NAME

stat, fstat, lstat − get file status

SYNOPSIS

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

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

DESCRIPTION

These functions return information about a file. No permissions are required on the file itself, but — in the case of **stat**() and **lstat**() — execute (search) permission is required on all of the directories in *path* that lead to the file.

stat() stats the file pointed to by *path* and fills in *buf* .

lstat() is identical to **stat**(), except that if *path* is a symbolic link, then the link itself is stat-ed, not the file that it refers to.

fstat() is identical to **stat**(), except that the file to be stat-ed is specified by the file descriptor *filedes*.

All of these system calls return a *stat* structure, which contains the following fields:

```
struct stat {
    dev_t     st_dev;     /* ID of device containing file */
    ino_t     st_ino;     /* inode number */
    mode_t    st_mode;    /* protection */
    nlink_t   st_nlink;   /* number of hard links */
    uid_t     st_uid;     /* user ID of owner */
    gid_t     st_gid;     /* group ID of owner */
    dev_t     st_rdev;    /* device ID (if special file) */
    off_t     st_size;    /* total size, in bytes */
    blksize_t st_blksize; /* blocksize for filesystem I/O */
    blkcnt_t  st_blocks;  /* number of blocks allocated */
    time_t    st_atime;   /* time of last access */
    time_t    st_mtime;   /* time of last modification */
    time_t    st_ctime;   /* time of last status change */
};
```

The *st_dev* field describes the device on which this file resides.

The *st_rdev* field describes the device that this file (inode) represents.

The *st_size* field gives the size of the file (if it is a regular file or a symbolic link) in bytes. The size of a symlink is the length of the pathname it contains, without a trailing null byte.

The *st_blocks* field indicates the number of blocks allocated to the file, 512-byte units. (This may be smaller than *st_size*/512, for example, when the file has holes.)

The *st_blksize* field gives the "preferred" blocksize for efficient file system I/O. (Writing to a file in smaller chunks may cause an inefficient read-modify-rewrite.)

Not all of the Linux filesystems implement all of the time fields. Some file system types allow mounting in such a way that file accesses do not cause an update of the *st_atime* field. (See 'noatime' in **mount**(8).)

The field *st_atime* is changed by file accesses, e.g. by **execve**(2), **mknod**(2), **pipe**(2), **utime**(2) and **read**(2)

(of more than zero bytes). Other routines, like **mmap**(2), may or may not update *st_atime*.

The field *st_mtime* is changed by file modifications, e.g. by **mknod**(2), **truncate**(2), **utime**(2) and **write**(2) (of more than zero bytes). Moreover, *st_mtime* of a directory is changed by the creation or deletion of files in that directory. The *st_mtime* field is *not* changed for changes in owner, group, hard link count, or mode.

The field *st_ctime* is changed by writing or by setting inode information (i.e., owner, group, link count, mode, etc.).

The following POSIX macros are defined to check the file type using the *st_mode* field:

S_ISREG(m)	is it a regular file?
S_ISDIR(m)	directory?
S_ISCHR(m)	character device?
S_ISBLK(m)	block device?
S_ISFIFO(m)	FIFO (named pipe)?
S_ISLNK(m)	symbolic link? (Not in POSIX.1-1996.)
S_ISSOCK(m)	socket? (Not in POSIX.1-1996.)

The following flags are defined for the *st_mode* field:

S_IFMT	0170000	bitmask for the file type bitfields
S_IFSOCK	0140000	socket
S_IFLNK	0120000	symbolic link
S_IFREG	0100000	regular file
S_IFBLK	0060000	block device
S_IFDIR	0040000	directory
S_IFCHR	0020000	character device
S_IFIFO	0010000	FIFO
S_ISUID	0004000	set UID bit
S_ISGID	0002000	set-group-ID bit (see below)
S_ISVTX	0001000	sticky bit (see below)
S_IRWXU	00700	mask for file owner permissions
S_IRUSR	00400	owner has read permission
S_IWUSR	00200	owner has write permission
S_IXUSR	00100	owner has execute permission
S_IRWXG	00070	mask for group permissions
S_IRGRP	00040	group has read permission
S_IWGRP	00020	group has write permission
S_IXGRP	00010	group has execute permission
S_IRWXO	00007	mask for permissions for others (not in group)
S_IROTH	00004	others have read permission
S_IWOTH	00002	others have write permission
S_IXOTH	00001	others have execute permission

The set-group-ID bit (S_ISGID) has several special uses. For a directory it indicates that BSD semantics is to be used for that directory: files created there inherit their group ID from the directory, not from the effective group ID of the creating process, and directories created there will also get the S_ISGID bit set. For a file that does not have the group execution bit (S_IXGRP) set, the set-group-ID bit indicates mandatory file/record locking.

The 'sticky' bit (S_ISVTX) on a directory means that a file in that directory can be renamed or deleted only by the owner of the file, by the owner of the directory, and by a privileged process.

LINUX NOTES

Since kernel 2.5.48, the *stat* structure supports nanosecond resolution for the three file timestamp fields. Glibc exposes the nanosecond component of each field using names either of the form *st_atim.tv_nsec*, if the _BSD_SOURCE or _SVID_SOURCE feature test macro is defined, or of the form *st_atimensec*, if neither of these macros is defined. On file systems that do not support sub-second timestamps, these nanosecond fields are returned with the value 0.

For most files under the */proc* directory, **stat**() does not return the file size in the *st_size* field; instead the field is returned with the value 0.

RETURN VALUE

On success, zero is returned. On error, −1 is returned, and *errno* is set appropriately.

ERRORS

EACCES
Search permission is denied for one of the directories in the path prefix of *path*. (See also **path_resolution**(2).)

EBADF
filedes is bad.

EFAULT
Bad address.

ELOOP
Too many symbolic links encountered while traversing the path.

ENAMETOOLONG
File name too long.

ENOENT
A component of the path *path* does not exist, or the path is an empty string.

ENOMEM
Out of memory (i.e. kernel memory).

ENOTDIR
A component of the path is not a directory.

CONFORMING TO

These system calls conform to SVr4, 4.3BSD, POSIX.1-2001.

Use of the *st_blocks* and *st_blksize* fields may be less portable. (They were introduced in BSD. The interpretation differs between systems, and possibly on a single system when NFS mounts are involved.)

POSIX does not describe the S_IFMT, S_IFSOCK, S_IFLNK, S_IFREG, S_IFBLK, S_IFDIR, S_IFCHR, S_IFIFO, S_ISVTX bits, but instead demands the use of the macros S_ISDIR(), etc. The S_ISLNK and S_ISSOCK macros are not in POSIX.1-1996, but both are present in POSIX.1-2001; the former is from SVID 4, the latter from SUSv2.

Unix V7 (and later systems) had S_IREAD, S_IWRITE, S_IEXEC, where POSIX prescribes the synonyms S_IRUSR, S_IWUSR, S_IXUSR.

OTHER SYSTEMS

Values that have been (or are) in use on various systems:

hex	name	ls	octal	description
f000	S_IFMT		170000	mask for file type
0000			000000	SCO out-of-service inode, BSD unknown type
				SVID-v2 and XPG2 have both 0 and 0100000 for ordinary file
1000	S_IFIFO	p|	010000	FIFO (named pipe)

2000	S_IFCHR	c	020000	character special (V7)
3000	S_IFMPC		030000	multiplexed character special (V7)
4000	S_IFDIR	d/	040000	directory (V7)
5000	S_IFNAM		050000	XENIX named special file
				with two subtypes, distinguished by st_rdev values 1, 2:
0001	S_INSEM	s	000001	XENIX semaphore subtype of IFNAM
0002	S_INSHD	m	000002	XENIX shared data subtype of IFNAM
6000	S_IFBLK	b	060000	block special (V7)
7000	S_IFMPB		070000	multiplexed block special (V7)
8000	S_IFREG	-	100000	regular (V7)
9000	S_IFCMP		110000	VxFS compressed
9000	S_IFNWK	n	110000	network special (HP-UX)
a000	S_IFLNK	l@	120000	symbolic link (BSD)
b000	S_IFSHAD		130000	Solaris shadow inode for ACL (not seen by userspace)
c000	S_IFSOCK	s=	140000	socket (BSD; also "S_IFSOC" on VxFS)
d000	S_IFDOOR	D>	150000	Solaris door
e000	S_IFWHT	w%	160000	BSD whiteout (not used for inode)
0200	S_ISVTX		001000	'sticky bit': save swapped text even after use (V7)
				reserved (SVID-v2)
				On non-directories: don't cache this file (SunOS)
				On directories: restricted deletion flag (SVID-v4.2)
0400	S_ISGID		002000	set-group-ID on execution (V7)
				for directories: use BSD semantics for propagation of GID
0400	S_ENFMT		002000	SysV file locking enforcement (shared with S_ISGID)
0800	S_ISUID		004000	set-user-ID on execution (V7)
0800	S_CDF		004000	directory is a context dependent file (HP-UX)

A sticky command appeared in Version 32V AT&T UNIX.

SEE ALSO

access(2), **chmod**(2), **chown**(2), **fstatat**(2), **readlink**(2), **utime**(2), **capabilities**(7)