MiscFile (Info)

Inherits From: **MiscGraphNode**

Declared In: <misckit/MiscFile.h>

Category Description

This category allows you to extract some useful information about a MiscFile instance. Most methods here are just more human readable names and invoke methods in the MiscFile+Unix category. If the file the instance of MiscFile represents is a symbolic link, some of the methods (the ones that get their information from either stat(2) or lstat(2)) will return either information about the file they represent, or information about the link itself. See **+useLinkInfo** (MiscFile+Unix category). The default is to return information about the file the MiscFile instance represents. If any of the methods that return a short or long int returns MISCFILE_ERROR, you can use the method ±**errorCode** (MiscFile+Unix) to find out what went wrong. For methods that don't return an integer (like -stat, -lstat, and -whichStat), they will return NULL if there was an error. You can still use the ±**errorCode** method to find out what went wrong.

Also, by default, stat caching is enabled (see **enableCaching**/**setEnableCaching**:), so the methods that get their information via stat or lstat may not get the most up to date information from the filesystem. If you need to make sure you get the most up to date information (though giving up speed), make sure to turn off caching. Here are some snippets of code illustrating enabling/disabling caching:

```
- addPermissions: (int)perms for: (int) who
{
   BOOL caching = [self enableCaching];

   ....
   // Turn off stat caching so we get the most up to date permissions.
   [self setEnableCaching: NO];
   realPerms = [self permissions];
   if (caching == YES)
      // Only reenable caching if it was set to YES before we set it to NO.
      [self setEnableCaching: YES];
   ....
}


- showAttributes: sender
{
   // If you are going to using many methods that get their information
   // from stat and you want the most up to date information, disable caching,
   // make one call, and reenable caching again so every call will not stat
   // the file again.
   BOOL caching = [self enableCaching];

   [self setEnableCaching: NO];
   size = [aFile size];
   if (caching == YES)
      [self setEnableCaching: YES];
```

```
    permissions = [aFile permissions];
    owner = [aFile owner];
    group = [aFile group];
    ...
}
```

Method Types

Stat information	- blocksAllocated - device - deviceType - group - hardLinks - inode - isDirectory - isSymbolicLink - lastAccessTime - lastModifyTime - lastStatusChangeTime ± optimalBlockSize - owner - permissions - size
Checking permissions	- hasPermissions:for: - doesExist - readAccess - executeAccess

- writeAccess

Instance Methods

blocksAllocated

- (long)**blocksAllocated**

Returns the actual blocks allocated for the file. This is the st_blocks member of the stat structure.

device

- (dev_t)**device**

Returns the device the inode resides on. This is the st_dev member of the stat structure.

See also: ± deviceType

deviceType

- (dev_t)**deviceType**

Returns the device type that the inode resides on. This is the st_rdev member of the stat structure.

See also: ± device

doesExist

- (BOOL)**doesExist**

Returns whether the receiver really still exists on the filesystem. It had to have existed when initially creating the instance, but something may have happened to it since then. You can use this method to make sure it is still there.

executeAccess

- (BOOL)**executeAccess**

Returns YES if you have execute access to the file the receiver represents.

See also: ± readAccess, ± writeAccess

group

- (MiscUserGroup *)**group**

Returns an instance of MiscUserGroup that represents the group-id of the file's owner. You are responsible for freeing the instance returned. This information comes from the st_gid member of the stat structure If something went wrong when stating the file, nil is returned.

See also: ± owner

hardLinks

- (short)**hardLinks**

Returns the number of hard links to the file the receiver represents. This is the st_nlink member of the stat structure.

hasPermissions:for:

- (BOOL)**hasPermissions:**(int)*perms*
  for:(int)*who*

Returns YES if *who* has the given *perms* on the file the receiver represents. *Who* can be either MISCFILE_OWNER, MISCFILE_GROUP, MISCFILE_OTHER or any ORed combination. The parameter *perms* can be either MISCFILE_READ, MISCFILE_WRITE, MISCFILE_EXECUTE or any ORed combination. For example, using the given file as an example:

```
-rw-r--r--  1 todd       90571 Jan 13 19:09 .newsrc

[newsrc hasPermissions: MISCFILE_READ for: MISCFILE_OWNER | MISCFILE_OTHER]
     // would return YES since both the owner and others can read the file.
[newsrc hasPermissions: MISCFILE_WRITE for: MISCFILE_OWNER  | MISCFILE_GROUP]
     // would return NO since the file is not group writable.
```

See also: ± setPermissions:for: (MiscFile+Modification)

inode

- (ino_t)**inode**

Returns the receiver's inode. This is the st_ino member of the stat structure.

isDirectory

- (BOOL)**isDirectory**

Returns YES if the file the receiver represents is a directory.

See also: ± **isSymbolicLink**

isSymbolicLink

- (BOOL)**isSymbolicLink**

Returns YES if the file the receiver represents is a symbolic link.

See also: ± **isDirectory**

lastAccessTime

- (MiscTime *)**lastAccessTime**

Returns an instance of MiscTime that represents the time the file was last accessed. The time is determined from the st_atime member of the stat structure. You are responsible for freeing the instance returned. If something went wrong when stating the file, nil is returned.

See also: ± **lastModifyTime,** ± **lastStatusChangeTime**

lastModifyTime

- (MiscTime *)**lastModifyTime**

Returns an instance of MiscTime that represents the time the file was last modified. The time is determined from the st_mtime member of the stat structure. You are responsible for freeing the instance returned. If something went wrong when stating the file, nil is returned.

See also: ± **lastAccessTime,** ± **lastStatusChangeTime**

lastStatusChangeTime

- (MiscTime *)**lastStatusChangeTime**

Returns an instance of MiscTime that represents the time the file's status was last changed. The time is determined from the st_ctime member of the stat structure. You are responsible for freeing the instance returned. If something went wrong when stating the file, nil is returned.

See also: ± lastAccessTime, ± lastModifyTime

optimalBlockSize

- (long)**optimalBlockSize**

Returns the optimal block size for file i/o operations. The information is derived from the st_blksize member of the stat structure.

owner

- (MiscUser *)**owner**

Returns an instance of MiscUser that represents the owner of the file. The owner information is derived from the st_uid member of the stat structure. You are responsible for freeing the returned instance. If something goes wrong when stating the file, nil will be returned.

permissions

- (u_short)**permissions**

Returns an unsigned short integer describing the permissions the file has. The permissions are derived from the st_mode member of the stat structure. Check the documentation for stat(2) to see what each individual bit represents.

See also: ± hasPermissions:for: (MiscFile+Info), ± setPermissions:for (MiscFile+Modification)

readAccess

- (BOOL)**readAccess**

Returns YES you have read access to the file the instance represents.

See also: ± writeAccess, ± executeAccess

size

- (off_t)**size**

Returns the total size of the file in bytes. This information comes from the st_size member of the stat structure.

writeAccess

- (BOOL)**writeAccess**

Returns YES if you have write access to the file the instance represents.

See also: ± executeAccess, ± readAccess