NAME

stat, lstat, fstat, fstatat - 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);

int fstatat(int fildes, const char *path, struct stat  *buf,
int flag);
```

DESCRIPTION

The stat() function 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() function 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.

The fstat() function obtains information about an open file known by the file descriptor fildes, obtained from a successful open(2), creat(2), dup(2), fcntl(2), or pipe(2) function.

The fstatat() function obtains file attributes similar to the stat(), lstat(), and fstat() functions. If the path argument is a relative path, it is resolved relative to the fildes argument rather than the current working directory. If path is absolute, the fildes argument is unused. If the fildes argument has the special value AT_FDCWD, defined in <fcntl.h>, relative paths are resolved from the current working directory. If the flag argument is AT_SYMLNK_NOFOLLOW, defined in <fcntl.h>, the function behaves like lstat() and does not automatically follow symbolic links. See fsattr(5).

The buf argument is a pointer to a stat structure into which information is placed concerning the file. A stat structure includes 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 */
blkcnt_t st_blocks;   /* Number of 512 byte blocks allocated*/
```

Descriptions of structure members are as follows:

st_mode

The mode of the file as described in mknod(2). In addition to the modes described in mknod(), the mode of a file can also be S_IFLNK if the file is a symbolic link. S_IFLNK can be returned either by lstat() or by fstat() when the AT_SYMLNK_NOFOLLOW flag is set.

```
     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  com-
           mands. 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  com-
           mands.

     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(2), and read(2).

     st_mtime
           Time when data was last modified. Changed by the  fol-
           lowing  functions:  creat(), mknod(), pipe(), utime(),
           and write(2).

     st_ctime
           Time when file status was last changed. Changed by the
           following   functions:   chmod(),   chown(),  creat(),
           link(2),  mknod(),  pipe(),  unlink(2),  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 charac-
           ter 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, 0 is returned. Otherwise, -1  is
     returned and errno is set to indicate the error.

ERRORS

 The stat(), fstat(), lstat(),  and  fstatat()functions  will fail if:

   EOVERFLOW
           The file size in bytes or the number of  blocks  allo-
           cated  to the file or the file serial number cannot be
           represented correctly in the structure pointed  to  by
           buf.

 The stat(), lstat(), and fstatat() functions will fail if:

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

   EFAULT  The   buf  or  path  argument  points  to  an  illegal
           address.
```

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

ELOOP Too many symbolic links were encountered in translating path.

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 The path argument 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, or the fildes argument does not refer to a valid directory when given a non-null relative path.

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

The fstat() and fstatat() functions will fail if:

EBADF The fildes argument is not a valid open file descriptor. Note that in fstatat() the fildes argument may also have the valid value of AT_FDCWD.

EFAULT The buf argument points to an illegal address.

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

ENOLINK The fildes argument 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.

USAGE

The stat(), fstat(), and lstat() functions have transitional interfaces for 64-bit file offsets. See lf64(5).

ATTRIBUTES

See attributes(5) for descriptions of the following attributes:

ATTRIBUTE TYPE	ATTRIBUTE VALUE
Interface Stability	stat() is Standard; fstatat() is Evolving
MT-Level	stat(), fstat() and fstatat() are Async-Signal-Safe

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(3HEAD), attributes(5), fsattr(5), lf64(5)

NOTES

If chmod(2) is used to change the file group owner permissions on a file with ACL entries, both the file group owner permissions and the ACL mask are changed to the new permissions. The new ACL mask permissions might change the effective permissions for additional users and groups who have ACL entries on the file.