NAME

statfs, fstatfs − get filesystem statistics

LIBRARY

Standard C library (*libc*, *−lc*)

SYNOPSIS

#include <sys/vfs.h> /* or <sys/statfs.h> */

[[deprecated]] int statfs(const char ***_path_, struct statfs ***_buf_**);**
[[deprecated]] int fstatfs(int _fd_**, struct statfs ***_buf_**);**

DESCRIPTION

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

```
struct statfs {
    __fsword_t f_type;    /* Type of filesystem (see below) */
    __fsword_t f_bsize;   /* Optimal transfer block size */
    fsblkcnt_t f_blocks;  /* Total data blocks in filesystem */
    fsblkcnt_t f_bfree;   /* Free blocks in filesystem */
    fsblkcnt_t f_bavail;  /* Free blocks available to
                             unprivileged user */
    fsfilcnt_t f_files;   /* Total inodes in filesystem */
    fsfilcnt_t f_ffree;   /* Free inodes in filesystem */
    fsid_t     f_fsid;    /* Filesystem ID */
    __fsword_t f_namelen; /* Maximum length of filenames */
    __fsword_t f_frsize;  /* Fragment size (since Linux 2.6) */
    __fsword_t f_flags;   /* Mount flags of filesystem
                             (since Linux 2.6.36) */
    __fsword_t f_spare[xxx];
                    /* Padding bytes reserved for future use */
};
```

The following filesystem types may appear in *f_type*:

```
ADFS_SUPER_MAGIC      0xadf5
AFFS_SUPER_MAGIC      0xadff
AFS_SUPER_MAGIC       0x5346414f
ANON_INODE_FS_MAGIC   0x09041934 /* Anonymous inode FS (for
                                    pseudofiles that have no name;
                                    e.g., epoll, signalfd, bpf) */
AUTOFS_SUPER_MAGIC    0x0187
BDEVFS_MAGIC          0x62646576
BEFS_SUPER_MAGIC      0x42465331
BFS_MAGIC             0x1badface
BINFMTFS_MAGIC        0x42494e4d
BPF_FS_MAGIC          0xcafe4a11
BTRFS_SUPER_MAGIC     0x9123683e
BTRFS_TEST_MAGIC      0x73727279
CGROUP_SUPER_MAGIC    0x27e0eb   /* Cgroup pseudo FS */
CGROUP2_SUPER_MAGIC   0x63677270 /* Cgroup v2 pseudo FS */
CIFS_MAGIC_NUMBER     0xff534d42
CODA_SUPER_MAGIC      0x73757245
COH_SUPER_MAGIC       0x012ff7b7
CRAMFS_MAGIC          0x28cd3d45
DEBUGFS_MAGIC         0x64626720
DEVFS_SUPER_MAGIC     0x1373     /* Linux 2.6.17 and earlier */
DEVPTS_SUPER_MAGIC    0x1cd1
```

```
ECRYPTFS_SUPER_MAGIC  0xf15f
EFIVARFS_MAGIC        0xde5e81e4
EFS_SUPER_MAGIC       0x00414a53
EXT_SUPER_MAGIC       0x137d     /* Linux 2.0 and earlier */
EXT2_OLD_SUPER_MAGIC  0xef51
EXT2_SUPER_MAGIC      0xef53
EXT3_SUPER_MAGIC      0xef53
EXT4_SUPER_MAGIC      0xef53
F2FS_SUPER_MAGIC      0xf2f52010
FUSE_SUPER_MAGIC      0x65735546
FUTEXFS_SUPER_MAGIC   0xbad1dea  /* Unused */
HFS_SUPER_MAGIC       0x4244
HOSTFS_SUPER_MAGIC    0x00c0ffee
HPFS_SUPER_MAGIC      0xf995e849
HUGETLBFS_MAGIC       0x958458f6
ISOFS_SUPER_MAGIC     0x9660
JFFS2_SUPER_MAGIC     0x72b6
JFS_SUPER_MAGIC       0x3153464a
MINIX_SUPER_MAGIC     0x137f     /* original minix FS */
MINIX_SUPER_MAGIC2    0x138f     /* 30 char minix FS */
MINIX2_SUPER_MAGIC    0x2468     /* minix V2 FS */
MINIX2_SUPER_MAGIC2   0x2478     /* minix V2 FS, 30 char names */
MINIX3_SUPER_MAGIC    0x4d5a     /* minix V3 FS, 60 char names */
MQUEUE_MAGIC          0x19800202 /* POSIX message queue FS */
MSDOS_SUPER_MAGIC     0x4d44
MTD_INODE_FS_MAGIC    0x11307854
NCP_SUPER_MAGIC       0x564c
NFS_SUPER_MAGIC       0x6969
NILFS_SUPER_MAGIC     0x3434
NSFS_MAGIC            0x6e736673
NTFS_SB_MAGIC         0x5346544e
OCFS2_SUPER_MAGIC     0x7461636f
OPENPROM_SUPER_MAGIC  0x9fa1
OVERLAYFS_SUPER_MAGIC 0x794c7630
PIPEFS_MAGIC          0x50495045
PROC_SUPER_MAGIC      0x9fa0     /* /proc FS */
PSTOREFS_MAGIC        0x6165676c
QNX4_SUPER_MAGIC      0x002f
QNX6_SUPER_MAGIC      0x68191122
RAMFS_MAGIC           0x858458f6
REISERFS_SUPER_MAGIC  0x52654973
ROMFS_MAGIC           0x7275
SECURITYFS_MAGIC      0x73636673
SELINUX_MAGIC         0xf97cff8c
SMACK_MAGIC           0x43415d53
SMB_SUPER_MAGIC       0x517b
SMB2_MAGIC_NUMBER     0xfe534d42
SOCKFS_MAGIC          0x534f434b
SQUASHFS_MAGIC        0x73717368
SYSFS_MAGIC           0x62656572
SYSV2_SUPER_MAGIC     0x012ff7b6
SYSV4_SUPER_MAGIC     0x012ff7b5
TMPFS_MAGIC           0x01021994
TRACEFS_MAGIC         0x74726163
```

```
UDF_SUPER_MAGIC       0x15013346
UFS_MAGIC             0x00011954
USBDEVICE_SUPER_MAGIC 0x9fa2
V9FS_MAGIC            0x01021997
VXFS_SUPER_MAGIC      0xa501fcf5
XENFS_SUPER_MAGIC     0xabba1974
XENIX_SUPER_MAGIC     0x012ff7b4
XFS_SUPER_MAGIC       0x58465342
_XIAFS_SUPER_MAGIC    0x012fd16d /* Linux 2.0 and earlier */
```

Most of these MAGIC constants are defined in */usr/include/linux/magic.h*, and some are hardcoded in kernel sources.

The *f_flags* field is a bit mask indicating mount options for the filesystem. It contains zero or more of the following bits:

ST_MANDLOCK
Mandatory locking is permitted on the filesystem (see **fcntl**(2)).

ST_NOATIME
Do not update access times; see **mount**(2).

ST_NODEV
Disallow access to device special files on this filesystem.

ST_NODIRATIME
Do not update directory access times; see **mount**(2).

ST_NOEXEC
Execution of programs is disallowed on this filesystem.

ST_NOSUID
The set-user-ID and set-group-ID bits are ignored by **exec**(3) for executable files on this filesystem

ST_RDONLY
This filesystem is mounted read-only.

ST_RELATIME
Update atime relative to mtime/ctime; see **mount**(2).

ST_SYNCHRONOUS
Writes are synched to the filesystem immediately (see the description of **O_SYNC** in **open**(2)).

ST_NOSYMFOLLOW (since Linux 5.10)
Symbolic links are not followed when resolving paths; see **mount**(2).

Nobody knows what *f_fsid* is supposed to contain (but see below).

Fields that are undefined for a particular filesystem are set to 0.

fstatfs() returns the same information about an open file referenced by descriptor *fd*.

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set to indicate the error.

ERRORS

EACCES
(**statfs**()) Search permission is denied for a component of the path prefix of *path*. (See also **path_resolution**(7).)

EBADF
(**fstatfs**()) *fd* is not a valid open file descriptor.

EFAULT
buf or *path* points to an invalid address.

EINTR
The call was interrupted by a signal; see **signal**(7).

EIO An I/O error occurred while reading from the filesystem.

ELOOP
(**statfs**()) Too many symbolic links were encountered in translating *path*.

ENAMETOOLONG
(**statfs**()) *path* is too long.

ENOENT
(**statfs**()) The file referred to by *path* does not exist.

ENOMEM
Insufficient kernel memory was available.

ENOSYS
The filesystem does not support this call.

ENOTDIR
(**statfs**()) A component of the path prefix of *path* is not a directory.

EOVERFLOW
Some values were too large to be represented in the returned struct.

STANDARDS

Linux-specific. The Linux **statfs**() was inspired by the 4.4BSD one (but they do not use the same structure).

NOTES

The *__fsword_t* type used for various fields in the *statfs* structure definition is a glibc internal type, not intended for public use. This leaves the programmer in a bit of a conundrum when trying to copy or compare these fields to local variables in a program. Using *unsigned int* for such variables suffices on most systems.

The original Linux **statfs**() and **fstatfs**() system calls were not designed with extremely large file sizes in mind. Subsequently, Linux 2.6 added new **statfs64**() and **fstatfs64**() system calls that employ a new structure, *statfs64*. The new structure contains the same fields as the original *statfs* structure, but the sizes of various fields are increased, to accommodate large file sizes. The glibc **statfs**() and **fstatfs**() wrapper functions transparently deal with the kernel differences.

Some systems have only *<sys/vfs.h>*, other systems also have *<sys/statfs.h>*, where the former includes the latter. So it seems including the former is the best choice.

LSB has deprecated the library calls **statfs**() and **fstatfs**() and tells us to use **statvfs**(3) and **fstatvfs**(3) instead.

The f_fsid field

Solaris, Irix, and POSIX have a system call **statvfs**(2) that returns a *struct statvfs* (defined in *<sys/statvfs.h>*) containing an *unsigned long f_fsid*. Linux, SunOS, HP-UX, 4.4BSD have a system call **statfs**() that returns a *struct statfs* (defined in *<sys/vfs.h>*) containing a *fsid_t f_fsid*, where *fsid_t* is defined as *struct { int val[2]; }*. The same holds for FreeBSD, except that it uses the include file *<sys/mount.h>*.

The general idea is that *f_fsid* contains some random stuff such that the pair (*f_fsid*,*ino*) uniquely determines a file. Some operating systems use (a variation on) the device number, or the device number combined with the filesystem type. Several operating systems restrict giving out the *f_fsid* field to the superuser only (and zero it for unprivileged users), because this field is used in the filehandle of the filesystem when NFS-exported, and giving it out is a security concern.

Under some operating systems, the *fsid* can be used as the second argument to the **sysfs**(2) system call.

BUGS

From Linux 2.6.38 up to and including Linux 3.1, **fstatfs**() failed with the error **ENOSYS** for file descriptors created by **pipe**(2).

SEE ALSO

stat(2), **statvfs**(3), **path_resolution**(7)