Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'fstatat.3p' command

$ man fstatat.3p

FSTATAT(3P) POSIX Programmer's Manual FSTATAT(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

fstatat, lstat, stat ? get file status

SYNOPSIS

```
#include <fcntl.h>
#include <sys/stat.h>
int fstatat(int fd, const char *restrict path,
    struct stat *restrict buf, int flag);
int lstat(const char *restrict path, struct stat *restrict buf);
int stat(const char *restrict path, struct stat *restrict buf);
```

DESCRIPTION

The stat() function shall obtain information about the named file and write it to the area pointed to by the buf argument. The path argument points to a pathname naming a file. Read, write, or execute permission of the named file is not required. An implementation that provides ad? ditional or alternate file access control mechanisms may, under imple? mentation-defined conditions, cause stat() to fail. In particular, the system may deny the existence of the file specified by path.

If the named file is a symbolic link, the stat() function shall con?
tinue pathname resolution using the contents of the symbolic link, and
shall return information pertaining to the resulting file if the file
exists.
The buf argument is a pointer to a stat structure, as defined in the
<sys/stat.h> header, into which information is placed concerning the
file.
The stat() function shall update any time-related fields (as described
in the Base Definitions volume of POSIX.1?2017, Section 4.9, File Times
Update), before writing into the stat structure.
If the named file is a shared memory object, the implementation shall
update in the stat structure pointed to by the buf argument the st_uid,
st_gid, st_size, and st_mode fields, and only the S_IRUSR, S_IWUSR,
S_IRGRP, S_IWGRP, S_IROTH, and S_IWOTH file permission bits need be
valid. The implementation may update other fields and flags.
If the named file is a typed memory object, the implementation shall
update in the stat structure pointed to by the buf argument the st_uid,
st_gid, st_size, and st_mode fields, and only the S_IRUSR, S_IWUSR,
S_IRGRP, S_IWGRP, S_IROTH, and S_IWOTH file permission bits need be
valid. The implementation may update other fields and flags.
For all other file types defined in this volume of POSIX.1?2017, the
structure members st_mode, st_ino, st_dev, st_uid, st_gid, st_atim,
st_ctim, and st_mtim shall have meaningful values and the value of the
member st_nlink shall be set to the number of links to the file.
The lstat() function shall be equivalent to stat(), except when path
refers to a symbolic link. In that case lstat() shall return informa?
tion about the link, while stat() shall return information about the
file the link references.
For symbolic links, the st_mode member shall contain meaningful infor?
mation when used with the file type macros. The file mode bits in
st_mode are unspecified. The structure members st_ino, st_dev, st_uid,
st_gid, st_atim, st_ctim, and st_mtim shall have meaningful values and
the value of the st_nlink member shall be set to the number of (hard)

links to the symbolic link. The value of the st_size member shall be set to the length of the pathname contained in the symbolic link not including any terminating null byte.

The fstatat() function shall be equivalent to the stat() or lstat() function, depending on the value of flag (see below), except in the case where path specifies a relative path. In this case the status shall be retrieved from a file relative to the directory associated with the file descriptor fd instead of the current working directory.

If the access mode of the open file description associated with the file descriptor is not O_SEARCH, the function shall check whether di? rectory searches are permitted using the current permissions of the di? rectory underlying the file descriptor. If the access mode is O_SEARCH, the function shall not perform the check.

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

AT_SYMLINK_NOFOLLOW

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

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

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set errno to indicate the error.

ERRORS

These functions shall fail if:

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

EIO An error occurred while reading from the file system.

ELOOP A loop exists in symbolic links encountered during resolution of the path argument.

ENAMETOOLONG

The length of a component of a pathname 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 names an existing file that is neither a directory nor a symbolic link to a directory, or the path argument contains at least one non-<slash> character and ends with one or more trailing <slash> characters and the last pathname component names an existing file that is neither a di? rectory nor a symbolic link to 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 buf.

The fstatat() function shall fail if:

EACCES The access mode of the open file description associated with fd is not O_SEARCH and the permissions of the directory underlying fd do not permit directory searches.

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 reading or searching.

ENOTDIR

The path argument is not an absolute path and fd is a file de? scriptor associated with a non-directory file.

These functions may fail if:

ELOOP More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the path argument.

ENAMETOOLONG

The length of a pathname exceeds {PATH_MAX}, or pathname resolu? tion of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

EOVERFLOW

A value to be stored would overflow one of the members of the

stat structure.

The fstatat() function may fail if:

EINVAL The value of the flag argument is not valid.

The following sections are informative.

EXAMPLES

Obtaining File Status Information

The following example shows how to obtain file status information for a file named /home/cnd/mod1. The structure variable buffer is defined for the stat structure.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
struct stat buffer;
int         status;
...
status = stat("/home/cnd/mod1", &buffer);
```

Getting Directory Information

The following example fragment gets status information for each entry in a directory. The call to the stat() function stores file information in the stat structure pointed to by statbuf. The lines that follow the stat() call format the fields in the stat structure for presentation to the user of the program.

```
#include <sys/types.h>
#include <sys/stat.h>
#include <dirent.h>
#include <pwd.h>
#include <grp.h>
#include <time.h>
#include <locale.h>
#include <langinfo.h>
#include <stdio.h>
#include <stdint.h>
struct dirent  *dp;
```

```
struct stat     statbuf;
struct passwd  *pwd;
struct group   *grp;
struct tm      *tm;
char            datestring[256];
...
/* Loop through directory entries. */
while ((dp = readdir(dir)) != NULL) {
    /* Get entry's information. */
    if (stat(dp->d_name, &statbuf) == -1)
        continue;
    /* Print out type, permissions, and number of links. */
    printf("%10.10s", sperm (statbuf.st_mode));
    printf("%4d", statbuf.st_nlink);
    /* Print out owner's name if it is found using getpwuid(). */
    if ((pwd = getpwuid(statbuf.st_uid)) != NULL)
        printf(" %-8.8s", pwd->pw_name);
    else
        printf(" %-8d", statbuf.st_uid);
    /* Print out group name if it is found using getgrgid(). */
    if ((grp = getgrgid(statbuf.st_gid)) != NULL)
        printf(" %-8.8s", grp->gr_name);
    else
        printf(" %-8d", statbuf.st_gid);
    /* Print size of file. */
    printf(" %9jd", (intmax_t)statbuf.st_size);
    tm = localtime(&statbuf.st_mtime);
    /* Get localized date string. */
    strftime(datestring, sizeof(datestring), nl_langinfo(D_T_FMT), tm);
    printf(" %s %s\n", datestring, dp->d_name);
}
```

Obtaining Symbolic Link Status Information

The following example shows how to obtain status information for a sym?

bolic link named /modules/pass1. The structure variable buffer is de?
fined for the stat structure. If the path argument specified the path?
name for the file pointed to by the symbolic link (/home/cnd/mod1), the
results of calling the function would be the same as those returned by
a call to the stat() function.

```
#include <sys/stat.h>
struct stat buffer;
int status;
...
status = lstat("/modules/pass1", &buffer);
```

APPLICATION USAGE

None.

RATIONALE

The intent of the paragraph describing ``additional or alternate file access control mechanisms'' is to allow a secure implementation where a process with a label that does not dominate the file's label cannot perform a stat() function. This is not related to read permission; a process with a label that dominates the file's label does not need read permission. An implementation that supports write-up operations could fail fstat() function calls even though it has a valid file descriptor open for writing.

The purpose of the fstatat() function is to obtain the status of files in directories other than the current working directory without expo? sure to race conditions. Any part of the path of a file could be changed in parallel to a call to stat(), resulting in unspecified be? havior. By opening a file descriptor for the target directory and using the fstatat() function it can be guaranteed that the file for which status is returned is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

access(), chmod(), fdopendir(), fstat(), mknod(), readlink(), symlink()

The Base Definitions volume of POSIX.1?2017, Section 4.9, File Times

Update, <fcntl.h>, <sys_stat.h>, <sys_types.h>

COPYRIGHT