

**NAME**

statvfs, fstatvfs – get filesystem statistics

**SYNOPSIS**

```
#include <sys/statvfs.h>
```

```
int statvfs(const char *path, struct statvfs *buf);
```

```
int fstatvfs(int fd, struct statvfs *buf);
```

**DESCRIPTION**

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

```
struct statvfs {
    unsigned long  f_bsize; /* Filesystem block size */
    unsigned long  f_frsize; /* Fragment size */
    fsblkcnt_t     f_blocks; /* Size of fs in f_frsize units */
    fsblkcnt_t     f_bfree; /* Number of free blocks */
    fsblkcnt_t     f_bavail; /* Number of free blocks for
                             unprivileged users */
    fsfilcnt_t     f_files; /* Number of inodes */
    fsfilcnt_t     f_ffree; /* Number of free inodes */
    fsfilcnt_t     f_favail; /* Number of free inodes for
                             unprivileged users */
    unsigned long  f_fsid; /* Filesystem ID */
    unsigned long  f_flag; /* Mount flags */
    unsigned long  f_namemax; /* Maximum filename length */
};
```

Here the types *fsblkcnt\_t* and *fsfilcnt\_t* are defined in *<sys/types.h>*. Both used to be *unsigned long*.

The field *f\_flag* is a bit mask indicating various options that were employed when mounting this filesystem. It contains zero or more of the following flags:

**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)**).

It is unspecified whether all members of the returned struct have meaningful values on all filesystems.

**fstatvfs()** 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 appropriately.

## ERRORS

### EACCES

(**statvfs()**) Search permission is denied for a component of the path prefix of *path*. (See also **path\_resolution(7)**.)

### EBADF

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

### EFAULT

*Buf* or *path* points to an invalid address.

### EINTR

This call was interrupted by a signal.

### EIO

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

### ELOOP

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

### ENAMETOOLONG

(**statvfs()**) *path* is too long.

### ENOENT

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

### ENOMEM

Insufficient kernel memory was available.

### ENOSYS

The filesystem does not support this call.

### ENOTDIR

(**statvfs()**) 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.

## ATTRIBUTES

### Multithreading (see **pthread(7)**)

The **statvfs()** and **fstatvfs()** functions are thread-safe.

## CONFORMING TO

POSIX.1-2001.

Only the **ST\_NOSUID** and **ST\_RDONLY** flags of the *f\_flag* field are specified in POSIX.1. To obtain definitions of the remaining flags, one must define **\_GNU\_SOURCE**.

## NOTES

The Linux kernel has system calls **statfs(2)** and **fstatfs(2)** to support this library call.

In glibc versions before 2.13, **statvfs()** populated the bits of the *f\_flag* field by scanning the mount options shown in */proc/mounts*. However, starting with Linux 2.6.36, the underlying **statfs(2)** system call provides the necessary information via the *f\_flags* field, and since glibc version 2.13, the **statvfs()** function will use information from that field rather than scanning */proc/mounts*.

The glibc implementations of

```
pathconf(path, _PC_REC_XFER_ALIGN);
pathconf(path, _PC_ALLOC_SIZE_MIN);
```

```
pathconf(path, _PC_REC_MIN_XFER_SIZE);
```

respectively use the *f\_fsize*, *f\_frsize*, and *f\_bsize* fields returned by a call to **statvfs()** with the argument *path*.

**SEE ALSO**

**statfs(2)**