also

+ [setThreadPriority](#): (page 11)

Related Sample Code

ExtractMovieAudioToAIFF

QTAudioExtractionPanel

QTExtractAndConvertToAIFF

QTExtractAndConvertToMovieFile

Declared In

NSThread.h

Instance Methods

cancel

Changes the cancelled state of the receiver to indicate that it should exit.

```
- (void)cancel
```

Discussion

The semantics of this method are the same as those used for the `NSOperation` object. This method sets state information in the receiver that is then reflected by the `isCancelled` method. Threads that support cancellation should periodically call the `isCancelled` method to determine if the thread has in fact been cancelled, and exit if it has been.

For more information about cancellation and operation objects, see *NSOperation Class Reference*.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [isCancelled](#) (page 15)

Declared In

NSThread.h

init

Returns an initialized `NSThread` object.

```
- (id)init
```

Return Value

An initialized NSThread object.

Discussion

This is the designated initializer for NSThread.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [initWithTarget:selector:object:](#) (page 14)
- [start](#) (page 18)

Declared In

NSThread.h

initWithTarget:selector:object:

Returns an NSThread object initialized with the given arguments.

```
- (id)initWithTarget:(id)target
    selector:(SEL)selector
    object:(id)argument
```

Parameters

target

The object to which the message specified by *selector* is sent.

selector

The selector for the message to send to *target*. This selector must take only one argument and must not have a return value.

argument

The single argument passed to the target. May be *nil*.

Return Value

An NSThread object initialized with the given arguments.

Discussion

For non garbage-collected applications, the method *selector* is responsible for setting up an autorelease pool for the newly detached thread and freeing that pool before it exits. Garbage-collected applications do not need to create an autorelease pool.

The objects *target* and *argument* are retained during the execution of the detached thread. They are released when the thread finally exits.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [init](#) (page 13)
- [start](#) (page 18)

Declared In

NSThread.h

isCancelled

Returns a Boolean value that indicates whether the receiver is cancelled.

- (BOOL)isCancelled

Return Value

YES if the receiver has been cancelled, otherwise NO.

Discussion

If your thread supports cancellation, it should call this method periodically and exit if it ever returns YES.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [cancel](#) (page 13)
- [isExecuting](#) (page 15)
- [isFinished](#) (page 15)

Declared In

NSThread.h

isExecuting

Returns a Boolean value that indicates whether the receiver is executing.

- (BOOL)isExecuting

Return Value

YES if the receiver is executing, otherwise NO.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [isCancelled](#) (page 15)
- [isFinished](#) (page 15)

Declared In

NSThread.h

isFinished

Returns a Boolean value that indicates whether the receiver has finished execution.

- (BOOL)isFinished

Return Value

YES if the receiver has finished execution, otherwise NO.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [isCancelled](#) (page 15)
- [isExecuting](#) (page 15)

Declared In

NSThread.h

isMainThread

Returns a Boolean value that indicates whether the receiver is the main thread.

- (BOOL)isMainThread

Return Value

YES if the receiver is the main thread, otherwise NO.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSThread.h

main

The main entry point routine for the thread.

- (void)main

Discussion

The default implementation of this method takes the target and selector used to initialize the receiver and invokes the selector on the specified target. If you subclass `NSThread`, you can override this method and use it to implement the main body of your thread instead. If you do so, you do not need to invoke `super`.

You should never invoke this method directly. You should always start your thread by invoking the `start` method.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [start](#) (page 18)

Declared In

NSThread.h

name

Returns the name of the receiver.

- (NSString *)name

Return Value

The name of the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [setName:](#) (page 17)

Declared In

NSThread.h

setName:

Sets the name of the receiver.

```
- (void)setName:(NSString *)n
```

Parameters

n

The name for the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [name](#) (page 16)

Declared In

NSThread.h

setStackSize:

Sets the stack size of the receiver.

```
- (void)setStackSize:(NSUInteger)s
```

Parameters

s

The stack size for the receiver. This value must be a multiple of 4KB.

Discussion

You must call this method before starting your thread. Setting the stack size after the thread has started changes the attribute size (which is reflected by the [stackSize](#) (page 18) method), but it does not affect the actual number of pages set aside for the thread.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [stackSize](#) (page 18)

Declared In

NSThread.h

stackSize

Returns the stack size of the receiver.

- (NSUInteger)stackSize

Return Value

The stack size of the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [setStackSize:](#) (page 17)

Declared In

NSThread.h

start

Starts the receiver.

- (void)start

Discussion

This method spawns the new thread and invokes the receiver's `main` method on the new thread. If you initialized the receiver with a target and selector, the default `main` method invokes that selector automatically.

If this thread is the first thread detached in the application, this method posts the [NSWillBecomeMultiThreadedNotification](#) (page 19) with object `nil` to the default notification center.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [init](#) (page 13)
- [initWithTarget:selector:object:](#) (page 14)
- [main](#) (page 16)

Declared In

NSThread.h

threadDictionary

Returns the thread object's dictionary.

- (NSMutableDictionary *)threadDictionary

Return Value

The thread object's dictionary.

Discussion

You can use the returned dictionary to store thread-specific data. The thread dictionary is not used during any manipulations of the `NSThread` object—it is simply a place where you can store any interesting data. For example, Foundation uses it to store the thread's default `NSConnection` and `NSAssertionHandler` instances. You may define your own keys for the dictionary.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSThread.h`

Notifications

NSDidBecomeSingleThreadedNotification

Not implemented.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSThread.h`

NSThreadWillExitNotification

An `NSThread` object posts this notification when it receives the `exit` (page 9) message, before the thread exits. Observer methods invoked to receive this notification execute in the exiting thread, before it exits.

The notification object is the exiting `NSThread` object. This notification does not contain a `userInfo` dictionary.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSThread.h`

NSWillBecomeMultiThreadedNotification

Posted when the first thread is detached from the current thread. The `NSThread` class posts this notification at most once—the first time a thread is detached using `detachNewThreadSelector:toTarget:withObject:` (page 8) or the `start` (page 18) method.

Subsequent invocations of those methods do not post this notification. Observers of this notification have their notification method invoked in the main thread, not the new thread. The observer notification methods always execute before the new thread begins executing.

This notification does not contain a notification object or a `userInfo` dictionary.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSThread.h

Document Revision History

This table describes the changes to *NSThread Class Reference*.

Date	Notes
2007-12-11	Corrected some erroneous method descriptions and updated the class discussion.
2007-02-21	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

C

callStackReturnAddresses **class method** 8
cancel **instance method** 13
currentThread **class method** 8

D

detachNewThreadSelector:toTarget:withObject:
class method 8

E

exit **class method** 9

I

init **instance method** 13
initWithTarget:selector:object: **instance method**
14
isCancelled **instance method** 15
isExecuting **instance method** 15
isFinished **instance method** 15
isMainThread **class method** 10
isMainThread **instance method** 16
isMultiThreaded **class method** 10

M

main **instance method** 16
mainThread **class method** 10

N

name **instance method** 16
NSDidBecomeSingleThreadedNotification
notification 19
NSThreadWillExitNotification **notification** 19
NSWillBecomeMultiThreadedNotification
notification 19

S

setName: **instance method** 17
setStackSize: **instance method** 17
setThreadPriority: **class method** 11
sleepForTimeInterval: **class method** 11
sleepUntilDate: **class method** 12
stackSize **instance method** 18
start **instance method** 18

T

threadDictionary **instance method** 18
threadPriority **class method** 12