NAME

stat() - get file status

SYNOPSIS

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

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

DESCRIPTION

The **stat()** function obtains information about the named file and writes it to the area pointed to by the *buf* argument. The *path* argument is a pointer to a path name of any file within the mounted file system. (All directories listed in the path name must be searchable). Read, write or execute permission of the named file is not required, but all directories listed in the pathname leading to all directories listed in the pathname leading to the file must be searchable. An implementation that provides additional or alternate file access control mechanisms may, under implementation-dependent conditions, cause **stat()** to fail. In particular, the system may deny the existence of the file specified by *path* .

The *buf* argument is a pointer to a **stat** structure, as defined in the header **<sys/stat.h>**, into which information is placed concerning the file. The **stat** structure contains the following members:

```
dev_t      st_dev;       /* ID of device containing a */
                         /* directory entry for this file */
ino_t      st_ino;       /* Inode number */
short      st_fstype;    /* Type of filesystem this file  */
                         /* is in; see sysfs(2) */
ushort     st_mode;      /* File type, attributes, and */
                         /* access control summary */
ushort     st_basemode   /* Permission bits (see chmod(1)) */
ushort     st_nlink;     /* Number of links */
uid_t      st_uid;       /* User ID of file owner */
gid_t      st_gid;       /* Group ID of file group */
dev_t      st_rdev;      /* Device ID; this entry defined */
                         /* only for char or blk spec files */
off_t      st_size;      /* File size (bytes) */
time_t     st_atime;     /* Time of last access */
int        st_natime;    /* Reserved, DO NOT USE; field may change. */
time_t     st_mtime;     /* Last modification time */
int        st_nmtime;    /* Reserved, DO NOT USE, field may change. */
time_t     st_ctime;     /* Last file status change time */
                         /* Measured in secs since */
                         /* 00:00:00 GMT, Jan 1, 1970 */
int        st_nctime;    /* Reserved, DO NOT USE; field may change */
long       st_blksize;   /* File system block size */
blkcnt_t   st_blocks;    /* Number of blocks of a */
                         /* file-system-specific size */
                         /* allocated for this object */
uint       st_acl:1;     /* Set if the file has optional */
                         /* access control list entries */
                         /* HFS File Systems only */
uint       st_aclv:1;    /* Set if the file has optional */
                         /* access control list entries */
                         /* JFS File Systems only */
```

(Note that the position of items in this list does not necessarily reflect the order of the members in the structure.)

If the chosen path name or file descriptor refers to a Multi-Level Directory (MLD), and the process does not have the multilevel effective privilege, the *i-node* number returned in *st_ino* is the *i-node* of the MLD itself.

The **stat()** function updates any time-related fields (as described in the definition of File Times Update in the XBD specification), before writing into the **stat** structure.

The structure members *st_mode* , *st_ino* , *st_dev* , *st_uid* , *st_gid* , *st_atime* , *st_ctime* , and *st_mtime* will have meaningful values for all file types defined in this document. The value of the member *st_nlink* will

be set to the number of links to the file.

Note: The *st_natime*, *st_nmtime*, and *st_nctime* fields are currently reserved. To avoid compatibility problems, these fields should not be used.

RETURN VALUE

Upon successful completion, 0 is returned. Otherwise, −1 is returned and **errno** is set to indicate the error.

ERRORS

The **stat()** function will fail if:

[EACCES]	Search permission is denied for a component of the *path* prefix.
[EFAULT]	*buf* or *path* points to an invalid address. The reliable detection of this error is implementation dependent.
[EIO]	An error occurred while reading from the file system.
[ELOOP]	Too many symbolic links were encountered in resolving path.
[ENAMETOOLONG]	The length of the *path* argument exceeds **{PATH_MAX}** or a pathname component is longer than **{NAME_MAX}**.
[ENOENT]	A component of *path* does not name an existing file or path is an empty string.
[ENOTDIR]	A component of the *path* prefix is not a directory.
[EOVERFLOW]	The file size in bytes or the number of blocks allocated to the file cannot be represented correctly in the structure pointed to by *buf*.

The **stat()** function may fail if:

[ENAMETOOLONG]	Pathname resolution of a symbolic link produced an intermediate result whose length exceeds **{PATH_MAX}**.
[EOVERFLOW]	A 32-bit application is making this call on a file where the **st_size** or other field(s) would need to hold a 64-bit value. Use **stat64()** instead.

NETWORKING FEATURES

NFS

The *st_basemode* is equal to *st_mode* and *st_acl* and the *st_aclv* fields are zero on files accessed remotely. The *st_acl* field is applicable to HFS File Systems only. The *st_aclv* field is applicable to JFS File Systems only.

WARNINGS

S

Access Control Lists - HFS and JFS File Systems only

Access control list descriptions in this entry apply only to HFS and JFS file systems on standard HP-UX operating systems.

For 32-bit applications, **st_ino** will be truncated to its least significant 32-bits for filesystems that use 64-bit values.

DEPENDENCIES

CD-ROM

The *st_uid* and *st_gid* fields are set to −1 if they are not specified on the disk for a given file.

AUTHOR

stat() and **fstat()** were developed by AT&T. **lstat()** was developed by the University of California, Berkeley.

SEE ALSO

touch(1), acl(2), chmod(2), chown(2), creat(2), fstat(2), link(2), lstat(2), mknod(2), pipe(2), read(2), rename(2), setacl(2), stat64(2), sysfs(2), time(2), truncate(2), unlink(2), utime(2), write(2), acl(5), aclv(5), privileges(5), stat(5), <sys/stat.h>, <sys/types.h>.

STANDARDS CONFORMANCE

stat(): AES, SVID2, SVID3, XPG2, XPG3, XPG4, FIPS 151-2, POSIX.1