MiscFile (Unix)

Inherits From: **MiscGraphNode**

Declared In: <misckit/MiscFile+Unix.h>

Category Description

The idea behind this category was to isolate all the lower level UNIX calls that all the other MiscFile categories would need, so if/when the underlying operating system changes (with some of the OpenStep releases), code changes would only have to be made here. Therefore if you add any other categories of MiscFile, please don't make any direct Unix calls unless you absolutely have to. Add a method here, which will make the actual call to the system for you. Also if you add any methods to this category make sure to reset the errorCode ivar depending if the call was successful or not. See the source code to any of the existing methods to see how it is done (there is a private method to help out).

Method Types

Use stat(2) or lstat(2)	+ useLinkInfo

	+ setUseLinkInfo:
Caching the stat structure	- enableCaching - setEnableCaching:
File manipulation	- access: - chmod: - chown:group: - link: - readlink:pathLen: - lstat - stat - whichStat - symlink: - errorCode

Class Methods

useLinkInfo

+ (BOOL)**useLinkInfo**

This method only makes a difference when an instance of MiscFile represents a symbolic link. In most cases you will probably want the stat information to be from the file the link references, which is the default. (in that case you would use stat(2)). In some cases though, you may want the stat information to be about the link itself (you would use lstat(2)). By setting **setUseLinkInfo**: YES and using **whichStat**, lstat will be used to fill the stat structure instead of stat if the MiscFile represents a symbolic link. See **whichStat** for more information.

See also: + setUseLinkInfo:, ± whichStat

setUseLinkInfo:

+ **setUseLinkInfo:**(BOOL)*linkInfo*

Sets whether the stat structure passed back from the **whichStat** method contains information about the link, or the file the link references. The second choice is the default.

See also: + useLinkInfo, ± whichStat

Instance Methods

access:

- (int)**access:**(int)*mode*

Same as access(2). *mode* can be either be the same parameter that access(2) takes or it's easier to read equivalent of one or an ORed combination of one of the following: MISCFILE_READ, MISCFILE_WRITE, MISCFILE_EXECUTE and MISCFILE_EXISTS. If you have the access specified by *mode*, then MISCFILE_SUCCESS is returned. Otherwise MISCFILE_ERROR will be returned.

For example:

```
if ([myFile access: MISCFILE_READ | MISCFILE_WRITE] == MISCFILE_SUCCESS)
   // The file is both readable and writable by the current user.
else
   // The file is either not readable or not writable.
```

See also: ± permissions (MiscFile+Info)

chmod:

- (int)**chmod:**(unsigned short)*mode*

The parameter *mode* is the same octal number that chmod(1) and chmod(2) expect. See the man pages for a description of what each of the bits means, or alternatively have a look at the **setPermissions**::, **addPermissions**:: and **removePermissons**:: methods, which have much easier to understand arguments.

See also: ± *permissions (MiscFile+Modification)

chown:group:

- (int)**chown:**(uid_t)*owner*
  group:(gid_t)*group*

Same as chown(2). See the man pages for a description of the arguments.

enableCaching

- (BOOL)**enableCaching**

Returns YES if the stat structure is cached. If NO is returned, it means that every time a method goes through either the -stat or -lstat method, that the file will actually get stated again, instead of just taking the information from the last stat.

See also: ± setEnableCaching:

errorCode

- (int)**errorCode**

Returns the same as the global errno that is set when an error occurs during any one of the UNIX file functions. For instance if a method returns either MISCFILE_ERROR or NULL (in the case of stat, lstat, whichStat), then you could get the error number to see what went wrong.

link:

- (int)**link:**(const char *)*newpath*

Attempts to create a hard link of the receiver with path *newPath*. If successful, MISCFILE_SUCCESS is returned, otherwise MISCFILE_ERROR.

See also: ± symlink

lstat

- (struct stat *)**lstat**

Returns a pointer to the file's stat information if the file was a symbolic link. If not, then nil will be returned. You do not need to free the structure that is returned.

See also: ± stat, ± whichStat

setEnableCaching:

- **setEnableCaching:**(BOOL)*cache*

Sets whether caching of the stat information is enabled. If so, the file will only be stated once, and all other calls which need the stat information will use the previous information. If set to YES (which is the default) there exists the possibility that the stat information will not be completely up to date with the state of the filesystem.

See also: ± enableCaching

stat

- (struct stat *)**stat**

Returns a pointer to the file's stat information. If the file is a symbolic link then the stat information refers to the file that the link references and not information about the link itself. Use -lstat for information about the link. You do not need to free the structure returned.

See also: ± lstat, ± whichStat

symlink:

- (int)**symlink:**(const char *)*newpath*

Tries to create a symbolic link to the receiver called newpath. If successful, MISCFILE_SUCCESS is returned, otherwise MISCFILE_ERROR. This method uses the UNIX call symlink(2).

See also: ± link

whichStat

- (struct stat *)**whichStat**

This method only matters when the file the receiver represents is a symbolic link. It checks the value of [MiscFile useLinkInfo] to determine whether the stat structure returned should contain information about the symbolic link or the file it references. This method is for use when the stat information returned would matter, depending upon the file being a symbolic link or not. For example, see the -size method in MiscFile+Info. If useLinkInfo is YES and the file is a symbolic link, size will return the file size of the link. If useLinkInfo is NO, size would return the size of the file the link references. If the file the receiver represents is not a symbolic link, then this method will return stat information about the file itself.

See also: ± link