NAME

statvfs, fstatvfs – get file system information

SYNOPSIS

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

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

int fstatvfs(int *fildes*, **struct statvfs** ∗*buf***);**

DESCRIPTION

statvfs( ) returns a "generic superblock" describing a file system; it can be used to acquire information about mounted file systems. *buf* is a pointer to a structure (described below) that is filled by the function.

path should name a file that resides on that file system. The file system type is known to the operating system. Read, write, or execute permission for the named file is not required, but all directories listed in the path name leading to the file must be searchable.

The **statvfs( )** structure pointed to by *buf* includes the following members:

```
u_long   f_bsize;                /* preferred file system block size */
u_long   f_frsize;               /* fundamental filesystem block size
                                 (if supported) */
u_long   f_blocks;               /* total # of blocks on file system
                                 in units of f_frsize */
u_long   f_bfree;                /* total # of free blocks */
u_long   f_bavail;               /* # of free blocks avail to
                                 non-super-user */
u_long   f_files;                /* total # of file nodes (inodes) */
u_long   f_ffree;                /* total # of free file nodes */
u_long   f_favail;               /* # of inodes avail to
                                 non-super-user*/
u_long   f_fsid;                 /* file system id (dev for now) */
char     f_basetype[FSTYPSZ];    /*  target fs type name,
                                 null-terminated */
u_long   f_flag;                 /* bit mask of flags */
u_long   f_namemax;              /* maximum file name length */
char     f_fstr[32];             /* file system specific string */
u_long   f_filler[16];           /* reserved for future expansion */
```

f_basetype contains a null-terminated FSType name of the mounted target.

The following flags can be returned in the **f_flag** field:

```
ST_RDONLY      0x01      /* read-only file system */
ST_NOSUID      0x02      /* does not support setuid/setgid
                         semantics */
ST_NOTRUNC     0x04      /* does not truncate file names
                         longer than {NAME_MAX}*/
```

fstatvfs( ) is similar to **statvfs( )**, except that the file named by *path* in **statvfs( )** is instead identified by an open file descriptor *fildes* obtained from a successful **open**(2), **creat**(2), **dup**(2), **fcntl**(2), or **pipe**(2) function.

RETURN VALUES

Upon successful completion a value of 0 is returned. Otherwise, a value of –1 is returned and **errno** is set to indicate the error.

ERRORS

statvfs( ) fails if one or more of the following are true:

EACCES Search permission is denied on a component of the path prefix.

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

EINTR A signal was caught during **statvfs( )** execution.

EIO An I/O error occurred while reading the file system.

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

EMULTIHOP Components of *path* require hopping to multiple remote machines and file system type does not allow it.

ENAMETOOLONG The length of a *path* component exceeds {**NAME_MAX**} characters, or the length of *path* exceeds {**PATH_MAX**} characters.

ENOENT Either a component of the path prefix or the file referred to by *path* does not exist.

ENOLINK *path* points to a remote machine and the link to that machine is no longer active.

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

fstatvfs( ) fails if one or more of the following are true:

EBADF *fildes* is not an open file descriptor.

EFAULT *buf* points to an illegal address.

EINTR A signal was caught during **fstatvfs( )** execution.

EIO An I/O error occurred while reading the file system.

SEE ALSO

chmod(2), **chown**(2), **creat**(2), **dup**(2), **fcntl**(2), **link**(2), **mknod**(2), **open**(2), **pipe**(2), **read**(2), **time**(2), **unlink**(2), **utime**(2), **write**(2)

BUGS

The values returned for **f_files**, **f_ffree**, and **f_favail** may not be valid for NFS mounted file systems.