



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 'chown.3p' command

\$ man chown.3p

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

chown, fchownat ? change owner and group of a file

SYNOPSIS

```
#include <unistd.h>

int chown(const char *path, uid_t owner, gid_t group);

#include <fcntl.h>

int fchownat(int fd, const char *path, uid_t owner, gid_t group,
             int flag);
```

DESCRIPTION

The `chown()` function shall change the user and group ownership of a file.

The `path` argument points to a pathname naming a file. The user ID and group ID of the named file shall be set to the numeric values contained in `owner` and `group`, respectively.

Only processes with an effective user ID equal to the user ID of the file or with appropriate privileges may change the ownership of a file.

If `_POSIX_CHOWN_RESTRICTED` is in effect for `path`:

- * Changing the user ID is restricted to processes with appropriate privileges.
- * Changing the group ID is permitted to a process with an effective user ID equal to the user ID of the file, but without appropriate privileges, if and only if owner is equal to the file's user ID or (uid_t)-1 and group is equal either to the calling process' effective group ID or to one of its supplementary group IDs.

If the specified file is a regular file, one or more of the S_IXUSR, S_IXGRP, or S_IXOTH bits of the file mode are set, and the process does not have appropriate privileges, the set-user-ID (S_ISUID) and set-group-ID (S_ISGID) bits of the file mode shall be cleared upon successful return from chown(). If the specified file is a regular file, one or more of the S_IXUSR, S_IXGRP, or S_IXOTH bits of the file mode are set, and the process has appropriate privileges, it is implementation-defined whether the set-user-ID and set-group-ID bits are altered. If the chown() function is successfully invoked on a file that is not a regular file and one or more of the S_IXUSR, S_IXGRP, or S_IXOTH bits of the file mode are set, the set-user-ID and set-group-ID bits may be cleared.

If owner or group is specified as (uid_t)-1 or (gid_t)-1, respectively, the corresponding ID of the file shall not be changed.

Upon successful completion, chown() shall mark for update the last file status change timestamp of the file, except that if owner is (uid_t)-1 and group is (gid_t)-1, the file status change timestamp need not be marked for update.

The fchownat() function shall be equivalent to the chown() and lchown() functions 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 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, ownership of the symbolic link is changed.

If fchownat() 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 chown() or lchown() respectively, depending on whether or not the AT_SYMLINK_NOFOLLOW bit is set in the flag argument.

RETURN VALUE

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 changes are made in the user ID and group ID of the file.

ERRORS

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, or

the calling process does not have appropriate privileges and `_POSIX_CHOWN_RESTRICTED` indicates that such privilege is required.

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

The `fchownat()` 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:

EIO An I/O error occurred while reading or writing to the filesystem.

EINTR The `chown()` function was interrupted by a signal which was caught.

EINVAL The owner or group ID supplied is not a value supported by the implementation.

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 `fchownat()` function may fail if:

EINVAL The value of the flag argument is not valid.

The following sections are informative.

EXAMPLES

None.

APPLICATION USAGE

Although `chown()` can be used on some implementations by the file owner to change the owner and group to any desired values, the only portable use of this function is to change the group of a file to the effective GID of the calling process or to a member of its group set.

RATIONALE

System III and System V allow a user to give away files; that is, the owner of a file may change its user ID to anything. This is a serious problem for implementations that are intended to meet government security regulations. Version 7 and 4.3 BSD permit only the superuser to change the user ID of a file. Some government agencies (usually not ones concerned directly with security) find this limitation too confining. This volume of POSIX.1-2017 uses `may` to permit secure implementations while not disallowing System V.

System III and System V allow the owner of a file to change the group ID to anything. Version 7 permits only the superuser to change the group ID of a file. 4.3 BSD permits the owner to change the group ID of a file to its effective group ID or to any of the groups in the list of supplementary group IDs, but to no others.

The POSIX.1-1990 standard requires that the `chown()` function invoked by a non-appropriate privileged process clear the `S_ISGID` and the `S_ISUID` bits for regular files, and permits them to be cleared for other types of files. This is so that changes in accessibility do not accidentally cause files to become security holes. Unfortunately, requiring these bits to be cleared on non-executable data files also clears the mandatory file locking bit (shared with `S_ISGID`), which is an extension on many implementations (it first appeared in System V). These bits should only be required to be cleared on regular files that have one or more of their execute bits set.

The purpose of the `fchownat()` function is to enable changing ownership 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 `chown()` or `lchown()`, resulting in unspecified behavior. By opening a file descriptor for the target `di?`

rectory and using the `fchownat()` function it can be guaranteed that the changed file is located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

`chmod()`, `fpathconf()`, `lchown()`

The Base Definitions volume of POSIX.1-2017, `<fcntl.h>`, `<sys_types.h>`, `<unistd.h>`

COPYRIGHT

Portions of this text are reprinted and reproduced in electronic form from IEEE Std 1003.1-2017, Standard for Information Technology -- Portable Operating System Interface (POSIX), The Open Group Base Specifications Issue 7, 2018 Edition, Copyright (C) 2018 by the Institute of Electrical and Electronics Engineers, Inc and The Open Group. In the event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at <http://www.opengroup.org/unix/online.html>.

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

CHOWN(3P)