

lstat() — Get Status of File or Symbolic Link

Standards

Standards / Extensions	C or C++	Dependencies
z/OS UNIX XPG4.2 Single UNIX Specification, Version 3	both	

Format

```
#define _POSIX1_SOURCE 2
#include <sys/stat.h>

int lstat(const char * __restrict __pathname, struct stat * __restrict __buf);
```

General Description

Gets status information about a specified file and places it in the area of memory pointed to by the *buf* argument. You do not need permissions on the file itself, but you must have search permission on all directory components of the *pathname*.

If the named file is a symbolic link, `lstat()` returns information about the symbolic link itself.

The information is returned in the following `stat` structure, defined in the `sys/stat.h` header file.

Table 39. Elements of stat Structure

Structure	Description
<code>mode_t st_mode</code>	A bit string indicating the permissions and privileges of the file. Symbols are defined in the <code>sys/stat.h</code> header file to refer to bits in a <code>mode_t</code> value; these symbols are listed in chmod() — Change the Mode of a File or Directory .
<code>ino_t st_ino</code>	The serial number of the file.
<code>dev_t st_dev</code>	The numeric ID of the device containing the file.
<code>nlink_t st_nlink</code>	The number of links to the file.
<code>uid_t st_uid</code>	The numeric user ID of the file's owner.
<code>gid_t st_gid</code>	The numeric group ID of the file's group.
<code>off_t st_size</code>	For regular files, the file's size in bytes. For symbolic links, the length of the pathname contained therein not counting the trailing NULL. For other kinds of files, the value of this field is unspecified.
<code>time_t st_atime</code>	The most recent time the file was accessed.
<code>time_t st_ctime</code>	The most recent time the status of the file was changed.
<code>time_t st_mtime</code>	The most recent time the contents of the file were changed.

Values for `time_t` are given in terms of seconds that have elapsed since epoch.

If the named file is a symbolic link, `lstat()` updates the time-related fields before putting information in the `stat` structure.

You can examine properties of a `mode_t` value from the `st_mode` field by using a collection of macros defined in the `sys/modes.h` header file. If *mode* is a `mode_t` value, and *genvalue* is an unsigned `int` value from the `stat` structure, then:

```
S_ISBLK(mode)
    Is nonzero for block special files.
S_ISCHR(mode)
    Is nonzero for character special files.
S_ISDIR(mode)
    Is nonzero for directories.
S_ISEXTL(mode,genvalue)
    Is nonzero for external links.
```

S_ISFIFO(*mode*)
Is nonzero for pipes and FIFO special files.

S_ISLNK(*mode*)
Is nonzero for symbolic links.

S_ISREG(*mode*)
Is nonzero for regular files.

S_ISSOCK(*mode*)
Is nonzero for sockets.

If lstat() successfully determines all this information, it stores it in the area indicated by the *buf* argument.

Large Files for HFS

Note:

Large Files for HFS behavior is automatic for AMODE 64 applications

Applications that are compiled with the option LONGLVL(LONGLONG) and also define the Feature Test Macro (FTM) `_LARGE_FILES` before any headers are included will enable this function to operate on HFS files that are larger than 2 gig-1 in size. File size and offset fields will be enlarged to 63 bits in width so any other function operating on this file will have to be enabled with the same FTM.

Returned Value

If successful, lstat() returns 0.

If unsuccessful, lstat() returns -1 and sets errno to one of the following values:

Error Code

	Description
EACCES	The process does not have search permission on some component of the <i>pathname</i> prefix.
EINVAL	<i>buf</i> contains a NULL.
EIO	Added for XPG4.2: An I/O error occurred while reading from the file system.
ELOOP	A loop exists in symbolic links. This error is issued if the number of symbolic links encountered during resolution of the <i>pathname</i> argument is greater than POSIX_SYMLOOP.
ENAMETOOLONG	<i>pathname</i> is longer than PATH_MAX characters or some component of <i>pathname</i> is longer than NAME_MAX characters while <code>_POSIX_NO_TRUNC</code> is in effect. For symbolic links, the length of the pathname string substituted for a symbolic link exceeds PATH_MAX . The PATH_MAX and NAME_MAX values can be determined through <code>pathconf()</code> .
ENOENT	There is no file named <i>pathname</i> , or <i>pathname</i> is an empty string.
ENOTDIR	A component of the <i>pathname</i> prefix is not a directory.
EOVERFLOW	The file size in bytes or the number of blocks allocated to the file or the file serial number cannot be represented correctly in the structure pointed to by <i>buf</i> . Note: Starting with z/OS V1.9, environment variable <code>_EDC_EOVERFLOW</code> can be used to control behavior of <code>lstat()</code> with respect to detecting an <code>EOVERFLOW</code> condition for UNIX files. By default, <code>lstat()</code> will not set <code>EOVERFLOW</code> when the file size can not be represented correctly in structure pointed to by <i>buf</i> . When <code>_EDC_EOVERFLOW</code> is set to YES, <code>lstat()</code> will check for an overflow condition.