NAME

stat, lstat, fstat – get file status

SYNOPSIS

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

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

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

int fstat(int *fildes*, **struct stat** ∗*buf***);**

MT-LEVEL

stat( ) and **fstat( )** are Async-Signal-Safe

DESCRIPTION

stat( ) obtains information about the file pointed to by *path*. Read, write, or execute permission of the named file is not required, but all directories listed in the path name leading to the file must be searchable.

lstat( ) obtains file attributes similar to **stat( )**, except when the named file is a symbolic link; in that case **lstat( )** returns information about the link, while **stat( )** returns information about the file the link references.

fstat( ) obtains information about an open file known by the file descriptor *fildes*, obtained from a successful **open**, **creat**, **dup**, **fcntl**, or **pipe** function.

buf is a pointer to a **stat( )** structure into which information is placed concerning the file.

The contents of the structure pointed to by *buf* include the following members:

```
mode_t   st_mode;     /* File mode (see mknod(2)) */
ino_t    st_ino;      /* Inode number */
dev_t    st_dev;      /* ID of device containing */
                      /* a directory entry for this file */
dev_t    st_rdev;     /* ID of device */
                      /* This entry is defined only for */
                      /* char special or block special files */
nlink_t  st_nlink;    /* Number of links */
uid_t    st_uid;      /* User ID of the file's owner */
gid_t    st_gid;      /* Group ID of the file's group */
off_t    st_size;     /* File size in bytes */
time_t   st_atime;    /* Time of last access */
time_t   st_mtime;    /* Time of last data modification */
time_t   st_ctime;    /* Time of last file status change */
                      /* Times measured in seconds since */
                      /* 00:00:00 UTC, Jan. 1, 1970 */
long     st_blksize;  /* Preferred I/O block size */
long     st_blocks;   /* Number of 512 byte blocks allocated*/
```

st_mode The mode of the file as described in **mknod**(2). In addition to the modes described in **mknod**(2), the mode of a file may also be S_IFLNK if the file is a symbolic link. (Note that S_IFLNK may only be returned by **lstat( )**.)

st_ino This field uniquely identifies the file in a given file system. The pair **st_ino**

and **st_dev** uniquely identifies regular files.

st_dev This field uniquely identifies the file system that contains the file. Its value may be used as input to the **ustat( )** function to determine more information about this file system. No other meaning is associated with this value.

st_rdev This field should be used only by administrative commands. It is valid only for block special or character special files and only has meaning on the system where the file was configured.

st_nlink This field should be used only by administrative commands.

st_uid The user ID of the file's owner.

st_gid The group ID of the file's group.

st_size For regular files, this is the address of the end of the file. For block special or character special, this is not defined. See also **pipe**(2).

st_atime Time when file data was last accessed. Changed by the following functions: **creat**, **mknod**, **pipe**, **utime**, and **read**.

st_mtime Time when data was last modified. Changed by the following functions: **creat**, **mknod**, **pipe**, **utime**, and **write**.

st_ctime Time when file status was last changed. Changed by the following functions: **chmod**, **chown**, **creat**, **link**, **mknod**, **pipe**, **unlink**, **utime**, and **write**.

st_blksize A hint as to the "best" unit size for I/O operations. This field is not defined for block special or character special files.

st_blocks The total number of physical blocks of size 512 bytes actually allocated on disk. This field is not defined for block special or character special files.

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

stat( ) and **lstat( )** fail if one or more of the following are true:

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

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

EINTR A signal was caught during the **stat( )** or **lstat( )** function.

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

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

ENAMETOOLONG The length of the *path* argument exceeds {**PATH_MAX**}, or the length of a *path* component exceeds {**NAME_MAX**} while {**_POSIX_NO_TRUNC**} is in effect.

ENOENT The named file does not exist or is the null pathname.

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

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

EOVERFLOW A component is too large to store in the structure pointed to by *buf*.

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

EBADF *fildes* is not a valid open file descriptor.

EFAULT *buf* points to an illegal address.

EINTR A signal was caught during the **fstat( )** function.

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

EOVERFLOW A component is too large to store in the structure pointed to by *buf*.

SEE ALSO **chmod**(2), **chown**(2), **creat**(2), **link**(2), **mknod**(2), **pipe**(2), **read**(2), **time**(2), **unlink**(2), **utime**(2), **write**(2), **fattach**(3C), **stat**(5)