NAME

stat, **lstat**, **fstat**, **fstatat** - get file status

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/stat.h>

int
stat(*const char * restrict path, struct stat * restrict sb*);

int
lstat(*const char * restrict path, struct stat * restrict sb*);

int
fstat(*int fd, struct stat *sb*);

int
fstatat(*int fd, const char *path, struct stat *sb, int flag*);

DESCRIPTION

The **stat**() system call 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.

The **lstat**() system call is like **stat**() except when the named file is a symbolic link, in which case **lstat**() returns information about the link, while **stat**() returns information about the file the link references.

The **fstat**() system call obtains the same information about an open file known by the file descriptor *fd*.

The **fstatat**() system call is equivalent to **stat**() and **lstat**() except when the *path* specifies a relative path. For **fstatat**() and relative *path*, the status is retrieved from a file relative to the directory associated with the file descriptor *fd* instead of the current working directory.

The values for the *flag* are constructed by a bitwise-inclusive OR of flags from this list, defined in <*fcntl.h*>:

AT_SYMLINK_NOFOLLOW
If *path* names a symbolic link, the status of the symbolic link is returned.

AT_RESOLVE_BENEATH
: Only walk paths below the starting directory. See the description of the O_RESOLVE_BENEATH flag in the open(2) manual page.

AT_EMPTY_PATH
: If the *path* argument is an empty string, operate on the file or directory referenced by the descriptor *fd*. If *fd* is equal to AT_FDCWD, operate on the current working directory.

If **fstatat**() is passed the special value AT_FDCWD in the *fd* parameter, the current working directory is used and the behavior is identical to a call to **stat**() or **lstat**() respectively, depending on whether or not the AT_SYMLINK_NOFOLLOW bit is set in *flag*.

When **fstatat**() is called with an absolute *path*, it ignores the *fd* argument.

The *sb* argument is a pointer to a *stat* structure as defined by <*sys/stat.h*> and into which information is placed concerning the file.

The fields of *struct stat* related to the file system are:

st_dev Numeric ID of the device containing the file.

st_ino The file's inode number.

st_nlink Number of hard links to the file.

st_flags Flags enabled for the file. See chflags(2) for the list of flags and their description.

The *st_dev* and *st_ino* fields together identify the file uniquely within the system.

The time-related fields of *struct stat* are:

st_atim Time when file data was last accessed. Changed implicitly by syscalls such as read(2) and readv(2), and explicitly by utimes(2).

st_mtim Time when file data was last modified. Changed implicitly by syscalls such as truncate(2), write(2), and writev(2), and explicitly by utimes(2). Also, any syscall which modifies directory content changes the *st_mtim* for the affected directory. For instance, creat(2), mkdir(2), rename(2), link(2), and unlink(2).

st_ctim Time when file status was last changed (inode data modification). Changed implicitly by

any syscall that affects file metadata, including *st_mtim*, such as chflags(2), chmod(2), chown(2), truncate(2), utimes(2), and write(2). Also, any syscall which modifies directory content changes the *st_ctim* for the affected directory. For instance, creat(2), mkdir(2), rename(2), link(2), and unlink(2).

st_birthtim Time when the inode was created.

These time-related macros are defined for compatibility:

```
#define    st_atime            st_atim.tv_sec
#define    st_mtime            st_mtim.tv_sec
#define    st_ctime            st_ctim.tv_sec
#ifndef _POSIX_SOURCE
#define    st_birthtime                st_birthtim.tv_sec
#endif

#ifndef _POSIX_SOURCE
#define    st_atimespec                st_atim
#define    st_mtimespec                st_mtim
#define    st_ctimespec                st_ctim
#define    st_birthtimespec    st_birthtim
#endif
```

Size-related fields of the *struct stat* are:

st_size File size in bytes.

st_blksize Optimal I/O block size for the file.

st_blocks Actual number of blocks allocated for the file in 512-byte units. As short symbolic links are stored in the inode, this number may be zero.

The access-related fields of *struct stat* are:

st_uid User ID of the file’s owner.

st_gid Group ID of the file.

st_mode Status of the file (see below).

The status information word *st_mode* has these bits:

```
#define S_IFMT   0170000  /* type of file mask */
#define S_IFIFO  0010000  /* named pipe (fifo) */
#define S_IFCHR  0020000  /* character special */
#define S_IFDIR  0040000  /* directory */
#define S_IFBLK  0060000  /* block special */
#define S_IFREG  0100000  /* regular */
#define S_IFLNK  0120000  /* symbolic link */
#define S_IFSOCK 0140000  /* socket */
#define S_IFWHT  0160000  /* whiteout */
#define S_ISUID  0004000  /* set user id on execution */
#define S_ISGID  0002000  /* set group id on execution */
#define S_ISVTX  0001000  /* save swapped text even after use */
#define S_IRWXU  0000700  /* RWX mask for owner */
#define S_IRUSR  0000400  /* read permission, owner */
#define S_IWUSR  0000200  /* write permission, owner */
#define S_IXUSR  0000100  /* execute/search permission, owner */
#define S_IRWXG  0000070  /* RWX mask for group */
#define S_IRGRP  0000040  /* read permission, group */
#define S_IWGRP  0000020  /* write permission, group */
#define S_IXGRP  0000010  /* execute/search permission, group */
#define S_IRWXO  0000007  /* RWX mask for other */
#define S_IROTH  0000004  /* read permission, other */
#define S_IWOTH  0000002  /* write permission, other */
#define S_IXOTH  0000001  /* execute/search permission, other */
```

For a list of access modes, see <*sys/stat.h*>, access(2) and chmod(2). These macros are available to test whether a *st_mode* value passed in the *m* argument corresponds to a file of the specified type:

S_ISBLK(*m*)
Test for a block special file.

S_ISCHR(*m*)
Test for a character special file.

S_ISDIR(*m*) Test for a directory.

S_ISFIFO(*m*)
Test for a pipe or FIFO special file.

S_ISLNK(*m*)
Test for a symbolic link.

S_ISREG(*m*)
Test for a regular file.

S_ISSOCK(*m*)
Test for a socket.

S_ISWHT(*m*)
Test for a whiteout.

The macros evaluate to a non-zero value if the test is true or to the value 0 if the test is false.

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.

COMPATIBILITY

Previous versions of the system used different types for the *st_dev*, *st_uid*, *st_gid*, *st_rdev*, *st_size*, *st_blksize* and *st_blocks* fields.

ERRORS

The **stat**() and **lstat**() system calls will fail if:

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

[EFAULT] The *sb* 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.

[ELOOP] Too many symbolic links were encountered in translating the pathname.

[ENAMETOOLONG]
A component of a pathname exceeded 255 characters, or an entire path name exceeded 1023 characters.

[ENOENT] The named file does not exist.

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

[EOVERFLOW] The file size in bytes cannot be represented correctly in the structure pointed to by *sb*.

The **fstat**() system call will fail if:

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

[EFAULT] The *sb* 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.

[EOVERFLOW] The file size in bytes cannot be represented correctly in the structure pointed to by *sb*.

In addition to the errors returned by the **lstat**(), the **fstatat**() may fail if:

[EBADF] The *path* argument does not specify an absolute path and the *fd* argument is neither AT_FDCWD nor a valid file descriptor open for searching.

[EINVAL] The value of the *flag* argument is not valid.

[ENOTDIR] The *path* argument is not an absolute path and *fd* is neither AT_FDCWD nor a file descriptor associated with a directory.

[ENOTCAPABLE] *path* is an absolute path, or contained a ".." component leading to a directory outside of the directory hierarchy specified by *fd*, and the process is in capability mode or the AT_RESOLVE_BENEATH flag was specified.

SEE ALSO

access(2), chmod(2), chown(2), fhstat(2), statfs(2), utimes(2), sticky(7), symlink(7)

STANDARDS

The **stat**() and **fstat**() system calls are expected to conform to IEEE Std 1003.1-1990 ("POSIX.1"). The **fstatat**() system call follows The Open Group Extended API Set 2 specification.

HISTORY

The **stat**() and **fstat**() system calls appeared in Version 1 AT&T UNIX. The **lstat**() system call appeared in 4.2BSD. The **fstatat**() system call appeared in FreeBSD 8.0.

BUGS

Applying **fstat**() to a socket returns a zeroed buffer, except for the blocksize field, and a unique device and inode number.