NAME

chflags, **lchflags**, **fchflags**, **chflagsat** - set file flags

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/stat.h>
#include <unistd.h>

int
chflags(*const char *path*, *unsigned long flags*);

int
lchflags(*const char *path*, *unsigned long flags*);

int
fchflags(*int fd*, *unsigned long flags*);

int
chflagsat(*int fd*, *const char *path*, *unsigned long flags*, *int atflag*);

DESCRIPTION

The file whose name is given by *path* or referenced by the descriptor *fd* has its flags changed to *flags*.

The **lchflags**() system call is like **chflags**() except in the case where the named file is a symbolic link, in which case **lchflags**() will change the flags of the link itself, rather than the file it points to.

The **chflagsat**() is equivalent to either **chflags**() or **lchflags**() depending on the *atflag* except in the case where *path* specifies a relative path. In this case the file to be changed is determined relative to the directory associated with the file descriptor *fd* instead of the current working directory. The values for the *atflag* are constructed by a bitwise-inclusive OR of flags from the following list, defined in <*fcntl.h*>:

AT_SYMLINK_NOFOLLOW
: If *path* names a symbolic link, then the flags of the symbolic link are changed.

AT_RESOLVE_BENEATH
: Only walk paths below the directory specified by the *fd* descriptor. See the description of the O_RESOLVE_BENEATH flag in the open(2) manual page.

AT_EMPTY_PATH
: If the *path* argument is an empty string, operate on the file or directory referenced by the descriptor *fd*. If *fd* is equal to AT_FDCWD, operate on the current working directory.

If **chflagsat**() is passed the special value AT_FDCWD in the *fd* parameter, the current working directory is used. If also *atflag* is zero, the behavior is identical to a call to **chflags**().

The flags specified are formed by *or*'ing the following values

SF_APPEND	The file may only be appended to.
SF_ARCHIVED	The file has been archived. This flag means the opposite of the DOS, Windows and CIFS FILE_ATTRIBUTE_ARCHIVE attribute. This flag has been deprecated, and may be removed in a future release.
SF_IMMUTABLE	The file may not be changed.
SF_NOUNLINK	The file may not be renamed or deleted.
SF_SNAPSHOT	The file is a snapshot file.
UF_APPEND	The file may only be appended to.
UF_ARCHIVE	The file needs to be archived. This flag has the same meaning as the DOS, Windows and CIFS FILE_ATTRIBUTE_ARCHIVE attribute. Filesystems in FreeBSD may or may not have special handling for this flag. For instance, ZFS tracks changes to files and will set this bit when a file is updated. UFS only stores the flag, and relies on the application to change it when needed.
UF_HIDDEN	The file may be hidden from directory listings at the application's discretion. The file has the DOS, Windows and CIFS FILE_ATTRIBUTE_HIDDEN attribute.
UF_IMMUTABLE	The file may not be changed.
UF_NODUMP	Do not dump the file.
UF_NOUNLINK	The file may not be renamed or deleted.
UF_OFFLINE	The file is offline, or has the Windows and CIFS FILE_ATTRIBUTE_OFFLINE attribute. Filesystems in FreeBSD store and display this flag, but do not provide any special handling when it is set.
UF_OPAQUE	The directory is opaque when viewed through a union stack.
UF_READONLY	The file is read only, and may not be written or appended. Filesystems may use this flag to maintain compatibility with the DOS, Windows and CIFS FILE_ATTRIBUTE_READONLY attribute.
UF_REPARSE	The file contains a Windows reparse point and has the Windows and CIFS FILE_ATTRIBUTE_REPARSE_POINT attribute.
UF_SPARSE	The file has the Windows FILE_ATTRIBUTE_SPARSE_FILE attribute. This may also be used by a filesystem to indicate a sparse file.

UF_SYSTEM	The file has the DOS, Windows and CIFS FILE_ATTRIBUTE_SYSTEM attribute. Filesystems in FreeBSD may store and display this flag, but do not provide any special handling when it is set.

If one of SF_IMMUTABLE, SF_APPEND, or SF_NOUNLINK is set a non-super-user cannot change any flags and even the super-user can change flags only if securelevel is 0. (See init(8) for details.)

The UF_IMMUTABLE, UF_APPEND, UF_NOUNLINK, UF_NODUMP, and UF_OPAQUE flags may be set or unset by either the owner of a file or the super-user.

The SF_IMMUTABLE, SF_APPEND, SF_NOUNLINK, and SF_ARCHIVED flags may only be set or unset by the super-user. Attempts to toggle these flags by non-super-users are rejected. These flags may be set at any time, but normally may only be unset when the system is in single-user mode. (See init(8) for details.)

The implementation of all flags is filesystem-dependent. See the description of the UF_ARCHIVE flag above for one example of the differences in behavior. Care should be exercised when writing applications to account for support or lack of support of these flags in various filesystems.

The SF_SNAPSHOT flag is maintained by the system and cannot be toggled.

RETURN VALUES

Upon successful completion, the value 0 is returned; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **chflags**() system call will fail if:

[ENOTDIR]	A component of the path prefix is not a directory.
[ENAMETOOLONG]	A component of a pathname exceeded 255 characters, or an entire path name exceeded 1023 characters.
[ENOENT]	The named file does not exist.
[EACCES]	Search permission is denied for a component of the path prefix.
[ELOOP]	Too many symbolic links were encountered in translating the pathname.

[EPERM]	The effective user ID does not match the owner of the file and the effective user ID is not the super-user.
[EPERM]	One of SF_IMMUTABLE, SF_APPEND, or SF_NOUNLINK is set and the user is either not the super-user or securelevel is greater than 0.
[EPERM]	A non-super-user attempted to toggle one of SF_ARCHIVED, SF_IMMUTABLE, SF_APPEND, or SF_NOUNLINK.
[EPERM]	An attempt was made to toggle the SF_SNAPSHOT flag.
[EROFS]	The named file resides on a read-only file system.
[EFAULT]	The *path* argument points outside the process's allocated address space.
[EIO]	An I/O error occurred while reading from or writing to the file system.
[EINTEGRITY]	Corrupted data was detected while reading from the file system.
[EOPNOTSUPP]	The underlying file system does not support file flags, or does not support all of the flags set in *flags*.

The **fchflags**() system call will fail if:

[EBADF]	The descriptor is not valid.
[EINVAL]	The *fd* argument refers to a socket, not to a file.
[EPERM]	The effective user ID does not match the owner of the file and the effective user ID is not the super-user.
[EPERM]	One of SF_IMMUTABLE, SF_APPEND, or SF_NOUNLINK is set and the user is either not the super-user or securelevel is greater than 0.
[EPERM]	A non-super-user attempted to toggle one of SF_ARCHIVED, SF_IMMUTABLE, SF_APPEND, or SF_NOUNLINK.
[EPERM]	An attempt was made to toggle the SF_SNAPSHOT flag.
[EROFS]	The file resides on a read-only file system.

[EIO]	An I/O error occurred while reading from or writing to the file system.
[EINTEGRITY]	Corrupted data was detected while reading from the file system.
[EOPNOTSUPP]	The underlying file system does not support file flags, or does not support all of the flags set in *flags*.
[ENOTCAPABLE]	*path* is an absolute path, or contained a ".." component leading to a directory outside of the directory hierarchy specified by *fd*, and the process is in capability mode or the AT_RESOLVE_BENEATH flag was specified.

SEE ALSO

chflags(1), fflagstostr(3), strtofflags(3), init(8), mount_unionfs(8)

HISTORY

The **chflags**() and **fchflags**() system calls first appeared in 4.4BSD. The **lchflags**() system call first appeared in FreeBSD 5.0. The **chflagsat**() system call first appeared in FreeBSD 10.0.