chmod(3p) — Linux manual page

PROLOG top

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 top

chmod, fchmodat – change mode of a file

SYNOPSIS top

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

int chmod(const char *path, mode_t mode);

#include <fcntl.h>
```

```
int fchmodat(int fd, const char *path, mode_t mode, int flag);
```

DESCRIPTION top

The *chmod*() function shall change S_ISUID, S_ISGID, S_ISVTX, and the file permission bits of the file named by the pathname pointed to by the *path* argument to the corresponding bits in the *mode* argument. The application shall ensure that the effective user ID of the process matches the owner of the file or the process has appropriate privileges in order to do this.

S_ISUID, S_ISGID, S_ISVTX, and the file permission bits are described in *<sys/stat.h>*.

If the calling process does not have appropriate privileges, and if the group ID of the file does not match the effective group ID or one of the supplementary group IDs and if the file is a regular file, bit S_ISGID (set-group-ID on execution) in the file's mode shall be cleared upon successful return from *chmod*().

Additional implementation-defined restrictions may cause the S_ISUID and S_ISGID bits in *mode* to be ignored.

Upon successful completion, *chmod*() shall mark for update the last file status change timestamp of the file.

The *fchmodat*() function shall be equivalent to the *chmod*() function except in the case where *path* specifies a relative path. In this case the file to be changed is determined 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 directory searches are permitted using the current permissions of the directory 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, then the mode of the symbolic link is changed.

If *fchmodat*() is passed the special value AT_FDCWD in the *fd* parameter, the current working directory shall be used. If also *flag* is zero, the behavior shall be identical to a call to *chmod*().

RETURN VALUE top

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set *errno* to indicate the error. If -1 is returned, no change to the file mode occurs.

ERRORS top

These functions shall fail if:

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

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 directory nor a symbolic link to a directory.

EPERM The effective user ID does not match the owner of the file and the process does not have appropriate privileges.

EROFS The named file resides on a read-only file system.

The *fchmodat*() 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 descriptor associated with a non-directory file.

These functions may fail if:

EINTR A signal was caught during execution of the function.

EINVAL The value of the *mode* argument is invalid.

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 resolution of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

The *fchmodat*() function may fail if:

EINVAL The value of the *flag* argument is invalid.

EOPNOTSUPP
The AT_SYMLINK_NOFOLLOW bit is set in the *flag* argument, *path* names a symbolic link, and the system does not support changing the mode of a symbolic link.

The following sections are informative.

EXAMPLES top

Setting Read Permissions for User, Group, and Others

The following example sets read permissions for the owner, group, and others.

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

const char *path;
...
chmod(path, S_IRUSR|S_IRGRP|S_IROTH);
```

Setting Read, Write, and Execute Permissions for the Owner Only

The following example sets read, write, and execute permissions for the owner, and no permissions for group and others.

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

const char *path;
...
chmod(path, S_IRWXU);
```

Setting Different Permissions for Owner, Group, and Other

The following example sets owner permissions for CHANGEFILE to read, write, and execute, group permissions to read and execute, and other permissions to read.

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

#define CHANGEFILE "/etc/myfile"
...
chmod(CHANGEFILE, S_IRWXU|S_IRGRP|S_IXGRP|S_IROTH);
```

Setting and Checking File Permissions

The following example sets the file permission bits for a file named **/home/cnd/mod1**, then calls the *stat*() function to verify the permissions.

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

int status;
struct stat buffer
...
chmod("/home/cnd/mod1", S_IRWXU|S_IRWXG|S_IROTH|S_IWOTH);
status = stat("/home/cnd/mod1", &buffer);
```

APPLICATION USAGE top

In order to ensure that the S_ISUID and S_ISGID bits are set, an application requiring this should use *stat*() after a successful *chmod*() to verify this.

Any file descriptors currently open by any process on the file could possibly become invalid if the mode of the file is changed to a value which would deny access to that process. One situation where this could occur is on a stateless file system. This behavior will not occur in a conforming environment.

RATIONALE top

This volume of POSIX.1-2017 specifies that the S_ISGID bit is cleared by *chmod*() on a regular file under certain conditions. This is specified on the assumption that regular files may be executed, and the system should prevent users from making executable *setgid*() files perform with privileges that the caller does not have. On implementations that support execution of other file types, the S_ISGID bit should be cleared for those file types under the same circumstances.

Implementations that use the S_ISUID bit to indicate some other function (for example, mandatory record locking) on non-executable files need not clear this bit on writing. They should clear the bit for executable files and any other cases where the bit grants special powers to processes that change the file contents. Similar comments apply to the S_ISGID bit.

The purpose of the *fchmodat*() function is to enable changing the mode of files in directories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to a call to *chmod*(),

resulting in unspecified behavior. By opening a file descriptor for the target directory and using the *fchmodat*() function it can be guaranteed that the changed file is located relative to the desired directory. Some implementations might allow changing the mode of symbolic links. This is not supported by the interfaces in the POSIX specification. Systems with such support provide an interface named *lchmod*(). To support such implementations *fchmodat*() has a *flag* parameter.

FUTURE DIRECTIONS top

None.

SEE ALSO top

access(3p), chown(3p), exec(1p), fstatat(3p), fstatvfs(3p), mkdir(3p), mkfifo(3p), mknod(3p), open(3p)

The Base Definitions volume of POSIX.1-2017, fcntl.h(0p), sys_stat.h(0p), sys_types.h(0p)

COPYRIGHT top

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.kernel.org/doc/man-pages/reporting_bugs.html .

IEEE/The Open Group 2017 CHMOD(3P)

Pages that refer to this page: sys_stat.h(0p), chmod(1p), access(3p), chown(3p), exec(3p), fchmod(3p), fchmodat(3p), fstatat(3p), fstatvfs(3p), lockf(3p), mkdir(3p), mkfifo(3p), mknod(3p), open(3p), posix_spawn(3p), write(3p)