NAME

statfs - get file system statistics

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/param.h>
#include <sys/mount.h>

int
statfs(*const char *path, struct statfs *buf*);

int
fstatfs(*int fd, struct statfs *buf*);

DESCRIPTION

The **statfs**() system call returns information about a mounted file system. The *path* argument is the path name of any file within the mounted file system. The *buf* argument is a pointer to a *statfs* structure defined as follows:

```
typedef struct fsid { int32_t val[2]; } fsid_t; /* file system id type */

/*
 * filesystem statistics
 */

#define    MFSNAMELEN    16                /* length of type name including null */
#define    MNAMELEN      1024              /* size of on/from name bufs */
#define    STATFS_VERSION          0x20140518          /* current version number */

struct statfs {
uint32_t f_version;             /* structure version number */
uint32_t f_type;                /* type of filesystem */
uint64_t f_flags;               /* copy of mount exported flags */
uint64_t f_bsize;               /* filesystem fragment size */
uint64_t f_iosize;              /* optimal transfer block size */
uint64_t f_blocks;              /* total data blocks in filesystem */
uint64_t f_bfree;               /* free blocks in filesystem */
int64_t     f_bavail;           /* free blocks avail to non-superuser */
```

```
uint64_t f_files;               /* total file nodes in filesystem */
int64_t     f_ffree;            /* free nodes avail to non-superuser */
uint64_t f_syncwrites;                  /* count of sync writes since mount */
uint64_t f_asyncwrites;                 /* count of async writes since mount */
uint64_t f_syncreads;                   /* count of sync reads since mount */
uint64_t f_asyncreads;                  /* count of async reads since mount */
uint64_t f_spare[10];                   /* unused spare */
uint32_t f_namemax;                     /* maximum filename length */
uid_t       f_owner;                    /* user that mounted the filesystem */
fsid_t      f_fsid;             /* filesystem id */
char        f_charspare[80];        /* spare string space */
char        f_fstypename[MFSNAMELEN]; /* filesystem type name */
char        f_mntfromname[MNAMELEN];  /* mounted filesystem */
char        f_mntonname[MNAMELEN];    /* directory on which mounted */
};
```

The flags that may be returned include:

MNT_RDONLY — The file system is mounted read-only; Even the super-user may not write on it.

MNT_NOEXEC — Files may not be executed from the file system.

MNT_NOSUID — Setuid and setgid bits on files are not honored when they are executed.

MNT_SYNCHRONOUS — All I/O to the file system is done synchronously.

MNT_ASYNC — No file system I/O is done synchronously.

MNT_SOFTDEP — Soft updates being done (see ffs(7)).

MNT_GJOURNAL — Journaling with gjournal is enabled (see gjournal(8)).

MNT_SUIDDIR — Special handling of SUID bit on directories.

MNT_UNION — Union with underlying file system.

MNT_NOSYMFOLLOW — Symbolic links are not followed.

MNT_NOCLUSTERR	Read clustering is disabled.
MNT_NOCLUSTERW	Write clustering is disabled.
MNT_MULTILABEL	Mandatory Access Control (MAC) support for individual objects (see mac(4)).
MNT_ACLS	Access Control List (ACL) support enabled.
MNT_LOCAL	The file system resides locally.
MNT_QUOTA	The file system has quotas enabled on it.
MNT_ROOTFS	Identifies the root file system.
MNT_EXRDONLY	The file system is exported read-only.
MNT_NOATIME	Updating of file access times is disabled.
MNT_USER	The file system has been mounted by a user.
MNT_EXPORTED	The file system is exported for both reading and writing.
MNT_DEFEXPORTED	The file system is exported for both reading and writing to any Internet host.
MNT_EXPORTANON	The file system maps all remote accesses to the anonymous user.
MNT_EXKERB	The file system is exported with Kerberos uid mapping.
MNT_EXPUBLIC	The file system is exported publicly (WebNFS).

Fields that are undefined for a particular file system are set to -1. The **fstatfs**() system call returns the same information about an open file referenced by descriptor *fd*.

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 **statfs**() system call fails if one or more of the following are true:

[ENOTDIR] A component of the path prefix of *path* is not a directory.

[ENAMETOOLONG] The length of a component of *path* exceeds 255 characters, or the length of *path* exceeds 1023 characters.

[ENOENT] The file referred to by *path* does not exist.

[EACCES] Search permission is denied for a component of the path prefix of *path*.

[ELOOP] Too many symbolic links were encountered in translating *path*.

[EFAULT] The *buf* or *path* argument points to an invalid address.

[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.

The **fstatfs**() system call fails if one or more of the following are true:

[EBADF] The *fd* argument is not a valid open file descriptor.

[EFAULT] The *buf* argument points to an invalid address.

[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.

NOTES

The fields in the *statfs* structure have been defined to provide the parameters relevant for traditional file systems. For some other file systems, values that have similar, but not identical, semantics to those described above may be returned. An example is msdosfs, which in case of FAT12 or FAT16 file systems reports the number of available and of free root directory entries instead of inodes (where 1 to 21 such directory entries are required to store each file or directory name or disk label).

SEE ALSO

fhstatfs(2), getfsstat(2)

HISTORY

The **statfs**() system call first appeared in 4.4BSD.