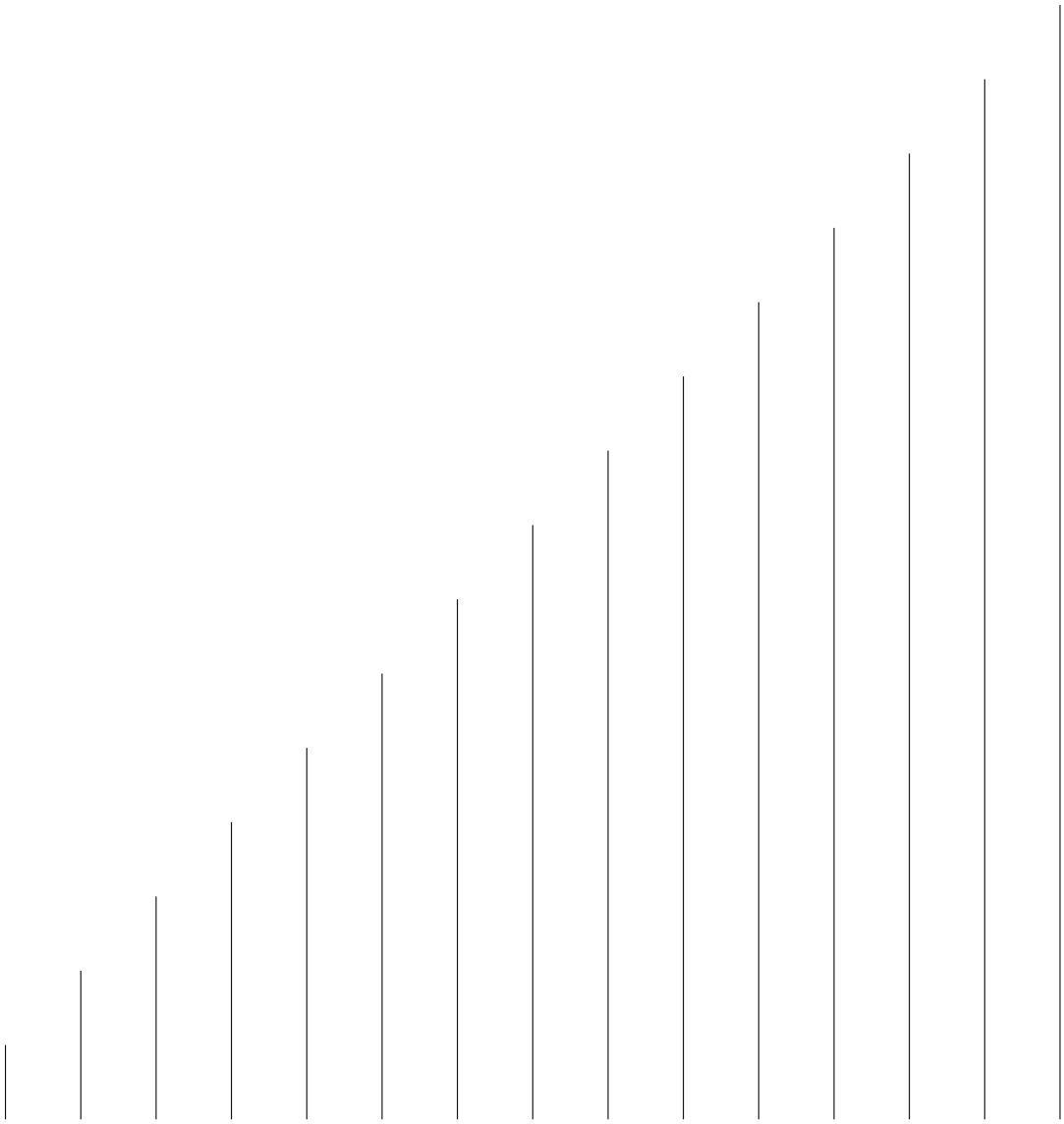
GNUstep Base



Author: Richard Frith-Macdonald (rfm@gnu.org)
Version: Revision: 1.7

Base

The GNUstep base library is a free software package implementing the API of the OpenStep Foundation Kit (tm), including later additions. This documentation package describes the core of the base library, for documentation on additional classes, see the BaseAdditions documentation package.

Compatibility

GNUstep is generally compatible with the OpenStep specification and with recent developments of the MacOS (cocoa) API. Where MacOS deviates from the OpenStep API, GNUstep generally attempts to support both versions. In some cases the newer MacOS APIs are incompatible with OpenStep, and GNUstep usually supports the richer version. See the section for more information on OpenStep Compliance.
In order to deal with compatiblity issues, GNUstep uses two mechanisms - it provides conditionally compiled sections of the library header files, so that software can be built that will conform strictly to a particular API, and it provides user default settings to control the behavior of the library at runtime.

Conditional compilation

Adding an option to a makefile to define one of the following preprocessor constants will modify the API visible to software being compiled -

NO_GNUSTEP GNUstep specific extensions to the OpenStep and MacOS cocoa APIs are excluded from the headers.

STRICT_MACOS_X Only methods and classes that are part of the MacOS cocoa API are made available in the headers.

STRICT_OPENSTEP Only methods and classes that are part of the OpenStep specification are made available in the headers.

NB These preprocessor constants are used in *developer code* (ie the code that users of GNUstep write) rather than by the GNUstep software itsself. They permit a developer to ensure that he/she does not write code which depends upon API not present on other implementations (in practice, MacOS-X or some old OPENSTEP systems).
The actual GNUstep libraries are always built with the full GNUstep API in place, so that the feature set is as consistent as possible.

User defaults

GSLogSyslog Setting the user default `GSLogSyslog` to `YES` will cause log/debug output to be sent to the syslog facility (on systems which support it), rather than to the standard error stream. This is useful in environments where stderr has been re-used strangely for some reason.

GSMacOSXCompatible Setting the user default `GSMacOSXCompatible` to `YES` will cause MacOS compatible behavior to be the default at runtime. This default may however be overridden to provide more fine grained control of system behavior.

GSOldStyleGeometry Specifies whether the functions for producing strings describing geometric structures (NSStringFromPoint(), NSStringFromSize(), and NSStringFromRect()) should produce strings conforming to the OpenStep specification or to MacOS-X behavior. The functions for parsing those strings should cope with both cases anyway.

GSSOCKS May be used to specify a default SOCKS5 server (and optionally a port separated from the server by a colon) to which tcp/ip connections made using the NSFileHandle extension methods should be directed.
This default overrides the SOCKS5_SERVER and SOCKS_SERVER environment variables.

Local Time Zone Used to specify the name of the timezone to be used by the NSTimeZone class.

NSWriteOldStylePropertyLists Specifies whether text property-list output should be in the default MacOS-X format (XML), or in the more human readable (but less powerful) original OpenStep format.
Reading of property lists is supported in either format, but *only* if GNUstep is built with the libxml library (which is needed to handle XML parsing).
NB. MacOS-X generates illegal XML for some strings - those which contain characters not legal in XML. GNUstep always generates legal XML, at the cost of a certain degree of compatibility. GNUstep XML property lists use a backslash to escape illegal chatracters, and consequently any string containing either a backslash or an illegal character will be written differently to the same string on MacOS-X.

NSLanguages An array of strings that lists the users

prefered languages, in order or preference. If not found the default is just English.

Environment variables

There are some environment variables used by GNUstep base, where there would be problems onbtaining data from the defaults asystem.

CRASH_ON_ABORT The default exception handler will either cause the program to simply terminate, or to crash - leaving a core dump. The standard behavior is to leave a core dump if the library was built for debugging, and to simply exit if it was not.
The CRASH_ON_ABORT environment variable can be used to override this behavior. If this is defined to *NO*, *FALSE*, or *0* then the program will simply exit when an exception occurs. Any other value of the variable will cause the program to generate a core dump.

GNUSTEP_STRING_ENCODING This is used to specify the default encoding for 8-bit strings. It defaults to NSISOLatin1StringEncoding, but may be any of the 8-bit encodings supported by your system (excluding multi-byte encodings).

GNUSTEP_HOST_CPU Used in place of GNUSTEP_TARGET_CPU if the other is missing.

GNUSTEP_HOST_DIR Used in place of GNUSTEP_TARGET_DIR if the other is missing.

GNUSTEP_HOST_OS Used in place of GNUSTEP_TARGET_OS if the other is missing.

GNUSTEP_LOCAL_ROOT Used to specify the GNUstep root directory for local (non-system) resources. Typically all locally produced or contributed software is installed relative to this.

GNUSTEP_NETWORK_ROOT Used to specify the GNUstep root directory for local (non-system) resources that are intended to be shared across a local network. Typically this is an NFS exported directory shared by many machines. It provides an alternative to GNUSTEP_LOCAL_

GNUSTEP_SYSTEM_ROOT Used to specify the GNUstep system root directory ... all system libraries, tools, applications, headers, resources in general are located relative to this.

GNUSTEP_USER_ROOT This environment variable,

commonly set by the make system, is **not** used by GNUstep programs. Instead values from `.GNUsteprc` are used (see later).

GNUSTEP_TARGET_CPU Overrides the default value of the machine (hardware) name used on this system.

GNUSTEP_TARGET_DIR Overrides the default path used to locate subdirectories for GNUstep binaries withing bundles and applications. This is normally equivalent to a path made up of the GNUSTEP_TARGET_CPU and GNUSTEP_TARGET_OS

GNUSTEP_TARGET_OS Overrides the default value of the operating system name used on this system.

GNUSTEP_TZ Used to specify the timezone to be used if there is no timezone specified in the user defaults system. The preferred mechanism is to use the 'Local Time Zone' value from the user defaults system.

HOMEDRIVE Used on windoze to locate the home directory.

HOMEPATH Used on windoze to locate the home directory.

LANGUAGES If there is no NSLanguages user default set, and there is no language infromation available in the native system locale mechanism, then this environment variable is used to provide a list of the languages that the user prefers to use. languages listed in this variable must be separated by semicolons.

LOGNAME This is used as the default value for the current user (as returned by the NSUserName() functions). If it is not specified, or contains an illegal value, other methods are used to get the user name.

LIBRARY_COMBO Used to override the default value of the combination of standard libraries used to build binaries. This value locates the final subdirectory used to locate binaries.

NSDeallocateZombies This may be used in conjunction with NSZombieEnabled to specify whether the objects should really be deallocated. If you set this to YES, the zombie logging will only work until the deallocated memory is re-used.

NSZombieEnabled If this is set to YES, then dealloca-

tion of an object causes the object to be morphed into a Zombie ... a special object which will call the GNUstep specific GSLogZombie() function to log the method call. You can set a breakpoint in this function and examine the process memory if you are running under a debugger.
As this overrides actual object deallocation, all memory allocated for objects will be leaked!

SOCKS5_SERVER Specifies the default socks server to be used when making outgoing tcp/ip connections using NSFileHandle. This may also specify a port after the host name (and spearated from it by a colon).
This environment variable is used only if the GSSOCKS user default is not set.

SOCKS_SERVER Equivalent to SOCKS5_SERVER, but used only if that is not defined.

TZ Used to specify the timezone to be used if there is no timezone specified by any other mechanism. The preferred mechanism is to use the 'Local Time Zone' value from the user defaults system.

.GNUsteprc files

The locations of the directories in which user specific files and the user defaults database are stored are defined in the `.GNUsteprc` files.
If no location is given for user specific files, they are stored in the GNUstep subdirectory of the users home directory.
If a separate location is not given for the defaults database, it is stored in the same directory as other user specific files.
The presence of a `.GNUsteprc` file in a users home directory premits the user to customize file locations using two commands -

GNUSTEP_DEFAULTS_ROOT=... The text after the '=' is taken to be the path to the users files. If it begins with a '~' character, the users home directory is prepended to it.

GNUSTEP_USER_ROOT=... The text after the '=' is taken to be the path to the users files. If it begins with a '~' character, the users home directory is prepended to it.

The presence of a `.GNUsteprc` file in the GNUstep system directory may provide default paths for all users and may even override user specific files. The content of this file is as for the user specific file, but permits two additional commands -

FORCE_DEFAULTS_ROOT If this line is present, and the file specifies a GNUSTEP_DEFAULTS_ROOT, then the value given in the system-wide file is used irrespective of the user specific file. Otherwise, the value in the user specific file takes precedence.

FORCE_USER_ROOT If this line is present, and the file specifies a GNUSTEP_USER_ROOT, then the value given in the system-wide file is used irrespective of the user specific file. Otherwise, the value in the user specific file takes precedence.

NSArchiver *XOG*

Inherits From: NSCoder : NSObject
Declared in: Foundation/NSArchiver.h

Description

Description forthcoming.

Class Methods

archiveRootObject:toFile: *XOG*

`+ (BOOL)` **archiveRootObject:** `(id)` *rootObject*
toFile: `(NSString*)` *path*

Description forthcoming.

archivedDataWithRootObject: *XOG*

`+ (NSData*)` **archivedDataWithRootObject:** `(id)` *rootObject*

Description forthcoming.

Instance Methods

archiverData *XOG*

`- (NSMutableData*)` **archiverData**

Description forthcoming.

classNameEncodedForTrueClassName: *XOG*

- (NSString*)**classNameEncodedForTrueClassName:**(NSString*) *trueName*

Description forthcoming.

encodeClassName:intoClassName: *XOG*

- (void)**encodeClassName:**(NSString*) *trueName*
intoClassName:(NSString*) *inArchiveName*

Description forthcoming.

initForWritingWithMutableData: *XOG*

- (id)**initForWritingWithMutableData:**(NSMutableData*) *mdata*

Description forthcoming.

replaceObject:withObject: *X¬OG*

- (void)**replaceObject:**(id) *object*
withObject:(id) *newObject*

Description forthcoming.

NSArray *XOG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying NSMutableCopying
Declared in:	Foundation/NSArray.h

Description

A simple, low overhead, ordered container for objects. All the objects in the container are retained by it. The container may not contain `nil` (though it may contain [NSNull +null]).

Class Methods

array *XOG*

`+ (id)` **array**

Returns an empty autoreleased array.

arrayWithArray: *XOG*

`+ (id)` **arrayWithArray:** `(NSArray*)` *array*

Returns a new autoreleased NSArray instance containing all the objects from *array* , in the same order as the original.

arrayWithContentsOfFile: *XOG*

`+ (id)` **arrayWithContentsOfFile:** `(NSString*)` *file*

Returns an autoreleased array based upon the *file*. The new array is created using +allocWithZone: and initialised using the -initWithContentsOfFile: method. See the documentation for those methods for more detail.

arrayWithObject: *XOG*

`+ (id)`**arrayWithObject:** `(id)` *anObject*

Returns an autoreleased array containing *anObject*.

arrayWithObjects: *XOG*

`+ (id)`**arrayWithObjects:** `(id)` *firstObject*

Returns an autoreleased array containing the list of objects, preserving order.

arrayWithObjects:count: *XOG*

`+ (id)`**arrayWithObjects:** `(id*)` *objects*
count: `(unsigned)` *count*

Returns an autoreleased array containing the specified *objects*, preserving order.

Instance Methods

arrayByAddingObject: *XOG*

`- (NSArray*)`**arrayByAddingObject:**`(id)` *anObject*

Returns an autoreleased array formed from the contents of the receiver and adding *anObject* as the last item.

arrayByAddingObjectsFromArray: *XOG*

`- (NSArray*)`**arrayByAddingObjectsFromArray:**`(NSArray*)` *anotherArray*

Returns a new array which is the concatenation of self and otherArray (in this precise order).

componentsJoinedByString: *XOG*

- (NSString*)**componentsJoinedByString:**(NSString*) *separator*

Returns a string formed by concatenating the objects in the receiver, with the specified *separator* string inserted between each part.

containsObject: *XOG*

- (BOOL)**containsObject:**(id) *anObject*

Returns YES if *anObject* belongs to self. No otherwise.
The -isEqual: method of *anObject* is used to test for equality.

count *XOG*

- (unsigned)**count**

Returns the number of elements contained in the receiver.

description *XOG*

- (NSString*)**description**

Returns the result of invoking -descriptionWithLocale:indent: with a nil locale and zero indent.

descriptionWithLocale: *XOG*

- (NSString*)**descriptionWithLocale:**(NSDictionary*) *locale*

Returns the result of invoking -descriptionWithLocale:indent: with a zero indent.

descriptionWithLocale:indent: *XOG*

- (NSString*)**descriptionWithLocale:**(NSDictionary*) *locale*
indent:(unsigned int) *level*

Returns the receiver as a text property list in the traditional format.
See [NSString -propertyList] for details.
If *locale* is nil, no formatting is done, otherwise entries are formatted according to the *locale*, and indented according to *level*.

Unless *locale* is `nil`, a *level* of zero indents items by four spaces, while a *level* of one indents them by a tab.

The items in the property list string appear in the same order as they appear in the receiver.

firstObjectCommonWithArray: *XOG*

- `(id)`**firstObjectCommonWithArray:**`(NSArray*)` *otherArray*

Returns the first object found in the receiver (starting at index 0) which is present in the *otherArray* as determined by using the -containsObject: method.

getObjects: *XOG*

- `(void)`**getObjects:**`(id*)` *aBuffer*

Copies the objects from the receiver to *aBuffer*, which must be an area of memory large enough to hold them.

getObjects:range: *XOG*

- `(void)`**getObjects:**`(id*)` *aBuffer*
range:`(NSRange)` *aRange*

Copies the objects from the range *aRange* of the receiver to *aBuffer*, which must be an area of memory large enough to hold them.

indexOfObject: *XOG*

- `(unsigned)`**indexOfObject:**`(id)` *anObject*

Returns the index of the first object found in the receiver which is equal to *anObject* (using anObject's -isEqual: method). Returns NSNotFound on failure.

indexOfObject:inRange: *XOG*

- `(unsigned)`**indexOfObject:**`(id)` *anObject*
inRange:`(NSRange)` *aRange*

Returns the index of the first object found in *aRange* of receiver which is equal to *anObject* (using anObject's -isEqual: method). Returns NSNotFound on failure.

indexOfObjectIdenticalTo: *XOG*

- (unsigned)**indexOfObjectIdenticalTo:**(id) *anObject*

Returns the index of the specified object in the receiver, or NSNotFound if the object is not present.

indexOfObjectIdenticalTo:inRange: *XOG*

- (unsigned)**indexOfObjectIdenticalTo:**(id) *anObject*
  inRange:(NSRange) *aRange*

Returns the index of the specified object in the range of the receiver, or NSNot-Found if the object is not present.

initWithArray: *XOG*

- (id)**initWithArray:**(NSArray*) *array*

Initialize the receiver with the contents of *array*. The order of *array* is preserved.
Invokes -initWithObjects:count:

initWithArray:copyItems: *X¬OG*

- (id)**initWithArray:**(NSArray*) *array*
  copyItems:(BOOL) *shouldCopy*

Initialize the receiver with the contents of *array*. The order of *array* is preserved.
If *shouldCopy* is YES then the objects are copied rather than simply retained.
Invokes -initWithObjects:count:

initWithContentsOfFile: *XOG*

- (id)**initWithContentsOfFile:**(NSString*) *file*

Initialises the array with the contents of the specified *file*, which must contain an array in property-list format.

In GNUstep, the property-list format may be either the OpenStep format (ASCII data), or the MacOS-X format (URF8 XML data)... this method will recognise which it is.
If there is a failure to load the *file* for any reason, the receiver will be released, the method will return `nil`, and a warning may be logged.
Works by invoking [NSString -initWithContentsOfFile:] and [NSString -propertyList] then checking that the result is an array.

initWithObjects: *XOG*

- (id)initWithObjects:(id) *firstObject*

Initialize the array the list of objects.
May change the value of self before returning it.

initWithObjects:count: *XOG*

- (id)initWithObjects:(id*) *objects*
count:(unsigned) *count*

Initialize the array with *count objects*.
Retains each object placed in the array.
Like all initializers, may change the value of self before returning it.

isEqualToArray: *XOG*

- (BOOL)isEqualToArray:(NSArray*) *otherArray*

Returns `YES` if the receiver is equal to *otherArray*, `NO` otherwise.

lastObject *XOG*

- (id)lastObject

Returns the last object in the receiver, or `nil` if the receiver is empty.

makeObjectsPerform: *O¬XG*

- (void)makeObjectsPerform:(SEL) *aSelector*

Obsolete version of -makeObjectsPerformSelector:

makeObjectsPerform:withObject: $O\neg XG$

- (void)**makeObjectsPerform:**(SEL) *aSelector*
  withObject:(id) *argument*

Obsolete version of -makeObjectsPerformSelector:withObject:

makeObjectsPerformSelector: $X\neg OG$

- (void)**makeObjectsPerformSelector:**(SEL) *aSelector*

Makes each object in the array perform *aSelector*.
This is done sequentially from the last to the first object.

makeObjectsPerformSelector:withObject: $X\neg OG$

- (void)**makeObjectsPerformSelector:**(SEL) *aSelector*
  withObject:(id) *arg*

Makes each object in the array perform *aSelector* with *arg*.
This is done sequentially from the last to the first object.

objectAtIndex: XOG

- (id)**objectAtIndex:**(unsigned) *index*

Returns the object at the specified *index*. Raises an exception of the *index* is beyond the array.

objectEnumerator XOG

- (NSEnumerator*)**objectEnumerator**

Returns an enumerator describing the array sequentially from the first to the last element.
If you use a mutable subclass of NSArray, you should not modify the array during enumeration.

pathsMatchingExtensions: *XOG*

- (NSArray*)**pathsMatchingExtensions:**(NSArray*) *extensions*

Assumes that the receiver is an array of paths, and returns an array formed by selecting the subset of those patch matching the specified array of *extensions*.

reverseObjectEnumerator *XOG*

- (NSEnumerator*)**reverseObjectEnumerator**

Returns an enumerator describing the array sequentially from the last to the first element.
If you use a mutable subclass of NSArray, you should not modify the array during enumeration.

sortedArrayHint *XOG*

- (NSData*)**sortedArrayHint**

Subclasses may provide a hint for sorting... The default GNUstep implementation just returns `nil`.

sortedArrayUsingFunction:context: *XOG*

- (NSArray*)**sortedArrayUsingFunction:**(NSComparisonResult(*)(id,id,void*)) c
context:(void*) *context*

Returns an autoreleased array in which the objects are ordered according to a sort with *comparator*. This invokes -sortedArrayUsingFunction:context:hint: with a `nil` hint.

sortedArrayUsingFunction:context:hint: *XOG*

- (NSArray*)**sortedArrayUsingFunction:**(NSComparisonResult(*)(id,id,void*)) c
context:(void*) *context*
hint:(NSData*) *hint*

Returns an autoreleased array in which the objects are ordered according to a sort with *comparator*, where the *comparator* function is passed two objects to compare, and the copntext as the third argument.

sortedArrayUsingSelector: *XOG*

- (NSArray*)**sortedArrayUsingSelector:**(SEL) *comparator*

Returns an autoreleased array in which the objects are ordered according to a sort with *comparator*.

subarrayWithRange: *XOG*

- (NSArray*)**subarrayWithRange:**(NSRange) *aRange*

Returns a subarray of the receiver containing the objects found in the specified range *aRange*.

writeToFile:atomically: *XOG*

- (BOOL)**writeToFile:**(NSString*) *path*
  atomically:(BOOL) *useAuxiliaryFile*

Writes the contents of the array to the file specified by *path*. The file contents will be in property-list format... under GNUstep this is either OpenStep style (ASCII characters using \U hexadecimal escape sequences for unicode), or MacOS-X style (XML in the UTF8 character set).

If the *useAuxiliaryFile* flag is YES, the file write operation is atomic... the data is written to a temporary file, which is then renamed to the actual file name.

If the conversion of data into the correct property-list format fails or the write operation fails, the method returns NO, otherwise it returns YES.

NB. The fact that the file is in property-list format does not necessarily mean that it can be used to reconstruct the array using the -initWithContentsOfFile: method. If the original array contains non-property-list objects, the descriptions of those objects will have been written, and reading in the file as a property-list will result in a new array containing the string descriptions.

writeToURL:atomically: *X¬OG*

- (BOOL)**writeToURL:**(NSURL*) *url*
  atomically:(BOOL) *useAuxiliaryFile*

Writes the contents of the array to the specified *url*. This functions just like -writeToFile:atomically: except that the output may be written to any URL, not just a local file.

NSAssertionHandler *XOG*

Inherits From: NSObject
Declared in: Foundation/NSException.h

Description

NSAssertionHandler objects are used to raise exceptions on behalf of macros implementing assertions.

Each thread has its own assertion handler instance.

The macros work together with the assertion handler object to produce meaningful exception messages containing the name of the source file, the position within that file, and the name of the ObjC method or C function in which the assertion failed. The assertion macros are: NSAssert() , NSCAssert(), NSAssert1() , NSCAssert1(), NSAssert2() , NSCAssert2(), NSAssert3() , NSCAssert3(), NSAssert4() , NSCAssert4(), NSAssert5() , NSCAssert5(), NSParameterAssert() , NSCParameterAssert()

Class Methods

currentHandler *XOG*

`+ (NSAssertionHandler*)`**currentHandler**

Returns the assertion handler object for the current thread.

If none exists, creates one and returns it.

Instance Methods

handleFailureInFunction:file:lineNumber:description: *XOG*

`- (void)`**handleFailureInFunction:**`(NSString*)` *functionName*
file:`(NSString*)` *fileName*
lineNumber:`(int)` *line*
description:`(NSString*)` *format*

Handles an assertion failure by using NSLogv() to print an error message built from the supplied arguments, and then raising an NSInternalInconsistencyException

handleFailureInMethod:object:file:lineNumber:description: *XOG*

- (void)**handleFailureInMethod:**(SEL) *aSelector*
object:(id) *object*
file:(NSString*) *fileName*
lineNumber:(int) *line*
description:(NSString*) *format*

Handles an assertion failure by using NSLogv() to print an error message built from the supplied arguments, and then raising an NSInternalInconsistencyException

NSAttributedString *X¬OG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying NSMutableCopying
Declared in:	Foundation/NSAttributedString.h

Description

Description forthcoming.

Instance Methods

attribute:atIndex:effectiveRange: *X¬OG*

- (id) **attribute:**(NSString*) *attributeName*
atIndex:(unsigned int) *index*
effectiveRange:(NSRange*) *aRange*

Description forthcoming.

attribute:atIndex:longestEffectiveRange:inRange: *X¬OG*

- (id) **attribute:**(NSString*) *attributeName*
atIndex:(unsigned int) *index*
longestEffectiveRange:(NSRange*) *aRange*
inRange:(NSRange) *rangeLimit*

Description forthcoming.

attributedSubstringFromRange: *X¬OG*

- (NSAttributedString*)**attributedSubstringFromRange:**(NSRange) *aRange*

Description forthcoming.

attributedSubstringWithRange: ¬X¬OG

- (NSAttributedString*)**attributedSubstringWithRange:**(NSRange) *aRange*

Description forthcoming.

attributesAtIndex:effectiveRange: X¬OG

- (NSDictionary*)**attributesAtIndex:**(unsigned int) *index*
effectiveRange:(NSRange*) *aRange*

Description forthcoming.

attributesAtIndex:longestEffectiveRange:inRange: X¬OG

- (NSDictionary*)**attributesAtIndex:**(unsigned int) *index*
longestEffectiveRange:(NSRange*) *aRange*
inRange:(NSRange) *rangeLimit*

Description forthcoming.

initWithAttributedString: X¬OG

- (id)**initWithAttributedString:**(NSAttributedString*) *attributedString*

Description forthcoming.

initWithString: X¬OG

- (id)**initWithString:**(NSString*) *aString*

Description forthcoming.

initWithString:attributes: X¬OG

- (id)**initWithString:**(NSString*) *aString*
attributes:(NSDictionary*) *attributes*

Description forthcoming.

isEqualToAttributedString: $X\neg OG$

- `(BOOL)`**isEqualToAttributedString:**`(NSAttributedString*)` *otherString*

Description forthcoming.

length $X\neg OG$

- `(unsigned int)`**length**

Description forthcoming.

string $X\neg OG$

- `(NSString*)`**string**

Description forthcoming.

NSAutoreleasePool *XOG*

Inherits From: NSObject
Declared in: Foundation/NSAutoreleasePool.h

Description

This class maintains a stack of autorelease pools objects in each thread.

When an autorelease pool is created, it is automatically added to the stack of pools in the thread.

When a pool is destroyed, it (and any pool later in the stack) is removed from the stack.

Class Methods

_endThread: *¬O¬XG*

`+ (void)_endThread: (NSThread*)` *thread*

Warning the underscore at the start of the name of this method indicates that it is private, for internal use only, and you should not use the method in your code.

addObject: *XOG*

`+ (void)addObject: (id)` *anObj*

Adds the specified object to the current autorelease pool. If there is no autorelease pool in the thread, a warning is logged.

autoreleaseCountForObject: *¬O¬XG*

`+ (unsigned)autoreleaseCountForObject: (id)` *anObject*

Counts the number of times that the specified object occurs in autorelease pools in the current thread.

This method is *slow* and should probably only be used for debugging purposes.

enableRelease: $\neg O \neg XG$

`+ (void)`**enableRelease:** `(BOOL)` *enable*

Specifies whether objects contained in autorelease pools are to be released when the pools are deallocated (by default `YES`).
You can set this to `NO` for debugging purposes.

freeCache $\neg O \neg XG$

`+ (void)`**freeCache**

When autorelease pools are deallocated, the memory they used is retained in a cache for re-use so that new polls can be created very quickly.
This method may be used to empty that cache, ensuring that the minimum memory is used by the application.

resetTotalAutoreleasedObjects $\neg O \neg XG$

`+ (void)`**resetTotalAutoreleasedObjects**

Description forthcoming.

setPoolCountThreshhold: $\neg O \neg XG$

`+ (void)`**setPoolCountThreshhold:** `(unsigned)` *c*

Specifies a limit to the number of objects that may be added to an autorelease pool. When this limit is reached an exception is raised.
You can set this to a smallish value to catch problems with code that autoreleases too many objects to operate efficiently.
Default value is maxint.

totalAutoreleasedObjects $\neg O \neg XG$

`+ (unsigned)`**totalAutoreleasedObjects**

Description forthcoming.

Instance Methods

addObject: *XOG*

- `(void)`**addObject:**`(id)` *anObj*

Adds the specified object to this autorelease pool.

NSBitmapCharSet *XOG*

Inherits From: NSCharacterSet : NSObject
Declared in: Foundation/NSBitmapCharSet.h

Description

Description forthcoming.

Instance Methods

initWithBitmap: *XOG*

- (id)**initWithBitmap:**(NSData*) *bitmap*

Description forthcoming.

NSBundle *XOG*

Inherits From: NSObject
Declared in: Foundation/NSBundle.h

Description

NSBundle provides methods for locating and handling application (and tool) resources at runtime. Resources includes any time of file that the application might need, such as images, nib (gorm or gmodel) files, localization files, and any other type of file that an application might need to use to function. Resources also include executable code, which can be dynamically linked into the application at runtime. These files and executable code are commonly put together into a directory called a bundle.

NSBundle knows how these bundles are organized and can search for files inside a bundle. NSBundle also handles locating the executable code, linking this in and initializing any classes that are located in the code. NSBundle also handles Frameworks, which are basically a bundle that contains a library archive. The organization of a framework is a little difference, but in most respects there is no difference between a bundle and a framework.

There is one special bundle, called the mainBundle, which is basically the application itself. The mainBundle is always loaded (of course), but you can still perform other operations on the mainBundle, such as searching for files, just as with any other bundle.

Class Methods

allBundles *XOG*

`+ (NSArray*)allBundles`

Return an array enumerating all the bundles in the application. This does not include frameworks.

allFrameworks *XOG*

`+ (NSArray*)allFrameworks`

Return an array enumerating all the frameworks in the application. This does not include normal bundles.

bundleForClass: *XOG*

`+ (NSBundle*)`**bundleForClass:** `(Class)` *aClass*

Return the bundle to which *aClass* belongs. If *aClass* was loaded from a bundle, return the bundle; if it belongs to a framework (either a framework linked into the application, or loaded dynamically), return the framework; in all other cases, return the main bundle.
Please note that GNUstep supports plain shared libraries, while the openstep standard, and other openstep-like systems, do not; the behaviour when *aClass* belongs to a plain shared library is at the moment still under investigation – you should consider it undefined since it might be changed. :-)

bundleWithPath: *XOG*

`+ (NSBundle*)`**bundleWithPath:** `(NSString*)` *path*

Return a bundle for the *path* at *path*. If *path* doesn't exist or is not readable, return `nil`. If you want the main bundle of an application or a tool, it's better if you use +mainBundle .

mainBundle *XOG*

`+ (NSBundle*)`**mainBundle**

Return the bundle containing the resources for the executable. If the executable is an application, this is the main application bundle (the xxx.app directory); if the executable is a tool, this is a bundle 'naturally' associated with the tool: if the tool executable is xxx/Tools/ix86/linux-gnu/gnu-gnu-gnu/Control then the tool's main bundle directory is xxx/Tools/Resources/Control.
NB: traditionally tools didn't have a main bundle – this is a recent GNUstep extension, but it's quite nice and it's here to stay.
The main bundle is where the application should put all of its resources, such as support files (images, html, rtf, txt,...), localization tables,.gorm (.nib) files, etc. gnustep-make (/ProjectCenter) allows you to easily specify the resource files to put in the main bundle when you create an application or a tool.

pathForResource:ofType:inDirectory: *XOG*

```
+ (NSString*)pathForResource: (NSString*) name
                      ofType: (NSString*) ext
                 inDirectory: (NSString*) bundlePath
```

Returns an absolute path for a resource *name* with the extension *ext* in the specified *bundlePath*. See also -pathForResource:ofType:inDirectory: for more information on searching a bundle.

pathForResource:ofType:inDirectory:withVersion: *XOG*

```
+ (NSString*)pathForResource: (NSString*) name
                      ofType: (NSString*) ext
                 inDirectory: (NSString*) bundlePath
                 withVersion: (int) version
```

This method has been depreciated. Version numbers were never implemented so this method behaves exactly like +pathForResource:ofType:inDirectory:.

pathsForResourcesOfType:inDirectory: *XOG*

```
+ (NSArray*)pathsForResourcesOfType: (NSString*) extension
                        inDirectory: (NSString*) bundlePath
```

Description forthcoming.

preferredLocalizationsFromArray: *X¬OG*

```
+ (NSArray*)preferredLocalizationsFromArray: (NSArray*) localizationsArray
```

Description forthcoming.

preferredLocalizationsFromArray:forPreferences: *X¬OG*

```
+ (NSArray*)preferredLocalizationsFromArray: (NSArray*) localizationsArray
                              forPreferences: (NSArray*) preferencesArray
```

Description forthcoming.

Instance Methods

bundleIdentifier *XOG*

- `(NSString*)`**bundleIdentifier**

Returns the bundle identifier, as defined by the CFBundleIdentifier key in the infoDictionary

bundlePath *XOG*

- `(NSString*)`**bundlePath**

Return the path to the bundle - an absolute path.

bundleVersion *XOG*

- `(unsigned)`**bundleVersion**

Returns the bundle version.

classNamed: *XOG*

- `(Class)`**classNamed:**`(NSString*)` *className*

Returns the class in the bundle with the given name. If no class of this name exists in the bundle, then Nil is returned.

executablePath *X¬OG*

- `(NSString*)`**executablePath**

Returns the path to the executable code in the bundle

infoDictionary *X¬OG*

- `(NSDictionary*)`**infoDictionary**

Returns the info property list associated with the bundle.

initWithPath: *XOG*

- `(id)`**initWithPath:**`(NSString*)` *path*

Init the bundle for reading resources from *path*. *path* must be an absolute *path* to a directory on disk. If *path* is `nil` or doesn't exist, initWithPath: returns `nil`. If a bundle for that *path* already existed, it is returned in place of the receiver (and the receiver is deallocated).

isLoaded *X¬OG*

- `(BOOL)`**isLoaded**

Returns `YES` if the receiver's code is loaded, otherwise, returns `NO`.

load *X¬OG*

- `(BOOL)`**load**

Loads any executable code contained in the bundle into the application. Load will be called implicitly if any information about the bundle classes is requested, such as -principalClass or -classNamed: .

localizations *X¬OG*

- `(NSArray*)`**localizations**

Returns all the localizations in the bundle.

localizedInfoDictionary *X¬OG*

- `(NSDictionary*)`**localizedInfoDictionary**

Returns a localized info property list based on the preferred localization or the most appropriate localization if the preferred one cannot be found.

localizedStringForKey:value:table: *XOG*

- (NSString*)**localizedStringForKey:**(NSString*) *key*
value:(NSString*) *value*
table:(NSString*) *tableName*

Returns the *value* for the *key* found in the strings file *tableName*, or Localizable.strings if *tableName* is `nil`.

pathForResource:ofType:

XOG

- (NSString*)**pathForResource:**(NSString*) *name*
ofType:(NSString*) *ext*

Returns an absolute path for a resource *name* with the extension *ext* in the receivers bundle path. See -pathForResource:ofType:inDirectory:.

pathForResource:ofType:inDirectory:

XOG

- (NSString*)**pathForResource:**(NSString*) *name*
ofType:(NSString*) *ext*
inDirectory:(NSString*) *bundlePath*

Returns an absolute path for a resource *name* with the extension *ext* in the specified *bundlePath*. Directories in the bundle are searched in the following order:

```
root path/Resources/bundlePath
root path/Resources/bundlePath/"language.lproj"
root path/bundlePath
root path/bundlePath/"language.lproj"
```

where lanuage.lproj can be any localized language directory inside the bundle. If *ext* is `nil` or empty, then the first file with *name* and any extension is returned.

pathForResource:ofType:inDirectory:forLocalization:

X¬OG

- (NSString*)**pathForResource:**(NSString*) *name*
ofType:(NSString*) *ext*
inDirectory:(NSString*) *bundlePath*
forLocalization:(NSString*) *localizationName*

Description forthcoming.

pathsForResourcesOfType:inDirectory:

XOG

- (NSArray*)**pathsForResourcesOfType:**(NSString*) *extension*
inDirectory:(NSString*) *bundlePath*

Returns an array of paths for all resources with the specified *extension* and residing in the *bundlePath* directory. If *extension* is `nil` or empty, all bundle resources are returned.

pathsForResourcesOfType:inDirectory:forLocalization: *X¬OG*

- (NSArray*)**pathsForResourcesOfType:**(NSString*) *extension*
inDirectory:(NSString*) *bundlePath*
forLocalization:(NSString*) *localizationName*

Description forthcoming.

preferredLocalizations *X¬OG*

- (NSArray*)**preferredLocalizations**

Returns the list of localizations that the bundle uses to search for information. This is based on the user's preferences.

principalClass *XOG*

- (Class)**principalClass**

Returns the principal class of the bundle. This is the class specified by the NSPrincipalClass key in the Info-gnustep property list contained in the bundle. If this key is not found, the class returned is arbitrary, although it is typically the first class compiled into the archive.

resourcePath *XOG*

- (NSString*)**resourcePath**

Returns the absolute path to the resources directory of the bundle.

setBundleVersion: *XOG*

- (void)**setBundleVersion:**(unsigned) *version*

Set the bundle *version*

NSCalendarDate *XOG*

Inherits From: NSDate : NSObject
Declared in: Foundation/NSCalendarDate.h

Description

An NSDate subclass which understands about timezones and provides methods for dealing with date and time information by calendar and with hours minutes and seconds.

Class Methods

calendarDate *XOG*

`+ (id)`**calendarDate**

Return an NSCalendarDate for the current date and time using the default timezone.

dateWithString:calendarFormat: *XOG*

`+ (id)`**dateWithString:** `(NSString*)` *description*
calendarFormat: `(NSString*)` *format*

Return an NSCalendarDate generated from the supplied *description* using the *format* specified for parsing that string.
Calls -initWithString:calendarFormat: to create the date.

dateWithString:calendarFormat:locale: *XOG*

`+ (id)`**dateWithString:** `(NSString*)` *description*
calendarFormat: `(NSString*)` *format*
locale: `(NSDictionary*)` *dictionary*

Return an NSCalendarDate generated from the supplied *description* using the *format* specified for parsing that string and interpreting it according to the *dictionary* specified.

Calls -initWithString:calendarFormat:locale: to create the date.

dateWithYear:month:day:hour:minute:second:timeZone: *XOG*

```
+ (id)dateWithYear: (int) year
             month: (unsigned int) month
               day: (unsigned int) day
              hour: (unsigned int) hour
            minute: (unsigned int) minute
            second: (unsigned int) second
          timeZone: (NSTimeZone*) aTimeZone
```

Creates and returns an NSCalendarDate from the specified values by calling -initWithYear:month:day:hour:minute:second:timeZone:

Instance Methods

addYear:month:day:hour:minute:second: *XOG*

```
- (NSCalendarDate*)addYear:(int) year
                     month:(int) month
                       day:(int) day
                      hour:(int) hour
                    minute:(int) minute
                    second:(int) second
```

This method exists solely for conformance to the OpenStep spec. Its use is deprecated... it simply calls -dateByAddingYears:months:days:hours:minutes:seconds:

calendarFormat *XOG*

```
- (NSString*)calendarFormat
```

Returns the format string associated with the receiver.
See -descriptionWithCalendarFormat:locale: for details.

dayOfCommonEra *XOG*

```
- (int)dayOfCommonEra
```

Return the day number (ie number of days since the start of) in the 'common' era of the receiving date. The era starts at 1 A.D.

dayOfMonth *XOG*

- (int)dayOfMonth

Return the month (1 to 31) of the receiving date.

dayOfWeek *XOG*

- (int)dayOfWeek

Return the day of the week (0 to 6) of the receiving date.

- 0 is sunday
- 1 is monday
- 2 is tuesday
- 3 is wednesday
- 4 is thursday
- 5 is friday
- 6 is saturday

dayOfYear *XOG*

- (int)dayOfYear

Return the day of the year (1 to 366) of the receiving date.

description *XOG*

- (NSString*)description

Calls -descriptionWithCalendarFormat:locale: passing the receviers calendar format and a `nil` locale.

descriptionWithCalendarFormat: *XOG*

- `(NSString*)`**descriptionWithCalendarFormat:**`(NSString*)` *format*

Returns a string representation of the receiver using the specified *format* string. Calls -descriptionWithCalendarFormat:locale: with a `nil` locale.

descriptionWithCalendarFormat:locale: *XOG*

- `(NSString*)`**descriptionWithCalendarFormat:**`(NSString*)` *format*
locale:`(NSDictionary*)` *locale*

Returns a string representation of the receiver using the specified *format* string and *locale* dictionary.
Format specifiers are -

- %a abbreviated weekday name according to *locale*
- %A full weekday name according to *locale*
- %b abbreviated month name according to *locale*
- %B full month name according to *locale*
- %d day of month as decimal number (leading zero)
- %e day of month as decimal number (leading space)
- %F milliseconds (000 to 999)
- %H hour as a decimal number using 24-hour clock
- %I hour as a decimal number using 12-hour clock
- %j day of year as a decimal number
- %m month as decimal number
- %M minute as decimal number
- %p 'am' or 'pm'
- %S second as decimal number
- %U week of the current year as decimal number (Sunday first day)

- %W week of the current year as decimal number (Monday first day)
- %w day of the week as decimal number (Sunday = 0)
- %y year as a decimal number without century
- %Y year as a decimal number with century
- %z time zone offset (HHMM)
- %Z time zone
- %% literal % character

descriptionWithLocale: *XOG*

- (NSString*)descriptionWithLocale:(NSDictionary*) *locale*

Returns a description of the receiver using its normal format but with the specified *locale* dictionary.
Calls -descriptionWithCalendarFormat:locale: to do this.

hourOfDay *XOG*

- (int)hourOfDay

Return the hour of the day (0 to 23) of the receiving date.

initWithString: *XOG*

- (id)initWithString:(NSString*) *description*

Initializes an NSCalendarDate using the specified *description* and the default calendar format and locale.
Calls -initWithString:calendarFormat:locale:

initWithString:calendarFormat: *XOG*

- (id)initWithString:(NSString*) *description*
calendarFormat:(NSString*) *format*

Initializes an NSCalendarDate using the specified *description* and *format* string interpreted in the default locale.
Calls -initWithString:calendarFormat:locale:

initWithString:calendarFormat:locale: *XOG*

```
- (id)initWithString:(NSString*) description
      calendarFormat:(NSString*) fmt
              locale:(NSDictionary*) locale
```

Initializes an NSCalendarDate using the specified *description* and format string interpreted in the given *locale*.
Format specifiers are -

- %% literal % character
- %a abbreviated weekday name according to *locale*
- %A full weekday name according to *locale*
- %b abbreviated month name according to *locale*
- %B full month name according to *locale*
- %c same as '%X %x'
- %d day of month as decimal number
- %e same as %d without leading zero (you get a leading space instead)
- %F milliseconds as a decimal number
- %H hour as a decimal number using 24-hour clock
- %I hour as a decimal number using 12-hour clock
- %j day of year as a decimal number
- %m month as decimal number
- %M minute as decimal number
- %p 'am' or 'pm'
- %S second as decimal number

- %U week of the current year as decimal number (Sunday first day)
- %W week of the current year as decimal number (Monday first day)
- %w day of the week as decimal number (Sunday = 0)
- %x date with date representation for *locale*
- %X time with time representation for *locale*
- %y year as a decimal number without century
- %Y year as a decimal number with century
- %z time zone offset in hours and minutes from GMT (HHMM)
- %Z time zone abbreviation

initWithYear:month:day:hour:minute:second:timeZone: *XOG*

```
- (id)initWithYear:(int) year
            month:(unsigned int) month
              day:(unsigned int) day
             hour:(unsigned int) hour
           minute:(unsigned int) minute
           second:(unsigned int) second
         timeZone:(NSTimeZone*) aTimeZone
```

Returns an NSCalendarDate instance with the given *year*, *month*, *day*, *hour*, *minute*, and *second*, using *aTimeZone*.

The *year* includes the century (ie you can't just say '02' when you mean '2002').

The *month* is in the range 1 to 12,

The *day* is in the range 1 to 31,

The *hour* is in the range 0 to 23,

The *minute* is in the range 0 to 59,

The *second* is in the range 0 to 59.

If *aTimeZone* is `nil`, the [NSTimeZone +localTimeZone] value is used.

GNUstep checks the validity of the method arguments, and unless the base library was built with 'warn=no' it generates a warning for bad values. It tries to use those bad values to generate a date anyway though, rather than failing (this also appears to be the behavior of MacOS-X). The algorithm GNUstep uses to create the date is this ...

- Convert the broken out date values into a time interval since the reference date, as if those values represent a GMT date/time.

- Ask the time zone for the offset from GMT at the resulting date, and apply that offset to the time interval... so get the value for the specified timezone.
- Ask the time zone for the offset from GMT at the new date... in case the new date is in a different daylight savings time band from the original date. If this offset differs from the previous one, apply the difference so that the result is corrected for daylight savings. This is the final result used.
- After establishing the time interval we will use and completing initialisation, we ask the time zone for the offset from GMT again. If it is not the same as the last time, then the time specified by the broken out date does not really exist... since it's in the period lost by the transition to daylight savings. The resulting date is therefore not the date that was actually asked for, but is the best approximation we can do. If the base library was not built with 'warn=no' then a warning message is logged for this condition.

minuteOfHour *XOG*

- `(int)`**minuteOfHour**

Return the minute of the hour (0 to 59) of the receiving date.

monthOfYear *XOG*

- `(int)`**monthOfYear**

Return the month of the year (1 to 12) of the receiving date.

secondOfMinute *XOG*

- `(int)`**secondOfMinute**

Return the second of the minute (0 to 59) of the receiving date.

setCalendarFormat: *XOG*

- `(void)`**setCalendarFormat:**`(NSString*)` *format*

Sets the *format* string associated with the receiver.
See -descriptionWithCalendarFormat:locale: for details.

setTimeZone: XOG

- `(void)`**setTimeZone:**`(NSTimeZone*)` *aTimeZone*

Sets the time zone associated with the receiver.

timeZone $X\neg OG$

- `(NSTimeZone*)`**timeZone**

Returns the time zone associated with the receiver.

timeZoneDetail $O\neg XG$

- `(NSTimeZoneDetail*)`**timeZoneDetail**

Returns the time zone detail associated with the receiver.

yearOfCommonEra XOG

- `(int)`**yearOfCommonEra**

Return the year of the 'common' era of the receiving date. The era starts at 1 A.D.

NSCharacterSet *XOG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying NSMutableCopying
Declared in:	Foundation/NSCharacterSet.h

Description

Description forthcoming.

Class Methods

alphanumericCharacterSet *XOG*

`+ (NSCharacterSet*)` **alphanumericCharacterSet**

Description forthcoming.

characterSetWithBitmapRepresentation: *XOG*

`+ (NSCharacterSet*)` **characterSetWithBitmapRepresentation:** `(NSData*)` *data*

Returns a character set containing characters as encoded in the *data* object.

characterSetWithCharactersInString: *XOG*

`+ (NSCharacterSet*)` **characterSetWithCharactersInString:** `(NSString*)` *aString*

Description forthcoming.

characterSetWithContentsOfFile: *O¬XG*

`+ (NSCharacterSet*)`**characterSetWithContentsOfFile:** `(NSString*)` *aFile*

Description forthcoming.

characterSetWithRange: *XOG*

`+ (NSCharacterSet*)`**characterSetWithRange:** `(NSRange)` *aRange*

Description forthcoming.

controlCharacterSet *XOG*

`+ (NSCharacterSet*)`**controlCharacterSet**

Description forthcoming.

decimalDigitCharacterSet *XOG*

`+ (NSCharacterSet*)`**decimalDigitCharacterSet**

Returns a character set containing characters that represent the decimal digits 0 through 9.

decomposableCharacterSet *XOG*

`+ (NSCharacterSet*)`**decomposableCharacterSet**

Returns a character set containing individual charactars that can be represented also by a composed character sequence.

illegalCharacterSet *XOG*

`+ (NSCharacterSet*)`**illegalCharacterSet**

Returns a character set containing unassigned (illegal) character values.

letterCharacterSet *XOG*

+ (NSCharacterSet*)**letterCharacterSet**

Description forthcoming.

lowercaseLetterCharacterSet *XOG*

+ (NSCharacterSet*)**lowercaseLetterCharacterSet**

Returns a character set that contains the lowercase characters. This set does not include caseless characters, only those that have corresponding characters in uppercase and/or titlecase.

nonBaseCharacterSet *XOG*

+ (NSCharacterSet*)**nonBaseCharacterSet**

Description forthcoming.

punctuationCharacterSet *XOG*

+ (NSCharacterSet*)**punctuationCharacterSet**

Description forthcoming.

symbolAndOperatorCharacterSet *XOG*

+ (NSCharacterSet*)**symbolAndOperatorCharacterSet**

Description forthcoming.

uppercaseLetterCharacterSet *XOG*

+ (NSCharacterSet*)**uppercaseLetterCharacterSet**

Returns a character set that contains the uppercase characters. This set does not include caseless characters, only those that have corresponding characters in lowercase and/or titlecase.

whitespaceAndNewlineCharacterSet *XOG*

`+ (NSCharacterSet*)`**whitespaceAndNewlineCharacterSet**

Returns a character set that contains the whitespace characters, plus the newline characters, values 0x000A and 0x000D.

whitespaceCharacterSet *XOG*

`+ (NSCharacterSet*)`**whitespaceCharacterSet**

Returns a character set that contains the whitespace characters.

Instance Methods

bitmapRepresentation *XOG*

`- (NSData*)`**bitmapRepresentation**

Returns a bitmap representation of the receiver's character set suitable for archiving or writing to a file, in an NSData object.

characterIsMember: *XOG*

`- (BOOL)`**characterIsMember:**`(unichar)` *aCharacter*

Returns `YES` if the receiver contains *aCharacter*, `NO` if it does not.

invertedSet *XOG*

`- (NSCharacterSet*)`**invertedSet**

Returns a character set containing only characters that the receiver does not contain.

NSClassDescription

X¬OG

Inherits From: NSObject
Declared in: Foundation/NSClassDescription.h

Description

Description forthcoming.

Class Methods

classDescriptionForClass:

X¬OG

`+ (NSClassDescription*)`**classDescriptionForClass:** `(Class)` *aClass*

Returns the class descriptuion for *aClass*. If there is no such description available, sends an NSClassDescriptionNeededForClassNotification (with *aClass* as its object) so that objects providing class descriptions can register one, and tries again to find one.

Returns `nil` if there is no description found.

Handles locking to ensure thread safety and ensures that the returned object will not be destroyed by other threads.

invalidateClassDescriptionCache

X¬OG

`+ (void)`**invalidateClassDescriptionCache**

Invalidates the cache of class descriptions so the new descriptions will be fetched as required and begin to refil the cache. You need this only if you suspect that a class description should have changed.

registerClassDescription:forClass:

X¬OG

`+ (void)`**registerClassDescription:** `(NSClassDescription*)` *aDescription*
forClass: `(Class)` *aClass*

Registers *aDescription* for *aClass*... placing it in the cache and replacing any previous version.

Instance Methods

attributeKeys

$X\neg OG$

- `(NSArray*)`**attributeKeys**

Returns the attribute keys - default implementation returns `nil`.

inverseForRelationshipKey:

$X\neg OG$

- `(NSString*)`**inverseForRelationshipKey:**`(NSString*)` *aKey*

Returns the inverse relationship keys - default implementation returns `nil`.

toManyRelationshipKeys

$X\neg OG$

- `(NSArray*)`**toManyRelationshipKeys**

Returns the to many relationship keys - default implementation returns `nil`.

toOneRelationshipKeys

$X\neg OG$

- `(NSArray*)`**toOneRelationshipKeys**

Returns the to one relationship keys - default implementation returns `nil`.

NSCoder *XOG*

Inherits From: NSObject
Declared in: Foundation/NSCoder.h

Description

Description forthcoming.

Instance Methods

decodeArrayOfObjCType:count:at: *XOG*

- `(void)`**decodeArrayOfObjCType:**`(const char*)` *type*
count:`(unsigned)` *count*
at:`(void*)` *address*

Description forthcoming.

decodeBytesWithReturnedLength: *XOG*

- `(void*)`**decodeBytesWithReturnedLength:**`(unsigned*)` *l*

Description forthcoming.

decodeDataObject *XOG*

- `(NSData*)`**decodeDataObject**

Description forthcoming.

decodeObject *XOG*

- `(id)`**decodeObject**

Description forthcoming.

decodePoint *XOG*

- `(NSPoint)`**decodePoint**

Description forthcoming.

decodePropertyList *XOG*

- `(id)`**decodePropertyList**

Description forthcoming.

decodeRect *XOG*

- `(NSRect)`**decodeRect**

Description forthcoming.

decodeSize *XOG*

- `(NSSize)`**decodeSize**

Description forthcoming.

decodeValueOfObjCType:at: *XOG*

- `(void)`**decodeValueOfObjCType:**`(const char*)` *type*
at:`(void*)` *address*

Description forthcoming.

decodeValuesOfObjCTypes: *XOG*

- `(void)`**decodeValuesOfObjCTypes:**`(const char*)` *types*

Description forthcoming.

encodeArrayOfObjCType:count:at: *XOG*

- (void)**encodeArrayOfObjCType:**(const char*) *type*
count:(unsigned) *count*
at:(const void*) *array*

Description forthcoming.

encodeBycopyObject: *XOG*

- (void)**encodeBycopyObject:**(id) *anObject*

Description forthcoming.

encodeByrefObject: *XOG*

- (void)**encodeByrefObject:**(id) *anObject*

Description forthcoming.

encodeBytes:length: *XOG*

- (void)**encodeBytes:**(void*) *d*
length:(unsigned) *l*

Description forthcoming.

encodeConditionalObject: *XOG*

- (void)**encodeConditionalObject:**(id) *anObject*

Description forthcoming.

encodeDataObject: *XOG*

- (void)**encodeDataObject:**(NSData*) *data*

Description forthcoming.

encodeObject: *XOG*

- `(void)`**encodeObject:**`(id)` *anObject*

Description forthcoming.

encodePoint: *XOG*

- `(void)`**encodePoint:**`(NSPoint)` *point*

Description forthcoming.

encodePropertyList: *XOG*

- `(void)`**encodePropertyList:**`(id)` *plist*

Description forthcoming.

encodeRect: *XOG*

- `(void)`**encodeRect:**`(NSRect)` *rect*

Description forthcoming.

encodeRootObject: *XOG*

- `(void)`**encodeRootObject:**`(id)` *rootObject*

Description forthcoming.

encodeSize: *XOG*

- `(void)`**encodeSize:**`(NSSize)` *size*

Description forthcoming.

encodeValueOfObjCType:at: *XOG*

- `(void)`**encodeValueOfObjCType:**`(const char*)` *type*
at:`(const void*)` *address*

Description forthcoming.

encodeValuesOfObjCTypes: *XOG*

- `(void)`**encodeValuesOfObjCTypes:**`(const char*)` *types*

Description forthcoming.

objectZone *XOG*

- `(NSZone*)`**objectZone**

Description forthcoming.

setObjectZone: *XOG*

- `(void)`**setObjectZone:**`(NSZone*)` *zone*

Description forthcoming.

systemVersion *XOG*

- `(unsigned int)`**systemVersion**

Description forthcoming.

versionForClassName: *XOG*

- `(unsigned int)`**versionForClassName:**`(NSString*)` *className*

Description forthcoming.

NSConditionLock *XOG*

Inherits From:	NSObject
Conforms To:	NSLocking GCFinalization
Declared in:	Foundation/NSLock.h

Description

Description forthcoming.

Instance Methods

condition *XOG*

- `(int)`**condition**

Description forthcoming.

initWithCondition: *XOG*

- `(id)`**initWithCondition:**`(int)` *value*

Description forthcoming.

lock *XOG*

- `(void)`**lock**

Description forthcoming.

lockBeforeDate: *XOG*

- `(BOOL)`**lockBeforeDate:**`(NSDate*)` *limit*

Description forthcoming.

lockWhenCondition: *XOG*

- `(void)`**lockWhenCondition:**`(int)` *value*

Description forthcoming.

lockWhenCondition:beforeDate: *XOG*

- `(BOOL)`**lockWhenCondition:**`(int)` *condition_to_meet*
beforeDate:`(NSDate*)` *limitDate*

Description forthcoming.

tryLock *XOG*

- `(BOOL)`**tryLock**

Description forthcoming.

tryLockWhenCondition: *XOG*

- `(BOOL)`**tryLockWhenCondition:**`(int)` *value*

Description forthcoming.

unlock *XOG*

- `(void)`**unlock**

Description forthcoming.

unlockWithCondition: *XOG*

- `(void)`**unlockWithCondition:**`(int)` *value*

Description forthcoming.

NSConnection *XOG*

Inherits From: NSObject
Declared in: Foundation/NSConnection.h

Description

NSConnection objects are used to manage communications between objects in different processes, in different machines, or in different threads.

Class Methods

allConnections *XOG*

`+ (NSArray*)`**allConnections**

Returns an array containing all the NSConnection objects known to the system. These connections will be valid at the time that the array was created, but may be invalidated by other threads before you get to examine the array.

connectionWithReceivePort:sendPort: *XOG*

`+ (NSConnection*)`**connectionWithReceivePort:** `(NSPort*)` *r*
sendPort: `(NSPort*)` *s*

Returns a connection initialised using -initWithReceivePort:sendPort:

connectionWithRegisteredName:host: *XOG*

`+ (NSConnection*)`**connectionWithRegisteredName:** `(NSString*)` *n*
host: `(NSString*)` *h*

Returns an NSConnection object whose send port is that of the NSConnection registered under the name *n* on the host *h*

This method calls +connectionWithRegisteredName:host:usingNameServer: using the default system name server.

connectionWithRegisteredName:host:usingNameServer: *XOG*

```
+ (NSConnection*)connectionWithRegisteredName: (NSString*) n
                                        host: (NSString*) h
                               usingNameServer: (NSPortNameServer*) s
```

Returns an NSConnection object whose send port is that of the NSConnection registered under *name* on *host*.

The nameserver *server* is used to look up the send port to be used for the connection.

If *host* is `nil` or an empty string, the host is taken to be the local machine. If it is an asterisk ('*') then the nameserver checks all hosts on the local subnet (unless the nameserver is one that only manages local ports). In the GNUstep implementation, the local host is searched before any other hosts.

If no NSConnection can be found for *name* and *host* host, the method returns `nil`.

The returned object has the default NSConnection of the current thread as its parent (it has the same receive port as the default connection).

currentConversation *XOG*

```
+ (id)currentConversation
```

Not used in GNUstep

defaultConnection *XOG*

```
+ (NSConnection*)defaultConnection
```

Returns the default connection for a thread.

Creates a new instance if necessary.

The default connection has a single NSPort object used for both sending and receiving - this it can't be used to connect to a remote process, but can be used to vend objects.

Possible problem - if the connection is invalidated, it won't be cleaned up until this thread calls this method again. The connection and it's ports could hang around for a very long time.

rootProxyForConnectionWithRegisteredName:host: *XOG*

+ (NSDistantObject*)**rootProxyForConnectionWithRegisteredName:** (NSString*) *n*
host: (NSString*) *h*

This method calls +rootProxyForConnectionWithRegisteredName:host:usingNameServer: to return a proxy for a root object on the remote connection with the send port registered under name *n* on host *h*.

rootProxyForConnectionWithRegisteredName:host:usingNameServer: *XOG*

+ (NSDistantObject*)**rootProxyForConnectionWithRegisteredName:** (NSString*) *n*
host: (NSString*) *h*
usingNameServer: (NSPortNameSe

This method calls +connectionWithRegisteredName:host:usingNameServer: to get a connection, then sends it a -rootProxy message to get a proxy for the root object being vended by the remote connection. Returns the proxy or `nil` if it couldn't find a connection or if the root object for the connection has not been set.

Instance Methods

addRequestMode: *XOG*

- (void)**addRequestMode:**(NSString*) *mode*

Adds *mode* to the run loop modes that the NSConnection will listen to for incoming messages.

addRunLoop: *XOG*

- (void)**addRunLoop:**(NSRunLoop*) *loop*

Adds *loop* to the set of run loops that the NSConnection will listen to for incoming messages.

delegate *XOG*

- (id)**delegate**

Returns the delegate of the NSConnection.

enableMultipleThreads *XOG*

- (void)enableMultipleThreads

Sets the NSConnection configuration so that multiple threads may use the connection to send requests to the remote connection.

This option is inherited by child connections.

NB. A connection with multiple threads enabled will run slower than a normal connection.

independentConversationQueueing *XOG*

- (BOOL)independentConversationQueueing

Returns YES if the NSConnection is configured to handle remote messages atomically, NO otherwise.

This option is inherited by child connections.

initWithReceivePort:sendPort: *XOG*

- (id)initWithReceivePort:(NSPort*) *r*
sendPort:(NSPort*) *s*

Initialises an NSConnection with the receive port *r* and the send port *s*.

Behavior varies with the port values as follows -

r **is** nil The NSConnection is released and the method returns nil.

s **is** nil The NSConnection uses *r* as the send port as well as the receive port.

s **is the same as** *r* The NSConnection is usable only for vending objects.

A connection with the same ports exists The new connection is released and the old connection is retained and returned.

A connection with the same ports (swapped) exists The new connection is initialised as normal, and will communicate with the old connection.

If a connection exists whose send and receive ports are both the same as the new connections receive port, that existing connection is deemed to be the parent of the new connection. The new connection inherits configuration information from the parent, and the delegate of the parent has a chance to adjust ythe configuration of the new connection or veto its creation.

NSConnectionDidInitializeNotification is posted once a new connection is initialised.

invalidate *XOG*

`- (void)` **invalidate**

Marks the receiving NSConnection as invalid.
Removes the NSConnections ports from any run loops.
Posts an NSConnectionDidDieNotification.
Invalidates all remote objects and local proxies.

isValid *XOG*

`- (BOOL)` **isValid**

Returns `YES` if the connection is valid, `NO` otherwise. A connection is valid until it has been sent an -invalidate message.

localObjects *XOG*

`- (NSArray*)` **localObjects**

Returns an array of all the local proxies to objects that are retained by the remote connection.

multipleThreadsEnabled *XOG*

`- (BOOL)` **multipleThreadsEnabled**

Returns `YES` if the connection permits multiple threads to use it to send requests, `NO` otherwise.
See the -enableMultipleThreads method.

receivePort *XOG*

`- (NSPort*)` **receivePort**

Returns the NSPort object on which incoming messages are recieved.

registerName: *XOG*

- (BOOL)**registerName:**(NSString*) *name*

Simply invokes -registerName:withNameServer: passing it the default system nameserver.

registerName:withNameServer: *XOG*

- (BOOL)**registerName:**(NSString*) *name*
  withNameServer:(NSPortNameServer*) *svr*

Registers the recieve port of the NSConnection as *name* and unregisters the previous value (if any).
Returns YES on success, NO on failure.
On failure, the connection remains registered under the previous *name*.
Supply nil as *name* to unregister the NSConnection.

remoteObjects *XOG*

- (NSArray*)**remoteObjects**

Returns an array of proxies to all the remote objects known to the NSConnection.

removeRequestMode: *XOG*

- (void)**removeRequestMode:**(NSString*) *mode*

Removes *mode* from the run loop modes used to recieve incoming messages.

removeRunLoop: *XOG*

- (void)**removeRunLoop:**(NSRunLoop*) *loop*

Removes *loop* from the run loops used to recieve incoming messages.

replyTimeout *XOG*

- (NSTimeInterval)**replyTimeout**

Returns the timeout interval used when waiting for a reply to a request sent on the NSConnection. This value is inherited from the parent connection or may be set using the -setReplyTimeout: method.
Under MacOS-X the default value is documented as the maximum delay (effectively infinite), but under GNUstep it is set to a more useful 300 seconds.

requestModes *XOG*

- (NSArray*)**requestModes**

Returns an array of all the run loop modes that the NSConnection uses when waiting for an incoming request.

requestTimeout *XOG*

- (NSTimeInterval)**requestTimeout**

Returns the timeout interval used when trying to send a request on the NSConnection. This value is inherited from the parent connection or may be set using the -setRequestTimeout: method.
Under MacOS-X the default value is documented as the maximum delay (effectively infinite), but under GNUstep it is set to a more useful 300 seconds.

rootObject *XOG*

- (id)**rootObject**

Returns the object that is made available by this connection or by its parent (the object is associated with the receive port).
Returns `nil` if no root object has been set.

rootProxy *XOG*

- (NSDistantObject*)**rootProxy**

Returns the proxy for the root object of the remote NSConnection.

runInNewThread *XOG*

- (void)**runInNewThread**

Removes the NSConnection from the current threads default run loop, then creates a new thread and runs the NSConnection in it.

sendPort *XOG*

- (NSPort*)**sendPort**

Returns the port on which the NSConnection sends messages.

setDelegate: *XOG*

- (void)**setDelegate:**(id) *anObj*

Sets the NSConnection's delegate (without retaining it).
The delegate is able to control some of the NSConnection's behavior by implementing methods in an informal protocol.

setIndependentConversationQueueing: *XOG*

- (void)**setIndependentConversationQueueing:**(BOOL) *flag*

Sets whether or not the NSConnection should handle requests arriving from the remote NSConnection atomically.
By default, this is set to NO... if set to YES then any messages arriving while one message is being dealt with, will be queued.
NB. careful - use of this option can cause deadlocks.

setReplyTimeout: *XOG*

- (void)**setReplyTimeout:**(NSTimeInterval) *to*

Sets the time interval that the NSConnection will wait for a reply for one of its requests before raising an NSPortTimeoutException.
NB. In GNUstep you may also get such an exception if the connection becomes invalidated while waiting for a reply *to* a request.

setRequestMode: *XOG*

- (void)**setRequestMode:**(NSString*) *mode*

Sets the runloop *mode* in which requests will be sent to the remote end of the connection.

setRequestTimeout: *XOG*

- (void)**setRequestTimeout:**(NSTimeInterval) *to*

Sets the time interval that the NSConnection will wait *to* send one of its requests before raising an NSPortTimeoutException.

setRootObject: *XOG*

- (void)**setRootObject:**(id) *anObj*

Sets the root object that is vended by the connection.

statistics *XOG*

- (NSDictionary*)**statistics**

Returns an object containing various statistics for the NSConnection.
On GNUstep the dictionary contains -

NSConnectionRepliesReceived The number of messages replied to by the remote NSConnection

NSConnectionRepliesSent The number of replies sent to the remote NSConnection

NSConnectionRequestsReceived The number of messages recieved from the remote NSConnection

NSConnectionRequestsSent The number of messages sent to the remote NSConnection

NSConnectionLocalCount The number of local objects currently vended

NSConnectionProxyCount The number of remote objects currently in use

NSCountedSet *XOG*

Inherits From: NSMutableSet : NSSet : NSObject
Declared in: Foundation/NSSet.h

Description

The NSCountedSet class is used to maintain a set of objects where the number of times each object has been added (wiithout a corresponding removal) is kept track of.
In GNUstep, extra methods are provided to make use of a counted set for *uniquing* objects easier.

Instance Methods

countForObject: *XOG*

`- (unsigned int)countForObject:(id)` *anObject*

Returns the number of times that an object that is equal to the specified object (as determined byt the [-isEqual:] method) has been added to the set and not removed from it.

NSData *XOG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying NSMutableCopying
Declared in:	Foundation/NSData.h

Description

Description forthcoming.

Class Methods

data *XOG*

`+ (id)data`

Returns an empty data object.

dataWithBytes:length: *XOG*

`+ (id)dataWithBytes:` `(const void*)` *bytes*
`length:` `(unsigned int)` *length*

Returns an autoreleased data object containing data copied from *bytes* and with the specified *length*. Invokes -initWithBytes:length:

dataWithBytesNoCopy:length: *XOG*

`+ (id)dataWithBytesNoCopy:` `(void*)` *bytes*
`length:` `(unsigned int)` *length*

Returns an autoreleased data object encapsulating the data at *bytes* and with the specified *length*. Invokes -initWithBytesNoCopy:length:freeWhenDone: with YES

dataWithBytesNoCopy:length:freeWhenDone: *X¬OG*

`+ (id)`**dataWithBytesNoCopy:** `(void*)` *aBuffer*
length: `(unsigned int)` *bufferSize*
freeWhenDone: `(BOOL)` *shouldFree*

Returns an autoreleased data object encapsulating the data at bytes and with the specified length. Invokes -initWithBytesNoCopy:length:freeWhenDone:

dataWithContentsOfFile: *XOG*

`+ (id)`**dataWithContentsOfFile:** `(NSString*)` *path*

Returns a data object encapsulating the contents of the specified file. Invokes -initWithContentsOfFile:

dataWithContentsOfMappedFile: *XOG*

`+ (id)`**dataWithContentsOfMappedFile:** `(NSString*)` *path*

Returns a data object encapsulating the contents of the specified file mapped directly into memory. Invokes -initWithContentsOfMappedFile:

dataWithContentsOfURL: *X¬OG*

`+ (id)`**dataWithContentsOfURL:** `(NSURL*)` *url*

Retrieves the information at the specified *url* and returns an NSData instance encapsulating it.

dataWithData: *XOG*

`+ (id)`**dataWithData:** `(NSData*)` *data*

Returns an autoreleased instance initialised by copying the contents of *data*.

Instance Methods

bytes *XOG*

- (const void*)**bytes**

Returns a pointer to the data encapsulated by the receiver.

description *XOG*

- (NSString*)**description**

Description forthcoming.

deserializeAlignedBytesLengthAtCursor: *XOG*

- (unsigned int)**deserializeAlignedBytesLengthAtCursor:**(unsigned int*) *cursor*

Description forthcoming.

deserializeBytes:length:atCursor: *XOG*

- (void)**deserializeBytes:**(void*) *buffer*
length:(unsigned int) *bytes*
atCursor:(unsigned int*) *cursor*

Description forthcoming.

deserializeDataAt:ofObjCType:atCursor:context: *XOG*

- (void)**deserializeDataAt:**(void*) *data*
ofObjCType:(const char*) *type*
atCursor:(unsigned int*) *cursor*
context:(id<NSObjCTypeSerializationCallBack>) *callback*

Description forthcoming.

deserializeIntAtCursor: *XOG*

- `(int)`**deserializeIntAtCursor:**`(unsigned int*)` *cursor*

Description forthcoming.

deserializeIntAtIndex: *XOG*

- `(int)`**deserializeIntAtIndex:**`(unsigned int)` *index*

Description forthcoming.

deserializeInts:count:atCursor: *XOG*

- `(void)`**deserializeInts:**`(int*)` *intBuffer*
count:`(unsigned int)` *numInts*
atCursor:`(unsigned int*)` *cursor*

Description forthcoming.

deserializeInts:count:atIndex: *XOG*

- `(void)`**deserializeInts:**`(int*)` *intBuffer*
count:`(unsigned int)` *numInts*
atIndex:`(unsigned int)` *index*

Description forthcoming.

getBytes: *XOG*

- `(void)`**getBytes:**`(void*)` *buffer*

Copies the contents of the memory encapsulated by the receiver into the specified *buffer*. The *buffer* must be large enough to contain -length bytes of data... if it isn't then a crash is likely to occur.

Invokes -getBytes:range: with the range set to the whole of the receiver.

getBytes:length: *XOG*

- (void)**getBytes:**(void*) *buffer*
length:(unsigned int) *length*

Copies *length* bytes of data from the memory encapsulated by the receiver into the specified *buffer*. The *buffer* must be large enough to contain *length* bytes of data... if it isn't then a crash is likely to occur.

Invokes -getBytes:range: with the range set to iNSMakeRange(0, *length*)

getBytes:range:

XOG

- (void)**getBytes:**(void*) *buffer*
range:(NSRange) *aRange*

Copies data from the memory encapsulated by the receiver (in the range specified by *aRange*) into the specified *buffer*.

The *buffer* must be large enough to contain the data... if it isn't then a crash is likely to occur.

If *aRange* specifies a range which does not entirely lie within the receiver, an exception is raised.

initWithBytes:length:

XOG

- (id)**initWithBytes:**(const void*) *aBuffer*
length:(unsigned int) *bufferSize*

Makes a copy of *bufferSize* bytes of data at *aBuffer*, and passes it to -initWithBytesNoCopy:le with a YES argument in order to initialise the receiver. Returns the result.

initWithBytesNoCopy:length:

XOG

- (id)**initWithBytesNoCopy:**(void*) *aBuffer*
length:(unsigned int) *bufferSize*

Invokes -initWithBytesNoCopy:length:freeWhenDone: with the last argument set to YES. Returns the resulting initialised data object (which may not be the receiver).

IO initWithBytesNoCopy:length:freeWhenDone:

X¬OG

- (id)**initWithBytesNoCopy:**(void*) *aBuffer*
length:(unsigned int) *bufferSize*
freeWhenDone:(BOOL) *shouldFree*

Initialises the receiver.
The value of *aBuffer* is a pointer to something to be stored.
The value of *bufferSize* is the number of bytes to use.
The value fo *shouldFree* specifies whether the receiver should attempt to free the memory pointer to by *aBuffer* when the receiver is deallocated ... ie. it says whether the receiver *owns* the memory. Supplying the wrong value here will lead to memory leaks or crashes.

initWithContentsOfFile: *XOG*

- `(id)`**initWithContentsOfFile:**`(NSString*)` *path*

Initialises the receiver with the contents of the specified file.
Returns the resulting object.
Returns `nil` if the file does not exist or can not be read for some reason.

initWithContentsOfMappedFile: *XOG*

- `(id)`**initWithContentsOfMappedFile:**`(NSString*)` *path*

Description forthcoming.

initWithContentsOfURL: *X¬OG*

- `(id)`**initWithContentsOfURL:**`(NSURL*)` *url*

Description forthcoming.

initWithData: *XOG*

- `(id)`**initWithData:**`(NSData*)` *data*

Description forthcoming.

isEqualToData: *XOG*

- `(BOOL)`**isEqualToData:**`(NSData*)` *other*

Returns a boolean value indicating if the receiver and *other* contain identical data (using a byte by byte comparison). Assumes that the *other* object is an NSData instance... may raise an exception if it isn't.

length

XOG

`- (unsigned int)` **length**

Returns the number of bytes of data encapsulated by the receiver.

subdataWithRange:

XOG

`- (NSData*)` **subdataWithRange:**`(NSRange)` *aRange*

Returns an NSData instance encapsulating the memory from the reciever specified by the range *aRange*.
If *aRange* specifies a range which does not entirely lie within the receiver, an exception is raised.

writeToFile:atomically:

XOG

`- (BOOL)` **writeToFile:**`(NSString*)` *path*
atomically:`(BOOL)` *useAuxiliaryFile*

Writes a copy of the data encapsulated by the receiver to a file at *path*. If the *useAuxiliaryFile* flag is `YES`, this writes to a temporary file and then renames that to the file at *path*, thus ensuring that *path* exists and does not contain partially written data at any point.
On success returns `YES`, on failure returns `NO`.

writeToURL:atomically:

X¬OG

`- (BOOL)` **writeToURL:**`(NSURL*)` *anURL*
atomically:`(BOOL)` *flag*

Writes a copy of the contents of the receiver to the specified URL.

NSDate *XOG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying
Declared in:	Foundation/NSDate.h

Description

Description forthcoming.

Class Methods

date *XOG*

`+ (id)date`

Description forthcoming.

dateWithNaturalLanguageString: *X¬OG*

`+ (id)dateWithNaturalLanguageString: (NSString*)` *string*

Description forthcoming.

dateWithNaturalLanguageString:locale: *X¬OG*

`+ (id)dateWithNaturalLanguageString: (NSString*)` *string*
`locale: (NSDictionary*)` *locale*

Description forthcoming.

dateWithString: *XOG*

`+ (id)dateWithString: (NSString*)` *description*

Description forthcoming.

dateWithTimeIntervalSince1970: *XOG*

`+ (id)`**dateWithTimeIntervalSince1970:** `(NSTimeInterval)` *seconds*
Description forthcoming.

dateWithTimeIntervalSinceNow: *XOG*

`+ (id)`**dateWithTimeIntervalSinceNow:** `(NSTimeInterval)` *seconds*
Description forthcoming.

dateWithTimeIntervalSinceReferenceDate: *XOG*

`+ (id)`**dateWithTimeIntervalSinceReferenceDate:** `(NSTimeInterval)` *seconds*

Description forthcoming.

distantFuture *XOG*

`+ (id)`**distantFuture**
Description forthcoming.

distantPast *XOG*

`+ (id)`**distantPast**
Description forthcoming.

timeIntervalSinceReferenceDate *XOG*

`+ (NSTimeInterval)`**timeIntervalSinceReferenceDate**
Description forthcoming.

Instance Methods

addTimeInterval: *XOG*

- `(id)`**addTimeInterval:**`(NSTimeInterval)` *seconds*

Description forthcoming.

compare: *XOG*

- `(NSComparisonResult)`**compare:**`(NSDate*)` *otherDate*

Description forthcoming.

dateWithCalendarFormat:timeZone: *XOG*

- `(NSCalendarDate*)`**dateWithCalendarFormat:**`(NSString*)` *formatString*
timeZone:`(NSTimeZone*)` *timeZone*

Description forthcoming.

description *XOG*

- `(NSString*)`**description**

Description forthcoming.

descriptionWithCalendarFormat:timeZone:locale: *XOG*

- `(NSString*)`**descriptionWithCalendarFormat:**`(NSString*)` *format*
timeZone:`(NSTimeZone*)` *aTimeZone*
locale:`(NSDictionary*)` *l*

Description forthcoming.

descriptionWithLocale: *XOG*

- `(NSString*)`**descriptionWithLocale:**`(NSDictionary*)` *locale*

Description forthcoming.

earlierDate: *XOG*

- `(NSDate*)`**earlierDate:**`(NSDate*)` *otherDate*

Description forthcoming.

initWithString: *XOG*

- `(id)`**initWithString:**`(NSString*)` *description*

Description forthcoming.

initWithTimeInterval:sinceDate: *XOG*

- `(id)`**initWithTimeInterval:**`(NSTimeInterval)` *secsToBeAdded*
sinceDate:`(NSDate*)` *anotherDate*

Description forthcoming.

initWithTimeIntervalSince1970: *X¬OG*

- `(id)`**initWithTimeIntervalSince1970:**`(NSTimeInterval)` *seconds*

Description forthcoming.

initWithTimeIntervalSinceNow: *XOG*

- `(id)`**initWithTimeIntervalSinceNow:**`(NSTimeInterval)` *secsToBeAdded*

Description forthcoming.

initWithTimeIntervalSinceReferenceDate: *XOG*

- `(id)`**initWithTimeIntervalSinceReferenceDate:**`(NSTimeInterval)` *secs*

Description forthcoming.

isEqualToDate: *XOG*

- `(BOOL)`**isEqualToDate:**`(NSDate*)` *other*

Description forthcoming.

laterDate: *XOG*

- `(NSDate*)`**laterDate:**`(NSDate*)` *otherDate*

Description forthcoming.

timeIntervalSince1970 *XOG*

- `(NSTimeInterval)`**timeIntervalSince1970**

Description forthcoming.

timeIntervalSinceDate: *XOG*

- `(NSTimeInterval)`**timeIntervalSinceDate:**`(NSDate*)` *otherDate*

Description forthcoming.

timeIntervalSinceNow *XOG*

- `(NSTimeInterval)`**timeIntervalSinceNow**

Description forthcoming.

timeIntervalSinceReferenceDate *XOG*

- `(NSTimeInterval)`**timeIntervalSinceReferenceDate**

Description forthcoming.

NSDateFormatter

X¬OG

Inherits From:	NSFormatter : NSObject
Conforms To:	NSCoding NSCopying
Declared in:	Foundation/NSDateFormatter.h

Description

Description forthcoming.

Instance Methods

allowsNaturalLanguage

X¬OG

- `(BOOL)` **allowsNaturalLanguage**

Description forthcoming.

dateFormat

X¬OG

- `(NSString*)` **dateFormat**

Description forthcoming.

initWithDateFormat:allowNaturalLanguage:

X¬OG

- `(id)` **initWithDateFormat:**`(NSString*)` *format*
allowNaturalLanguage:`(BOOL)` *flag*

Description forthcoming.

NSDecimalNumber *X¬OG*

Inherits From:	NSNumber : NSValue : NSObject
Conforms To:	NSDecimalNumberBehaviors
Declared in:	Foundation/NSDecimalNumber.h

Description

Description forthcoming.

Class Methods

decimalNumberWithDecimal: *X¬OG*

`+ (NSDecimalNumber*)`**decimalNumberWithDecimal:** `(NSDecimal)` *decimal*

Description forthcoming.

decimalNumberWithMantissa:exponent:isNegative: *X¬OG*

`+ (NSDecimalNumber*)`**decimalNumberWithMantissa:** `(unsigned long long)` *mantiss*
exponent: `(short)` *exponent*
isNegative: `(BOOL)` *isNegative*

Description forthcoming.

decimalNumberWithString: *X¬OG*

`+ (NSDecimalNumber*)`**decimalNumberWithString:** `(NSString*)` *numericString*

Description forthcoming.

decimalNumberWithString:locale: *X¬OG*

+ (NSDecimalNumber*)**decimalNumberWithString:** (NSString*) *numericString*
locale: (NSDictionary*) *locale*

Description forthcoming.

defaultBehavior

X¬OG

+ (id<NSDecimalNumberBehaviors>)**defaultBehavior**

Description forthcoming.

maximumDecimalNumber

X¬OG

+ (NSDecimalNumber*)**maximumDecimalNumber**

Description forthcoming.

minimumDecimalNumber

X¬OG

+ (NSDecimalNumber*)**minimumDecimalNumber**

Description forthcoming.

notANumber

X¬OG

+ (NSDecimalNumber*)**notANumber**

Description forthcoming.

one

X¬OG

+ (NSDecimalNumber*)**one**

Description forthcoming.

setDefaultBehavior:

X¬OG

+ (void)**setDefaultBehavior:** (id<NSDecimalNumberBehaviors>) *behavior*

Description forthcoming.

zero *X¬OG*

+ (NSDecimalNumber*)**zero**

Description forthcoming.

Instance Methods

compare: *X¬OG*

- (NSComparisonResult)**compare:**(NSNumber*) *decimalNumber*

Description forthcoming.

decimalNumberByAdding: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByAdding:**(NSDecimalNumber*) *decimalNumber*

Description forthcoming.

decimalNumberByAdding:withBehavior: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByAdding:**(NSDecimalNumber*) *decimalNumber*
withBehavior:(id<NSDecimalNumberBehavi

Description forthcoming.

decimalNumberByDividingBy: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByDividingBy:**(NSDecimalNumber*) *decimalNu*

Description forthcoming.

decimalNumberByDividingBy:withBehavior: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByDividingBy:**(NSDecimalNumber*) *decimalNu*
withBehavior:(id<NSDecimalNumberBeh

Description forthcoming.

decimalNumberByMultiplyingBy: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByMultiplyingBy:**(NSDecimalNumber*) *decimal*

Description forthcoming.

decimalNumberByMultiplyingBy:withBehavior: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByMultiplyingBy:**(NSDecimalNumber*) *decimal*
withBehavior:(id<NSDecimalNumberB

Description forthcoming.

decimalNumberByMultiplyingByPowerOf10: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByMultiplyingByPowerOf10:**(short) *power*

Description forthcoming.

decimalNumberByMultiplyingByPowerOf10:withBehavior: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByMultiplyingByPowerOf10:**(short) *power*
withBehavior:(id<NSDecim

Description forthcoming.

decimalNumberByRaisingToPower: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByRaisingToPower:**(unsigned) *power*

Description forthcoming.

decimalNumberByRaisingToPower:withBehavior: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByRaisingToPower:**(unsigned) *power*
withBehavior:(id<NSDecimalNumber

Description forthcoming.

decimalNumberByRoundingAccordingToBehavior: *X¬OG*

- (NSDecimalNumber*)**decimalNumberByRoundingAccordingToBehavior:**(id<NSD

Description forthcoming.

decimalNumberBySubtracting: *X¬OG*

- (NSDecimalNumber*)**decimalNumberBySubtracting:**(NSDecimalNumber*) *decimalNu*

Description forthcoming.

decimalNumberBySubtracting:withBehavior: *X¬OG*

- (NSDecimalNumber*)**decimalNumberBySubtracting:**(NSDecimalNumber*) *decimalNu*
withBehavior:(id<NSDecimalNumberBeh

Description forthcoming.

decimalValue *X¬OG*

- (NSDecimal)decimalValue

Description forthcoming.

descriptionWithLocale: $X\neg OG$

- (NSString*)descriptionWithLocale:(NSDictionary*) *locale*

Description forthcoming.

doubleValue $X\neg OG$

- (double)doubleValue

Description forthcoming.

initWithDecimal: $X\neg OG$

- (id)initWithDecimal:(NSDecimal) *decimal*

Description forthcoming.

initWithMantissa:exponent:isNegative: $X\neg OG$

- (id)initWithMantissa:(unsigned long long) *mantissa*
exponent:(short) *exponent*
isNegative:(BOOL) *flag*

Description forthcoming.

initWithString: $X\neg OG$

- (id)initWithString:(NSString*) *numberValue*

Description forthcoming.

initWithString:locale: $X\neg OG$

- (id)**initWithString:**(NSString*) *numberValue*
locale:(NSDictionary*) *locale*

Description forthcoming.

objCType

$X\neg OG$

- (const char*)**objCType**

Description forthcoming.

NSDecimalNumberHandler

X¬OG

Inherits From:	NSObject
Conforms To:	NSDecimalNumberBehaviors
Declared in:	Foundation/NSDecimalNumber.h

Description

Description forthcoming.

Class Methods

decimalNumberHandlerWithRoundingMode:scale:raiseOnExactness:raiseOnOverflow:r

X¬OG

`+ (id)` **decimalNumberHandlerWithRoundingMode:** `(NSRoundingMode)` *roundingMode*
scale: `(short)` *scale*
raiseOnExactness: `(BOOL)` *raiseOnExactness*
raiseOnOverflow: `(BOOL)` *raiseOnOverflow*
raiseOnUnderflow: `(BOOL)` *raiseOnUnderflow*
raiseOnDivideByZero: `(BOOL)` *raiseOnDivideByZero*

Description forthcoming.

defaultDecimalNumberHandler

X¬OG

`+ (id)` **defaultDecimalNumberHandler**

Description forthcoming.

Instance Methods

initWithRoundingMode:scale:raiseOnExactness:raiseOnOverflow:raiseOnUnderflow:ra

- (id)**initWithRoundingMode:**(NSRoundingMode) *roundingMode*
scale:(short) *scale*
raiseOnExactness:(BOOL) *raiseOnExactness*
raiseOnOverflow:(BOOL) *raiseOnOverflow*
raiseOnUnderflow:(BOOL) *raiseOnUnderflow*
raiseOnDivideByZero:(BOOL) *raiseOnDivideByZero*

Description forthcoming.

NSDictionary *XOG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying NSMutableCopying
Declared in:	Foundation/NSDictionary.h

Description

Description forthcoming.

Class Methods

dictionary *XOG*

`+ (id)`**dictionary**

Description forthcoming.

dictionaryWithContentsOfFile: *XOG*

`+ (id)`**dictionaryWithContentsOfFile:** `(NSString*)` *path*

Returns a dictionary using the file located at *path*. The file must be a property list containing a dictionary as its root object.

dictionaryWithDictionary: *XOG*

`+ (id)`**dictionaryWithDictionary:** `(NSDictionary*)` *otherDictionary*

Returns a newly created dictionary with the keys and objects of *otherDictionary*. (The keys and objects are not copied.)

dictionaryWithObject:forKey: *XOG*

+ (id)**dictionaryWithObject:** (id) *object*
forKey: (id) *key*

Returns a dictionary containing only one *object* which is associated with a *key*.

dictionaryWithObjects:forKeys: *XOG*

+ (id)**dictionaryWithObjects:** (NSArray*) *objects*
forKeys: (NSArray*) *keys*

Description forthcoming.

dictionaryWithObjects:forKeys:count: *XOG*

+ (id)**dictionaryWithObjects:** (id*) *objects*
forKeys: (id*) *keys*
count: (unsigned) *count*

Returns a dictionary created using the given *objects* and *keys*. The two arrays must have the same size. The n th element of the *objects* array is associated with the n th element of the *keys* array.

dictionaryWithObjectsAndKeys: *XOG*

+ (id)**dictionaryWithObjectsAndKeys:** (id) *firstObject*

Returns a dictionary created using the list given as argument. The list is alernately composed of objects and keys. Thus, the list's length must be pair.

Instance Methods

allKeys *XOG*

- (NSArray*)**allKeys**

Returns an array containing all the dictionary's keys.

allKeysForObject: *XOG*

- (NSArray*)**allKeysForObject:**(id) *anObject*

Returns an array containing all the dictionary's keys that are associated with *anObject*.

allValues *XOG*

- (NSArray*)**allValues**

Returns an array containing all the dictionary's objects.

count *XOG*

- (unsigned)**count**

Returns an unsigned integer which is the number of elements stored in the dictionary.

description *XOG*

- (NSString*)**description**

Returns the result of invoking -descriptionWithLocale:indent: with a `nil` locale and zero indent.

descriptionInStringsFileFormat *XOG*

- (NSString*)**descriptionInStringsFileFormat**

Returns the receiver as a text property list strings file format.
See [NSString -propertyListFromStringsFileFormat] for details.
The order of the items is undefined.

descriptionWithLocale: *XOG*

- (NSString*)**descriptionWithLocale:**(NSDictionary*) *locale*

Returns the result of invoking -descriptionWithLocale:indent: with a zero indent.

descriptionWithLocale:indent: *XOG*

- (NSString*)**descriptionWithLocale:**(NSDictionary*) *locale*
indent:(unsigned int) *level*

Returns the receiver as a text property list in the traditional format.

See [NSString -propertyList] for details.

If *locale* is nil, no formatting is done, otherwise entries are formatted according to the *locale*, and indented according to *level*.

Unless *locale* is nil, a *level* of zero indents items by four spaces, while a *level* of one indents them by a tab.

If the keys in the dictionary respond to -compare: , the items are listed by key in ascending order. If not, the order in which the items are listed is undefined.

initWithContentsOfFile: *XOG*

- (id)**initWithContentsOfFile:**(NSString*) *path*

Initialises the dictionary with the contents of the specified file, which must contain a dictionary in property-list format.

In GNUstep, the property-list format may be either the OpenStep format (ASCII data), or the MacOS-X format (URF8 XML data)... this method will recognise which it is.

If there is a failure to load the file for any reason, the receiver will be released and the method will return nil.

Works by invoking [NSString -initWithContentsOfFile:] and [NSString -propertyList] then checking that the result is a dictionary.

initWithDictionary: *XOG*

- (id)**initWithDictionary:**(NSDictionary*) *otherDictionary*

Description forthcoming.

initWithDictionary:copyItems: *XOG*

- (id)**initWithDictionary:**(NSDictionary*) *other*
copyItems:(BOOL) *shouldCopy*

Initialise dictionary with the keys and values of otherDictionary. If the *shouldCopy* flag is YES then the values are copied into the newly initialised dictionary, otherwise they are simply retained.

initWithObjects:forKeys: *XOG*

- `(id)`**initWithObjects:**`(NSArray*)` *objects*
forKeys:`(NSArray*)` *keys*

Initialises a dictionary created using the given *objects* and *keys*. The two arrays must have the same size. The n th element of the *objects* array is associated with the n th element of the *keys* array.

initWithObjects:forKeys:count: *XOG*

- `(id)`**initWithObjects:**`(id*)` *objects*
forKeys:`(id*)` *keys*
count:`(unsigned)` *count*

Description forthcoming.

initWithObjectsAndKeys: *XOG*

- `(id)`**initWithObjectsAndKeys:**`(id)` *firstObject*

Initialises a dictionary created using the list given as argument. The list is alernately composed of objects and keys. Thus, the list's length must be pair.

isEqualToDictionary: *XOG*

- `(BOOL)`**isEqualToDictionary:**`(NSDictionary*)` *other*

Description forthcoming.

keyEnumerator *XOG*

- `(NSEnumerator*)`**keyEnumerator**

Return an enumerator object containing all the keys of the dictionary.

keysSortedByValueUsingSelector: *XOG*

- (NSArray*)**keysSortedByValueUsingSelector:**(SEL) *comp*

Description forthcoming.

objectEnumerator *XOG*

- (NSEnumerator*)**objectEnumerator**

Return an enumerator object containing all the objects of the dictionary.

objectForKey: *XOG*

- (id)**objectForKey:**(id) *aKey*

Returns the object in the dictionary corresponding to *aKey*, or `nil` if the key is not present.

objectsForKeys:notFoundMarker: *XOG*

- (NSArray*)**objectsForKeys:**(NSArray*) *keys*
  notFoundMarker:(id) *marker*

Description forthcoming.

valueForKey: *X¬OG*

- (id)**valueForKey:**(NSString*) *key*

Default implementation for this class is to return the value stored in the dictionary under the specified *key*, or `nil` if there is no value.

writeToFile:atomically: *XOG*

- (BOOL)**writeToFile:**(NSString*) *path*
  atomically:(BOOL) *useAuxiliaryFile*

Writes the contents of the dictionary to the file specified by *path*. The file contents will be in property-list format... under GNUstep this is either OpenStep style (ASCII characters using \U hexadecimal escape sequences for unicode), or MacOS-X style (XML in the UTF8 character set).

If the *useAuxiliaryFile* flag is YES, the file write operation is atomic... the data is written to a temporary file, which is then renamed to the actual file name.

If the conversion of data into the correct property-list format fails or the write operation fails, the method returns NO, otherwise it returns YES.

NB. The fact that the file is in property-list format does not necessarily mean that it can be used to reconstruct the dictionary using the -initWithContentsOfFile: method. If the original dictionary contains non-property-list objects, the descriptions of those objects will have been written, and reading in the file as a property-list will result in a new dictionary containing the string descriptions.

writeToURL:atomically: *X¬OG*

- (BOOL)**writeToURL:**(NSURL*) *url*
  atomically:(BOOL) *useAuxiliaryFile*

Writes the contents of the dictionary to the specified *url*. This functions just like -writeToFile:atomically: except that the output may be written to any URL, not just a local file.

NSDirectoryEnumerator $X\neg OG$

Inherits From: NSEnumerator : NSObject
Declared in: Foundation/NSFileManager.h

Description

NSDirectoryEnumerator implementation

The Objective-C interface hides a traditional C implementation. This was the only way I could get near the speed of standard unix tools for big directories.

Instance Methods

directoryAttributes $X\neg OG$

- (NSDictionary*)**directoryAttributes**

Returns a dictionary containing the attributes of the directory at which enumeration started.

The contents of this dictionary are as produced by [NSFileManager -fileAttributesAtPath:tra

fileAttributes $X\neg OG$

- (NSDictionary*)**fileAttributes**

Returns a dictionary containing the attributes of the file currently being enumerated.

The contents of this dictionary are as produced by [NSFileManager -fileAttributesAtPath:tra

initWithDirectoryPath:recurseIntoSubdirectories:followSymlinks:justContents: $X\neg OG$

- (id)**initWithDirectoryPath:**(NSString*) *path*
  recurseIntoSubdirectories:(BOOL) *recurse*
  followSymlinks:(BOOL) *follow*
  justContents:(BOOL) *justContents*

Description forthcoming.

skipDescendents

$X\neg OG$

- (void)skipDescendents

Informs the receiver that any descendents of the current directory should be skipped rather than enumerated. Use this to avaoid enumerating the contents of directories you are not interested in.

NSDistantObject *XOG*

Inherits From: NSProxy
Conforms To: NSCoding
Declared in: Foundation/NSDistantObject.h

Description

Description forthcoming.

Class Methods

proxyWithLocal:connection: *XOG*

`+ (NSDistantObject*)`**proxyWithLocal:** `(id)` *anObject*
connection: `(NSConnection*)` *aConnection*

Description forthcoming.

proxyWithTarget:connection: *XOG*

`+ (NSDistantObject*)`**proxyWithTarget:** `(unsigned)` *anObject*
connection: `(NSConnection*)` *aConnection*

Description forthcoming.

Instance Methods

connectionForProxy *XOG*

`- (NSConnection*)`**connectionForProxy**

Description forthcoming.

initWithLocal:connection: *XOG*

- (id)initWithLocal:(id) *anObject*
connection:(NSConnection*) *aConnection*

Description forthcoming.

initWithTarget:connection: *XOG*

- (id)initWithTarget:(unsigned) *target*
connection:(NSConnection*) *aConnection*

Description forthcoming.

setProtocolForProxy: *XOG*

- (void)setProtocolForProxy:(Protocol*) *aProtocol*

Description forthcoming.

NSDistributedLock *XOG*

Inherits From: NSObject
Declared in: Foundation/NSDistributedLock.h

Description

Description forthcoming.

Class Methods

lockWithPath: *XOG*

`+ (NSDistributedLock*)` **lockWithPath:** `(NSString*)` *aPath*

Return a distributed lock for *aPath*. See -initWithPath: for details.

Instance Methods

breakLock *XOG*

`- (void)` **breakLock**

Forces release of the lock whether the receiver owns it or not.
Raises an NSGenericException if unable to remove the lock.

initWithPath: *XOG*

`- (NSDistributedLock*)` **initWithPath:** `(NSString*)` *aPath*

Initialises the reciever with the specified filesystem path.
The location in the filesystem must be accessible for this to be usable. That is, the processes using the lock must be able to access, create, and destroy files at the path.
The directory in which the last path component resides must already exist... create it using NSFileManager if you need to.

lockDate *XOG*

- (NSDate*)lockDate

Returns the date at which the lock was aquired by any NSDistributedLock using the same path. If nothing has the lock, this returns `nil`.

tryLock *XOG*

- (BOOL)tryLock

Attempt to aquire the lock and return `YES` on success, `NO` on failure.
May raise an NSGenericException if a problem occurs.

unlock *XOG*

- (void)unlock

Releases the lock. Raises an NSGenericException if unable to release the lock (for instance if the receiver does not own it or another process has broken it).

NSDistributedNotificationCenter

$X\neg OG$

Inherits From: NSNotificationCenter : NSObject
Declared in: Foundation/NSDistributedNotificationCenter.h

Description

The NSDistributedNotificationCenter provides a versatile yet simple mechanism for objects in different processes to communicate effectively while knowing very little about each others internals.

A distributed notification center acts much like a normal notification center, but it handles notifications on a machine-wide (or local network wide) basis rather than just notifications within a single process. Objects are able to register themselves as observers for particular notification names and objects, and they will then receive notifications (including optional user information consisting of a dictionary of property-list objects) as they are posted.

Since posting of distributed notifications involves inter-process (and sometimes inter-host) communication, it is fundamentally slower than normal notifications, and should be used relatively sparingly. In order to help with this, the NSDistributedNotificationCenter provides a notion of 'suspension', whereby a center can be suspended causing notifications for observers in the process where the center was suspended to cease receiving notifications. Observers can specify how notifications are to be handled in this case (queued or discarded) and posters can specify that particular notifications are to be delivered immediately irrespective of suspension.

Distributed notifications are mediated by a server process which handles all notifications for a particular center type. In GNUstep this process is the gdnc tool, and when started without special options, a gdnc process acts as the local centre for the host it is running on. When started with the GSNetwork user default set to YES, the gdnc tool acts as a local network wide server (you should only run one copy of gdnc like this on your LAN).

The gdnc process should be started at machine boot time, but GNUstep will attempt to start it automatically if it can't find it.

MacOS-X currently defines only a notification center for the local host. GNUstep also defines a local network center which can be used from multiple hosts. By default the system sends this to any gdnc process it can find which is configured as a network-wide server, but the GDNCHost user default may be used to specify a particular host to be contacted... this may be of use where you wish to have logically separate clusters of machines on a shared LAN.

Class Methods

defaultCenter

X¬OG

`+ (NSNotificationCenter*)`**defaultCenter**

Returns the default notification center... a shared notification center for the local host. This is simply a convenience method equivalent to calling +notificationCenterForType: with NSLocalNotificationCenterType as its argument.

notificationCenterForType:

X¬OG

`+ (NSNotificationCenter*)`**notificationCenterForType:** `(NSString*)` *type*

Returns a notification center of the specified *type*.

The NSLocalNotificationCenterType provides a shared access to a notificatiuon center used by processes on the local host.

The GSNetworkNotificationCenterType provides a shared access to a notificatiuon center used by processes on the local network.

MacOS-X supports only NSLocalNotificationCenterType.

Instance Methods

addObserver:selector:name:object:

X¬OG

`- (void)`**addObserver:**`(id)` *anObserver*
selector:`(SEL)` *aSelector*
name:`(NSString*)` *notificationName*
object:`(NSString*)` *anObject*

Adds an observer to the receiver. Calls -addObserver:selector:name:object:suspensionBehavi with NSNotificationSuspensionBehaviorCoalesce.

addObserver:selector:name:object:suspensionBehavior:

X¬OG

`- (void)`**addObserver:**`(id)` *anObserver*
selector:`(SEL)` *aSelector*
name:`(NSString*)` *notificationName*
object:`(NSString*)` *anObject*
suspensionBehavior:`(NSNotificationSuspensionBehavior)` *suspensionBehavior*

Adds an observer to the receiver.

When a notification matching *notificationName* and *anObject* is sent to the center, the object *anObserver* is sent the message *aSelector* with the notification info dictionary as its argument.

The *suspensionBehavior* governs how the center deals with notifications when the process to which the notification should be delivered is suspended:

NSNotificationSuspensionBehaviorDrop Discards the notification if the observing process is suspended.

NSNotificationSuspensionBehaviorCoalesce Discards previously queued notifications when the observing process is suspended, leaving only the last notification posted in the queue. Delivers this single notification when the process becomes unsuspended.

NSNotificationSuspensionBehaviorHold Queues notifications when the observing process is suspended, delivering all the queued notifications when the process becomes unsuspended again.

NSNotificationSuspensionBehaviorDeliverImmediately Deliver the notification immediately, even if the destination process is suspended.

postNotification: $X\neg OG$

- (void)**postNotification:**(NSNotification*) *notification*

Posts the *notification* to the center using postNotificationName:object:userInfo:deliverImmedi
with the delivery flag set to NO.

postNotificationName:object: $X\neg OG$

- (void)**postNotificationName:**(NSString*) *notificationName*
object:(NSString*) *anObject*

Posts the *notificationName* and *anObject* to the center using postNotificationName:object:user
with the user info set to nil and the delivery flag set to NO.

postNotificationName:object:userInfo: $X\neg OG$

- (void)**postNotificationName:**(NSString*) *notificationName*
object:(NSString*) *anObject*
userInfo:(NSDictionary*) *userInfo*

Posts the *notificationName*, *anObject* and *userInfo* to the center using postNotificationName:object:userInfo:deliverImmediately: with the delivery flag set to `NO`.

postNotificationName:object:userInfo:deliverImmediately: *X¬OG*

- (void)postNotificationName:(NSString*) *notificationName*
object:(NSString*) *anObject*
userInfo:(NSDictionary*) *userInfo*
deliverImmediately:(BOOL) *deliverImmediately*

The primitive notification posting method...

The *userInfo* dictionary may contain only property-list objects.

The *deliverImmediately* flag specifies whether the suspension state of the receiving process is to be ignored.

removeObserver:name:object: *X¬OG*

- (void)removeObserver:(id) *anObserver*
name:(NSString*) *notificationName*
object:(NSString*) *anObject*

Removes the observer from the center.

setSuspended: *X¬OG*

- (void)setSuspended:(BOOL) *flag*

Sets the suspension state of the receiver... if the receiver is suspended, it won't handle notification until it is unsuspended again, unless the notifications are posted to be delivered immediately.

suspended *X¬OG*

- (BOOL)suspended

Returns the current suspension state of the receiver.

NSEnumerator *XOG*

Inherits From: NSObject
Declared in: Foundation/NSEnumerator.h

Description

Description forthcoming.

Instance Methods

allObjects *XOG*

- (NSArray*)allObjects

Description forthcoming.

nextObject *XOG*

- (id)nextObject

Description forthcoming.

NSException *XOG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying
Declared in:	Foundation/NSException.h

Description

The NSException class helps manage errors in a program. It provides a mechanism for lower-level methods to provide information about problems to higher-level methods, which more often than not, have a better ability to decide what to do about the problems.

Exceptions are typically handled by enclosing a sensitive section of code inside the macros NS_DURING and NS_HANDLER, and then handling any problems after this, up to the NS_ENDHANDLER macro:

```
NS_DURING
  code that might cause an exception
NS_HANDLER
  code that deals with the exception. If this code cannot deal
with
  it, you can re-raise the exception like this
  [localException raise]
  so the next higher level of code can handle it
NS_ENDHANDLER
```

The local variable localException is the name of the exception object you can use in the NS_HANDLER section. The easiest way to cause an exeption is using the +raise:format:,... method.

Class Methods

exceptionWithName:reason:userInfo: *XOG*

```
+ (NSException*)exceptionWithName: (NSString*) name
                           reason: (NSString*) reason
                         userInfo: (NSDictionary*) userInfo
```

Create an an exception object with a *name* , *reason* and a dictionary *userInfo* which can be used to provide additional information or access to objects needed to handle the exception. After the exception is created you must -raise it.

raise:format: *XOG*

```
+ (void)raise: (NSString*) name
       format: (NSString*) format
```

Creates an exception with a *name* and a reason using the *format* string and any additional arguments. The exception is then raised.

raise:format:arguments: *XOG*

```
+ (void)raise: (NSString*) name
       format: (NSString*) format
    arguments: (va_list) argList
```

Creates an exception with a *name* and a reason string using the *format* string and additional arguments specified as a variable argument list *argList*. The exception is then raised.

Instance Methods

initWithName:reason:userInfo: *XOG*

```
- (id)initWithName:(NSString*) name
           reason:(NSString*) reason
         userInfo:(NSDictionary*) userInfo
```

Initializes a newly allocated NSException object with a *name*, *reason* and a dictionary *userInfo*.

name *XOG*

```
- (NSString*)name
```

Returns the name of the exception

raise *XOG*

- (void)**raise**

Raises the exception. All code following the raise will not be executed and program control will be transfered to the closest calling method which encapsulates the exception code in an NS_DURING macro, or to the uncaught exception handler if there is no other handling code.

reason *XOG*

- (NSString*)**reason**

Returns the exception reason

userInfo *XOG*

- (NSDictionary*)**userInfo**

Returns the exception userInfo dictionary

NSFileHandle *XOG*

Inherits From: NSObject
Declared in: Foundation/NSFileHandle.h

Description

NSFileHandler is a Class that provides a wrapper for accessing system files and other connections. You can open connections to a file using class methods such as +fileHandleForReadingAtPath: .

GNUstep extends the use of this Class to allow you to create network connections (sockets), secure connections and also allows you to use compression with these files and connections (as long as GNUstep Base was compiled with the zlib library).

Class Methods

fileHandleForReadingAtPath: *XOG*

`+ (id)` **fileHandleForReadingAtPath:** `(NSString*)` *path*

Returns an NSFileHandle object setup for reading from the file listed at *path*. If the file does not exist or cannot be opened for some other reason, `nil` is returned.

fileHandleForUpdatingAtPath: *XOG*

`+ (id)` **fileHandleForUpdatingAtPath:** `(NSString*)` *path*

Returns an NSFileHandle object setup for updating (reading and writing) from the file listed at *path*. If the file does not exist or cannot be opened for some other reason, `nil` is returned.

fileHandleForWritingAtPath: *XOG*

`+ (id)` **fileHandleForWritingAtPath:** `(NSString*)` *path*

Returns an NSFileHandle object setup for writing to the file listed at *path*. If the file does not exist or cannot be opened for some other reason, `nil` is returned.

fileHandleWithNullDevice *XOG*

+ (id)fileHandleWithNullDevice

Returns a file handle object that is connected to the null device (i.e. a device that does nothing.) It is typically used in arrays and other collections of file handle objects as a place holder (null) object, so that all objects can respond to the same messages.

fileHandleWithStandardError *XOG*

+ (id)fileHandleWithStandardError

Returns an NSFileHandle object for the standard error descriptor. The returned object is a shared instance as there can only be one standard error per process.

fileHandleWithStandardInput *XOG*

+ (id)fileHandleWithStandardInput

Returns an NSFileHandle object for the standard input descriptor. The returned object is a shared instance as there can only be one standard input per process.

fileHandleWithStandardOutput *XOG*

+ (id)fileHandleWithStandardOutput

Returns an NSFileHandle object for the standard output descriptor. The returned object is a shared instance as there can only be one standard output per process.

Instance Methods

acceptConnectionInBackgroundAndNotify *XOG*

- (void)acceptConnectionInBackgroundAndNotify

Description forthcoming.

acceptConnectionInBackgroundAndNotifyForModes: *XOG*

- `(void)`**acceptConnectionInBackgroundAndNotifyForModes:**`(NSArray*)` *modes*

Description forthcoming.

availableData *XOG*

- `(NSData*)`**availableData**

Description forthcoming.

closeFile *XOG*

- `(void)`**closeFile**

Description forthcoming.

fileDescriptor *XOG*

- `(int)`**fileDescriptor**

Description forthcoming.

initWithFileDescriptor: *XOG*

- `(id)`**initWithFileDescriptor:**`(int)` *desc*

Description forthcoming.

initWithFileDescriptor:closeOnDealloc: *XOG*

- `(id)`**initWithFileDescriptor:**`(int)` *desc*
closeOnDealloc:`(BOOL)` *flag*

Description forthcoming.

initWithNativeHandle: *XOG*

- `(id)`**initWithNativeHandle:**`(void*)` *hdl*

Description forthcoming.

initWithNativeHandle:closeOnDealloc: *XOG*

- `(id)`**initWithNativeHandle:**`(void*)` *hdl*
closeOnDealloc:`(BOOL)` *flag*

Description forthcoming.

nativeHandle *XOG*

- `(void*)`**nativeHandle**

Description forthcoming.

offsetInFile *XOG*

- `(unsigned long long)`**offsetInFile**

Description forthcoming.

readDataOfLength: *XOG*

- `(NSData*)`**readDataOfLength:**`(unsigned int)` *len*

Description forthcoming.

readDataToEndOfFile *XOG*

- `(NSData*)`**readDataToEndOfFile**

Description forthcoming.

readInBackgroundAndNotify *XOG*

- `(void)`**readInBackgroundAndNotify**

Call -readInBackgroundAndNotifyForModes: with `nil` modes.

readInBackgroundAndNotifyForModes: *XOG*

- `(void)`**readInBackgroundAndNotifyForModes:**`(NSArray*)` *modes*

Set up an asynchonous read operation which will cause a notification to be sent when any amount of data (or end of file) is read. Note that the file handle will not continuously send notifications when data is available. If you want to continue to receive notifications, you need to send this message again after receiving a notification.

readToEndOfFileInBackgroundAndNotify *XOG*

- `(void)`**readToEndOfFileInBackgroundAndNotify**

Call -readToEndOfFileInBackgroundAndNotifyForModes: with `nil` modes.

readToEndOfFileInBackgroundAndNotifyForModes: *XOG*

- `(void)`**readToEndOfFileInBackgroundAndNotifyForModes:**`(NSArray*)` *modes*

Set up an asynchonous read operation which will cause a notification to be sent when end of file is read.

seekToEndOfFile *XOG*

- `(unsigned long long)`**seekToEndOfFile**

Description forthcoming.

seekToFileOffset: *XOG*

- `(void)`**seekToFileOffset:**`(unsigned long long)` *pos*

Description forthcoming.

synchronizeFile *XOG*

- `(void)`**synchronizeFile**

Description forthcoming.

truncateFileAtOffset: *XOG*

- `(void)`**truncateFileAtOffset:**`(unsigned long long)` *pos*

Description forthcoming.

waitForDataInBackgroundAndNotify *XOG*

- `(void)`**waitForDataInBackgroundAndNotify**

Call -waitForDataInBackgroundAndNotifyForModes: with `nil` modes.

waitForDataInBackgroundAndNotifyForModes: *XOG*

- `(void)`**waitForDataInBackgroundAndNotifyForModes:**`(NSArray*)` *modes*

Set up to provide a notification when data can be read from the handle.

writeData: *XOG*

- `(void)`**writeData:**`(NSData*)` *item*

Description forthcoming.

NSFileManager

X¬OG

Inherits From: NSObject
Declared in: Foundation/NSFileManager.h

Description

This is the main class for management of the local filesystem.

Class Methods

defaultManager

X¬OG

`+ (NSFileManager*)`**defaultManager**

Returns a shared default file manager which may be used throughout an application.

Instance Methods

changeCurrentDirectoryPath:

X¬OG

`- (BOOL)`**changeCurrentDirectoryPath:**`(NSString*)` *path*

Changes the current directory used for all subsequent operations.
All non-absolute paths are interpreted relative to this directory.
The current directory is set on a per-task basis, so the current directory for other file manager instances will also be changed by this method.

changeFileAttributes:atPath:

X¬OG

`- (BOOL)`**changeFileAttributes:**`(NSDictionary*)` *attributes*
atPath:`(NSString*)` *path*

Change the *attributes* of the file at *path* to those specified.

Returns YES if all requested changes were made (or if the dictionary was nil or empty, so no changes were requested), NO otherwise.
On failure, some fo the requested changes may have taken place.

componentsToDisplayForPath: *X¬OG*

- (NSArray*)**componentsToDisplayForPath:**(NSString*) *path*

Returns an array of *path* components suitably modified for display to the end user. This modification may render the returned strings unusable for *path* manipulation, so you should work with two arrays... one returned by this method (for display tio the user), and a parallel one returned by [NSString -pathComponents] (for *path* manipulation).

contentsAtPath: *X¬OG*

- (NSData*)**contentsAtPath:**(NSString*) *path*

Reads the file at *path* an returns its contents as an NSData object.
If an error occurs or if *path* specifies a directory etc then nil is returned.

contentsEqualAtPath:andPath: *X¬OG*

- (BOOL)**contentsEqualAtPath:**(NSString*) *path1*
andPath:(NSString*) *path2*

Returns YES if the contents of the file or directory at *path1* are the same as those at *path2*.
If *path1* and *path2* are files, this is a simple comparison. If they are directories, the contents of the files in those subdirectories are compared recursively.
Symbolic links are not followed.
A comparison checks first file identity, then size, then content.

copyPath:toPath:handler: *X¬OG*

- (BOOL)**copyPath:**(NSString*) *source*
toPath:(NSString*) *destination*
handler:(id) *handler*

Copies the file or directory at *source* to *destination*, using a *handler* object which should respond to [NSObject -fileManager:willProcessPath:] and [NSObject -fileManager:shouldProceedAfterError:] messages.

createDirectoryAtPath:attributes: *X¬OG*

- (BOOL)**createDirectoryAtPath:**(NSString*) *path*
attributes:(NSDictionary*) *attributes*

Creates a new directory, and sets its *attributes* as specified.
Creates other directories in the *path* as necessary.
Returns YES on success, NO on failure.

createFileAtPath:contents:attributes: *X¬OG*

- (BOOL)**createFileAtPath:**(NSString*) *path*
contents:(NSData*) *contents*
attributes:(NSDictionary*) *attributes*

Creates a new file, and sets its *attributes* as specified.
Initialises the file content with the specified data.
Returns YES on success, NO on failure.

createSymbolicLinkAtPath:pathContent: *X¬OG*

- (BOOL)**createSymbolicLinkAtPath:**(NSString*) *path*
pathContent:(NSString*) *otherPath*

Creates a symbolic link at *path* which links to the location specified by *otherPath*.

currentDirectoryPath *X¬OG*

- (NSString*)**currentDirectoryPath**

Returns the current working directory used by all instance of the file manager in the current task.

directoryContentsAtPath: *X¬OG*

- (NSArray*)**directoryContentsAtPath:**(NSString*) *path*

Returns an array of the contents of the specified directory.
The listing does **not** recursively list subdirectories.

The special files '.' and '..' are not listed.
Indicates an error by returning `nil` (eg. if *path* is not a directory or it can't be read for some reason).

displayNameAtPath:

X¬OG

`-(NSString*)`**displayNameAtPath:**`(NSString*)` *path*

Returns the name of the file or directory at *path*. Converts it into a format for display to an end user. This may render it unusable as part of a file/path name.
For instance, if a user has elected not to see file extensions, this method may return filenames with the extension removed.
The default operation is to return the result of calling [NSString -lastPathComponent] on the *path*.

enumeratorAtPath:

X¬OG

`-(NSDirectoryEnumerator*)`**enumeratorAtPath:**`(NSString*)` *path*

Returns an enumerator which can be used to return each item with the directory at *path* in turn.
The enumeration is recursive... following all nested subdirectories.

fileAttributesAtPath:traverseLink:

X¬OG

`-(NSDictionary*)`**fileAttributesAtPath:**`(NSString*)` *path*
traverseLink:`(BOOL)` *flag*

If a file (or directory etc) exists at the specified *path*, and can be queriesd for its attributes, this method returns a dictionary containing the various attributes of that file. Otherwise `nil` is returned.
If the *flag* is `NO` and the file is a symbolic link, the attributes of the link itsself (rather than the file it points to) are returned.
The dictionary keys for attributes are -

NSFileAppendOnly NSNumber... boolean

NSFileCreationDate NSDate when the file was created (if supported)

NSFileDeviceIdentifier NSNumber (identifies the device on which the file is stored)

NSFileExtensionHidden NSNumber... boolean

NSFileGroupOwnerAccountName NSString name of the file group

NSFileGroupOwnerAccountID NSNumber ID of the file group

NSFileHFSCreatorCode NSNumber not used

NSFileHFSTypeCode NSNumber not used

NSFileImmutable NSNumber... boolean

NSFileModificationDate NSDate when the file was last modified

NSFileOwnerAccountName NSString name of the file owner

NSFileOwnerAccountID NSNumber ID of the file owner

NSFilePosixPermissions NSNumber posix access permissions mask

NSFileReferenceCount NSNumber number of links to this file

NSFileSize NSNumber size of the file in bytes

NSFileSystemFileNumber NSNumber the identifier for the file on the filesystem

NSFileSystemNumber NSNumber the filesystem on which the file is stored

NSFileType NSString the type of file

The NSDictionary class also has a set of accessor methods which enable you to get at file attribute information more efficiently than using the keys above to extract it. You should generally use the accessor methods where they are available.

- [NSDictionary -fileCreationDate]
- [NSDictionary -fileExtensionHidden]
- [NSDictionary -fileHFSCreatorCode]
- [NSDictionary -fileHFSTypeCode]
- [NSDictionary -fileIsAppendOnly]
- [NSDictionary -fileIsImmutable]
- [NSDictionary -fileSize]

- [NSDictionary -fileType]
- [NSDictionary -fileOwnerAccountName]
- [NSDictionary -fileOwnerAccountID]
- [NSDictionary -fileGroupOwnerAccountName]
- [NSDictionary -fileGroupOwnerAccountID]
- [NSDictionary -fileModificationDate]
- [NSDictionary -filePosixPermissions]
- [NSDictionary -fileSystemNumber]
- [NSDictionary -fileSystemFileNumber]

fileExistsAtPath: *X¬OG*

`-(BOOL)`**fileExistsAtPath:**`(NSString*)` *path*

Returns `YES` if a file (or directory etc) exists at the specified *path*.

fileExistsAtPath:isDirectory: *X¬OG*

`-(BOOL)`**fileExistsAtPath:**`(NSString*)` *path*
isDirectory:`(BOOL*)` *isDirectory*

Returns `YES` if a file (or directory etc) exists at the specified *path*.
If the *isDirectory* argument is not a nul pointer, stores a flag in the location it points to, to indicate whether the file is a directory or not.

fileSystemAttributesAtPath: *X¬OG*

`-(NSDictionary*)`**fileSystemAttributesAtPath:**`(NSString*)` *path*

Returns a dictionary containing the filesystem attributes for the specified *path* (or `nil` if the *path* is not valid).

NSFileSystemSize NSNumber the size of the filesystem in bytes

NSFileSystemFreeSize NSNumber the amount of unused space on the filesystem in bytes

NSFileSystemNodes NSNumber the number of nodes in use to store files

NSFileSystemFreeNodes NSNumber the number of nodes available to create files

NSFileSystemNumber NSNumber the identifying number for the filesystem

fileSystemRepresentationWithPath: *X¬OG*

- (const char*)**fileSystemRepresentationWithPath:**(NSString*) *path*

Convert from OpenStep internal *path* format (unix-style) to a string in the local filesystem format, suitable for passing to system functions.

Under unix, this simply standardizes the *path* and converts to a C string.

Under windoze, this attempts to use local conventions to convert to a windows *path*. In GNUstep, the conventional unix syntax '~user/...' can be used to indicate a windoze drive specification by using the drive letter in place of the username.

isDeletableFileAtPath: *X¬OG*

- (BOOL)**isDeletableFileAtPath:**(NSString*) *path*

Returns YES if a file (or directory etc) exists at the specified *path* and is deletable.

isExecutableFileAtPath: *X¬OG*

- (BOOL)**isExecutableFileAtPath:**(NSString*) *path*

Returns YES if a file (or directory etc) exists at the specified *path* and is executable (if a directory is executable, you can access its contents).

isReadableFileAtPath: *X¬OG*

- (BOOL)**isReadableFileAtPath:**(NSString*) *path*

Returns YES if a file (or directory etc) exists at the specified *path* and is readable.

isWritableFileAtPath: *X¬OG*

- (BOOL)**isWritableFileAtPath:**(NSString*) *path*

Returns YES if a file (or directory etc) exists at the specified *path* and is writable.

linkPath:toPath:handler: *X¬OG*

- (BOOL)**linkPath:**(NSString*) *source*
toPath:(NSString*) *destination*
handler:(id) *handler*

Links the file or directory at *source* to *destination*, using a *handler* object which should respond to [NSObject -fileManager:willProcessPath:] and [NSObject -fileManager:shouldProceedAfterError:] messages.

movePath:toPath:handler: *X¬OG*

- (BOOL)**movePath:**(NSString*) *source*
toPath:(NSString*) *destination*
handler:(id) *handler*

Moves the file or directory at *source* to *destination*, using a *handler* object which should respond to [NSObject -fileManager:willProcessPath:] and [NSObject -fileManager:shouldProceedAfterError:] messages.

pathContentOfSymbolicLinkAtPath: *X¬OG*

- (NSString*)**pathContentOfSymbolicLinkAtPath:**(NSString*) *path*

Returns the name of the file or directory that the symbolic link at *path* points to.

removeFileAtPath:handler: *X¬OG*

- (BOOL)**removeFileAtPath:**(NSString*) *path*
handler:(id) *handler*

Removes the file or directory at *path*, using a *handler* object which should respond to [NSObject -fileManager:willProcessPath:] and [NSObject -fileManager:shouldProce
messages.

stringWithFileSystemRepresentation:length: *X¬OG*

- (NSString*)**stringWithFileSystemRepresentation:**(const char*) *string*
length:(unsigned int) *len*

This method converts from a local system specific filename representation to the internal OpenStep representation (unix-style). This should be used whenever a filename is read in from the local system.

In GNUstep, windoze drive specifiers are encoded in the internal path using the conventuional unix syntax of '~user/...' where the drive letter is used instead of a username.

subpathsAtPath: *X¬OG*

- (NSArray*)**subpathsAtPath:**(NSString*) *path*

Returns an array containing the (relative) paths of all the items in the directory at *path*.

The listing follows all subdirectories, so it can produce a very large array... use with care.

NSFormatter

X¬OG

Inherits From:	NSObject
Conforms To:	NSCopying NSCoding
Declared in:	Foundation/NSFormatter.h

Description

Description forthcoming.

Instance Methods

attributedStringForObjectValue:withDefaultAttributes:

X¬OG

- (NSAttributedString*)**attributedStringForObjectValue:**(id) *anObject*
withDefaultAttributes:(NSDictionary*) *attr*

Description forthcoming.

editingStringForObjectValue:

X¬OG

- (NSString*)**editingStringForObjectValue:**(id) *anObject*

Description forthcoming.

getObjectValue:forString:errorDescription:

X¬OG

- (BOOL)**getObjectValue:**(id*) *anObject*
forString:(NSString*) *string*
errorDescription:(NSString**) *error*

Description forthcoming.

isPartialStringValid:newEditingString:errorDescription:

X¬OG

```
- (BOOL)isPartialStringValid:(NSString*) partialString
          newEditingString:(NSString**) newString
          errorDescription:(NSString**) error
```

Description forthcoming.

isPartialStringValid:proposedSelectedRange:originalString:originalSelectedRange:erro

X¬OG

```
- (BOOL)isPartialStringValid:(NSString**) partialStringPtr
      proposedSelectedRange:(NSRange*) proposedSelRangePtr
             originalString:(NSString*) origString
      originalSelectedRange:(NSRange) originalSelRangePtr
          errorDescription:(NSString**) error
```

Description forthcoming.

stringForObjectValue:

X¬OG

```
- (NSString*)stringForObjectValue:(id) anObject
```

Description forthcoming.

NSHost *XOG*

Inherits From: NSObject
Declared in: Foundation/NSHost.h

Description

Description forthcoming.

Class Methods

currentHost *XOG*

`+ (NSHost*)` **currentHost**

Description forthcoming.

flushHostCache *XOG*

`+ (void)` **flushHostCache**

Description forthcoming.

hostWithAddress: *XOG*

`+ (NSHost*)` **hostWithAddress:** `(NSString*)` *address*

Description forthcoming.

hostWithName: *XOG*

`+ (NSHost*)` **hostWithName:** `(NSString*)` *name*

Description forthcoming.

isHostCacheEnabled *XOG*

`+ (BOOL)`**isHostCacheEnabled**

Description forthcoming.

setHostCacheEnabled: *XOG*

`+ (void)`**setHostCacheEnabled:** `(BOOL)` *flag*

Description forthcoming.

Instance Methods

address *XOG*

`- (NSString*)`**address**

Description forthcoming.

addresses *XOG*

`- (NSArray*)`**addresses**

Description forthcoming.

isEqualToHost: *XOG*

`- (BOOL)`**isEqualToHost:**`(NSHost*)` *aHost*

Description forthcoming.

name *XOG*

`- (NSString*)`**name**

Description forthcoming.

`-(NSArray*)`**names**

Description forthcoming.

NSInvocation *XOG*

Inherits From: NSObject
Declared in: Foundation/NSInvocation.h

Description

The NSInvocation class implements a mechanism of constructing messages (as NSInvocation instances), sending these to other objects, and handling the returned values.

An NSInvocation object may contain a target object to which a message can be sent, or may send the message to an arbitrary object.

Each message consists of a selector for that method and an argument list. Once the message has been sent, the invocation will contain a return value whose contents may be copied out of it.

The target, selector, and arguments of an instance be constructed dynamically, providing a great deal of power/flexibility.

The sending of the message to the target object (using the -invoke or -invokeWithTarget: method) can be done at any time, but a standard use of this is by the [NSObject -forwardInvocation:] method which is called whenever a method is not implemented by the class of the object to which it was sent.

Related to the class are two convenience macros... NS_MESSAGE() and NS_INVOCATIO to allow easy construction of invocations with all the arguments set up.

Class Methods

invocationWithMethodSignature: *XOG*

`+ (NSInvocation*)`**invocationWithMethodSignature:** `(NSMethodSignature*)` *_signatu*

Returns an invocation instance which can be used to send messages to a target object using the described signature.

Raises an NSInvalidArgumentException if the signature is `nil`.

Instance Methods

argumentsRetained *XOG*

- (BOOL)argumentsRetained

Returns a flag to indicate whether object arguments of the invocation (including its target) are retained by the invocation.

getArgument:atIndex: *XOG*

- (void)getArgument:(void*) *buffer*
atIndex:(int) *index*

Copies the argument identified by *index* into the memory location specified by the *buffer* argument.
An *index* of zero is the target object, an *index* of one is the selector, so the actual method arguments start at *index* 2.

getReturnValue: *XOG*

- (void)getReturnValue:(void*) *buffer*

Copies the invocations return value to the location pointed to by *buffer* if a return value has been set (see the -setReturnValue: method).
If there isn't a return value then this method raises an exception.

invoke *XOG*

- (void)invoke

Sends the message encapsulated in the invocation to its target.

invokeWithTarget: *XOG*

- (void)invokeWithTarget:(id) *anObject*

Sends the message encapsulated in the invocation to *anObject*.

methodSignature *XOG*

- (NSMethodSignature*)**methodSignature**

Returns the method signature of the invocation.

retainArguments *XOG*

- (void)**retainArguments**

Instructs the invocation to retain its object arguments (including the target). The default is not to retain them.

selector *XOG*

- (SEL)**selector**

Returns the selector of the invocation (the argument at index 1)

setArgument:atIndex: *XOG*

- (void)**setArgument:**(void*) *buffer*
  atIndex:(int) *index*

Sets the argument identified by *index* from the memory location specified by the *buffer* argument.

Using an *index* of 0 is equivalent to calling -setTarget: and using an argument of 1 is equivalent to -setSelector:

Proper arguments start at *index* 2.

NB. Unlike -setTarget: and -setSelector: the value of *buffer* must be *a pointer to* the argument to be set in the invocation.

If -retainArguments was called, then any object argument set in the receiver is retained by it.

setReturnValue: *XOG*

- (void)**setReturnValue:**(void*) *buffer*

Sets the return value of the invocation to the item that *buffer* points to.

setSelector: *XOG*

- `(void)`**setSelector:**`(SEL)` *aSelector*

Sets the selector for the invocation.

setTarget: *XOG*

- `(void)`**setTarget:**`(id)` *anObject*

Sets the target object for the invocation.
If -retainArguments was called, then the target is retained.

target *XOG*

- `(id)`**target**

Returns the target object of the invocation.

NSLock *XOG*

Inherits From:	NSObject
Conforms To:	NSLocking GCFinalization
Declared in:	Foundation/NSLock.h

Description

An NSLock is used in multi-threaded applications to protect critical pieces of code. While one thread holds a lock within a piece of code, another thread cannot execute that code until the first thread has given up it's hold on the lock. The limitation of NSLock is that you can only lock an NSLock once and it must be unlocked before it can be aquired again.

Other lock classes, notably NSRecursiveLock, have different restrictions.

Instance Methods

lock *XOG*

`- (void)` **lock**

Attempts to aquire a lock, and waits until it can do so.

lockBeforeDate: *XOG*

`- (BOOL)` **lockBeforeDate:** `(NSDate*)` *limit*

Attempts to aquire a lock before the date *limit* passes. It returns `YES` if it can. It returns `NO` if it cannot, or if the current thread already has the lock (but it waits until the time *limit* is up before returning `NO`).

tryLock *XOG*

`- (BOOL)` **tryLock**

Attempts to aquire a lock, but returns immediately if the lock cannot be aquired. It returns `YES` if the lock is aquired. It returns `NO` if the lock cannot be aquired or if the current thread already has the lock.

unlock *XOG*

`- (void)`**unlock**

Description forthcoming.

NSMethodSignature *XOG*

Inherits From: NSObject
Declared in: Foundation/NSMethodSignature.h

Description

Description forthcoming.

Class Methods

signatureWithObjCTypes: *XOG*

`+ (NSMethodSignature*)` **signatureWithObjCTypes:** `(const char*)` *t*

Description forthcoming.

Instance Methods

argumentInfoAtIndex: *O¬XG*

`- (NSArgumentInfo)` **argumentInfoAtIndex:** `(unsigned)` *index*

Description forthcoming.

frameLength *XOG*

`- (unsigned)` **frameLength**

Description forthcoming.

getArgumentTypeAtIndex: *XOG*

`- (const char*)` **getArgumentTypeAtIndex:** `(unsigned)` *index*

Description forthcoming.

isOneway *XOG*

- `(BOOL)`**isOneway**

Description forthcoming.

methodReturnLength *XOG*

- `(unsigned)`**methodReturnLength**

Description forthcoming.

methodReturnType *XOG*

- `(const char*)`**methodReturnType**

Description forthcoming.

numberOfArguments *XOG*

- `(unsigned)`**numberOfArguments**

Description forthcoming.

NSMutableArray *XOG*

Inherits From: NSArray : NSObject
Declared in: Foundation/NSArray.h

Description

Description forthcoming.

Class Methods

arrayWithCapacity: *XOG*

`+ (id)` **arrayWithCapacity:** `(unsigned)` *numItems*

Creates an autoreleased mutable array anble to store at least *numItems*. See the -initWithCapacity: method.

Instance Methods

addObject: *XOG*

`- (void)` **addObject:**`(id)` *anObject*

Adds *anObject* at the end of the array, thus increasing the size of the array. The object is retained upon addition.

addObjectsFromArray: *XOG*

`- (void)` **addObjectsFromArray:**`(NSArray*)` *otherArray*

Adds each object from *otherArray* to the receiver, in first to last order.

exchangeObjectAtIndex:withObjectAtIndex: *X¬OG*

- (void)**exchangeObjectAtIndex:**(unsigned int) *i1*
withObjectAtIndex:(unsigned int) *i2*

Swaps the positions of two objects in the array. Raises an exception if either array index is out of bounds.

initWithCapacity: *XOG*

- (id)**initWithCapacity:**(unsigned) *numItems*

Initialise the array with the specified capacity ... this should ensure that the array can have *numItems* added efficiently.

insertObject:atIndex: *XOG*

- (void)**insertObject:**(id) *anObject*
atIndex:(unsigned) *index*

Inserts an object into the receiver at the specified location.
Raises an exception if given an array *index* which is too large.
The size of the array increases by one.
The object is retained by the array.

removeAllObjects *XOG*

- (void)**removeAllObjects**

Removes all objects from the receiver, leaving an empty array.

removeLastObject *XOG*

- (void)**removeLastObject**

Removes the last object in the array. Raises an exception if the array is already empty.

removeObject: *XOG*

- (void)**removeObject:**(id) *anObject*

Removes all occurrances of *anObject* (found by anObjects -isEqual: method) from the receiver.

removeObject:inRange: *XOG*

- (void)**removeObject:**(id) *anObject*
  inRange:(NSRange) *aRange*

Removes all occurrances of *anObject* (found by the -isEqual: method of *anObject*) *aRange* in the receiver.

removeObjectAtIndex: *XOG*

- (void)**removeObjectAtIndex:**(unsigned) *index*

Removes an object from the receiver at the specified location.
The size of the array decreases by one.
Raises an exception if given an array *index* which is too large.

removeObjectIdenticalTo: *XOG*

- (void)**removeObjectIdenticalTo:**(id) *anObject*

Removes all occurrances of *anObject* (found by pointer equality) from the receiver.

removeObjectIdenticalTo:inRange: *XOG*

- (void)**removeObjectIdenticalTo:**(id) *anObject*
  inRange:(NSRange) *aRange*

Removes all occurrances of *anObject* (found by pointer equality) from *aRange* in the receiver.

removeObjectsFromIndices:numIndices: *XOG*

- (void)**removeObjectsFromIndices:**(unsigned*) *indices*
  numIndices:(unsigned) *count*

Supplied with a C array of *indices* containing *count* values, this method removes all corresponding objects from the receiver. The objects are removed in such a way that the removal is *safe* irrespective of the order in which they are specified in the *indices* array.

removeObjectsInArray: *XOG*

- `(void)`**removeObjectsInArray:**`(NSArray*)` *otherArray*

Removes from the receiver, all the objects present in *otherArray*, as determined by using the -isEqual: method.

removeObjectsInRange: *XOG*

- `(void)`**removeObjectsInRange:**`(NSRange)` *aRange*

Removes all the objects in *aRange* from the receiver.

⊙ replaceObjectAtIndex:withObject: *XOG*

- `(void)`**replaceObjectAtIndex:**`(unsigned)` *index*
withObject:`(id)` *anObject*

Places an object into the receiver at the specified location.
Raises an exception if given an array *index* which is too large.
The object is retained by the array.

replaceObjectsInRange:withObjectsFromArray: *XOG*

- `(void)`**replaceObjectsInRange:**`(NSRange)` *aRange*
withObjectsFromArray:`(NSArray*)` *anArray*

Replaces objects in the receiver with those from *anArray*.
Raises an exception if given a range extending beyond the array.

replaceObjectsInRange:withObjectsFromArray:range: *XOG*

- `(void)`**replaceObjectsInRange:**`(NSRange)` *aRange*
withObjectsFromArray:`(NSArray*)` *anArray*
range:`(NSRange)` *anotherRange*

Replaces objects in the receiver with some of those from *anArray*.
Raises an exception if given a range extending beyond the array.

setArray: *XOG*

- (void)**setArray:**(NSArray*) *otherArray*

Sets the contents of the receiver to be identical to the contents of othrArray.

sortUsingFunction:context: *XOG*

- (void)**sortUsingFunction:**(NSComparisonResult(*)(id,id,void*)) *compare*
context:(void*) *context*

Sorts the array according to the supplied *compare* function with the *context* information.

sortUsingSelector: *XOG*

- (void)**sortUsingSelector:**(SEL) *comparator*

Sorts the array according to the supplied *comparator*.

NSMutableAttributedString *X¬OG*

Inherits From: NSAttributedString : NSObject
Declared in: Foundation/NSAttributedString.h

Description

Description forthcoming.

Instance Methods

addAttribute:value:range: *X¬OG*

- (void)**addAttribute:**(NSString*) *name*
value:(id) *value*
range:(NSRange) *aRange*

Description forthcoming.

addAttributes:range: *X¬OG*

- (void)**addAttributes:**(NSDictionary*) *attributes*
range:(NSRange) *aRange*

Description forthcoming.

appendAttributedString: *X¬OG*

- (void)**appendAttributedString:**(NSAttributedString*) *attributedString*

Description forthcoming.

beginEditing *X¬OG*

- (void)**beginEditing**

Description forthcoming.

deleteCharactersInRange: *X¬OG*

- `(void)`**deleteCharactersInRange:**`(NSRange)` *aRange*

Description forthcoming.

endEditing *X¬OG*

- `(void)`**endEditing**

Description forthcoming.

insertAttributedString:atIndex: *X¬OG*

- `(void)`**insertAttributedString:**`(NSAttributedString*)` *attributedString*
atIndex:`(unsigned int)` *index*

Description forthcoming.

mutableString *X¬OG*

- `(NSMutableString*)`**mutableString**

Description forthcoming.

removeAttribute:range: *X¬OG*

- `(void)`**removeAttribute:**`(NSString*)` *name*
range:`(NSRange)` *aRange*

Description forthcoming.

replaceCharactersInRange:withAttributedString: *X¬OG*

- `(void)`**replaceCharactersInRange:**`(NSRange)` *aRange*
withAttributedString:`(NSAttributedString*)` *attributedString*

Description forthcoming.

replaceCharactersInRange:withString: *X¬OG*

- `(void)`**replaceCharactersInRange:**`(NSRange)` *aRange*
withString:`(NSString*)` *aString*

Description forthcoming.

setAttributedString: *X¬OG*

- `(void)`**setAttributedString:**`(NSAttributedString*)` *attributedString*

Description forthcoming.

setAttributes:range: *X¬OG*

- `(void)`**setAttributes:**`(NSDictionary*)` *attributes*
range:`(NSRange)` *aRange*

Description forthcoming.

NSMutableBitmapCharSet *XOG*

Inherits From: NSMutableCharacterSet : NSCharacterSet : NSObject
Declared in: Foundation/NSBitmapCharSet.h

Description

Description forthcoming.

Instance Methods

initWithBitmap: *XOG*

- (id)**initWithBitmap:**(NSData*) *bitmap*

Description forthcoming.

NSMutableCharacterSet *XOG*

Inherits From: NSCharacterSet : NSObject
Declared in: Foundation/NSCharacterSet.h

Description

Description forthcoming.

Instance Methods

addCharactersInRange: *XOG*

- `(void)`**addCharactersInRange:**`(NSRange)` *aRange*

Description forthcoming.

addCharactersInString: *XOG*

- `(void)`**addCharactersInString:**`(NSString*)` *aString*

Description forthcoming.

formIntersectionWithCharacterSet: *XOG*

- `(void)`**formIntersectionWithCharacterSet:**`(NSCharacterSet*)` *otherSet*

Description forthcoming.

formUnionWithCharacterSet: *XOG*

- `(void)`**formUnionWithCharacterSet:**`(NSCharacterSet*)` *otherSet*

Description forthcoming.

invert *XOG*

`- (void)`**invert**

Description forthcoming.

removeCharactersInRange: *XOG*

`- (void)`**removeCharactersInRange:**`(NSRange)` *aRange*

Description forthcoming.

removeCharactersInString: *XOG*

`- (void)`**removeCharactersInString:**`(NSString*)` *aString*

Description forthcoming.

NSMutableData *XOG*

Inherits From: NSData : NSObject
Declared in: Foundation/NSData.h

Description

Description forthcoming.

Class Methods

dataWithCapacity: *XOG*

`+ (id)` **dataWithCapacity:** `(unsigned int)` *numBytes*

Description forthcoming.

dataWithLength: *XOG*

`+ (id)` **dataWithLength:** `(unsigned int)` *length*

Description forthcoming.

Instance Methods

appendBytes:length: *XOG*

`- (void)` **appendBytes:**`(const void*)` *aBuffer*
length:`(unsigned int)` *bufferSize*

Description forthcoming.

appendData: *XOG*

- (void)appendData:(NSData*) *other*

Description forthcoming.

increaseLengthBy: *XOG*

- (void)increaseLengthBy:(unsigned int) *extraLength*

Description forthcoming.

initWithCapacity: *XOG*

- (id)initWithCapacity:(unsigned int) *capacity*

Description forthcoming.

initWithLength: *XOG*

- (id)initWithLength:(unsigned int) *length*

Description forthcoming.

mutableBytes *XOG*

- (void*)mutableBytes

Returns a pointer to the data storage of the receiver.
Modifications to the memory pointed to by this pointer will change the contents of the object. It is important that your code should not try to modify the memory beyond the number of bytes given by the `-length` method.
NB. if the object is released, or any method that changes its size or content is called, then the pointer previously returned by this method may cease to be valid.
This is a 'primitive' method... you need to implement it if you write a subclass of NSMutableData.

replaceBytesInRange:withBytes: *XOG*

- (void)replaceBytesInRange:(NSRange) *aRange*
withBytes:(const void*) *bytes*

Replaces the *bytes* of data in the specified range with a copy of the new *bytes* supplied.

If the location of the range specified lies beyond the end of the data (`[self length] < range.location`) then a range exception is raised.

Otherwise, if the range specified extends beyond the end of the data, then the size of the data is increased to accomodate the new *bytes*.

replaceBytesInRange:withBytes:length: *X¬OG*

```
- (void)replaceBytesInRange:(NSRange) aRange
                  withBytes:(const void*) bytes
                     length:(unsigned int) length
```

Replace the content of the receiver which lies in *aRange* with the specified *length* of data from the buffer pointed to by *bytes*.

The size of the receiver is adjusted to allow for the change.

resetBytesInRange: *XOG*

```
- (void)resetBytesInRange:(NSRange) aRange
```

Description forthcoming.

serializeAlignedBytesLength: *XOG*

```
- (void)serializeAlignedBytesLength:(unsigned int) length
```

Description forthcoming.

serializeDataAt:ofObjCType:context: *XOG*

```
- (void)serializeDataAt:(const void*) data
             ofObjCType:(const char*) type
                context:(id&#60;NSObjCTypeSerializationCallBack>) callback
```

Description forthcoming.

serializeInt: *XOG*

- (void)**serializeInt:**(int) *value*

Description forthcoming.

serializeInt:atIndex: *XOG*

- (void)**serializeInt:**(int) *value*
  atIndex:(unsigned int) *index*

Description forthcoming.

serializeInts:count: *XOG*

- (void)**serializeInts:**(int*) *intBuffer*
  count:(unsigned int) *numInts*

Description forthcoming.

serializeInts:count:atIndex: *XOG*

- (void)**serializeInts:**(int*) *intBuffer*
  count:(unsigned int) *numInts*
  atIndex:(unsigned int) *index*

Description forthcoming.

setData: *XOG*

- (void)**setData:**(NSData*) *data*

Description forthcoming.

setLength: *XOG*

- (void)**setLength:**(unsigned int) *size*

Sets the length of the NSMutableData object. If the length is increased, the newly allocated data area is filled with zero bytes.

This is a 'primitive' method... you need to implement it if you write a subclass of NSMutableData.

NSMutableDictionary *XOG*

Inherits From: NSDictionary : NSObject
Declared in: Foundation/NSDictionary.h

Description

Description forthcoming.

Class Methods

dictionaryWithCapacity: *XOG*

`+ (id)` **dictionaryWithCapacity:** `(unsigned)` *numItems*

Description forthcoming.

Instance Methods

addEntriesFromDictionary: *XOG*

`- (void)` **addEntriesFromDictionary:**`(NSDictionary*)` *otherDictionary*

Merges information from *otherDictionary* into the receiver. If a key exists in both dictionaries, the value from *otherDictionary* replaces that which was originally in the reciever.

initWithCapacity: *XOG*

`- (id)` **initWithCapacity:**`(unsigned)` *numItems*

Description forthcoming.

removeAllObjects *XOG*

- (void)**removeAllObjects**

Description forthcoming.

removeObjectForKey: *XOG*

- (void)**removeObjectForKey:**(id) *aKey*

Description forthcoming.

removeObjectsForKeys: *XOG*

- (void)**removeObjectsForKeys:**(NSArray*) *keyArray*

Description forthcoming.

setDictionary: *XOG*

- (void)**setDictionary:**(NSDictionary*) *otherDictionary*

Description forthcoming.

setObject:forKey: *XOG*

- (void)**setObject:**(id) *anObject*
  forKey:(id) *aKey*

Description forthcoming.

takeStoredValue:forKey: *X¬OG*

- (void)**takeStoredValue:**(id) *value*
  forKey:(NSString*) *key*

Default implementation for this class is equivalent to the -setObject:forKey: method unless *value* is nil, in which case it is equivalent to -removeObjectForKey:

takeValue:forKey: *X¬OG*

- (void)**takeValue:**(id) *value*
forKey:(NSString*) *key*

Default implementation for this class is equivalent to the -setObject:forKey: method unless *value* is nil, in which case it is equivalent to -removeObjectForKey:

NSMutableSet *XOG*

Inherits From: NSSet : NSObject
Declared in: Foundation/NSSet.h

Description

Description forthcoming.

Class Methods

setWithCapacity: *XOG*

`+ (id)setWithCapacity: (unsigned)` *numItems*

Description forthcoming.

Instance Methods

addObject: *XOG*

`- (void)addObject:(id)` *anObject*

Adds *anObject* to the set.
The object is retained by the set.

addObjectsFromArray: *XOG*

`- (void)addObjectsFromArray:(NSArray*)` *array*

Adds all the objects in the *array* to the receiver.

initWithCapacity: *XOG*

- (id)**initWithCapacity:**(unsigned) *numItems*

Initialises a newly allocated set to contain no objects but to have space available to hold the specified number of items.
Additions of items to a set initialised with an appropriate capacity will be more efficient than addition of items otherwise.

intersectSet: *XOG*

- (void)**intersectSet:**(NSSet*) *other*

Removes from the receiver all the objects it contains which are not also in *other*.

minusSet: *XOG*

- (void)**minusSet:**(NSSet*) *other*

Removes from the receiver all the objects that are in *other*.

removeAllObjects *XOG*

- (void)**removeAllObjects**

Removes all objects from the receiver.

removeObject: *XOG*

- (void)**removeObject:**(id) *anObject*

Removes the *anObject* from the receiver.

setSet: *X¬OG*

- (void)**setSet:**(NSSet*) *other*

Removes all objects from the receiver then adds the objects from *other*. If the receiver *is other*, the method has no effect.

unionSet: *XOG*

- (void)**unionSet:**(NSSet*) *other*

Adds all the objects from *other* to the receiver.

NSMutableString *XOG*

Inherits From: NSString : NSObject
Declared in: Foundation/NSString.h

Description

This is the mutable form of the NSString class.

Class Methods

string *XOG*

`+ (id)`**string**

Description forthcoming.

stringWithCString: *XOG*

`+ (id)`**stringWithCString:** `(const char*)` *byteString*

Description forthcoming.

stringWithCString:length: *XOG*

`+ (id)`**stringWithCString:** `(const char*)` *byteString*
length: `(unsigned int)` *length*

Description forthcoming.

stringWithCapacity: *XOG*

`+ (NSMutableString*)`**stringWithCapacity:** `(unsigned int)` *capacity*

Description forthcoming.

stringWithCharacters:length: *XOG*

+ (id)**stringWithCharacters:** (const unichar*) *characters*
length: (unsigned int) *length*

Description forthcoming.

stringWithContentsOfFile: *XOG*

+ (id)**stringWithContentsOfFile:** (NSString*) *path*

Description forthcoming.

stringWithFormat: *XOG*

+ (id)**stringWithFormat:** (NSString*) *format*

Description forthcoming.

Instance Methods

appendFormat: *XOG*

- (void)**appendFormat:**(NSString*) *format*

Description forthcoming.

appendString: *XOG*

- (void)**appendString:**(NSString*) *aString*

Description forthcoming.

deleteCharactersInRange: *XOG*

- (void)**deleteCharactersInRange:**(NSRange) *range*

Description forthcoming.

initWithCapacity: *XOG*

- (id)**initWithCapacity:**(unsigned int) *capacity*

Description forthcoming.

insertString:atIndex: *XOG*

- (void)**insertString:**(NSString*) *aString*
atIndex:(unsigned int) *loc*

Description forthcoming.

replaceCharactersInRange:withString: *XOG*

- (void)**replaceCharactersInRange:**(NSRange) *range*
withString:(NSString*) *aString*

Description forthcoming.

replaceOccurrencesOfString:withString:options:range: *XOG*

- (unsigned int)**replaceOccurrencesOfString:**(NSString*) *replace*
withString:(NSString*) *by*
options:(unsigned int) *opts*
range:(NSRange) *searchRange*

Replaces all occurrences of the *replace* string with the *by* string, for those cases where the entire *replace* string lies within the specified *searchRange* value.

The value of *opts* determines the direction of the search is and whether only leading/trailing occurrances (anchored search) of *replace* are substituted.

Raises NSInvalidArgumentException if either string argument is `nil`.

Raises NSRangeException if part of *searchRange* is beyond the end of the receiver.

setString: *XOG*

- (void)**setString:**(NSString*) *aString*

Description forthcoming.

NSNotification *XOG*

Inherits From:	NSObject
Conforms To:	NSCopying NSCoding
Declared in:	Foundation/NSNotification.h

Description

Description forthcoming.

Class Methods

notificationWithName:object: *XOG*

`+ (NSNotification*)`**notificationWithName:** `(NSString*)` *name*
object: `(id)` *object*

Create a new autoreleased notification by calling +notificationWithName:object:userInfo: with a `nil` user info argument.

notificationWithName:object:userInfo: *XOG*

`+ (NSNotification*)`**notificationWithName:** `(NSString*)` *name*
object: `(id)` *object*
userInfo: `(NSDictionary*)` *info*

Create a new autoreleased notification. Concrete subclasses override this method to create actual notification objects.

Instance Methods

name

XOG

- (NSString*)name

Concrete subclasses of NSNotification are responsible for implementing this method to return the notification name.

object

XOG

- (id)object

Concrete subclasses of NSNotification are responsible for implementing this method to return the notification object.

userInfo

XOG

- (NSDictionary*)userInfo

Concrete subclasses of NSNotification are responsible for implementing this method to return the notification user information.

NSNotificationCenter *XOG*

Inherits From:	NSObject
Conforms To:	GCFinalization
Declared in:	Foundation/NSNotification.h

Description

Description forthcoming.

Class Methods

defaultCenter *XOG*

`+ (NSNotificationCenter*)`**defaultCenter**

Description forthcoming.

Instance Methods

addObserver:selector:name:object: *XOG*

`- (void)`**addObserver:**`(id)` *observer*
selector:`(SEL)` *selector*
name:`(NSString*)` *name*
object:`(id)` *object*

Description forthcoming.

postNotification: *XOG*

`- (void)`**postNotification:**`(NSNotification*)` *notification*

Posts *notification* to all the observers that match its NAME and OBJECT.
The GNUstep implementation calls -postNotificationName:object:userInfo: to perform the actual posting.

postNotificationName:object:

XOG

- (void)**postNotificationName:**(NSString*) *name*
object:(id) *object*

Creates and posts a notification using the -postNotificationName:object:userInfo: passing a nil user info argument.

postNotificationName:object:userInfo:

XOG

- (void)**postNotificationName:**(NSString*) *name*
object:(id) *object*
userInfo:(NSDictionary*) *info*

The preferred method for posting a notification.

For performance reasons, we don't wrap an exception handler round every message sent to an observer. This means that, if one observer raises an exception, later observers in the lists will not get the notification.

removeObserver:

XOG

- (void)**removeObserver:**(id) *observer*

Description forthcoming.

removeObserver:name:object:

XOG

- (void)**removeObserver:**(id) *observer*
name:(NSString*) *name*
object:(id) *object*

Description forthcoming.

NSNotificationQueue *XOG*

Inherits From: NSObject
Declared in: Foundation/NSNotificationQueue.h

Description

Description forthcoming.

Class Methods

defaultQueue *XOG*

`+ (NSNotificationQueue*)` **defaultQueue**

Description forthcoming.

Instance Methods

dequeueNotificationsMatching:coalesceMask: *XOG*

`- (void)` **dequeueNotificationsMatching:**`(NSNotification*)` *notification*
coalesceMask:`(unsigned int)` *coalesceMask*

Description forthcoming.

enqueueNotification:postingStyle: *XOG*

`- (void)` **enqueueNotification:**`(NSNotification*)` *notification*
postingStyle:`(NSPostingStyle)` *postingStyle*

Description forthcoming.

enqueueNotification:postingStyle:coalesceMask:forModes: *XOG*

- (void)**enqueueNotification:**(NSNotification*) *notification*
postingStyle:(NSPostingStyle) *postingStyle*
coalesceMask:(unsigned int) *coalesceMask*
forModes:(NSArray*) *modes*

Description forthcoming.

initWithNotificationCenter: *XOG*

- (id)**initWithNotificationCenter:**(NSNotificationCenter*) *notificationCenter*

Description forthcoming.

NSNull *XOG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying
Declared in:	Foundation/NSNull.h

Description

An object to use as a placeholder - in collections for instance.

Class Methods

null *XOG*

`+ (NSNull*)null`

Return an object that can be used as a placeholder in a collection. This method always returns the same object.

NSNumber *XOG*

Inherits From:	NSValue : NSObject
Conforms To:	NSCopying NSCoding
Declared in:	Foundation/NSValue.h

Description

Description forthcoming.

Class Methods

numberWithBool: *XOG*

`+ (NSNumber*)` **numberWithBool:** `(BOOL)` *value*

Description forthcoming.

numberWithChar: *XOG*

`+ (NSNumber*)` **numberWithChar:** `(signed char)` *value*

Description forthcoming.

numberWithDouble: *XOG*

`+ (NSNumber*)` **numberWithDouble:** `(double)` *value*

Description forthcoming.

numberWithFloat: *XOG*

`+ (NSNumber*)` **numberWithFloat:** `(float)` *value*

Description forthcoming.

numberWithInt: *XOG*

`+ (NSNumber*)`**numberWithInt:** `(signed int)` *value*
Description forthcoming.

numberWithLong: *XOG*

`+ (NSNumber*)`**numberWithLong:** `(signed long)` *value*
Description forthcoming.

numberWithLongLong: *XOG*

`+ (NSNumber*)`**numberWithLongLong:** `(signed long long)` *value*
Description forthcoming.

numberWithShort: *XOG*

`+ (NSNumber*)`**numberWithShort:** `(signed short)` *value*
Description forthcoming.

numberWithUnsignedChar: *XOG*

`+ (NSNumber*)`**numberWithUnsignedChar:** `(unsigned char)` *value*
Description forthcoming.

numberWithUnsignedInt: *XOG*

`+ (NSNumber*)`**numberWithUnsignedInt:** `(unsigned int)` *value*
Description forthcoming.

numberWithUnsignedLong: *XOG*

`+ (NSNumber*)`**numberWithUnsignedLong:** `(unsigned long)` *value*

Description forthcoming.

numberWithUnsignedLongLong: *XOG*

`+ (NSNumber*)`**numberWithUnsignedLongLong:** `(unsigned long long)` *value*

Description forthcoming.

numberWithUnsignedShort: *XOG*

`+ (NSNumber*)`**numberWithUnsignedShort:** `(unsigned short)` *value*

Description forthcoming.

Instance Methods

boolValue *XOG*

`- (BOOL)`**boolValue**

Description forthcoming.

charValue *XOG*

`- (signed char)`**charValue**

Description forthcoming.

compare: *XOG*

`- (NSComparisonResult)`**compare:**`(NSNumber*)` *otherNumber*

Description forthcoming.

description *XOG*

- `(NSString*)`**description**

Description forthcoming.

descriptionWithLocale: *XOG*

- `(NSString*)`**descriptionWithLocale:**`(NSDictionary*)` *locale*

Description forthcoming.

doubleValue *XOG*

- `(double)`**doubleValue**

Description forthcoming.

floatValue *XOG*

- `(float)`**floatValue**

Description forthcoming.

initWithBool: *XOG*

- `(id)`**initWithBool:**`(BOOL)` *value*

Description forthcoming.

initWithChar: *XOG*

- `(id)`**initWithChar:**`(signed char)` *value*

Description forthcoming.

initWithDouble: *XOG*

- (id)**initWithDouble:**(double) *value*
Description forthcoming.

initWithFloat: *XOG*

- (id)**initWithFloat:**(float) *value*
Description forthcoming.

initWithInt: *XOG*

- (id)**initWithInt:**(signed int) *value*
Description forthcoming.

initWithLong: *XOG*

- (id)**initWithLong:**(signed long) *value*
Description forthcoming.

initWithLongLong: *XOG*

- (id)**initWithLongLong:**(signed long long) *value*
Description forthcoming.

initWithShort: *XOG*

- (id)**initWithShort:**(signed short) *value*
Description forthcoming.

initWithUnsignedChar: *XOG*

- (id)initWithUnsignedChar:(unsigned char) *value*
Description forthcoming.

initWithUnsignedInt: *XOG*

- (id)initWithUnsignedInt:(unsigned int) *value*
Description forthcoming.

initWithUnsignedLong: *XOG*

- (id)initWithUnsignedLong:(unsigned long) *value*
Description forthcoming.

initWithUnsignedLongLong: *XOG*

- (id)initWithUnsignedLongLong:(unsigned long long) *value*
Description forthcoming.

initWithUnsignedShort: *XOG*

- (id)initWithUnsignedShort:(unsigned short) *value*
Description forthcoming.

intValue *XOG*

- (signed int)intValue
Description forthcoming.

isEqualToNumber: *XOG*

- (BOOL)isEqualToNumber:(NSNumber*) *otherNumber*
Description forthcoming.

longLongValue *XOG*

- `(signed long long)`**longLongValue**

Description forthcoming.

longValue *XOG*

- `(signed long)`**longValue**

Description forthcoming.

shortValue *XOG*

- `(signed short)`**shortValue**

Description forthcoming.

stringValue *XOG*

- `(NSString*)`**stringValue**

Description forthcoming.

unsignedCharValue *XOG*

- `(unsigned char)`**unsignedCharValue**

Description forthcoming.

unsignedIntValue *XOG*

- `(unsigned int)`**unsignedIntValue**

Description forthcoming.

unsignedLongLongValue *XOG*

- `(unsigned long long)`**unsignedLongLongValue**

Description forthcoming.

unsignedLongValue *XOG*

- `(unsigned long)`**unsignedLongValue**

Description forthcoming.

unsignedShortValue *XOG*

- `(unsigned short)`**unsignedShortValue**

Description forthcoming.

NSNumberFormatter $X\neg OG$

Inherits From: NSFormatter : NSObject
Declared in: Foundation/NSNumberFormatter.h

Description

Description forthcoming.

Instance Methods

allowsFloats $X\neg OG$

- (BOOL)allowsFloats

Description forthcoming.

attributedStringForNil $X\neg OG$

- (NSAttributedString*)attributedStringForNil

Description forthcoming.

attributedStringForNotANumber $X\neg OG$

- (NSAttributedString*)attributedStringForNotANumber

Description forthcoming.

attributedStringForZero $X\neg OG$

- (NSAttributedString*)attributedStringForZero

Description forthcoming.

decimalSeparator $X\neg OG$

- `(NSString*)`**decimalSeparator**

Description forthcoming.

format $X\neg OG$

- `(NSString*)`**format**

Description forthcoming.

hasThousandSeparators $X\neg OG$

- `(BOOL)`**hasThousandSeparators**

Description forthcoming.

localizesFormat $X\neg OG$

- `(BOOL)`**localizesFormat**

Description forthcoming.

maximum $X\neg OG$

- `(NSDecimalNumber*)`**maximum**

Description forthcoming.

minimum $X\neg OG$

- `(NSDecimalNumber*)`**minimum**

Description forthcoming.

negativeFormat $X\neg OG$

- (NSString*)**negativeFormat**

Description forthcoming.

positiveFormat *X¬OG*

- (NSString*)**positiveFormat**

Description forthcoming.

roundingBehavior *X¬OG*

- (NSDecimalNumberHandler*)**roundingBehavior**

Description forthcoming.

setAllowsFloats: *X¬OG*

- (void)**setAllowsFloats:**(BOOL) *flag*

Description forthcoming.

setAttributedStringForNil: *X¬OG*

- (void)**setAttributedStringForNil:**(NSAttributedString*) *newAttributedString*

Description forthcoming.

setAttributedStringForNotANumber: *X¬OG*

- (void)**setAttributedStringForNotANumber:**(NSAttributedString*) *newAttributedStr*

Description forthcoming.

setAttributedStringForZero: *X¬OG*

- (void)**setAttributedStringForZero:**(NSAttributedString*) *newAttributedString*

Description forthcoming.

setDecimalSeparator: *X¬OG*

- (void)**setDecimalSeparator:**(NSString*) *newSeparator*

Description forthcoming.

setFormat: *X¬OG*

- (void)**setFormat:**(NSString*) *aFormat*

Description forthcoming.

setHasThousandSeparators: *X¬OG*

- (void)**setHasThousandSeparators:**(BOOL) *flag*

Description forthcoming.

setLocalizesFormat: *X¬OG*

- (void)**setLocalizesFormat:**(BOOL) *flag*

Description forthcoming.

setMaximum: *X¬OG*

- (void)**setMaximum:**(NSDecimalNumber*) *aMaximum*

Description forthcoming.

setMinimum: *X¬OG*

- (void)**setMinimum:**(NSDecimalNumber*) *aMinimum*

Description forthcoming.

setNegativeFormat: *X¬OG*

- (void)**setNegativeFormat:**(NSString*) *aFormat*

Description forthcoming.

setPositiveFormat: *X¬OG*

- (void)**setPositiveFormat:**(NSString*) *aFormat*

Description forthcoming.

setRoundingBehavior: *X¬OG*

- (void)**setRoundingBehavior:**(NSDecimalNumberHandler*) *newRoundingBehavior*

Description forthcoming.

setTextAttributesForNegativeValues: *X¬OG*

- (void)**setTextAttributesForNegativeValues:**(NSDictionary*) *newAttributes*

Description forthcoming.

setTextAttributesForPositiveValues: *X¬OG*

- (void)**setTextAttributesForPositiveValues:**(NSDictionary*) *newAttributes*

Description forthcoming.

setThousandSeparator: *X¬OG*

- (void)**setThousandSeparator:**(NSString*) *newSeparator*

Description forthcoming.

textAttributesForNegativeValues $X\neg OG$

- `(NSDictionary*)`**textAttributesForNegativeValues**

Description forthcoming.

textAttributesForPositiveValues $X\neg OG$

- `(NSDictionary*)`**textAttributesForPositiveValues**

Description forthcoming.

thousandSeparator $X\neg OG$

- `(NSString*)`**thousandSeparator**

Description forthcoming.

NSObject

XOG

Inherits From:	None, is a root class
Conforms To:	NSObject
Declared in:	Foundation/NSObject.h

Description

`NSObject` is the root class (a root class is a class with no superclass) of the gnustep base library class hierarchy, so all classes normally inherit from `NSObject`. There is an exception though: `NSProxy` (which is used for remote messaging) does not inherit from `NSObject`.

Unless you are really sure of what you are doing, all your own classes should inherit (directly or indirectly) from `NSObject` (or in special cases from `NSProxy`). `NSObject` provides the basic common functionality shared by all gnustep classes and objects.

The essential methods which must be implemented by all classes for their instances to be usable within gnustep are declared in a separate protocol, which is the `NSObject` protocol. Both `NSObject` and `NSProxy` conform to this protocol, which means all objects in a gnustep application will conform to this protocol (btw, if you don't find a method of `NSObject` you are looking for in this documentation, make sure you also look into the documentation for the `NSObject` protocol).

Theoretically, in special cases you might need to implement a new root class. If you do, you need to make sure that your root class conforms (at least) to the `NSObject` protocol, otherwise it will not interact correctly with the gnustep framework. Said that, I must note that I have never seen a case in which a new root class is needed.

`NSObject` is a root class, which implies that instance methods of `NSObject` are treated in a special way by the Objective-C runtime. This is an exception to the normal way messaging works with class and instance methods: if the Objective-C runtime can't find a class method for a class object, as a last resort it looks for an instance method of the root class with the same name, and executes it if it finds it. This means that instance methods of the root class (such as `NSObject`) can be performed by class objects which inherit from that root class ! This can only happen if the class doesn't have a class method with the same name, otherwise that method - of course - takes the precedence. Because of this exception, `NSObject` 's instance methods are written in such a way that they work both on `NSObject` 's instances and on class objects.

Class Methods

alloc *XOG*

`+ (id)`**alloc**

Allocates a new instance of the receiver from the default zone, by invoking +allocWithZone: with `NSDefaultMallocZone()` as the zone argument.
Returns the created instance.

allocWithZone: *XOG*

`+ (id)`**allocWithZone:** `(NSZone*)` *z*

This is the basic method to create a new instance. It allocates a new instance of the receiver from the specified memory zone.
Memory for an instance of the receiver is allocated; a pointer to this newly created instance is returned. All instance variables are set to 0 except the `isa` pointer which is set to point to the object class. No initialization of the instance is performed: it is your responsibility to initialize the instance by calling an appropriate `init` method. If you are not using the garbage collector, it is also your responsibility to make sure the returned instance is destroyed when you finish using it, by calling the `release` method to destroy the instance directly, or by using `autorelease` and autorelease pools.
You do not normally need to override this method in subclasses, unless you are implementing a class which for some reasons silently allocates instances of another class (this is typically needed to implement class clusters and similar design schemes).
If you have turned on debugging of object allocation (by calling the `GSDebugAllocationActive` function), this method will also update the various debugging counts and monitors of allocated objects, which you can access using the `GSDebugAllocation...` functions.

class *XOG*

`+ (Class)`**class**

Returns the receiver.

description *XOG*

`+ (NSString*)`**description**

Returns a string describing the receiving class. The default implementation gives the name of the class by calling NSStringFromClass() .

initialize *XOG*

`+ (void)initialize`

Description forthcoming.

instanceMethodForSelector: *XOG*

`+ (IMP)instanceMethodForSelector: (SEL)` *aSelector*

Returns a pointer to the C function implementing the method used to respond to messages with *aSelector* by instances of the receiving class.
Raises NSInvalidArgumentException if given a null selector.

instanceMethodSignatureForSelector: *XOG*

`+ (NSMethodSignature*)instanceMethodSignatureForSelector: (SEL)` *aSelector*

Returns a pointer to the C function implementing the method used to respond to messages with *aSelector* whihc are sent to instances of the receiving class.
Raises NSInvalidArgumentException if given a null selector.

instancesRespondToSelector: *XOG*

`+ (BOOL)instancesRespondToSelector: (SEL)` *aSelector*

Returns a flag to say if instances of the receiver class will respond to the specified selector. This ignores situations where a subclass implements -forwardInvocation: to respond to selectors not normally handled... in these cases the subclass may override this method to handle it.
If given a null selector, raises NSInvalidArgumentException when in MacOS-X compatibility more, or returns `NO` otherwise.

isSubclassOfClass: *XOG*

+ (BOOL)**isSubclassOfClass:** (Class) *aClass*

Returns YES if the receiver is *aClass* or a subclass of *aClass*.

new

XOG

+ (id)**new**

This method is a short-hand for alloc followed by init, that is,
`NSObject *object = [NSObject new];` is exactly the same as
`NSObject *object = [[NSObject alloc] init];`
This is a general convention: all `new...` methods are supposed to return a newly allocated and initialized instance, as would be generated by an `alloc` method followed by a corresponding `init...` method. Please note that if you are not using a garbage collector, this means that instances generated by the `new...` methods are not autoreleased, that is, you are responsible for releasing (autoreleasing) the instances yourself. So when you use `new` you typically do something like:

```
NSMutableArray *array = AUTORELEASE ([NSMutableArray new]);
```

You do not normally need to override `new` in subclasses, because if you override `init` (and optionally `allocWithZone:` if you really need), `new` will automatically use your subclass methods.

You might need instead to define new `new...` methods specific to your subclass to match any `init...` specific to your subclass. For example, if your subclass defines an instance method

`initWithName:`

it might be handy for you to have a class method

`newWithName:`

which combines `alloc` and `initWithName:`. You would implement it as follows:

```
+ (id) newWithName: (NSString *)aName {return [[self alloc] initWith-
Name: aName];}
```

poseAsClass:

XOG

+ (void)**poseAsClass:** (Class) *aClassObject*

Sets up the ObjC runtime so that the receiver is used wherever code calls for *aClassObject* to be used.

requiresTypedMemory

¬O¬XG

+ (BOOL)**requiresTypedMemory**

Description forthcoming.

setVersion: *XOG*

+ (id)**setVersion:** (int) *aVersion*

Sets the version number of the receiving class.

superclass *XOG*

+ (Class)**superclass**

Returns the super class from which the recevier was derived.

version *XOG*

+ (int)**version**

Returns the version number of the receiving class.

Instance Methods

autorelease *XOG*

- (id)**autorelease**

Adds the receiver to the current autorelease pool, so that it will be sent a -release message when the pool is destroyed.
Returns the receiver.
In GNUstep, the [NSObject +enableDoubleReleaseCheck:] method may be used to turn on checking for ratain/release errors in this method.

awakeAfterUsingCoder: *XOG*

- (id)**awakeAfterUsingCoder:**(NSCoder*) *aDecoder*

Called after the receiver has been created by decoding some sort of archive. Returns self. Subclasses may override this to perform some special initialisation upon being decoded.

class *XOG*

- (Class)class

Returns the class of which the receiver is an instance.
The default implementation returns the private `isa` instance variable of NSObject, which is used to store a pointer to the objects class.
NB. When NSZombie is enabled (see NSDebug.h) this pointer is changed upon object deallocation.

classForArchiver *XOG*

- (Class)classForArchiver

Description forthcoming.

classForCoder *XOG*

- (Class)classForCoder

Description forthcoming.

classForPortCoder *XOG*

- (Class)classForPortCoder

Description forthcoming.

className *X¬OG*

- (NSString*)className

Returns the name of the class of the receiving object by using the NSStringFromClass() function.
This is a MacOS-X addition for apple scripting, which is also generally useful.

conformsToProtocol: *XOG*

`- (BOOL)conformsToProtocol:(Protocol*)` *aProtocol*

Returns a flag to say whether the class of the receiver conforms to *aProtocol*.

copy *XOG*

`- (id)copy`

Creates and returns a copy of the reciever by calling -copyWithZone: passing NSDefaultMallocZone()

dealloc *XOG*

`- (void)dealloc`

Deallocates the receiver by calling NSDeallocateObject() with self as the argument.

You should normally call the superclass implementation of this method when you override it in a subclass, or the memory occupied by your object will not be released.

`NSObject` 's implementation of this method destroys the receiver, by returning the memory allocated to the receiver to the system. After this method has been called on an instance, you must not refer the instance in any way, because it does not exist any longer. If you do, it is a bug and your program might even crash with a segmentation fault.

If you have turned on the debugging facilities for instance allocation, `NSObject` 's implementation of this method will also update the various counts and monitors of allocated instances (see the `GSDebugAllocation...` functions for more info).

Normally you are supposed to manage the memory taken by objects by using the high level interface provided by the `retain`, `release` and `autorelease` methods (or better by the corresponding macros `RETAIN`, `RELEASE` and `AUTORELEASE`), and by autorelease pools and such; whenever the release/autorelease mechanism determines that an object is no longer needed (which happens when its retain count reaches 0), it will call the `dealloc` method to actually deallocate the object. This means that normally, you should not need to call `dealloc` directly as the gnustep base library automatically calls it for you when the retain count of an object reaches 0.

Because the `dealloc` method will be called when an instance is being destroyed, if instances of your subclass use objects or resources (as it happens for most useful classes), you must override `dealloc` in subclasses to release all objects and

resources which are used by the instance, otherwise these objects and resources would be leaked. In the subclass implementation, you should first release all your subclass specific objects and resources, and then invoke super's implementation (which will do the same, and so on up in the class hierarchy to `NSObject` 's implementation, which finally destroys the object). Here is an example of the implementation of `dealloc` for a subclass whose instances have a single instance variable `name` which needs to be released when an instance is deallocated:

```
- (void) dealloc {RELEASE (name); [super dealloc];}
```

`dealloc` might contain code to release not only objects, but also other resources, such as open files, network connections, raw memory allocated in other ways, etc.

If you have allocated the memory using a non-standard mechanism, you will not call the superclass (NSObject) implementation of the method as you will need to handle the deallocation specially.

In some circumstances, an object may wish to prevent itsself from being deallocated, it can do this simply be refraining from calling the superclass implementation.

description *XOG*

- (NSString*)description

Returns a string describing the receiver. The default implementation gives the class and memory location of the receiver.

doesNotRecognizeSelector: *XOG*

- (void)doesNotRecognizeSelector:(SEL) *aSelector*

Raises an invalid argument exception providing infomration about the receivers inability to handle *aSelector*.

forwardInvocation: *XOG*

- (void)forwardInvocation:(NSInvocation*) *anInvocation*

This method is called automatically to handle a message sent to the receiver for which the receivers class has no method.

The default implemnentation calls -doesNotRecognizeSelector:

hash *XOG*

- (unsigned)**hash**

Returns the hash of the receiver. Subclasses should ensure that their implementations of this method obey the rule that if the -isEqual: method returns YES for two instances of the class, the -hash method returns the same value fro both instances.
The default implementation returns the address of the instance.

init *XOG*

- (id)**init**

Initialises the receiver... the NSObject implementation simply returns self.

isEqual: *XOG*

- (BOOL)**isEqual:**(id) *anObject*

Tests *anObject* and the receiver for equality. The default implementation considers two objects to be equal only if they are the same object (ie occupy the same memory location).
If a subclass overrides this method, it should also override the -hash method so that if two objects are equal they both have the same hash.

isKindOfClass: *XOG*

- (BOOL)**isKindOfClass:**(Class) *aClass*

Returns YES if the class of the receiver is either the same as *aClass* or is derived from (a subclass of) *aClass*.

isMemberOfClass: *XOG*

- (BOOL)**isMemberOfClass:**(Class) *aClass*

Returns YES if the class of the receiver is *aClass*

isProxy *XOG*

- (BOOL)**isProxy**

Returns a flag to differentiate between 'true' objects, and objects which are proxies for other objects (ie they forward messages to the other objects).
The default implementation returns NO.

methodForSelector: *XOG*

- (IMP)**methodForSelector:**(SEL) *aSelector*

Returns a pointer to the C function implementing the method used to respond to messages with *aSelector*.
Raises NSInvalidArgumentException if given a null selector.

methodSignatureForSelector: *XOG*

- (NSMethodSignature*)**methodSignatureForSelector:**(SEL) *aSelector*

Returns the method signature describing how the receiver would handle a message with *aSelector*.
Raises NSInvalidArgumentException if given a null selector.

mutableCopy *XOG*

- (id)**mutableCopy**

Creates and rturns a mutable copy of the receiver by calling -mutableCopyWithZone: passing NSDefaultMallocZone() .

performSelector: *XOG*

- (id)**performSelector:**(SEL) *aSelector*

Causes the receiver to execute the method implementation corresponding to *aSelector* and returns the result.
The method must be one which takes no arguments and returns an object.
Raises NSInvalidArgumentException if given a null selector.

performSelector:withObject: *XOG*

- (id)**performSelector:**(SEL) *aSelector*
withObject:(id) *anObject*

Causes the receiver to execute the method implementation corresponding to *aSelector* and returns the result.
The method must be one which takes one argument and returns an object.
Raises NSInvalidArgumentException if given a null selector.

performSelector:withObject:withObject: *XOG*

- (id)**performSelector:**(SEL) *aSelector*
withObject:(id) *object1*
withObject:(id) *object2*

Causes the receiver to execute the method implementation corresponding to *aSelector* and returns the result.
The method must be one which takes two arguments and returns an object.
Raises NSInvalidArgumentException if given a null selector.

release *XOG*

- (void)**release**

Decrements the retain count for the receiver if greater than zeron, otherwise calls the dealloc method instead.
The default implementation calls the NSDecrementExtraRefCountWasZero() function to test the extra reference count for the receiver (and decrement it if non-zero) - if the extra reference count is zero then the retain count is one, and the dealloc method is called.
In GNUstep, the [NSObject +enableDoubleReleaseCheck:] method may be used to turn on checking for ratain/release errors in this method.

replacementObjectForArchiver: *XOG*

- (id)**replacementObjectForArchiver:**(NSArchiver*) *anArchiver*

Description forthcoming.

replacementObjectForCoder: *XOG*

- (id)**replacementObjectForCoder:**(NSCoder*) *anEncoder*

Description forthcoming.

replacementObjectForPortCoder: *XOG*

- (id)**replacementObjectForPortCoder:**(NSPortCoder*) *aCoder*

Returns the actual object to be encoded for sending over the network on a Distributed Objects connection.

The default implementation returns self if the receiver is being sent *bycopy* and returns a proxy otherwise.

Subclasses may override this method to change this behavior, eg. to ensure that they are always copied.

respondsToSelector: *XOG*

- (BOOL)**respondsToSelector:**(SEL) *aSelector*

Returns a flag to say if the receiver will respond to the specified selector. This ignores situations where a subclass implements -forwardInvocation: to respond to selectors not normally handled... in these cases the subclass may override this method to handle it.

If given a null selector, raises NSInvalidArgumentException when in MacOS-X compatibility more, or returns NO otherwise.

retain *XOG*

- (id)**retain**

Increments the reference count and returns the receiver.

The default implementation does this by calling NSIncrementExtraRefCount()

retainCount *XOG*

- (unsigned)**retainCount**

Returns the reference count for the receiver. Each instance has an implicit reference count of 1, and has an 'extra refrence count' returned by the NSExtraRefCount() function, so the value returned by this method is always greater than zero.

By convention, objects which should (or can) never be deallocated return the maximum unsigned integer value.

self *XOG*

- (id)self

Returns the reciever.

superclass *XOG*

- (Class)superclass

Returns the super class from which the receviers class was derived.

zone *XOG*

- (NSZone*)zone

Returns the memory allocation zone in which the receiver is located.

NSPipe *XOG*

Inherits From: NSObject
Declared in: Foundation/NSFileHandle.h

Description

The NSPipe provides an encapsulation of the UNIX concept of pipe. With NSPipe, it is possible to redirect the standard input or standard output.

Class Methods

pipe *XOG*

`+ (id)`**pipe**

Returns a newly allocated and initialized NSPipe object that has been sent an autorelease message.

Instance Methods

fileHandleForReading *XOG*

`- (NSFileHandle*)`**fileHandleForReading**

Description forthcoming.

fileHandleForWriting *XOG*

`- (NSFileHandle*)`**fileHandleForWriting**

Description forthcoming.

NSPort *XOG*

Inherits From:	NSObject
Conforms To:	NSCoding NSCopying
Declared in:	Foundation/NSPort.h

Description

Description forthcoming.

Class Methods

port *XOG*

`+ (NSPort*)`**port**

Description forthcoming.

portWithMachPort: *XOG*

`+ (NSPort*)`**portWithMachPort:** `(int)` *machPort*

Description forthcoming.

Instance Methods

addConnection:toRunLoop:forMode: *X¬OG*

`- (void)`**addConnection:**`(NSConnection*)` *aConnection*
toRunLoop:`(NSRunLoop*)` *aLoop*
forMode:`(NSString*)` *aMode*

Description forthcoming.

delegate *XOG*

- `(id)`**delegate**

Description forthcoming.

init *XOG*

- `(id)`**init**

Description forthcoming.

initWithMachPort: *XOG*

- `(id)`**initWithMachPort:**`(int)` *machPort*

Description forthcoming.

invalidate *XOG*

- `(void)`**invalidate**

Description forthcoming.

isValid *XOG*

- `(BOOL)`**isValid**

Description forthcoming.

machPort *XOG*

- `(int)`**machPort**

Description forthcoming.

removeConnection:fromRunLoop:forMode: *X¬OG*

- (void)**removeConnection:**(NSConnection*) *aConnection*
fromRunLoop:(NSRunLoop*) *aLoop*
forMode:(NSString*) *aMode*

Description forthcoming.

reservedSpaceLength

X¬OG

- (unsigned)**reservedSpaceLength**

Description forthcoming.

sendBeforeDate:components:from:reserved:

X¬OG

- (BOOL)**sendBeforeDate:**(NSDate*) *when*
components:(NSMutableArray*) *components*
from:(NSPort*) *receivingPort*
reserved:(unsigned) *length*

Description forthcoming.

sendBeforeDate:msgid:components:from:reserved:

X¬OG

- (BOOL)**sendBeforeDate:**(NSDate*) *when*
msgid:(int) *msgid*
components:(NSMutableArray*) *components*
from:(NSPort*) *receivingPort*
reserved:(unsigned) *length*

Description forthcoming.

setDelegate:

XOG

- (void)**setDelegate:**(id) *anObject*

Description forthcoming.

NSPortCoder *XOG*

Inherits From: NSCoder : NSObject
Declared in: Foundation/NSPortCoder.h

Description

Description forthcoming.

Class Methods

portCoderWithReceivePort:sendPort:components: *XOG*

`+ (NSPortCoder*)` **portCoderWithReceivePort:** `(NSPort*)` *recv*
sendPort: `(NSPort*)` *send*
components: `(NSArray*)` *comp*

Description forthcoming.

Instance Methods

connection *XOG*

`- (NSConnection*)` **connection**

Description forthcoming.

decodePortObject *XOG*

`- (NSPort*)` **decodePortObject**

Description forthcoming.

dispatch *XOG*

- (void)dispatch

Description forthcoming.

encodePortObject: *XOG*

- (void)encodePortObject:(NSPort*) *aPort*

Description forthcoming.

initWithReceivePort:sendPort:components: *XOG*

- (id)initWithReceivePort:(NSPort*) *recv*
sendPort:(NSPort*) *send*
components:(NSArray*) *comp*

Description forthcoming.

isBycopy *XOG*

- (BOOL)isBycopy

Description forthcoming.

isByref *XOG*

- (BOOL)isByref

Description forthcoming.

NSPortMessage *XOG*

Inherits From: NSObject
Declared in: Foundation/NSPortMessage.h

Description

Description forthcoming.

Instance Methods

components *XOG*

- (NSArray*)**components**

Description forthcoming.

initWithMachMessage: *XOG*

- (id)**initWithMachMessage:**(void*) *buffer*

Description forthcoming.

initWithSendPort:receivePort:components: *XOG*

- (id)**initWithSendPort:**(NSPort*) *aPort*
receivePort:(NSPort*) *anotherPort*
components:(NSArray*) *items*

Description forthcoming.

msgid *XOG*

- (unsigned)**msgid**

Description forthcoming.

receivePort *XOG*

- `(NSPort*)`**receivePort**

Description forthcoming.

sendBeforeDate: *XOG*

- `(BOOL)`**sendBeforeDate:**`(NSDate*)` *when*

Description forthcoming.

sendPort *XOG*

- `(NSPort*)`**sendPort**

Description forthcoming.

setMsgid: *XOG*

- `(void)`**setMsgid:**`(unsigned)` *anId*

Description forthcoming.

NSPortNameServer *XOG*

Inherits From: NSObject
Declared in: Foundation/NSPortNameServer.h

Description

Description forthcoming.

Class Methods

systemDefaultPortNameServer *XOG*

`+ (id)` **systemDefaultPortNameServer**

Description forthcoming.

Instance Methods

portForName: *XOG*

`- (NSPort*)` **portForName:** `(NSString*)` *name*

Description forthcoming.

portForName:onHost: *XOG*

`- (NSPort*)` **portForName:** `(NSString*)` *name*
onHost: `(NSString*)` *host*

Description forthcoming.

registerPort:forName: *XOG*

- (BOOL)**registerPort:**(NSPort*) *port*
forName:(NSString*) *name*

Description forthcoming.

removePortForName: *XOG*

- (BOOL)**removePortForName:**(NSString*) *name*

Description forthcoming.

NSProcessInfo *XOG*

Inherits From: NSObject
Declared in: Foundation/NSProcessInfo.h

Description

Description forthcoming.

Class Methods

processInfo *XOG*

`+ (NSProcessInfo*)` **processInfo**

Returns the shared NSProcessInfo object for the current process.

Instance Methods

arguments *XOG*

`- (NSArray*)` **arguments**

Returns an array containing the arguments supplied to start this process. NB. In GNUstep, any arguments of the form –GNU-Debug=... are *not* included in this array... they are part of the debug mechanism, and are hidden so that setting debug variables will not effect the normal operation of the program.

environment *XOG*

`- (NSDictionary*)` **environment**

Returns a dictionary giving the environment variables which were provided for the process to use.

globallyUniqueString *XOG*

- `(NSString*)` **globallyUniqueString**

Returns a string which may be used as a globally unique identifier.
The string contains the host name, the process ID, a timestamp and a counter.
The first three values identify the process in which the string is generated, while the fourth ensures that multiple strings generated within the same process are unique.

hostName *XOG*

- `(NSString*)` **hostName**

Returns the name of the machine on which this process is running.

operatingSystem *X¬OG*

- `(unsigned int)` **operatingSystem**

Return a number representing the operating system type.
The known types are listed in the header file, but not all of the listed types are actually implemented... some are present for MacOS-X compatibility only.

- NSWindowsNTOperatingSystem - used for windows NT, 2000, XP
- NSWindows95OperatingSystem - probably never to be implemented
- NSSolarisOperatingSystem - not yet recognised
- NSHPUXOperatingSystem - not implemented
- NSMACHOperatingSystem - perhaps the HURD in future?
- NSSunOSOperatingSystem - probably never to be implemented
- NSOSF1OperatingSystem - probably never to be implemented
- NSGNULinuxOperatingSystem - the GNUstep 'standard'
- NSBSDOperatingSystem - BSD derived operating systems

operatingSystemName

X¬OG

- (NSString*)operatingSystemName

Returns the name of the operating system in use.

processIdentifier

X¬OG

- (int)processIdentifier

Returns the process identifier number which identifies this process on this machine.

processName

XOG

- (NSString*)processName

Returns the process name for this process. This may have been set using the -setProcessName: method, or may be the default process name (the file name of the binary being executed).

setProcessName:

XOG

- (void)setProcessName:(NSString*) *newName*

Change the name of the current process to *newName*.

NSProtocolChecker *XOG*

Inherits From: NSObject
Declared in: Foundation/NSProtocolChecker.h

Description

Description forthcoming.

Class Methods

protocolCheckerWithTarget:protocol: *XOG*

`+ (id)` **protocolCheckerWithTarget:** `(NSObject*)` *anObject*
protocol: `(Protocol*)` *aProtocol*

Description forthcoming.

Instance Methods

forwardInvocation: *XOG*

`- (void)` **forwardInvocation:** `(NSInvocation*)` *anInvocation*

Description forthcoming.

initWithTarget:protocol: *XOG*

`- (id)` **initWithTarget:** `(NSObject*)` *anObject*
protocol: `(Protocol*)` *aProtocol*

Description forthcoming.

methodDescriptionForSelector: *XOG*

- (struct objc_method_description*)**methodDescriptionForSelector:**(SEL) *aSelector*

Description forthcoming.

protocol *XOG*

- (Protocol*)**protocol**

Description forthcoming.

target *XOG*

- (NSObject*)**target**

Description forthcoming.

NSProxy *XOG*

Inherits From: None, is a root class
Conforms To: NSObject
Declared in: Foundation/NSProxy.h

Description

The NSProxy class provides a basic implementation of a class whose instances are used to *stand in* for other objects.

The class provides the most basic methods of NSObject, and expects messages for other methods to be forwarded to the *real* object represented by the proxy. You must subclass NSProxy to implement -forwardInvocation: to these *real* objects.

Class Methods

alloc *XOG*

`+ (id)` **alloc**

Allocates and returns an NSProxy instance in the default zone.

allocWithZone: *XOG*

`+ (id)` **allocWithZone:** `(NSZone*)` *z*

Allocates and returns an NSProxy instance in the specified zone *z*.

autorelease *XOG*

`+ (id)` **autorelease**

Returns the receiver

class *XOG*

`+ (Class)`**class**

Returns the receiver

description *XOG*

`+ (NSString*)`**description**

Returns a string describing the receiver.

isKindOfClass: *XOG*

`+ (BOOL)`**isKindOfClass:** `(Class)` *aClass*

Returns `NO`... the NSProxy class cannot be an instance of any class.

isMemberOfClass: *XOG*

`+ (BOOL)`**isMemberOfClass:** `(Class)` *aClass*

Returns `YES` if *aClass* is identical to the receiver, `NO` otherwise.

load *XOG*

`+ (void)`**load**

A dummy method...

release *XOG*

`+ (void)`**release**

A dummy method to ensure that the class can safely be held in containers.

respondsToSelector: *XOG*

+ (BOOL)**respondsToSelector:** (SEL) *aSelector*

Returns YES if the receiver responds to *aSelector*, NO otherwise.

retain *XOG*

+ (id)**retain**

Returns the receiver.

retainCount *XOG*

+ (unsigned int)**retainCount**

Returns the maximum unsigned integer value.

Instance Methods

autorelease *XOG*

- (id)**autorelease**

Adds the receiver to the current autorelease pool and returns self.

class *XOG*

- (Class)**class**

Returns the class of the receiver.

conformsToProtocol: *XOG*

- (BOOL)**conformsToProtocol:**(Protocol*) *aProtocol*

Calls the -forwardInvocation: method to determine if the 'real' object referred to by the proxy conforms to *aProtocol*. Returns the result.
NB. The default operation of -forwardInvocation: is to raise an exception.

dealloc *XOG*

- `(void)`**dealloc**

Frees the memory used by the receiver.

description *XOG*

- `(NSString*)`**description**

Returns a text descrioption of the receiver.

forwardInvocation: *XOG*

- `(void)`**forwardInvocation:**`(NSInvocation*)` *anInvocation*

Raises an NSInvalidArgumentException

hash *XOG*

- `(unsigned int)`**hash**

Returns the address of the receiver... so it can be stored in a dictionary.

init *XOG*

- `(id)`**init**

Initialises the receiver and returns the resulting instance.

isEqual: *XOG*

- `(BOOL)`**isEqual:**`(id)` *anObject*

Tests for pointer equality with *anObject*

isKindOfClass: *XOG*

- (BOOL)**isKindOfClass:**(Class) *aClass*

Calls the -forwardInvocation: method to determine if the 'real' object referred to by the proxy is an instance of the specified class. Returns the result.
NB. The default operation of -forwardInvocation: is to raise an exception.

isMemberOfClass: *XOG*

- (BOOL)**isMemberOfClass:**(Class) *aClass*

Calls the -forwardInvocation: method to determine if the 'real' object referred to by the proxy is an instance of the specified class. Returns the result.
NB. The default operation of -forwardInvocation: is to raise an exception.

isProxy *XOG*

- (BOOL)**isProxy**

Returns YES

methodSignatureForSelector: *XOG*

- (NSMethodSignature*)**methodSignatureForSelector:**(SEL) *aSelector*

If we respond to the method directly, create and return a method signature. Otherwise raise an exception.

release *XOG*

- (void)**release**

Decrement the retain count for the receiver... deallocate if it would become negative.

respondsToSelector: *XOG*

- (BOOL)**respondsToSelector:**(SEL) *aSelector*

If we respond to the method directly, return YES, otherwise forward this request to the object we are acting as a proxy for.

retain *XOG*

- `(id)`**retain**

Increment the retain count for the receiver.

retainCount *XOG*

- `(unsigned int)`**retainCount**

Return the retain count for the receiver.

self *XOG*

- `(id)`**self**

Returns the receiver.

superclass *XOG*

- `(Class)`**superclass**

Returns the superclass of the receivers class.

zone *XOG*

- `(NSZone*)`**zone**

Returns the zone in which the receiver was allocated.

NSRecursiveLock *XOG*

Inherits From:	NSObject
Conforms To:	NSLocking GCFinalization
Declared in:	Foundation/NSLock.h

Description

See NSLock for more information about what a lock is. A recursive lock extends NSLock in that you can lock a recursive lock multiple times. Each lock must be balanced by a cooresponding unlock, and the lock is not released for another thread to aquire until the last unlock call is made (corresponding to the first lock message).

Instance Methods

lock *XOG*

- (void)lock

Description forthcoming.

lockBeforeDate: *XOG*

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

Attempts to aquire a lock before the date *limit* passes. It returns `YES` if it can. It returns `NO` if it cannot (but it waits until the time *limit* is up before returning `NO`).

tryLock *XOG*

- (BOOL)tryLock

Attempts to aquire a lock, but returns `NO` immediately if the lock cannot be aquired. It returns `YES` if the lock is aquired. Can be called multiple times to make nested locks.

unlock *XOG*

- `(void)`**unlock**

Description forthcoming.

NSRunLoop *XOG*

Inherits From: NSObject
Conforms To: GCFinalization
Declared in: Foundation/NSRunLoop.h

Description

Description forthcoming.

Class Methods

currentRunLoop *XOG*

`+ (NSRunLoop*)`**currentRunLoop**

Description forthcoming.

Instance Methods

acceptInputForMode:beforeDate: *XOG*

`- (void)`**acceptInputForMode:**`(NSString*)` *mode*
beforeDate:`(NSDate*)` *limit_date*

Listen to input sources.
If *limit_date* is `nil` or in the past, then don't wait; just poll inputs and return, otherwise block until input is available or until the earliest limit date has passed (whichever comes first).
If the supplied *mode* is `nil`, uses NSDefaultRunLoopMode.

addTimer:forMode: *XOG*

`- (void)`**addTimer:**`(NSTimer*)` *timer*
forMode:`(NSString*)` *mode*

Adds a *timer* to the loop in the specified *mode*.
Timers are removed automatically when they are invalid.

currentMode *XOG*

- (NSString*)**currentMode**

Returns the current mode of this runloop. If the runloop is not running then this method returns `nil`.

limitDateForMode: *XOG*

- (NSDate*)**limitDateForMode:**(NSString*) *mode*

Fire appropriate timers and determine the earliest time that anything watched for becomes useless.

run *XOG*

- (void)**run**

Description forthcoming.

runMode:beforeDate: *XOG*

- (BOOL)**runMode:**(NSString*) *mode*
  beforeDate:(NSDate*) *date*

Calls -acceptInputForMode:beforeDate: to run the loop once.
The specified *date* may be `nil`... in which case the loop runs until the first input or timeout.
If the limit dates for all of mode's input sources have passed, returns `NO` without running the loop, otherwise returns `YES`.

runUntilDate: *XOG*

- (void)**runUntilDate:**(NSDate*) *date*

Description forthcoming.

NSScanner *XOG*

Inherits From: NSObject
Conforms To: NSCopying
Declared in: Foundation/NSScanner.h

Description

The NSScanner class cluster (currently a single class in GNUstep) provides a mechanism to parse the contents of a string into number and string values by making a sequence of scan operations to step through the string retrieving successive items.

You can tell the scanner whether its scanning is supposed to be case sensitive or not, and you can specify a set of characters to be skipped before each scanning operation (by default, whitespace and newlines).

Class Methods

localizedScannerWithString: *XOG*

`+ (id)localizedScannerWithString: (NSString*)` *aString*

Returns an NSScanner instance set up to scan *aString* (using -initWithString: and with a locale set the default locale (using -setLocale:

scannerWithString: *XOG*

`+ (id)scannerWithString: (NSString*)` *aString*

Description forthcoming.

Instance Methods

caseSensitive *XOG*

- (BOOL)caseSensitive

If the scanner is set to be case-sensitive in its scanning of the string (other than characters to be skipped), this method returns YES, otherwise it returns NO.
The default is for a scanner to *not* be case sensitive.

charactersToBeSkipped *XOG*

- (NSCharacterSet*)charactersToBeSkipped

Returns a set of characters containing those characters that the scanner ignores when starting any scan operation. Once a character not in this set has been encountered during an operation, skipping is finished, and any further characters from this set that are found are scanned normally.
The default for this is the whitespaceAndNewlineCharacterSet.

initWithString: *XOG*

- (id)initWithString:(NSString*) *aString*

Initialises the scanner to scan *aString*. The GNUstep implementation may make an internal copy of the original string - so it is not safe to assume that if you modify a mutable string that you initialised a scanner with, the changes will be visible to the scanner.
Returns the scanner object.

isAtEnd *XOG*

- (BOOL)isAtEnd

Returns YES if no more characters remain to be scanned.
Returns YES if all characters remaining to be scanned are to be skipped.
Returns NO if there are characters left to scan.

locale *XOG*

- (NSDictionary*)**locale**

Returns the locale set for the scanner, or `nil` if no locale has been set. A scanner uses it's locale to alter the way it handles scanning - it uses the NSDecimalSeparator value for scanning numbers.

scanCharactersFromSet:intoString: *XOG*

- (BOOL)**scanCharactersFromSet:**(NSCharacterSet*) *aSet*
intoString:(NSString**) *value*

After initial skipping (if any), this method scans any characters from *aSet*, terminating when a character not in the set is found.

Returns `YES` if any character is scanned, `NO` otherwise.

If *value* is not null, any character scanned are stored in a string returned in this location.

scanDecimal: *X¬OG*

- (BOOL)**scanDecimal:**(NSDecimal*) *value*

Not implemented.

scanDouble: *XOG*

- (BOOL)**scanDouble:**(double*) *value*

After initial skipping (if any), this method scans a double *value*, placing it in *doubleValue* if that is not null. Returns `YES` if anything is scanned, `NO` otherwise.

On overflow, HUGE_VAL or - HUGE_VAL is put into *doubleValue*

On underflow, 0.0 is put into *doubleValue*

Scans past any excess digits

scanFloat: *XOG*

- (BOOL)**scanFloat:**(float*) *value*

After initial skipping (if any), this method scans a float *value*, placing it in *floatValue* if that is not null. Returns `YES` if anything is scanned, `NO` otherwise.

On overflow, HUGE_VAL or - HUGE_VAL is put into *floatValue*

On underflow, 0.0 is put into *floatValue*

Scans past any excess digits

scanHexInt: *XOG*

- `(BOOL)`**scanHexInt:**`(unsigned int*)` *value*

After initial skipping (if any), this method scans a hexadecimal integer *value* (optionally prefixed by "0x" or "0X"), placing it in *intValue* if that is not null.
Returns `YES` if anything is scanned, `NO` otherwise.
On overflow, INT_MAX or INT_MIN is put into *intValue*
Scans past any excess digits

scanInt: *XOG*

- `(BOOL)`**scanInt:**`(int*)` *value*

After initial skipping (if any), this method scans a integer *value*, placing it in *intValue* if that is not null.
Returns `YES` if anything is scanned, `NO` otherwise.
On overflow, INT_MAX or INT_MIN is put into *intValue*
Scans past any excess digits

scanLocation *XOG*

- `(unsigned)`**scanLocation**

Returns the current position that the scanner has reached in scanning the string. This is the position at which the next scan operation will begin.

scanLongLong: *XOG*

- `(BOOL)`**scanLongLong:**`(long long*)` *value*

After initial skipping (if any), this method scans a long decimal integer *value* placing it in *longLongValue* if that is not null.
Returns `YES` if anything is scanned, `NO` otherwise.
On overflow, LONG_LONG_MAX or LONG_LONG_MIN is put into *longLongValue*
Scans past any excess digits

scanRadixUnsignedInt: $\neg O \neg XG$

- (BOOL)**scanRadixUnsignedInt:**(unsigned int*) *value*

After initial skipping (if any), this method scans an unsigned integer *value* placing it in *intValue* if that is not null. If the number begins with "0x" or "0X" it is treated as hexadecimal, otherwise if the number begins with "0" it is treated as octal, otherwise the number is treated as decimal.

Returns YES if anything is scanned, NO otherwise.

On overflow, INT_MAX or INT_MIN is put into *intValue*

Scans past any excess digits

scanString:intoString: XOG

- (BOOL)**scanString:**(NSString*) *string*
intoString:(NSString**) *value*

After initial skipping (if any), this method scans for *aString* and places the *string* ound in *stringValue* if that is not null.

Returns YES if anything is scanned, NO otherwise.

scanUpToCharactersFromSet:intoString: XOG

- (BOOL)**scanUpToCharactersFromSet:**(NSCharacterSet*) *aSet*
intoString:(NSString**) *value*

After initial skipping (if any), this method scans characters until it finds one in *set*. The scanned characters are placed in *stringValue* if that is not null.

Returns YES if anything is scanned, NO otherwise.

scanUpToString:intoString: XOG

- (BOOL)**scanUpToString:**(NSString*) *string*
intoString:(NSString**) *value*

After initial skipping (if any), this method scans characters until it finds *aString*. The scanned characters are placed in *stringValue* if that is not null. If *aString* is not found, all the characters up to the end of the scanned *string* will be returned.

Returns YES if anything is scanned, NO otherwise.

setCaseSensitive: *XOG*

- `(void)`**setCaseSensitive:**`(BOOL)` *flag*

Sets the case sensitivity of the scanner.
Case sensitivity governs matching of characters being scanned, but does not effect the characters in the set to be skipped.
The default is for a scanner to *not* be case sensitive.

setCharactersToBeSkipped: *XOG*

- `(void)`**setCharactersToBeSkipped:**`(NSCharacterSet*)` *aSet*

Sets the set of characters that the scanner will skip over at the start of each scanning operation to be *skipSet*. Skipping is performed by literal character matchins - the case sensitivity of the scanner does not effect it. If this is set to `nil`, no skipping is done.
The default for this is the whitespaceAndNewlineCharacterSet.

setLocale: *XOG*

- `(void)`**setLocale:**`(NSDictionary*)` *localeDictionary*

This method sets the locale used by the scanner to *aLocale*. The locale may be set to `nil`.

setScanLocation: *XOG*

- `(void)`**setScanLocation:**`(unsigned int)` *anIndex*

This method sets the location in the scanned string at which the next scan operation begins. Raises an NSRangeException if index is beyond the end of the scanned string.

string *XOG*

- `(NSString*)`**string**

Returns the string being scanned.

NSSet *XOG*

Inherits From: NSObject
Conforms To: NSCoding
NSCopying
NSMutableCopying
Declared in: Foundation/NSSet.h

Description

Description forthcoming.

Class Methods

set *XOG*

`+ (id)set`

Description forthcoming.

setWithArray: *XOG*

`+ (id)setWithArray: (NSArray*)` *objects*

Description forthcoming.

setWithObject: *XOG*

`+ (id)setWithObject: (id)` *anObject*

Description forthcoming.

setWithObjects: *XOG*

`+ (id)setWithObjects: (id)` *firstObject*

Description forthcoming.

setWithObjects:count: $X\neg OG$

`+ (id)`**setWithObjects:** `(id*)` *objects*
count: `(unsigned)` *count*

Description forthcoming.

setWithSet: XOG

`+ (id)`**setWithSet:** `(NSSet*)` *aSet*

Description forthcoming.

Instance Methods

allObjects XOG

`- (NSArray*)`**allObjects**

Description forthcoming.

anyObject XOG

`- (id)`**anyObject**

Description forthcoming.

containsObject: XOG

`- (BOOL)`**containsObject:**`(id)` *anObject*

Description forthcoming.

count XOG

`- (unsigned)`**count**

Returns the number of objects stored in the set.

description *XOG*

- `(NSString*)`**description**

Description forthcoming.

descriptionWithLocale: *XOG*

- `(NSString*)`**descriptionWithLocale:**`(NSDictionary*)` *locale*

Description forthcoming.

initWithArray: *XOG*

- `(id)`**initWithArray:**`(NSArray*)` *other*

Initialises a newly allocated set by adding all the objects in the supplied array to the set.

initWithObjects: *XOG*

- `(id)`**initWithObjects:**`(id)` *firstObject*

Description forthcoming.

initWithObjects:count: *XOG*

- `(id)`**initWithObjects:**`(id*)` *objects*
count:`(unsigned)` *count*

Description forthcoming.

initWithSet: *XOG*

- `(id)`**initWithSet:**`(NSSet*)` *other*

Description forthcoming.

initWithSet:copyItems: *XOG*

- (id)**initWithSet:**(NSSet*) *other*
copyItems:(BOOL) *flag*

Initialises a newly allocated set by adding all the objects in the supplied set.

intersectsSet: *XOG*

- (BOOL)**intersectsSet:**(NSSet*) *otherSet*

Description forthcoming.

isEqualToSet: *XOG*

- (BOOL)**isEqualToSet:**(NSSet*) *other*

Description forthcoming.

isSubsetOfSet: *XOG*

- (BOOL)**isSubsetOfSet:**(NSSet*) *otherSet*

Description forthcoming.

makeObjectsPerform: *XOG*

- (void)**makeObjectsPerform:**(SEL) *aSelector*

Description forthcoming.

makeObjectsPerform:withObject: *XOG*

- (void)**makeObjectsPerform:**(SEL) *aSelector*
withObject:(id) *argument*

Description forthcoming.

makeObjectsPerformSelector: *X¬OG*

- `(void)`**makeObjectsPerformSelector:**`(SEL)` *aSelector*

Description forthcoming.

makeObjectsPerformSelector:withObject: *X¬OG*

- `(void)`**makeObjectsPerformSelector:**`(SEL)` *aSelector*
withObject:`(id)` *argument*

Description forthcoming.

member: *XOG*

- `(id)`**member:**`(id)` *anObject*

Description forthcoming.

objectEnumerator *XOG*

- `(NSEnumerator*)`**objectEnumerator**

Description forthcoming.

NSSocketPort

X¬OG

Inherits From: NSPort : NSObject
Conforms To: NSCoding
NSCopying
Declared in: Foundation/NSPort.h

Description

`NSSocketPort` on MacOS X(tm) is a concrete subclass of NSPort which implements Distributed Objects communication between hosts on a network. However, the GNUstep distributed objects system's NSPort class uses TCP/IP for all of its communication. The GNUstep `NSSocketPort`, then, is useful as a convenient method to create and encapsulate BSD sockets:

```
int fileDesc;
NSFileHandle *smtpHandle;
  fileDesc = [[NSSocketPort alloc] initRemoteWithTCPPort: 25
                                   host: @"mail.example.com"];
smtpHandle = [[NSFileHandle alloc] initWithFileDescriptor:
fileDesc];
```

Instance Methods

address

X¬OG

- (NSData*)address

Return the protocol family-specific socket address in an NSData object.

init

X¬OG

- (id)init

Initialize the receiver with a local socket to accept TCP connections on a non-conflicting port number chosen by the system.

initRemoteWithProtocolFamily:socketType:protocol:address: *X¬OG*

```
- (id)initRemoteWithProtocolFamily:(int) family
                        socketType:(int) type
                          protocol:(int) protocol
                           address:(NSData*) addrData
```

Initialize the receiver to connect to a remote socket of *type* with *protocol* from the *protocol family family*. The *addrData* should contain a copy of the *protocol* family-specific address data in an NSData object.

initRemoteWithTCPPort:host: *X¬OG*

```
- (id)initRemoteWithTCPPort:(unsigned short) portNumber
                       host:(NSString*) hostname
```

Initialize the receiver to connect to a remote TCP socket on port *portNumber* of host *hostname*. The receiver delays initiation of the connection until it has data to send.

NOTE: This method currently does not support IPv6 connections.

initWithProtocolFamily:socketType:protocol:address: *X¬OG*

```
- (id)initWithProtocolFamily:(int) family
                  socketType:(int) type
                    protocol:(int) protocol
                     address:(NSData*) addrData
```

Initialize the receiver as a local socket to accept connections on a socket of *type* with the *protocol* from the *protocol family family*. The *addrData* should contain a copy of the *protocol* family-specific address data in an NSData object.

initWithProtocolFamily:socketType:protocol:socket: *X¬OG*

```
- (id)initWithProtocolFamily:(int) family
                  socketType:(int) type
                    protocol:(int) protocol
                      socket:(NSSocketNativeHandle) socket
```

Initialize the receiver with *socket*, the platform-native handle to a previously initialized listen-mode *socket* of *type type* with the *protocol protocol* from the *protocol family family*.

The receiver will close the *socket* upon deallocation.

initWithTCPPort: $X\neg OG$

- (id)initWithTCPPort:(unsigned short) *portNumber*

Initialize the receiver as a local socket to accept connections on TCP port *portNumber*. If *portNumber* is zero, the system will chose a non-conflicting port number.
NOTE: This method currently does not support IPv6 connections.

protocol $X\neg OG$

- (int)protocol

Return the socket protocol.

protocolFamily $X\neg OG$

- (int)protocolFamily

Return the socket protocol family.

socket $X\neg OG$

- (NSSocketNativeHandle)socket

Return the platform-native socket handle.

socketType $X\neg OG$

- (int)socketType

Return the socket type.

NSString *XOG*

Inherits From: NSObject
Conforms To: NSCoding
NSCopying
NSMutableCopying
Declared in: Foundation/NSString.h

Description

NSString objects represent an immutable string of characters. NSString itself is an abstract class which provides factory methods to generate objects of unspecified subclasses.

A constant NSString can be created using the following syntax: `@"..."`, where the contents of the quotes are the string, using only ASCII characters.

To create a concrete subclass of NSString, you must have your class inherit from NSString and override at least the two primitive methods - length and characterAtIndex:

In general the rule is that your subclass must override any initialiser that you want to use with it. The GNUstep implementation relaxes that to say that, you may override only the *designated initialiser* and the other initialisation methods should work.

Class Methods

availableStringEncodings *X¬OG*

`+ (NSStringEncoding*)`**availableStringEncodings**

Returns an array of all available string encodings, terminated by a null value.

constantStringClass *¬O¬XG*

`+ (Class)`**constantStringClass**

Return the class used to store constant strings (those ascii strings placed in the source code using the @"this is a string" syntax).

Use this method to obtain the constant string class rather than using the obsolete name *NXConstantString* in your code... with more recent compiler versions the name of this class is variable (and will automatically be changed by GNUstep to avoid conflicts with the default implementation in the Objective-C runtime library).

defaultCStringEncoding

XOG

`+ (NSStringEncoding)` **defaultCStringEncoding**

Returns the encoding used for any method accepting a C string. This value is determined automatically from the programs environment and cannot be changed programmatically.

You should *NOT* override this method in an attempt to change the encoding being used... it won't work.

In GNUstep, this encoding is determined by the initial value of the `GNUSTEP_STRING_ENC` environment variable. If this is not defined, `NSISOLatin1StringEncoding` is assumed.

localizedNameOfStringEncoding:

X¬OG

`+ (NSString*)` **localizedNameOfStringEncoding:** `(NSStringEncoding)` *encoding*

Returns the localized name of the *encoding* specified.

localizedStringWithFormat:

X¬OG

`+ (NSString*)` **localizedStringWithFormat:** `(NSString*)` *format*

Description forthcoming.

pathWithComponents:

X¬OG

`+ (NSString*)` **pathWithComponents:** `(NSArray*)` *components*

Concatenates the strings in the *components* array placing a path separator between each one and returns the result.

string

XOG

+ (id)**string**

Description forthcoming.

stringWithCString: *XOG*

+ (id)**stringWithCString:** (const char*) *byteString*

Description forthcoming.

stringWithCString:length: *XOG*

+ (id)**stringWithCString:** (const char*) *byteString*
length: (unsigned int) *length*

Description forthcoming.

stringWithCharacters:length: *XOG*

+ (id)**stringWithCharacters:** (const unichar*) *chars*
length: (unsigned int) *length*

Description forthcoming.

stringWithContentsOfFile: *XOG*

+ (id)**stringWithContentsOfFile:** (NSString*) *path*

Description forthcoming.

stringWithContentsOfURL: *X¬OG*

+ (id)**stringWithContentsOfURL:** (NSURL*) *url*

Description forthcoming.

stringWithFormat: *XOG*

+ (id)**stringWithFormat:** (NSString*) *format*

Description forthcoming.

stringWithFormat:arguments:

X¬OG

+ (id)**stringWithFormat:** (NSString*) *format*
arguments: (va_list) *argList*

Description forthcoming.

stringWithString:

X¬OG

+ (id)**stringWithString:** (NSString*) *aString*

Description forthcoming.

stringWithUTF8String:

X¬OG

+ (id)**stringWithUTF8String:** (const char*) *bytes*

Description forthcoming.

Instance Methods

UTF8String

X¬OG

- (const char*)**UTF8String**

Description forthcoming.

_baseLength

XOG

- (int)**_baseLength**

Warning the underscore at the start of the name of this method indicates that it is private, for internal use only, and you should not use the method in your code.

boolValue

$\neg O \neg XG$

- `(BOOL)`**boolValue**

If the string consists of the words 'true' or 'yes' (case insensitive) or begins with a non-zero numeric value, return `YES`, otherwise return `NO`.

cString

XOG

- `(const char*)`**cString**

Returns a pointer to a null terminated string of 8-bit characters in the default encoding. The memory pointed to is not owned by the caller, so the caller must copy its contents to keep it.

cStringLength

XOG

- `(unsigned int)`**cStringLength**

Description forthcoming.

canBeConvertedToEncoding:

XOG

- `(BOOL)`**canBeConvertedToEncoding:**`(NSStringEncoding)` *encoding*

Description forthcoming.

capitalizedString

XOG

- `(NSString*)`**capitalizedString**

Description forthcoming.

caseInsensitiveCompare:

$X \neg OG$

- `(NSComparisonResult)`**caseInsensitiveCompare:**`(NSString*)` *aString*

Description forthcoming.

characterAtIndex: *XOG*

- (unichar)**characterAtIndex:**(unsigned int) *index*

Description forthcoming.

commonPrefixWithString:options: *XOG*

- (NSString*)**commonPrefixWithString:**(NSString*) *aString*
options:(unsigned int) *mask*

Description forthcoming.

compare: *XOG*

- (NSComparisonResult)**compare:**(NSString*) *aString*

Description forthcoming.

compare:options: *XOG*

- (NSComparisonResult)**compare:**(NSString*) *aString*
options:(unsigned int) *mask*

Description forthcoming.

compare:options:range: *XOG*

- (NSComparisonResult)**compare:**(NSString*) *aString*
options:(unsigned int) *mask*
range:(NSRange) *aRange*

Description forthcoming.

compare:options:range:locale: *X¬OG*

- (NSComparisonResult)**compare:**(NSString*) *string*
options:(unsigned int) *mask*
range:(NSRange) *compareRange*
locale:(NSDictionary*) *dict*

Description forthcoming.

completePathIntoString:caseSensitive:matchesIntoArray:filterTypes: *XOG*

- (unsigned int)**completePathIntoString:**(NSString**) *outputName*
caseSensitive:(BOOL) *flag*
matchesIntoArray:(NSArray**) *outputArray*
filterTypes:(NSArray*) *filterTypes*

Description forthcoming.

componentsSeparatedByString: *XOG*

- (NSArray*)**componentsSeparatedByString:**(NSString*) *separator*

Description forthcoming.

dataUsingEncoding: *XOG*

- (NSData*)**dataUsingEncoding:**(NSStringEncoding) *encoding*

Description forthcoming.

dataUsingEncoding:allowLossyConversion: *XOG*

- (NSData*)**dataUsingEncoding:**(NSStringEncoding) *encoding*
allowLossyConversion:(BOOL) *flag*

Description forthcoming.

description *XOG*

- (NSString*)**description**

Description forthcoming.

doubleValue *X¬OG*

- `(double)`**doubleValue**

Description forthcoming.

fastestEncoding *XOG*

- `(NSStringEncoding)`**fastestEncoding**

Description forthcoming.

fileSystemRepresentation *XOG*

- `(const char*)`**fileSystemRepresentation**

Description forthcoming.

floatValue *XOG*

- `(float)`**floatValue**

Description forthcoming.

getCString: *XOG*

- `(void)`**getCString:**`(char*)` *buffer*

Retrieve the contents of the receiver into the *buffer*.
The *buffer* must be large enought to contain the CString representation of the characters in the receiver, plus a null terminator which this method adds.

getCString:maxLength: *XOG*

- `(void)`**getCString:**`(char*)` *buffer*
maxLength:`(unsigned int)` *maxLength*

Retrieve up to *maxLength* characters from the receiver into the *buffer*.

The *buffer* must be at least *maxLength* characters long, so that it has room for the null terminator that this method adds.

getCString:maxLength:range:remainingRange: *XOG*

```
- (void)getCString:(char*) buffer
         maxLength:(unsigned int) maxLength
             range:(NSRange) aRange
    remainingRange:(NSRange*) leftoverRange
```

Description forthcoming.

getCharacters: *XOG*

```
- (void)getCharacters:(unichar*) buffer
```

Description forthcoming.

getCharacters:range: *XOG*

```
- (void)getCharacters:(unichar*) buffer
                range:(NSRange) aRange
```

Description forthcoming.

getFileSystemRepresentation:maxLength: *XOG*

```
- (BOOL)getFileSystemRepresentation:(char*) buffer
                          maxLength:(unsigned int) size
```

Description forthcoming.

getLineStart:end:contentsEnd:forRange: *X¬OG*

```
- (void)getLineStart:(unsigned int*) startIndex
                 end:(unsigned int*) lineEndIndex
         contentsEnd:(unsigned int*) contentsEndIndex
            forRange:(NSRange) aRange
```

Determines the smallest range of lines containing *aRange* and returns the locations in that range.

Lines are delimited by any of these character sequences, the longest (CRLF) sequence preferred.

- U+000A (linefeed)
- U+000D (carriage return)
- U+2028 (Unicode line separator)
- U+2029 (Unicode paragraph separator)
- U+000D U+000A (CRLF)

The index of the first character of the line at or before *aRange* is returned in *startIndex*.
The index of the first character of the next line after the line terminator is returned in endIndex.
The index of the last character before the line terminator is returned *contentsEndIndex*.
Raises an NSRangeException if the range is invalid, but permits the index arguments to be null pointers (in which case no value is returned in that argument).

hasPrefix: *XOG*

- `(BOOL)`**hasPrefix:**`(NSString*)` *aString*

Description forthcoming.

hasSuffix: *XOG*

- `(BOOL)`**hasSuffix:**`(NSString*)` *aString*

Description forthcoming.

hash *XOG*

- `(unsigned int)`**hash**

Return 28-bit hash value (in 32-bit integer). The top few bits are used for other purposes in a bitfield in the concrete string subclasses, so we must not use the full unsigned integer.

init *XOG*

- `(id)`**init**

Description forthcoming.

initWithCString: *XOG*

- `(id)`**initWithCString:**`(const char*)` *byteString*

Description forthcoming.

initWithCString:length: *XOG*

- `(id)`**initWithCString:**`(const char*)` *byteString*
length:`(unsigned int)` *length*

Description forthcoming.

initWithCStringNoCopy:length:freeWhenDone: *XOG*

- `(id)`**initWithCStringNoCopy:**`(char*)` *byteString*
length:`(unsigned int)` *length*
freeWhenDone:`(BOOL)` *flag*

Description forthcoming.

initWithCharacters:length: *XOG*

- `(id)`**initWithCharacters:**`(const unichar*)` *chars*
length:`(unsigned int)` *length*

Description forthcoming.

initWithCharactersNoCopy:length:freeWhenDone: *XOG*

- `(id)`**initWithCharactersNoCopy:**`(unichar*)` *chars*
length:`(unsigned int)` *length*
freeWhenDone:`(BOOL)` *flag*

This is the most basic initialiser for unicode strings. In the GNUstep implementation, your subclasses may override this initialiser in order to have all others function.

initWithContentsOfFile: *XOG*

- (id)initWithContentsOfFile:(NSString*) *path*

Initialises the receiver with the contents of the file at *path*.
Invokes [NSData -initWithContentsOfFile:] to read the file, then examines the data to infer its encoding type, and converts the data to a string using -initWithData:encodi
The encoding to use is determined as follows... if the data begins with the 16-bit unicode Byte Order Marker, then it is assumed to be unicode data in the appropriate ordering and converted as such.
If it begins with a UTF8 representation of the BOM, the UTF8 encoding is used.
Otherwise, the default C String encoding is used.
Releases the receiver and returns `nil` if the file could not be read and converted to a string.

initWithContentsOfURL: *X¬OG*

- (id)initWithContentsOfURL:(NSURL*) *url*

Description forthcoming.

initWithData:encoding: *XOG*

- (id)initWithData:(NSData*) *data*
encoding:(NSStringEncoding) *encoding*

Initialises the receiver with the supplied *data*, using the specified *encoding*.
For NSUnicodeStringEncoding and NSUTF8String *encoding*, a Byte Order Marker (if present at the start of the *data*) is removed automatically.
If the *data* can not be interpreted using the *encoding*, the receiver is released and `nil` is returned.

initWithFormat: *XOG*

- (id)initWithFormat:(NSString*) *format*

Invokes -initWithFormat:locale:arguments: with a `nil` locale.

initWithFormat:arguments: *XOG*

- (id)**initWithFormat:**(NSString*) *format*
  arguments:(va_list) *argList*

Invokes -initWithFormat:locale:arguments: with a nil locale.

initWithFormat:locale: *X¬OG*

- (id)**initWithFormat:**(NSString*) *format*
  locale:(NSDictionary*) *locale*

Invokes -initWithFormat:locale:arguments:

initWithFormat:locale:arguments: *X¬OG*

- (id)**initWithFormat:**(NSString*) *format*
  locale:(NSDictionary*) *locale*
  arguments:(va_list) *argList*

Initialises the string using the specified *format* and *locale* to *format* the following arguments.

initWithString: *XOG*

- (id)**initWithString:**(NSString*) *string*

Description forthcoming.

initWithUTF8String: *X¬OG*

- (id)**initWithUTF8String:**(const char*) *bytes*

Description forthcoming.

intValue *XOG*

- (int)**intValue**

Description forthcoming.

isAbsolutePath

X¬OG

- `(BOOL)`**isAbsolutePath**

Description forthcoming.

isEqual:

XOG

- `(BOOL)`**isEqual:**`(id)` *anObject*

Description forthcoming.

isEqualToString:

XOG

- `(BOOL)`**isEqualToString:**`(NSString*)` *aString*

Description forthcoming.

lastPathComponent

XOG

- `(NSString*)`**lastPathComponent**

Returns a string containing the last path component of the receiver.
The path component is the last non-empty substring delimited by the ends of the string or by path * separator (’/’) characters.
If the receiver is an empty string, it is simply returned.
If there are no non-empty substrings, the root string is returned.

length

XOG

- `(unsigned int)`**length**

Description forthcoming.

lineRangeForRange:

X¬OG

- (NSRange)**lineRangeForRange:**(NSRange) *aRange*

Determines the smallest range of lines containing *aRange* and returns the information as a range.
Calls -getLineStart:end:contentsEnd:forRange: to do the work.

localizedCaseInsensitiveCompare: *X¬OG*

- (NSComparisonResult)**localizedCaseInsensitiveCompare:**(NSString*) *string*

Description forthcoming.

localizedCompare: *X¬OG*

- (NSComparisonResult)**localizedCompare:**(NSString*) *string*

Description forthcoming.

lossyCString *X¬OG*

- (const char*)**lossyCString**

Description forthcoming.

lowercaseString *XOG*

- (NSString*)**lowercaseString**

Returns a copy of the receiver with all characters converted to lowercase.

pathComponents *X¬OG*

- (NSArray*)**pathComponents**

Returns the path components of the reciever separated into an array.
If the receiver begins with a '/' character then that is used as the first element in the array.
Empty components are removed.

pathExtension *XOG*

- (NSString*)pathExtension

Returns a new string containing the path extension of the receiver.
The path extension is a suffix on the last path component which starts with the extension separator (a '.') (for example.tiff is the pathExtension for /foo/bar.tiff).
Returns an empty string if no such extension exists.

propertyList *XOG*

- (id)propertyList

Attempts to interpret the receiver as a *property list* and returns the result. If the receiver does not contain a string representation of a *property list* then the method returns nil.
There are three readable *property list* storage formats - The binary format used by NSSerializer does not concern us here, but there are two 'human readable' formats, the *traditional* OpenStep format (which is extended in GNUstep) and the *XML* format.
The [NSArray -descriptionWithLocale:indent:] and [NSDictionary -descriptionWithLocale:in methods both generate strings containing traditional style *property lists*, but [NSArray -writeToFile:atomically:] and [NSDictionary -writeToFile:atomically:] generate either traditional or XML style *property lists* depending on the value of the GSMacOSXCompatible and NSWriteOldStylePropertyLists user defaults.
If GSMacOSXCompatible is YES then XML *property lists* are written unless NSWriteOldStylePropertyLists is also YES.
By default GNUstep writes old style data and always supports reading of either style.
The traditional format is more compact and more easily readable by people, but (without the GNUstep extensions) cannot represent date and number objects (except as strings). The XML format is more verbose and less readable, but can be fed into modern XML tools and thus used to pass data to non-OpenStep applications more readily.
The traditional format is strictly ascii encoded, with any unicode characters represented by escape sequences. The XML format is encoded as UTF8 data.
Both the traditional format and the XML format permit comments to be placed in *property list* documents. In traditional format the comment notations used in ObjectiveC programming are supported, while in XML format, the standard SGML comment sequences are used. A *property list* may only be one of the following classes -

NSArray An array which is either empty or contains only *property list* objects.

An array is delimited by round brackets and its contents are comma *separated* (there is no comma after the last array element).

```
( "one", "two", "three" )
```

In XML format, an array is an element whose name is `array` and whose content is the array content.

```
<array><string>one</string><string>two</string><string>three</
```

NSData An array is represented as a series of pairs of hexadecimal characters (each pair representing a byte of data) enclosed in angle brackets. Spaces are ignored).

```
< 54637374 696D67 >
```

In XML format, a data object is an element whose name is `data` and whose content is a stream of base64 encoded bytes.

NSDate Date objects were not traditionally allowed in *property lists* but were added when the XML format was intoroduced. GNUstep provides an extension to the traditional *property list* format to support date objects, but older code will not read *property lists* containing this extension.

This format consists of an asterisk follwed by the letter 'D' then a date/time in YYYY-MM-DD HH:MM:SS +/-ZZZZ format, all enclosed within angle brackets.

```
<*D2002-03-22 11:30:00 +0100>
```

In XML format, a date object is an element whose name is `date` and whose content is a date in the above format.

```
<date>2002-03-22 11:30:00 +0100</date>
```

NSDictionary A dictionary which is either empty or contains only *string* keys and *property list* objects.

A dictionary is delimited by curly brackets and its contents are semicolon *terminated* (there is a semicolon after each value). Each item in the dictionary is a key/value pair with an equals sign after the key and before the value.

```
{
  "key1" = "value1";
}
```

In XML format, a dictionary is an element whose name is `dictionary` and whose content consists of pairs of strings and other *property list* objects.

```
<dictionary>
  <string>key1</string>
  <string>value1</string>
</dictionary>
```

NSNumber Number objects were not traditionally allowed in *property lists* but were added when the XML format was intoroduced. GNUstep provides an extension to the traditional *property list* format to support number objects, but older code will not read *property lists* containing this extension.

Numbers are stored in a variety of formats depending on their values.

- boolean... either `<*BY>` for `YES` or `<*BN>` for `NO`.
  In XML format this is either `<true />` or `<false />`
- integer... `<*INNN>` where NNN is an integer.
  In XML format this is `<integer>NNN<integer>`
- real... `<*RNNN>` where NNN is a real number.
  In XML format this is `<real>NNN<real>`

NSString A string is either stored literally (if it contains no spaces or special characters), or is stored as a quoted string with special characters escaped where necessary.

Escape conventions are similar to those normally used in ObjectiveC programming, using a backslash followed by -

- `\` a backslash character
- **"** a quote character
- **b** a backspace character
- **n** a newline character
- **r** a carriage return character
- **t** a tab character
- **OOO** (three octal digits) an arbitrary ascii character
- **UXXXX** (where X is a hexadecimal digit) a an arbitrary unicode character

```
"hello world & others"
```

In XML format, the string is simply stored in UTF8 format as the content of a `string` element, and the only character escapes required are those used by XML such as the '<' markup representing a '<' character.

```
<string>hello world &amp;amp; others</string>"
```

propertyListFromStringsFileFormat *XOG*

- `(NSDictionary*)`**propertyListFromStringsFileFormat**

Reads a *property list* (see -propertyList) from a simplified file format. This format is a traditional style property list file containing a single dictionary, but with the leading '{' and trailing '}' characters omitted.

That is to say, the file contains only semicolon separated key/value pairs (and optionally comments). As a convenience, it is possible to omit the equals sign and the value, so an entry consists of a key string followed by a semicolon. In this case, the value for that key is assumed to be an empty string.

```
// Strings file entries follow -
  key1 = " a string value";
  key2; // This key has an empty string as a value.
  "Another key" = "a longer string value for th third key";
```

rangeOfCharacterFromSet: *XOG*

- `(NSRange)`**rangeOfCharacterFromSet:**`(NSCharacterSet*)` *aSet*

Description forthcoming.

rangeOfCharacterFromSet:options: *XOG*

- `(NSRange)`**rangeOfCharacterFromSet:**`(NSCharacterSet*)` *aSet*
options:`(unsigned int)` *mask*

Description forthcoming.

rangeOfCharacterFromSet:options:range: *XOG*

- `(NSRange)`**rangeOfCharacterFromSet:**`(NSCharacterSet*)` *aSet*
options:`(unsigned int)` *mask*
range:`(NSRange)` *aRange*

Description forthcoming.

rangeOfComposedCharacterSequenceAtIndex: *XOG*

- (NSRange)**rangeOfComposedCharacterSequenceAtIndex:**(unsigned int) *anIndex*

Description forthcoming.

rangeOfString: *XOG*

- (NSRange)**rangeOfString:**(NSString*) *string*

Invokes -rangeOfString:options: with the options mask set to zero.

rangeOfString:options: *XOG*

- (NSRange)**rangeOfString:**(NSString*) *string*
options:(unsigned int) *mask*

Invokes -rangeOfString:options:range: with the range set set to the range of the whole of the reciever.

rangeOfString:options:range: *XOG*

- (NSRange)**rangeOfString:**(NSString*) *aString*
options:(unsigned int) *mask*
range:(NSRange) *aRange*

Returns the range giving the location and length of the first occurrence of *aString* within *aRange*.
If *aString* does not exist in the receiver (an empty string is never considered to exist in the receiver), the length of the returned range is zero.
If *aString* is `nil`, an exception is raised.
If any part of *aRange* lies outside the range of the receiver, an exception is raised.
The options *mask* may contain the following options -

- NSCaseInsensitiveSearch
- NSLiteralSearch
- NSBackwardsSearch
- NSAnchoredSearch

smallestEncoding *XOG*

- `(NSStringEncoding)`**smallestEncoding**

Description forthcoming.

stringByAbbreviatingWithTildeInPath *XOG*

- `(NSString*)`**stringByAbbreviatingWithTildeInPath**

Returns a string where a prefix of the current user's home directory is abbreviated by '~', or returns the receiver if it was not found to have the home directory as a prefix.

stringByAppendingFormat: *XOG*

- `(NSString*)`**stringByAppendingFormat:**`(NSString*)` *format*

Description forthcoming.

stringByAppendingPathComponent: *XOG*

- `(NSString*)`**stringByAppendingPathComponent:**`(NSString*)` *aString*

Returns a new string with the path component given in *aString* appended to the receiver. Removes trailing separators and multiple separators.

stringByAppendingPathExtension: *XOG*

- `(NSString*)`**stringByAppendingPathExtension:**`(NSString*)` *aString*

Returns a new string with the path extension given in *aString* appended to the receiver after the extensionSeparator ('.').
If the receiver has trailing '/' characters which are not part of the root directory, those '/' characters are stripped before the extension separator is added.

stringByAppendingString: *XOG*

- `(NSString*)`**stringByAppendingString:**`(NSString*)` *aString*

Description forthcoming.

stringByDeletingLastPathComponent *XOG*

- (NSString*)stringByDeletingLastPathComponent

Returns a new string with the last path component (including any final path separators) removed from the receiver.

A string without a path component other than the root is returned without alteration.

See -lastPathComponent for a definition of a path component.

stringByDeletingPathExtension *XOG*

- (NSString*)stringByDeletingPathExtension

Returns a new string with the path extension removed from the receiver.

Strips any trailing path separators before checking for the extension separator.

Does not consider a string starting with the extension separator ('.') to be a path extension.

stringByExpandingTildeInPath *XOG*

- (NSString*)stringByExpandingTildeInPath

Returns a string created by expanding the initial tilde ('~') and any following username to be the home directory of the current user or the named user.

Returns the receiver if it was not possible to expand it.

stringByPaddingToLength:withString:startingAtIndex: *X¬OG*

- (NSString*)stringByPaddingToLength:(unsigned int) *newLength*
withString:(NSString*) *padString*
startingAtIndex:(unsigned int) *padIndex*

Returns a string formed by extending or truncating the receiver to *newLength* characters. If the new string is larger, it is padded by appending characters from *padString* (appending it as many times as required). The first character from *padString* to be appended is specified by *padIndex*.

stringByResolvingSymlinksInPath *XOG*

- (NSString*)stringByResolvingSymlinksInPath

Description forthcoming.

stringByStandardizingPath *XOG*

- (NSString*)stringByStandardizingPath

Returns a standardised form of the receiver, with unnecessary parts removed, tilde characters expanded, and symbolic links resolved where possible.

If the string is an invalid path, the unmodified receiver is returned.

Uses -stringByExpandingTildeInPath to expand tilde expressions.

Simplifies '//' and '/./' sequences.

Removes any '/private' prefix.

For absolute paths, uses -stringByResolvingSymlinksInPath to resolve any links, then gets rid of '/../' sequences.

stringByTrimmingCharactersInSet: *X¬OG*

- (NSString*)stringByTrimmingCharactersInSet:(NSCharacterSet*) *aSet*

Return a string formed by removing characters from the ends of the receiver. Characters are removed only if they are in *aSet*.

If the string consists entirely of characters in *aSet* , an empty string is returned.

The *aSet* argument nust not be nil.

stringsByAppendingPaths: *X¬OG*

- (NSArray*)stringsByAppendingPaths:(NSArray*) *paths*

Returns an array of strings made by appending the values in *paths* to the receiver.

substringFromIndex: *XOG*

- (NSString*)substringFromIndex:(unsigned int) *index*

Returns a substring of the receiver from character at the specified *index* to the end of the string.

So, supplying an *index* of 3 would return a substring consisting of the entire string apart from the first three character (those would be at *index* 0, 1, and 2).
If the supplied *index* is greater than or equal to the length of the receiver an exception is raised.

substringFromRange: *XOG*

`- (NSString*)`**substringFromRange:**`(NSRange)` *aRange*

An obsolete name for -substringWithRange: ... deprecated.

substringToIndex: *XOG*

`- (NSString*)`**substringToIndex:**`(unsigned int)` *index*

Returns a substring of the receiver from the start of the string to (but not including) the specified *index* position.
So, supplying an *index* of 3 would return a substring consisting of the first three characters of the receiver.
If the supplied *index* is greater than the length of the receiver an exception is raised.

substringWithRange: *X¬OG*

`- (NSString*)`**substringWithRange:**`(NSRange)` *aRange*

Returns a substring of the receiver containing the characters in *aRange*.
If *aRange* specifies any character position not present in the receiver, an exception is raised.
If *aRange* has a length of zero, an empty string is returned.

uppercaseString *XOG*

`- (NSString*)`**uppercaseString**

Returns a copy of the receiver with all characters converted to uppercase.

writeToFile:atomically: *X¬OG*

- (BOOL)**writeToFile:**(NSString*) *filename*
atomically:(BOOL) *useAuxiliaryFile*

Description forthcoming.

writeToURL:atomically:

$X\neg OG$

- (BOOL)**writeToURL:**(NSURL*) *anURL*
atomically:(BOOL) *atomically*

Description forthcoming.

NSTask *XOG*

Inherits From: NSObject
Conforms To: GCFinalization
Declared in: Foundation/NSTask.h

Description

The NSTask class provides a mechanism to run separate tasks under (limited) control of your program.

Class Methods

launchedTaskWithLaunchPath:arguments: *XOG*

`+ (NSTask*)` **launchedTaskWithLaunchPath:** `(NSString*)` *path*
arguments: `(NSArray*)` *args*

Creates and launches a task, returning an autoreleased task object. Supplies the *path* to the executable and an array of argument. The task inherits the parents environment and I/O.

Instance Methods

arguments *XOG*

`- (NSArray*)` **arguments**

Returns the arguments set for the task.

currentDirectoryPath *XOG*

`- (NSString*)` **currentDirectoryPath**

Returns the working directory set for the task.

environment *XOG*

- `(NSDictionary*)`**environment**

Returns the environment set for the task.

interrupt *XOG*

- `(void)`**interrupt**

Sends an interrupt signal to the receiver and any subtasks.
If the task has not been launched, raises an NSInvalidArgumentException.
Has no effect on a task that has already terminated.
This is rather like the terminate method, but the child process may not choose to terminate in response to an interrupt.

isRunning *XOG*

- `(BOOL)`**isRunning**

Checks to see if the task is currently running.

launch *XOG*

- `(void)`**launch**

Launches the task.
Raises an NSInvalidArgumentException if the launch path is not set or if the subtask cannot be started for some reason (eg. the executable does not exist).

launchPath *XOG*

- `(NSString*)`**launchPath**

Returns the launch path set for the task.

processIdentifier *X¬OG*

- (int)**processIdentifier**

Returns the number identifying the child process on this system.

resume

X¬OG

- (BOOL)**resume**

Sends a cont signal to the receiver and any subtasks.
If the task has not been launched, raises an NSInvalidArgumentException.

setArguments:

XOG

- (void)**setArguments:**(NSArray*) *args*

Sets an array of arguments to be supplied to the task when it is launched. The default is an empty array. This method cannot be used after a task is launched... it raises an NSInvalidArgumentException.

setCurrentDirectoryPath:

XOG

- (void)**setCurrentDirectoryPath:**(NSString*) *path*

Sets the home directory in which the task is to be run. The default is the parent processes directory. This method cannot be used after a task is launched... it raises an NSInvalidArgumentException.

setEnvironment:

XOG

- (void)**setEnvironment:**(NSDictionary*) *env*

Sets the environment variables for the task to be run. The default is the parent processes environment. This method cannot be used after a task is launched... it raises an NSInvalidArgumentException.

setLaunchPath:

XOG

- (void)**setLaunchPath:**(NSString*) *path*

Sets the *path* to the executable file to be run. There is no default for this - you must set the launch *path*. This method cannot be used after a task is launched... it raises an NSInvalidArgumentException.

setStandardError: *XOG*

- `(void)`**setStandardError:**`(id)` *hdl*

Sets the standard error stream for the task.

This is normally a writable NSFileHandle object. If this is an NSPipe, the write end of the pipe is automatically closed on launching.

The default behavior is to inherit the parent processes stderr output.

This method cannot be used after a task is launched... it raises an NSInvalidArgumentException.

setStandardInput: *XOG*

- `(void)`**setStandardInput:**`(id)` *hdl*

Sets the standard input stream for the task.

This is normally a readable NSFileHandle object. If this is an NSPipe, the read end of the pipe is automatically closed on launching.

The default behavior is to inherit the parent processes stdin stream.

This method cannot be used after a task is launched... it raises an NSInvalidArgumentException.

setStandardOutput: *XOG*

- `(void)`**setStandardOutput:**`(id)` *hdl*

Sets the standard output stream for the task.

This is normally a writable NSFileHandle object. If this is an NSPipe, the write end of the pipe is automatically closed on launching.

The default behavior is to inherit the parent processes stdout stream.

This method cannot be used after a task is launched... it raises an NSInvalidArgumentException.

standardError *XOG*

- `(id)`**standardError**

Returns the standard error stream for the task - an NSFileHandle unless an NSPipe was passed to -setStandardError:

standardInput *XOG*

- (id)standardInput

Returns the standard input stream for the task - an NSFileHandle unless an NSPipe was passed to -setStandardInput:

standardOutput *XOG*

- (id)standardOutput

Returns the standard output stream for the task - an NSFileHandle unless an NSPipe was passed to -setStandardOutput:

suspend *X¬OG*

- (BOOL)suspend

Sends a stop signal to the receiver and any subtasks.
If the task has not been launched, raises an NSInvalidArgumentException.

terminate *XOG*

- (void)terminate

Sends a terminate signal to the receiver and any subtasks.
If the task has not been launched, raises an NSInvalidArgumentException.
Has no effect on a task that has already terminated.
When a task temrinates, either due to this method being called, or normal termination, an NSTaskDidTerminateNotification is posted.

terminationStatus *XOG*

- (int)terminationStatus

Returns the termination status of the task.

If the task has not completed running, raises an NSInvalidArgumentException.

usePseudoTerminal

$\neg O \neg XG$

- (BOOL)usePseudoTerminal

If the system supports it, this method sets the standard input, output, and error streams to a pseudo-terminal so that, when launched, the child task will act as if it was running interactively on a terminal. The file handles can then be used to communicate with the child.

This method cannot be used after a task is launched... it raises an NSInvalidArgumentException.

The standard input, output and error streams cannot be changed after calling this method.

The method returns `YES` on success, `NO` on failure.

validatedLaunchPath

$\neg O \neg XG$

- (NSString*)validatedLaunchPath

Returns a validated launch path or `nil` .

Allows for the GNUstep host/operating system, and library combination subdirectories in a path, appending them as necessary to try to locate the actual binary to be used.

Checks that the binary file exists and is executable.

Even tries searching the directories in the PATH environment variable to locate a binary if the original alunch path set was not absolute.

waitUntilExit

XOG

- (void)waitUntilExit

Suspends the current thread until the task terminates, by waiting in NSRunLoop (NSDefaultRunLoopMode) for the task termination.

Returns immediately if the task is not running.

NSThread *XOG*

Inherits From: NSObject
Declared in: Foundation/NSThread.h

Description

This class encapsulates OpenStep threading. See NSLock and its subclasses for handling synchronisation between threads.

Each process begins with a main thread and additional threads can be created using NSThread. The GNUstep implementation of OpenStep has been carefully designed so that the internals of the base library do not use threading (except for methods which explicitly deal with threads of course) so that you can write applications without threading. Non-threaded applications re more efficient (no locking is required) and are easier to debug during development.

Class Methods

currentThread *XOG*

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

Returns the NSThread object corresponding to the current thread.

NB. In GNUstep the library internals use the GSCurrentThread() function as a more efficient mechanism for doing this job - so you cannot use a category to override this method and expect the library internals to use your implementation.

detachNewThreadSelector:toTarget:withObject: *XOG*

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

Create a new thread - use this method rather than alloc-init

exit *XOG*

+ (void)**exit**

Terminating a thread What happens if the thread doesn't call +exit - it doesn't terminate!

isMultiThreaded *XOG*

+ (BOOL)**isMultiThreaded**

Returns a flag to say whether the application is multi-threaded or not. An application is considered to be multi-threaded if any thread other than the main thread has been started, irrespective of whether that thread has since terminated.

setThreadPriority: *XOG*

+ (void)**setThreadPriority:** (double) *pri*

Set the priority of the current thread. This is a value in the range 0.0 (lowest) to 1.0 (highest) which is mapped to the underlying system priorities. The current gnu objc runtime supports three priority levels which you can obtain using values of 0.0, 0.5, and 1.0

sleepUntilDate: *XOG*

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

Delaying a thread... pause until the specified *date*.

threadPriority *XOG*

+ (double)**threadPriority**

Return the priority of the current thread.

Instance Methods

threadDictionary *XOG*

- (NSMutableDictionary*)**threadDictionary**

Return the thread dictionary. This dictionary can be used to store arbitrary thread specific data.

NB. This cannot be autoreleased, since we cannot be sure that the autorelease pool for the thread will continue to exist for the entire life of the thread!

NSTimeZone *XOG*

Inherits From: NSObject
Declared in: Foundation/NSTimeZone.h

Description

If the GNUstep time zone datafiles become too out of date, one can download an updated database from

and compile it as specified in the README file in the NSTimeZones directory.

Time zone names in NSDates should be GMT, MET etc. not Europe/Berlin, America/Washington etc.

The problem with this is that various time zones may use the same abbreviation (e.g. Australia/Brisbane and America/New_York both use EST), and some time zones may have different rules for daylight saving time even if the abbreviation and offsets from UTC are the same.

The problems with depending on the OS for providing time zone info are that some methods for the NSTimeZone classes might be difficult to implement, and also that time zone names may vary wildly between OSes (this could be a big problem when archiving is used between different systems).

Class Methods

abbreviationDictionary *XOG*

`+ (NSDictionary*)`**abbreviationDictionary**

Description forthcoming.

abbreviationMap *¬O¬XG*

`+ (NSDictionary*)`**abbreviationMap**

Description forthcoming.

defaultTimeZone *XOG*

+ (NSTimeZone*)**defaultTimeZone**

Return the default time zone for this process.

localTimeZone *XOG*

+ (NSTimeZone*)**localTimeZone**

Return a proxy to the default time zone for this process.

resetSystemTimeZone *X¬OG*

+ (void)**resetSystemTimeZone**

Destroy the system time zone so that it will be recreated next time it is used.

setDefaultTimeZone: *XOG*

+ (void)**setDefaultTimeZone:** (NSTimeZone*) *aTimeZone*

Set the default time zone to be used for this process.

systemTimeZone *X¬OG*

+ (NSTimeZone*)**systemTimeZone**

Returns the current system time zone for the process.

timeZoneArray *XOG*

+ (NSArray*)**timeZoneArray**

Description forthcoming.

timeZoneForSecondsFromGMT: *XOG*

+ (NSTimeZone*)**timeZoneForSecondsFromGMT:** (int) *seconds*

Return a timezone for the specified offset from GMT.
The timezone returned does *not* use daylight savings time. The actual timezone returned has an offset rounded to the nearest minute.
Time zones with an offset of more than +/- 18 hours are disallowed, and `nil` is returned.

timeZoneWithAbbreviation: *XOG*

`+ (NSTimeZone*)`**timeZoneWithAbbreviation:** `(NSString*)` *abbreviation*

Returns a timezone for the specified abbrevition,

timeZoneWithName: *XOG*

`+ (NSTimeZone*)`**timeZoneWithName:** `(NSString*)` *aTimeZoneName*

Returns a timezone for the specified name.

timeZoneWithName:data: *X¬OG*

`+ (NSTimeZone*)`**timeZoneWithName:** `(NSString*)` *name*
data: `(NSData*)` *data*

Returns a timezone for the specified *name*, created from the supplied *data*.

Instance Methods

abbreviation *X¬OG*

`- (NSString*)`**abbreviation**

Returns the abbreviation for this timezone now. Invokes -abbreviationForDate:

abbreviationForDate: *X¬OG*

`- (NSString*)`**abbreviationForDate:**`(NSDate*)` *aDate*

Returns the abbreviation for this timezone at *aDate*. This may differ depending on whether daylight savings time is in effect or not.

data

$X\neg OG$

`- (NSData*)`**data**

Returns the data with which the receiver was initialised.

initWithName:

$X\neg OG$

`- (id)`**initWithName:**`(NSString*)` *name*

Initialise a timezone with the supplied *name*. May return a cached timezone object rather than the newly created one.

initWithName:data:

$X\neg OG$

`- (id)`**initWithName:**`(NSString*)` *name*
data:`(NSData*)` *data*

Initialises a time zone object using the supplied *data* object.
This method is intended for internal use by the NSTimeZone class cluster. Don't use it... use -initWithName: instead.

isDaylightSavingTime

$X\neg OG$

`- (BOOL)`**isDaylightSavingTime**

Returns a boolean indicating whether daylight savings time is in effect now. Invokes -isDaylightSavingTimeForDate:

isDaylightSavingTimeForDate:

$X\neg OG$

`- (BOOL)`**isDaylightSavingTimeForDate:**`(NSDate*)` *aDate*

Returns a boolean indicating whether daylight savings time is in effect for this time zone at *aDate*.

isEqualToTimeZone:

$X\neg OG$

- (BOOL)**isEqualToTimeZone:**(NSTimeZone*) *aTimeZone*

Description forthcoming.

name

X¬OG

- (NSString*)**name**

Description forthcoming.

secondsFromGMT

X¬OG

- (int)**secondsFromGMT**

Returns the number of seconds by which the receiver differs from Greenwich Mean Time at the current date and time.
Invokes -secondsFromGMTForDate:

secondsFromGMTForDate:

X¬OG

- (int)**secondsFromGMTForDate:**(NSDate*) *aDate*

Returns the number of seconds by which the receiver differs from Greenwich Mean Time at the date *aDate*.
If the time zone uses dayl;ight savings time, the returned value will vary at different times of year.

timeZoneDetailArray

XOG

- (NSArray*)**timeZoneDetailArray**

Description forthcoming.

timeZoneDetailForDate:

O¬XG

- (NSTimeZoneDetail*)**timeZoneDetailForDate:**(NSDate*) *date*

Description forthcoming.

timeZoneName $O\neg XG$

- (NSString*)timeZoneName

Description forthcoming.

NSTimeZoneDetail $O\neg XG$

Inherits From: NSTimeZone : NSObject
Declared in: Foundation/NSTimeZone.h

Description

This class serves no useful purpose in GNUstep, and is provided solely for backward compatibility with the OpenStep spec. It is missing entirely from MacOS-X.

Instance Methods

isDaylightSavingTimeZone $O\neg XG$

- (BOOL)isDaylightSavingTimeZone

Description forthcoming.

timeZoneAbbreviation $O\neg XG$

- (NSString*)timeZoneAbbreviation

Description forthcoming.

timeZoneSecondsFromGMT $O\neg XG$

- (int)timeZoneSecondsFromGMT

Description forthcoming.

NSTimer *XOG*

Inherits From: NSObject
Declared in: Foundation/NSTimer.h

Description

Description forthcoming.

Class Methods

scheduledTimerWithTimeInterval:invocation:repeats: *XOG*

```
+ (NSTimer*)scheduledTimerWithTimeInterval: (NSTimeInterval) ti
                                invocation: (NSInvocation*) invocation
                                   repeats: (BOOL) f
```

Create a timer which will fire after *ti* seconds and, if *f* is YES, every *ti* seconds thereafter. On firing, *invocation* will be performed.

This timer will automatically be added to the current run loop and will fire in the default run loop mode.

scheduledTimerWithTimeInterval:target:selector:userInfo:repeats: *XOG*

```
+ (NSTimer*)scheduledTimerWithTimeInterval: (NSTimeInterval) ti
                                   target: (id) object
                                 selector: (SEL) selector
                                 userInfo: (id) info
                                  repeats: (BOOL) f
```

Create a timer which will fire after *ti* seconds and, if *f* is YES, every *ti* seconds thereafter. On firing, the target *object* will be sent a message specified by *selector* and with the *object info* as an argument.

This timer will automatically be added to the current run loop and will fire in the default run loop mode.

timerWithTimeInterval:invocation:repeats: *XOG*

```
+ (NSTimer*)timerWithTimeInterval: (NSTimeInterval) ti
                       invocation: (NSInvocation*) invocation
                          repeats: (BOOL) f
```

Create a timer wchich will fire after *ti* seconds and, if *f* is YES, every *ti* seconds thereafter. On firing, *invocation* will be performed.

NB. To make the timer operate, you must add it to a run loop.

timerWithTimeInterval:target:selector:userInfo:repeats: *XOG*

```
+ (NSTimer*)timerWithTimeInterval: (NSTimeInterval) ti
                           target: (id) object
                         selector: (SEL) selector
                         userInfo: (id) info
                          repeats: (BOOL) f
```

Create a timer wchich will fire after *ti* seconds and, if *f* is YES, every *ti* seconds thereafter. On firing, the target *object* will be sent a message specified by *selector* and with the *object info* as an argument.

NB. To make the timer operate, you must add it to a run loop.

Instance Methods

fire *XOG*

```
- (void)fire
```

Fires the timer... either performs an invocation or ssends a message to a target object, depending on how the timer was set up.

If the timer is not set to repeat, it is automatically invalidated.

fireDate *XOG*

```
- (NSDate*)fireDate
```

Returns the date/time at which the timer is next due to fire.

initWithFireDate:interval:target:selector:userInfo:repeats: *X¬OG*

- (id)**initWithFireDate:**(NSDate*) *fd*
interval:(NSTimeInterval) *ti*
target:(id) *object*
selector:(SEL) *selector*
userInfo:(id) *info*
repeats:(BOOL) *f*

Initialise the receive, a newly allocated NSTimer *object*.

The *fd* argument specifies an initial fire date... if it is not supplied (a `nil` *object*) then the *ti* argument is used to create a start date relative to the current time.

The *ti* argument specifies the time (in seconds) between the firing. If it is less than or equal to 0.0 then a small interval is chosen automatically.

The *f* argument specifies whether the timer will fire repeatedly or just once.

If the *selector* argument is zero, then then *object* is an invocation to be used when the timer fires. otherwise, the *object* is sent the message specified by the *selector* and with the timer as an argument.

The *fd*, *object* and *info* arguments will be retained until the timer is invalidated.

invalidate

XOG

- (void)**invalidate**

Marks the timer as invalid, causing its target/invocation and user info objects to be released.

Invalidated timers are automatically removed from the run loop when it detects them.

isValid

X¬OG

- (BOOL)**isValid**

Checks to see if the timer has been invalidated.

setFireDate:

X¬OG

- (void)**setFireDate:**(NSDate*) *fireDate*

Change the fire date for the receiver.

NB. You should *NOT* use this method for a timer which has been added to a run loop. The only time when it is safe to modify the fire date of a timer in a run loop is for a repeating timer when the timer is actually in the process of firing.

timeInterval

$X\neg OG$

- `(NSTimeInterval)`**timeInterval**

Returns the interval beteen firings.

userInfo

XOG

- `(id)`**userInfo**

Returns the user info which was set for the timer when it was created, or `nil` if none was set or the timer is invalid.

NSURL

X¬OG

Inherits From: NSObject
Conforms To: NSCoding
NSCopying
NSURLHandleClient
Declared in: Foundation/NSURL.h

Description

This class permits manipulation of URLs and the resources to which they refer. They can be used to represent absolute URLs or relative URLs which are based upon an absolute URL. The relevant RFCs describing how a URL is formatted, and what is legal in a URL are - 1808, 1738, and 2396.

Handling of the underlying resources is carried out by NSURLHandle objects, but NSURL provides a simoplified API wrapping these objects.

Class Methods

URLWithString:

X¬OG

`+ (id)` **URLWithString:** `(NSString*)` *aUrlString*

Create and return a URL with the supplied string, which should be a string (containing percent escape codes where necessary) conforming to the description (in RFC2396) of an absolute URL.
Calls -initWithString:

URLWithString:relativeToURL:

X¬OG

`+ (id)` **URLWithString:** `(NSString*)` *aUrlString*
relativeToURL: `(NSURL*)` *aBaseUrl*

Create and return a URL with the supplied string, which should be a string (containing percent escape codes where necessary) conforming to the description (in RFC2396) of a relative URL.
Calls -initWithString:relativeToURL:

fileURLWithPath: $X\neg OG$

`+ (id)`**fileURLWithPath:** `(NSString*)` *aPath*

Create and return a file URL with the supplied path.
The value of *aPath* must be a valid filesystem path.
Calls -initFileURLWithPath:

Instance Methods

URLHandleUsingCache: $X\neg OG$

`- (NSURLHandle*)`**URLHandleUsingCache:**`(BOOL)` *shouldUseCache*

Returns an NSURLHandle instance which may be used to write data to the resource represented by the receiver URL, or read data from it.
The *shouldUseCache* flag indicates whether a cached handle may be returned or a new one should be created.

absoluteString $X\neg OG$

`- (NSString*)`**absoluteString**

Returns the full string describing the receiver resiolved against its base.

absoluteURL $X\neg OG$

`- (NSURL*)`**absoluteURL**

If the receiver is an absolute URL, returns self. Otherwise returns an absolute URL referring to the same resource as the receiver.

baseURL $X\neg OG$

`- (NSURL*)`**baseURL**

If the receiver is a relative URL, returns its base URL.
Otherwise, returns `nil`.

fragment

X¬OG

- (NSString*)**fragment**

Returns the fragment portion of the receiver or `nil` if there is no fragment supplied in the URL.
The fragment is everything in the original URL string after a '#'
File URLs do not have fragments.

host

X¬OG

- (NSString*)**host**

Returns the host portion of the receiver or `nil` if there is no host supplied in the URL.
Percent escape sequences in the user string are translated and the string treated as UTF8.

initFileURLWithPath:

X¬OG

- (id)**initFileURLWithPath:**(NSString*) *aPath*

Initialise as a file URL with the specified path (which must be a valid path on the local filesystem).
Converts relative paths to absolute ones.
Appends a trailing slash to the path when necessary if it specifies a directory.
Calls -initWithScheme:host:path:

initWithScheme:host:path:

X¬OG

- (id)**initWithScheme:**(NSString*) *aScheme*
  host:(NSString*) *aHost*
  path:(NSString*) *aPath*

Initialise by building a URL string from the supplied parameters and calling -initWithString:relativeToURL:

initWithString:

X¬OG

- (id)**initWithString:**(NSString*) *aUrlString*

Initialise as an absolute URL.
Calls -initWithString:relativeToURL:

initWithString:relativeToURL: *X¬OG*

- (id)**initWithString:**(NSString*) *aUrlString*
  relativeToURL:(NSURL*) *aBaseUrl*

Initialised using *aUrlString* and *aBaseUrl*. The value of *aBaseUrl* may be nil, but *aUrlString* must be non-nil.
If the string cannot be parsed the method returns nil.

isFileURL *X¬OG*

- (BOOL)**isFileURL**

Returns YES if the recevier is a file URL, NO otherwise.

loadResourceDataNotifyingClient:usingCache: *X¬OG*

- (void)**loadResourceDataNotifyingClient:**(id) *client*
  usingCache:(BOOL) *shouldUseCache*

Loads resource data for the specified *client*.
If *shouldUseCache* is YES then an attempt will be made to locate a cached NSURL-Handle to provide the resource data, otherwise a new handle will be created and cached.
If the handle does not have the data available, it will be asked to load the data in the background by calling its loadInBackground method.
The specified *client* (if non-nil) will be set up to receive notifications of the progress of the background load process.
The processes current run loop must be run in order for the background load operation to operate!

parameterString *X¬OG*

- (NSString*)**parameterString**

Returns the parameter portion of the receiver or nil if there is no parameter supplied in the URL.

The parameters are everything in the original URL string after a ';' but before the query.

File URLs do not have parameters.

password

X¬OG

- (NSString*)**password**

Returns the password portion of the receiver or `nil` if there is no password supplied in the URL.

Percent escape sequences in the user string are translated and the string treated as UTF8 in GNUstep but this appears to be broken in MacOS-X.

NB. because of its security implications it is recommended that you do not use URLs with users and passwords unless necessary.

path

X¬OG

- (NSString*)**path**

Returns the path portion of the receiver.

Replaces percent escapes with unescaped values, interpreting non-ascii character sequences as UTF8.

NB. This does not conform strictly to the RFCs, in that it includes a leading slash ('/') character (wheras the path part of a URL strictly should not) and the interpretation of non-ascii character is (strictly speaking) undefined.

Also, this breaks strict conformance in that a URL of file scheme is treated as having a path (contrary to RFCs)

port

X¬OG

- (NSNumber*)**port**

Returns the port portion of the receiver or `nil` if there is no port supplied in the URL.

Percent escape sequences in the user string are translated in GNUstep but this appears to be broken in MacOS-X.

propertyForKey:

X¬OG

- (id)**propertyForKey:**(NSString*) *propertyKey*

Asks a URL handle to return the property for the specified key and returns the result.

query

$X\neg OG$

- (NSString*)**query**

Returns the query portion of the receiver or `nil` if there is no query supplied in the URL.
The query is everything in the original URL string after a '?' but before the fragment.
File URLs do not have queries.

relativePath

$X\neg OG$

- (NSString*)**relativePath**

Returns the path of the receiver, without taking any base URL into account. If the receiver is an absolute URL, -relativePath is the same as -path .
Returns `nil` if there is no path specified for the URL.

relativeString

$X\neg OG$

- (NSString*)**relativeString**

Returns the relative portion of the URL string. If the receiver is not a relative URL, this returns the same as absoluteString.

resourceDataUsingCache:

$X\neg OG$

- (NSData*)**resourceDataUsingCache:**(BOOL) *shouldUseCache*

Loads the resource data for the represented URL and returns the result. The shoulduseCache flag determines whether an existing cached NSURLHandle can be used to provide the data.

resourceSpecifier

$X\neg OG$

- (NSString*)**resourceSpecifier**

Returns the resource specifier of the URL... the part which lies after the scheme.

scheme

X¬OG

- (NSString*)**scheme**

Returns the scheme of the receiver.

setProperty:forKey:

X¬OG

- (BOOL)**setProperty:**(id) *property*
  forKey:(NSString*) *propertyKey*

Calls [NSURLHandle -writeProperty:forKey:] to set the named *property*.

setResourceData:

X¬OG

- (BOOL)**setResourceData:**(NSData*) *data*

Calls [NSURLHandle -writeData:] to write the specified *data* object to the resource identified by the receiver URL.
Returns the result.

standardizedURL

X¬OG

- (NSURL*)**standardizedURL**

Returns a URL with '/./' and '/../' sequences resolved etc.

user

X¬OG

- (NSString*)**user**

Returns the user portion of the receiver or nil if there is no user supplied in the URL.
Percent escape sequences in the user string are translated and the whole is treated as UTF8 data.
NB. because of its security implications it is recommended that you do not use URLs with users and passwords unless necessary.

NSURLHandle

X¬OG

Inherits From: NSObject
Declared in: Foundation/NSURLHandle.h

Description

An NSURLHandle instance is used to manage the resource data corresponding to an NSURL object. A single NSURLHandle can be used to manage multiple NSURL objects as long as those objects all refer to the same resource.

Different NSURLHandle subclasses are used to manage different types of URL (usually based on the scheme of the URL), and you can register new subclasses to extend (or replace) the standard ones.

GNUstep comes with private subclasses to handle the common URL schemes -

- `file:` (local file I/O)
- `http:` and `shttp:` (webserver) access.
- `ftp:` (FTP server) access.

Class Methods

URLHandleClassForURL:

X¬OG

`+ (Class)`**URLHandleClassForURL:** `(NSURL*)` *url*

Returns the most recently registered NSURLHandle subclass that responds to +canInitWithURL: with `YES`. If there is no such subclass, returns `nil`.

cachedHandleForURL:

X¬OG

`+ (NSURLHandle*)`**cachedHandleForURL:** `(NSURL*)` *url*

Return a handle for the specified URL from the cache if possible. If the cache does not contain a matching handle, returns `nil`.

canInitWithURL: $X\neg OG$

`+ (BOOL)`**canInitWithURL:** `(NSURL*)` *url*

Implemented by subclasses to say which URLs they can handle. This method is used to determine which subclasses can be used to handle a particular URL.

registerURLHandleClass: $X\neg OG$

`+ (void)`**registerURLHandleClass:** `(Class)` *urlHandleSubclass*

Used to register a subclass as being available to handle URLs.

Instance Methods

addClient: $X\neg OG$

`- (void)`**addClient:**`(id&#60;NSURLHandleClient>)` *client*

Add a *client* object, making sure that it doesn't occur more than once.
The *client* object will receive messages notifying it of events on the handle.

availableResourceData $X\neg OG$

`- (NSData*)`**availableResourceData**

Returns the resource data that is currently available for the handle. This may be a partially loaded resource or may be empty of no data has been loaded yet.

backgroundLoadDidFailWithReason: $X\neg OG$

`- (void)`**backgroundLoadDidFailWithReason:**`(NSString*)` *reason*

This method should be called when a background load fails.
The method passes the failure notification to the clients of the handle - so subclasses should call super's implementation at the end of their implementation of this method.

beginLoadInBackground

X¬OG

- (void)**beginLoadInBackground**

This method is called by when a background load begins. Subclasses should call super's implementation at the end of their implementation of this method.

cancelLoadInBackground

X¬OG

- (void)**cancelLoadInBackground**

This method should be called to cancel a load currently in progress. The method calls -endLoadInBackground Subclasses should call super's implementation at the end of their implementation of this method.

didLoadBytes:loadComplete:

X¬OG

- (void)**didLoadBytes:**(NSData*) *newData*
  loadComplete:(BOOL) *loadComplete*

Method called by subclasses during process of loading a resource. The base class maintains a copy of the data being read in and accumulates separate parts of the data.

endLoadInBackground

X¬OG

- (void)**endLoadInBackground**

This method is called to stop any background loading process. -cancelLoadInBackground uses this method to cancel loading. Subclasses should call super's implementation at the end of their implementation of this method.

failureReason

X¬OG

- (NSString*)**failureReason**

Returns the failure reason for the last failure to load the resource data.

flushCachedData

X¬OG

- (void)**flushCachedData**

Flushes any cached resource data.

initWithURL:cached: $X\neg OG$

- (id)**initWithURL:**(NSURL*) *url*
  cached:(BOOL) *cached*

Initialises a handle with the specified URL.
The flag determines whether the handle will cache resource data and respond to requests from equivalent URLs for the *cached* data.

loadInBackground $X\neg OG$

- (void)**loadInBackground**

Starts (or queues) loading of the handle's resource data in the background (asynchronously).
The default implementation uses loadInForeground - if this method is not overridden, loadInForeground MUST be.

loadInForeground $X\neg OG$

- (NSData*)**loadInForeground**

Loads the handle's resource data in the foreground (synchronously).
The default implementation starts a background load and waits for it to complete - if this method is not overridden, loadInBackground MUST be.

propertyForKey: $X\neg OG$

- (id)**propertyForKey:**(NSString*) *propertyKey*

Returns the property for the specified key, or `nil` if the key does not exist.

propertyForKeyIfAvailable: $X\neg OG$

- (id)**propertyForKeyIfAvailable:**(NSString*) *propertyKey*

Returns the property for the specified key, but only if the handle does not need to do any work to retrieve it.

removeClient: $X\neg OG$

- (void)**removeClient:**(id<NSURLHandleClient>) *client*

Removes an object from them list of clients notified of resource loading events by the URL handle.

resourceData $X\neg OG$

- (NSData*)**resourceData**

Returns the resource data belonging to the handle. Calls -loadInForeground if necessary.
The GNUstep implementation treats an *ftp:* request for a directory as a request to list the names of the directory contents.

status $X\neg OG$

- (NSURLHandleStatus)**status**

Returns the current status of the handle.

writeData: $X\neg OG$

- (BOOL)**writeData:**(NSData*) *data*

Writes resource *data* to the handle. Returns YES on success, NO on failure.
The GNUstep implementation for *file:* writes the *data* directly to the local filesystem, and the return status reflects the result of that write operation.
The GNUstep implementation for *http:* and *https:* sets the specified *data* as information to be POSTed to the URL next time it is loaded - so the method always returns YES.
The GNUstep implementation for *ftp:* sets the specified *data* as information to be weitten to the URL next time it is loaded - so the method always returns YES.

writeProperty:forKey: $X\neg OG$

- (BOOL)**writeProperty:**(id) *propertyValue*
forKey:(NSString*) *propertyKey*

Sets a property for handle. Returns YES on success, NO on failure.

The GNUstep implementation sets the property as a header to be sent the next time the URL is loaded, and recognizes some special property keys which control the behavior of the next load.

NSUnarchiver *XOG*

Inherits From: NSCoder : NSObject
Declared in: Foundation/NSArchiver.h

Description

This class reconstructs objects from an archive.

Re-using the archiver

The -resetUnarchiverWithData:atIndex: method lets you re-use the archive to decode a new data object or, in conjunction with the 'cursor' method (which reports the current decoding position in the archive), decode a second archive that exists in the data object after the first one. **Subclassing with different input format.**

NSUnarchiver normally reads directly from an NSData object using the methods -

-deserializeTypeTag:andCrossRef:atCursor: to decode type tags for data items, the tag is the first byte of the character encoding string for the data type (as provided by '@encode(xxx)'), possibly with the top bit set to indicate that what follows is a crossreference to an item already encoded.

Also decode a crossreference number either to identify the following item, or to refer to a previously encoded item. Objects, Classes, Selectors, CStrings and Pointer items have crossreference encoding, other types do not.

[NSData -deserializeDataAt:ofObjCType:atCursor:context:] to decode all other information.

And uses other NSData methods to read the archive header information from within the method: [-deserializeHeaderAt:version:classes:objects:pointers:] to read a fixed size header including archiver version (obtained by [self systemVersion]) and crossreference table sizes.

To subclass NSUnarchiver, you must implement your own versions of the four methods above, and override the 'directDataAccess' method to return `NO` so that the archiver knows to use your serialization methods rather than those in the NSData object.

Class Methods

classNameDecodedForArchiveClassName: *XOG*

`+ (NSString*)`**classNameDecodedForArchiveClassName:** `(NSString*)` *nameInArchive*

Description forthcoming.

decodeClassName:asClassName: *XOG*

`+ (void)`**decodeClassName:** `(NSString*)` *nameInArchive*
asClassName: `(NSString*)` *trueName*

Description forthcoming.

unarchiveObjectWithData: *XOG*

`+ (id)`**unarchiveObjectWithData:** `(NSData*)` *anObject*

Description forthcoming.

unarchiveObjectWithFile: *XOG*

`+ (id)`**unarchiveObjectWithFile:** `(NSString*)` *path*

Description forthcoming.

Instance Methods

classNameDecodedForArchiveClassName: *XOG*

`- (NSString*)`**classNameDecodedForArchiveClassName:**`(NSString*)` *nameInArchive*

Description forthcoming.

decodeClassName:asClassName: *XOG*

- (void)**decodeClassName:**(NSString*) *nameInArchive*
asClassName:(NSString*) *trueName*

Description forthcoming.

initForReadingWithData: *XOG*

- (id)**initForReadingWithData:**(NSData*) *anObject*

Description forthcoming.

isAtEnd *XOG*

- (BOOL)**isAtEnd**

Description forthcoming.

objectZone *XOG*

- (NSZone*)**objectZone**

Description forthcoming.

replaceObject:withObject: *X¬OG*

- (void)**replaceObject:**(id) *anObject*
withObject:(id) *replacement*

Description forthcoming.

setObjectZone: *XOG*

- (void)**setObjectZone:**(NSZone*) *aZone*

Description forthcoming.

systemVersion *XOG*

- (unsigned int)**systemVersion**

Description forthcoming.

NSUndoManager *XOG*

Inherits From: NSObject
Declared in: Foundation/NSUndoManager.h

Description

Description forthcoming.

Instance Methods

beginUndoGrouping *XOG*

- (void)beginUndoGrouping

Description forthcoming.

canRedo *XOG*

- (BOOL)canRedo

Description forthcoming.

canUndo *XOG*

- (BOOL)canUndo

Description forthcoming.

disableUndoRegistration *XOG*

- (void)disableUndoRegistration

Description forthcoming.

enableUndoRegistration *XOG*

- `(void)`**enableUndoRegistration**

Description forthcoming.

endUndoGrouping *XOG*

- `(void)`**endUndoGrouping**

Description forthcoming.

forwardInvocation: *XOG*

- `(void)`**forwardInvocation:**`(NSInvocation*)` *anInvocation*

Description forthcoming.

groupingLevel *XOG*

- `(int)`**groupingLevel**

Description forthcoming.

groupsByEvent *XOG*

- `(BOOL)`**groupsByEvent**

Description forthcoming.

isRedoing *XOG*

- `(BOOL)`**isRedoing**

Description forthcoming.

isUndoRegistrationEnabled *XOG*

- `(BOOL)`**isUndoRegistrationEnabled**

Description forthcoming.

isUndoing *XOG*

- `(BOOL)`**isUndoing**

Description forthcoming.

levelsOfUndo *XOG*

- `(unsigned int)`**levelsOfUndo**

Description forthcoming.

prepareWithInvocationTarget: *XOG*

- `(id)`**prepareWithInvocationTarget:**`(id)` *target*

Description forthcoming.

redo *XOG*

- `(void)`**redo**

Description forthcoming.

redoActionName *XOG*

- `(NSString*)`**redoActionName**

Description forthcoming.

redoMenuItemTitle *XOG*

- (NSString*)**redoMenuItemTitle**

Description forthcoming.

redoMenuTitleForUndoActionName: *XOG*

- (NSString*)**redoMenuTitleForUndoActionName:**(NSString*) *name*

Description forthcoming.

registerUndoWithTarget:selector:object: *XOG*

- (void)**registerUndoWithTarget:**(id) *target*
selector:(SEL) *aSelector*
object:(id) *anObject*

Description forthcoming.

removeAllActions *XOG*

- (void)**removeAllActions**

Description forthcoming.

removeAllActionsWithTarget: *XOG*

- (void)**removeAllActionsWithTarget:**(id) *target*

Description forthcoming.

runLoopModes *XOG*

- (NSArray*)**runLoopModes**

Description forthcoming.

setActionName: *XOG*

- (void)**setActionName:**(NSString*) *name*

Description forthcoming.

setGroupsByEvent: *XOG*

- (void)**setGroupsByEvent:**(BOOL) *flag*

Description forthcoming.

setLevelsOfUndo: *XOG*

- (void)**setLevelsOfUndo:**(unsigned) *num*

Description forthcoming.

setRunLoopModes: *XOG*

- (void)**setRunLoopModes:**(NSArray*) *newModes*

Description forthcoming.

undo *XOG*

- (void)**undo**

Description forthcoming.

undoActionName *XOG*

- (NSString*)**undoActionName**

Description forthcoming.

undoMenuItemTitle *XOG*

- (NSString*)**undoMenuItemTitle**

Description forthcoming.

undoMenuTitleForUndoActionName: *XOG*

- `(NSString*)`**undoMenuTitleForUndoActionName:**`(NSString*)` *name*

Description forthcoming.

undoNestedGroup *XOG*

- `(void)`**undoNestedGroup**

Description forthcoming.

NSUserDefaults *XOG*

Inherits From: NSObject
Declared in: Foundation/NSUserDefaults.h

Description

NSUserDefaults provides an interface to the defaults system, which allows an application access to global and/or application specific defualts set by the user. A particular instance of NSUserDefaults, standardUserDefaults, is provided as a convenience. Most of the information described below pertains to the standardUserDefaults. It is unlikely that you would want to instantiate your own userDefaults object, since it would not be set up in the same way as the standardUserDefaults.

Defaults are managed based on *domains*. Certain domains, such as `NSGlobalDomain`, are persistant. These domains have defaults that are stored externally. Other domains are volitale. The defaults in these domains remain in effect only during the existance of the application and may in fact be different for applications running at the same time. When asking for a default value from standardUserDefaults, NSUserDefaults looks through the various domains in a particular order.

`NSArgumentDomain`... **volatile** Contains defaults read from the arguments provided to the application at startup.

Application (name of the current process)... persistent Contains application specific defaults, such as window positions.

`NSGlobalDomain`... **persistent** Global defaults applicable to all applications.

Language (name based on users's language)... volatile Constants that help with localization to the users's language.

`NSRegistrationDomain`... **volatile** Temporary defaults set up by the application.

The *NSLanguages* default value is used to set up the constants for localization. GNUstep will also look for the `LANGUAGES` environment variable if it is not set in the defaults system. If it exists, it consists of an array of languages that the user prefers. At least one of the languages should have a corresponding localization file (typically located in the `Languages` directory of the GNUstep resources).

As a special extension, on systems that support locales (e.g. GNU/Linux and Solaris), GNUstep will use information from the user specified locale, if the *NSLanguages* default value is not found. Typically the locale is specified in the environment with the `LANG` environment variable.

The first change to a persistent domain after a -synchronize will cause an NSUserDefaultsDidChangeNotification to be posted (as will any change caused by reading new values from disk), so your application can keep track of changes made to the defaults by other software.

NB. The GNUstep implementation differs from the Apple one in that it is thread-safe while Apples (as of MacOS-X 10.1) is not.

Class Methods

resetStandardUserDefaults

$X\neg OG$

`+ (void)`**resetStandardUserDefaults**

Resets the shared user defaults object to reflect the current user ID. Needed by setuid processes which change the user they are running as.

In GNUstep you should call GSSetUserName() when changing your effective user ID, and that class will call this function for you.

setUserLanguages:

$\neg X\neg OG$

`+ (void)`**setUserLanguages:** `(NSArray*)` *languages*

Sets the array of user *languages* preferences. Places the specified array in the *NSLanguages* user default.

standardUserDefaults

XOG

`+ (NSUserDefaults*)`**standardUserDefaults**

Returns the shared defaults object. If it doesn't exist yet, it's created. The defaults are initialized for the current user. The search list is guaranteed to be standard only the first time this method is invoked. The shared instance is provided as a convenience; other instances may also be created.

userLanguages

$\neg X\neg OG$

`+ (NSArray*)`**userLanguages**

Returns the array of user languages preferences. Uses the *NSLanguages* user default if available, otherwise tries to infer setup from operating system information etc (in particular, uses the *LANGUAGES* environment variable).

Instance Methods

arrayForKey: *XOG*

`- (NSArray*)arrayForKey:(NSString*)` *defaultName*

Looks up a value for a specified default using -objectForKey: and checks that it is an NSArray object. Returns `nil` if it is not.

boolForKey: *XOG*

`- (BOOL)boolForKey:(NSString*)` *defaultName*

Looks up a value for a specified default using -objectForKey: and returns its boolean representation.

Returns `NO` if it is not a boolean.

The text 'yes' or 'true' or any non zero numeric value is considered to be a boolean `YES`. Other string values are `NO`.

NB. This differs slightly from the documented behavior for MacOS-X (August 2002) in that the GNUstep version accepts the string 'TRUE' as equivalent to 'YES'.

dataForKey: *XOG*

`- (NSData*)dataForKey:(NSString*)` *defaultName*

Looks up a value for a specified default using -objectForKey: and checks that it is an NSData object. Returns `nil` if it is not.

dictionaryForKey: *XOG*

`- (NSDictionary*)dictionaryForKey:(NSString*)` *defaultName*

Looks up a value for a specified default using -objectForKey: and checks that it is an NSDictionary object. Returns `nil` if it is not.

dictionaryRepresentation *XOG*

- `(NSDictionary*)`**dictionaryRepresentation**

Returns a dictionary representing the current state of the defaults system... this is a merged version of all the domains in the search list.

floatForKey: *XOG*

- `(float)`**floatForKey:**`(NSString*)` *defaultName*

Looks up a value for a specified default using -objectForKey: and checks that it is a float. Returns 0.0 if it is not.

init *XOG*

- `(id)`**init**

Initializes defaults for current user calling initWithUser:

initWithContentsOfFile: *XOG*

- `(id)`**initWithContentsOfFile:**`(NSString*)` *path*

Initializes defaults for the specified *path*. Returns an object with an empty search list.

initWithUser: *XOG*

- `(id)`**initWithUser:**`(NSString*)` *userName*

Initializes defaults for the specified user calling -initWithContentsOfFile:

integerForKey: *XOG*

- `(int)`**integerForKey:**`(NSString*)` *defaultName*

Looks up a value for a specified default using -objectForKey: and checks that it is an integer. Returns 0 if it is not.

objectForKey: *XOG*

- `(id)`**objectForKey:**`(NSString*)` *defaultName*

Looks up a value for a specified default using. The lookup is performed by accessing the domains in the order given in the search list.
Returns `nil` if *defaultName* cannot be found.

persistentDomainForName: *XOG*

- `(NSDictionary*)`**persistentDomainForName:**`(NSString*)` *domainName*

Returns the persistent domain specified by *domainName*.

persistentDomainNames *XOG*

- `(NSArray*)`**persistentDomainNames**

Returns an array listing the name of all the persistent domains.

registerDefaults: *XOG*

- `(void)`**registerDefaults:**`(NSDictionary*)` *newVals*

Merges the contents of the dictionary *newVals* into the registration domain. Registration defaults may be added to or replaced using this method, but may never be removed. Thus, setting registration defaults at any point in your program guarantees that the defaults will be available thereafter.

removeObjectForKey: *XOG*

- `(void)`**removeObjectForKey:**`(NSString*)` *defaultName*

Removes the default with the specified name from the application domain.

removePersistentDomainForName: *XOG*

- (void)**removePersistentDomainForName:**(NSString*) *domainName*

Removes the persistent domain specified by *domainName* from the user defaults.
Causes a NSUserDefaultsDidChangeNotification to be posted if this is the first change to a persistent-domain since the last -synchronize .

removeVolatileDomainForName: *XOG*

- (void)**removeVolatileDomainForName:**(NSString*) *domainName*

Removes the volatile domain specified by *domainName* from the user defaults.

searchList *XOG*

- (NSMutableArray*)**searchList**

Returns an array listing the domains searched in order to look up a value in the defaults system. The order of the names in the array is the order in which the domains are searched.

setBool:forKey: *XOG*

- (void)**setBool:**(BOOL) *value*
  forKey:(NSString*) *defaultName*

Sets a boolean *value* for *defaultName* in the application domain.
The boolean *value* is stored as a string - either YES or NO. Calls -setObject:forKey: to make the change.

setFloat:forKey: *XOG*

- (void)**setFloat:**(float) *value*
  forKey:(NSString*) *defaultName*

Sets a float *value* for *defaultName* in the application domain.
Calls -setObject:forKey: to make the change.

setInteger:forKey: *XOG*

- (void)**setInteger:**(int) *value*
forKey:(NSString*) *defaultName*

Sets an integer *value* for *defaultName* in the application domain.
Calls -setObject:forKey: to make the change.

setObject:forKey: *XOG*

- (void)**setObject:**(id) *value*
forKey:(NSString*) *defaultName*

Sets an object *value* for *defaultName* in the application domain.
The *defaultName* must be a non-empty string.
The *value* must be an instance of one of the [NSString -propertyList] classes.
Causes a NSUserDefaultsDidChangeNotification to be posted if this is the first change to a persistent-domain since the last -synchronize .

setPersistentDomain:forName: *XOG*

- (void)**setPersistentDomain:**(NSDictionary*) *domain*
forName:(NSString*) *domainName*

Replaces the persistent-domain specified by *domainName* with *domain*... a dictionary containing keys and defaults values.
Raises an NSInvalidArgumentException if *domainName* already exists as a volatile-domain.
Causes a NSUserDefaultsDidChangeNotification to be posted if this is the first change to a persistent-domain since the last -synchronize .

setSearchList: *XOG*

- (void)**setSearchList:**(NSArray*) *newList*

Sets the list of the domains searched in order to look up a value in the defaults system. The order of the names in the array is the order in which the domains are searched.
On lookup, the first match is used.

setVolatileDomain:forName: *XOG*

- (void)**setVolatileDomain:**(NSDictionary*) *domain*
forName:(NSString*) *domainName*

Sets the volatile-domain specified by *domainName* to *domain*... a dictionary containing keys and defaults values.

Raises an NSInvalidArgumentException if *domainName* already exists as either a volatile-domain or a persistent-domain.

stringArrayForKey: *XOG*

- (NSArray*)**stringArrayForKey:**(NSString*) *defaultName*

Calls -arrayForKey: to get an array value for *defaultName* and checks that the array contents are string objects... if not, returns nil.

stringForKey: *XOG*

- (NSString*)**stringForKey:**(NSString*) *defaultName*

Looks up a value for a specified default using -objectForKey: and checks that it is an NSString. Returns nil if it is not.

synchronize *XOG*

- (BOOL)**synchronize**

Ensures that the in-memory and on-disk representations of the defaults are in sync. You may call this yourself, but probably don't need to since it is invoked at intervals whenever a runloop is running.

If any persistent domain is changed by reading new values from disk, an NSUserDefaultsDidChangeNotification is posted.

volatileDomainForName: *XOG*

- (NSDictionary*)**volatileDomainForName:**(NSString*) *domainName*

Returns the volatile domain specified by *domainName*.

volatileDomainNames *XOG*

- `(NSArray*)`**volatileDomainNames**

Returns an array listing the name of all the volatile domains.

NSValue *XOG*

Inherits From:	NSObject
Conforms To:	NSCopying NSCoding
Declared in:	Foundation/NSValue.h

Description

Description forthcoming.

Class Methods

value:withObjCType: *XOG*

`+ (NSValue*)`**value:** `(const void*)` *value*
withObjCType: `(const char*)` *type*

Description forthcoming.

valueWithBytes:objCType: *X¬OG*

`+ (NSValue*)`**valueWithBytes:** `(const void*)` *value*
objCType: `(const char*)` *type*

Description forthcoming.

valueWithNonretainedObject: *XOG*

`+ (NSValue*)`**valueWithNonretainedObject:** `(id)` *anObject*

Description forthcoming.

valueWithPoint: *XOG*

+ (NSValue*)**valueWithPoint:** (NSPoint) *point*
Description forthcoming.

valueWithPointer: *XOG*

+ (NSValue*)**valueWithPointer:** (const void*) *pointer*
Description forthcoming.

valueWithRange: *XOG*

+ (NSValue*)**valueWithRange:** (NSRange) *range*
Description forthcoming.

valueWithRect: *XOG*

+ (NSValue*)**valueWithRect:** (NSRect) *rect*
Description forthcoming.

valueWithSize: *XOG*

+ (NSValue*)**valueWithSize:** (NSSize) *size*
Description forthcoming.

Instance Methods

getValue: *XOG*

- (void)**getValue:**(void*) *value*
Description forthcoming.

initWithBytes:objCType: *X¬OG*

- (id)**initWithBytes:**(const void*) *data*
objCType:(const char*) *type*

Description forthcoming.

isEqualToValue: *X¬OG*

- (BOOL)**isEqualToValue:**(NSValue*) *other*

Description forthcoming.

nonretainedObjectValue *XOG*

- (id)**nonretainedObjectValue**

Description forthcoming.

objCType *XOG*

- (const char*)**objCType**

Description forthcoming.

pointValue *XOG*

- (NSPoint)**pointValue**

Description forthcoming.

pointerValue *XOG*

- (void*)**pointerValue**

Description forthcoming.

rangeValue *XOG*

- (NSRange)**rangeValue**

Description forthcoming.

rectValue *XOG*

- (NSRect)rectValue

Description forthcoming.

sizeValue *XOG*

- (NSSize)sizeValue

Description forthcoming.

NXConstantString

XOG

Inherits From: NSString : NSObject
Declared in: Foundation/NSString.h

Description

The NXConstantString class is used to hold constant 8-bit character string objects produced by the compiler where it sees @"..." in the source. The compiler generates the instances of this class - which has three instance variables -

- a pointer to the class (this is the sole ivar of NSObject)
- a pointer to the 8-bit data
- the length of the string

In older versions of the compiler, the isa variable is always set to the NXConstantString class. In newer versions a compiler option was added for GNUstep, to permit the isa variable to be set to another class, and GNUstep uses this to avoid conflicts with the default implementation of NXConstantString in the ObjC runtime library (the preprocessor is used to change all occurances of NXConstantString in the source code to NSConstantString).

Since GNUstep will generally use the GNUstep extension to the compiler, you should never refer to the constnat string class by name, but should use the [NSString +constantStringClass] method to get the actual class being used for constant strings. What follows is a dummy declaration of the class to keep the compiler happy.

GCFinalization Protocol

Declared in: Foundation/NSObject.h

Description

Description forthcoming.

XOG

Instance Methods

gcFinalize *XOG*

- (void)gcFinalize

Description forthcoming.

NSCoding Protocol

Declared in: Foundation/NSObject.h

Description

This protocol must be adopted by any class wishing to support saving and restoring instances to an archive, or copying them to remote processes via the Distributed Objects mechanism.

XOG

Instance Methods

encodeWithCoder: *XOG*

- `(void)`**encodeWithCoder:**`(NSCoder*)` *aCoder*

Description forthcoming.

initWithCoder: *XOG*

- `(id)`**initWithCoder:**`(NSCoder*)` *aDecoder*

Description forthcoming.

NSCopying Protocol

Declared in: Foundation/NSObject.h

Description

This protocol must be adopted by any class wishing to support copying - ie where instances of the class should be able to create new instances which are copies of the original and, where a class has mutable and immutable versions, where the copies are immutable.

XOG

Instance Methods

copyWithZone: *XOG*

`- (id)` **copyWithZone:**`(NSZone*)` *zone*

Called by [NSObject -copy] passing NSDefaultMallocZone() as *zone*.

This method returns a copy of the receiver and, where the receiver is a mutable variant of a class which has an immutable partner class, the object returned is an instance of that immutable class.

The new object is *not* autoreleased, and is considered to be 'owned' by the calling code... which is therefore responsible for releasing it.

In the case where the receiver is an instance of a container class, it is undefined whether contained objects are merely retained in the new copy, or are themselves copied, or whether some other mechanism entirely is used.

NSDecimalNumberBehaviors Protocol

Declared in: Foundation/NSDecimalNumber.h

Description

Description forthcoming.

X¬OG

Instance Methods

exceptionDuringOperation:error:leftOperand:rightOperand:

X¬OG

- (NSDecimalNumber*)**exceptionDuringOperation:**(SEL) *method*
error:(NSCalculationError) *error*
leftOperand:(NSDecimalNumber*) *leftOperand*
rightOperand:(NSDecimalNumber*) *rightOperand*

Description forthcoming.

roundingMode

X¬OG

- (NSRoundingMode)**roundingMode**

Description forthcoming.

scale

X¬OG

- (short)**scale**

Description forthcoming.

NSLocking Protocol

Declared in: Foundation/NSLock.h

Description

Description forthcoming.

XOG

Instance Methods

lock

XOG

- `(void)`**lock**

Description forthcoming.

unlock

XOG

- `(void)`**unlock**

Description forthcoming.

NSMutableCopying Protocol

Declared in: Foundation/NSObject.h

Description

This protocol must be adopted by any class wishing to support mutable copying - ie where instances of the class should be able to create mutable copies of themselves.

XOG

Instance Methods

mutableCopyWithZone: *XOG*

`- (id)` **mutableCopyWithZone:** `(NSZone*)` *zone*

Called by [NSObject -mutableCopy] passing NSDefaultMallocZone() as *zone*.
This method returns a copy of the receiver and, where the receiver is an immmutable variant of a class which has a mutable partner class, the object returned is an instance of that mutable class. The new object is *not* autoreleased, and is considered to be 'owned' by the calling code... which is therefore responsible for releasing it.
In the case where the receiver is an instance of a container class, it is undefined whether contained objects are merely retained in the new copy, or are themselves copied, or whether some other mechanism entirely is used.

NSObject Protocol

Declared in: Foundation/NSObject.h

Description

The NSObject protocol describes a minimal set of methods that all objects are expected to support. You should be able to send any of the messages listed in this protocol to an object, and be safe in assuming that the receiver can handle it.

XOG

Instance Methods

autorelease *XOG*

- (id)autorelease

See [NSObject -autorelease]

class *XOG*

- (Class)class

See [NSObject -class]

conformsToProtocol: *XOG*

- (BOOL)conformsToProtocol:(Protocol*) *aProtocol*

See [NSObject -conformsToProtocol:]

description *XOG*

- (NSString*)description

See [NSObject -description]

hash *XOG*

- (unsigned)**hash**

See [NSObject -hash]

isEqual: *XOG*

- (BOOL)**isEqual:**(id) *anObject*

See [NSObject -isEqual:]

isKindOfClass: *XOG*

- (BOOL)**isKindOfClass:**(Class) *aClass*

See [NSObject -isKindOfClass:]

isMemberOfClass: *XOG*

- (BOOL)**isMemberOfClass:**(Class) *aClass*

See [NSObject -isMemberOfClass:]

isProxy *XOG*

- (BOOL)**isProxy**

See [NSObject -isProxy]

performSelector: *XOG*

- (id)**performSelector:**(SEL) *aSelector*

See [NSObject -performSelector:]

performSelector:withObject: *XOG*

- (id)**performSelector:**(SEL) *aSelector*
  withObject:(id) *anObject*

See [NSObject -performSelector:withObject:]

performSelector:withObject:withObject: *XOG*

- (id)**performSelector:**(SEL) *aSelector*
  withObject:(id) *object1*
  withObject:(id) *object2*

See [NSObject -performSelector:withObject:withObject:]

release *XOG*

- (oneway void)**release**

See [NSObject -release]

respondsToSelector: *XOG*

- (BOOL)**respondsToSelector:**(SEL) *aSelector*

See [NSObject -respondsToSelector:]

retain *XOG*

- (id)**retain**

See [NSObject -retain]

retainCount *XOG*

- (unsigned)**retainCount**

See [NSObject -retainCount]

self

XOG

- `(id)`**self**

See [NSObject -self]

superclass

XOG

- `(Class)`**superclass**

See [NSObject -superclass]

zone

XOG

- `(NSZone*)`**zone**

See [NSObject -zone]

NSURLHandleClient Protocol

Declared in: Foundation/NSURLHandle.h

Description

A protocol to which clients of a handle must conform in order to rfeceive notification of events on the handle.

X¬OG

Instance Methods

URLHandle:resourceDataDidBecomeAvailable: *X¬OG*

- (void) **URLHandle:**(NSURLHandle*) *sender*
resourceDataDidBecomeAvailable:(NSData*) *newData*

Sent by the NSURLHandle object when some data becomes available from the handle. Note that this does not mean that all data has become available, only that a chunk of data has arrived.

URLHandle:resourceDidFailLoadingWithReason: *X¬OG*

- (void) **URLHandle:**(NSURLHandle*) *sender*
resourceDidFailLoadingWithReason:(NSString*) *reason*

Sent by the NSURLHandle object on resource load failure. Supplies a human readable failure *reason*.

URLHandleResourceDidBeginLoading: *X¬OG*

- (void)**URLHandleResourceDidBeginLoading:**(NSURLHandle*) *sender*

Sent by the NSURLHandle object when it begins loading resource data.

URLHandleResourceDidCancelLoading: *X¬OG*

- (void)**URLHandleResourceDidCancelLoading:**(NSURLHandle*) *sender*

Sent by the NSURLHandle object when resource loading is canceled by programmatic request (rather than by failure).

URLHandleResourceDidFinishLoading: $X\neg OG$

- (void)**URLHandleResourceDidFinishLoading:**(NSURLHandle*) *sender*

Sent by the NSURLHandle object when it completes loading resource data.

RunLoopEvents Protocol

Declared in: Foundation/NSRunLoop.h

Description

Description forthcoming.

XOG

Instance Methods

receivedEvent:type:extra:forMode:

XOG

```
- (void)receivedEvent:(void*) data
                 type:(RunLoopEventType) type
                extra:(void*) extra
              forMode:(NSString*) mode
```

Description forthcoming.

timedOutEvent:type:forMode:

XOG

```
- (NSDate*)timedOutEvent:(void*) data
                    type:(RunLoopEventType) type
                 forMode:(NSString*) mode
```

Description forthcoming.