NAME

ioctl_iflags – ioctl() operations for inode flags

DESCRIPTION

Various Linux filesystems support the notion of *inode flags*—attributes that modify the semantics of files and directories. These flags can be retrieved and modified using two **ioctl**(2) operations:

```
int attr;
fd = open("pathname", ...);

ioctl(fd, FS_IOC_GETFLAGS, &attr);  /* Place current flags
                                       in 'attr' */
attr |= FS_NOATIME_FL;              /* Tweak returned bit mask */
ioctl(fd, FS_IOC_SETFLAGS, &attr);  /* Update flags for inode
                                       referred to by 'fd' */
```

The **lsattr**(1) and **chattr**(1) shell commands provide interfaces to these two operations, allowing a user to view and modify the inode flags associated with a file.

The following flags are supported (shown along with the corresponding letter used to indicate the flag by **lsattr**(1) and **chattr**(1)):

FS_APPEND_FL 'a'
: The file can be opened only with the **O_APPEND** flag. (This restriction applies even to the superuser.) Only a privileged process (**CAP_LINUX_IMMUTABLE**) can set or clear this attribute.

FS_COMPR_FL 'c'
: Store the file in a compressed format on disk. This flag is *not* supported by most of the mainstream filesystem implementations; one exception is **btrfs**(5).

FS_DIRSYNC_FL 'D' (since Linux 2.6.0)
: Write directory changes synchronously to disk. This flag provides semantics equivalent to the **mount**(2) **MS_DIRSYNC** option, but on a per-directory basis. This flag can be applied only to directories.

FS_IMMUTABLE_FL 'i'
: The file is immutable: no changes are permitted to the file contents or metadata (permissions, timestamps, ownership, link count and so on). (This restriction applies even to the superuser.) Only a privileged process (**CAP_LINUX_IMMUTABLE**) can set or clear this attribute.

FS_JOURNAL_DATA_FL 'j'
: Enable journaling of file data on **ext3**(5) and **ext4**(5) filesystems. On a filesystem that is journaling in *ordered* or *writeback* mode, a privileged (**CAP_SYS_RESOURCE**) process can set this flag to enable journaling of data updates on a per-file basis.

FS_NOATIME_FL 'A'
: Don't update the file last access time when the file is accessed. This can provide I/O performance benefits for applications that do not care about the accuracy of this timestamp. This flag provides functionality similar to the **mount**(2) **MS_NOATIME** flag, but on a per-file basis.

FS_NOCOW_FL 'C' (since Linux 2.6.39)
: The file will not be subject to copy-on-write updates. This flag has an effect only on filesystems that support copy-on-write semantics, such as Btrfs. See **chattr**(1) and **btrfs**(5).

FS_NODUMP_FL 'd'
: Don't include this file in backups made using **dump**(8).

FS_NOTAIL_FL 't'
: This flag is supported only on Reiserfs. It disables the Reiserfs tail-packing feature, which tries to pack small files (and the final fragment of larger files) into the same disk block as the file metadata.

FS_PROJINHERIT_FL 'P' (since Linux 4.5)
Inherit the quota project ID. Files and subdirectories will inherit the project ID of the directory. This flag can be applied only to directories.

FS_SECRM_FL 's'
Mark the file for secure deletion. This feature is not implemented by any filesystem, since the task of securely erasing a file from a recording medium is surprisingly difficult.

FS_SYNC_FL 'S'
Make file updates synchronous. For files, this makes all writes synchronous (as though all opens of the file were with the **O_SYNC** flag). For directories, this has the same effect as the **FS_DIRSYNC_FL** flag.

FS_TOPDIR_FL 'T'
Mark a directory for special treatment under the Orlov block-allocation strategy. See **chattr**(1) for details. This flag can be applied only to directories and has an effect only for ext2, ext3, and ext4.

FS_UNRM_FL 'u'
Allow the file to be undeleted if it is deleted. This feature is not implemented by any filesystem, since it is possible to implement file-recovery mechanisms outside the kernel.

In most cases, when any of the above flags is set on a directory, the flag is inherited by files and subdirectories created inside that directory. Exceptions include **FS_TOPDIR_FL**, which is not inheritable, and **FS_DIRSYNC_FL**, which is inherited only by subdirectories.

CONFORMING TO
Inode flags are a nonstandard Linux extension.

NOTES
In order to change the inode flags of a file using the **FS_IOC_SETFLAGS** operation, the effective user ID of the caller must match the owner of the file, or the caller must have the **CAP_FOWNER** capability.

SEE ALSO
chattr(1), **lsattr**(1), **mount**(2), **btrfs**(5), **ext4**(5), **xfs**(5), **xattr**(7), **mount**(8)

COLOPHON
This page is part of release 5.02 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at https://www.kernel.org/doc/man−pages/.