



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 'fcntl.h.0p' command

\$ man fcntl.h.0p

fcntl.h(0P) POSIX Programmer's Manual fcntl.h(0P)

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

fcntl.h ? file control options

SYNOPSIS

```
#include <fcntl.h>
```

DESCRIPTION

The <fcntl.h> header shall define the following symbolic constants for the cmd argument used by fcntl(). The values shall be unique and shall be suitable for use in #if preprocessing directives.

F_DUPFD Duplicate file descriptor.

F_DUPFD_CLOEXEC

 Duplicate file descriptor with the close-on-exec flag

 FD_CLOEXEC set.

F_GETFD Get file descriptor flags.

F_SETFD Set file descriptor flags.

F_GETFL Get file status flags and file access modes.

F_SETFL Set file status flags.

F_GETLK Get record locking information.

F_SETLK Set record locking information.

F_SETLKW Set record locking information; wait if blocked.

F_GETOWN Get process or process group ID to receive SIGURG signals.

F_SETOWN Set process or process group ID to receive SIGURG signals.

The `<fcntl.h>` header shall define the following symbolic constant used for the `fcntl()` file descriptor flags, which shall be suitable for use in `#if` preprocessing directives.

FD_CLOEXEC Close the file descriptor upon execution of an exec family function.

The `<fcntl.h>` header shall also define the following symbolic constants for the `l_type` argument used for record locking with `fcntl()`. The values shall be unique and shall be suitable for use in `#if` preprocessing directives.

F_RDLCK Shared or read lock.

F_UNLCK Unlock.

F_WRLCK Exclusive or write lock.

The `<fcntl.h>` header shall define the values used for `l_whence`, `SEEK_SET`, `SEEK_CUR`, and `SEEK_END` as described in `<stdio.h>`.

The `<fcntl.h>` header shall define the following symbolic constants as file creation flags for use in the `oflag` value to `open()` and `openat()`.

The values shall be bitwise-distinct and shall be suitable for use in `#if` preprocessing directives.

O_CLOEXEC The `FD_CLOEXEC` flag associated with the new descriptor shall be set to close the file descriptor upon execution of an exec family function.

O_CREAT Create file if it does not exist.

O_DIRECTORY Fail if file is a non-directory file.

O_EXCL Exclusive use flag.

O_NOCTTY Do not assign controlling terminal.

O_NOFOLLOW Do not follow symbolic links.

O_TRUNC Truncate flag.

O_TTY_INIT Set the `termios` structure terminal parameters to a state that provides conforming behavior; see Section 11.2, Param?

eters that Can be Set.

The `O_TTY_INIT` flag can have the value zero and in this case it need not be bitwise-distinct from the other flags.

The `<fcntl.h>` header shall define the following symbolic constants for use as file status flags for `open()`, `openat()`, and `fcntl()`. The values shall be suitable for use in `#if` preprocessing directives.

`O_APPEND` Set append mode.

`O_DSYNC` Write according to synchronized I/O data integrity completion.

`O_NONBLOCK` Non-blocking mode.

`O_RSYNC` Synchronized read I/O operations.

`O_SYNC` Write according to synchronized I/O file integrity completion.

The `<fcntl.h>` header shall define the following symbolic constant for use as the mask for file access modes. The value shall be suitable for use in `#if` preprocessing directives.

`O_ACCMODE` Mask for file access modes.

The `<fcntl.h>` header shall define the following symbolic constants for use as the file access modes for `open()`, `openat()`, and `fcntl()`. The values shall be unique, except that `O_EXEC` and `O_SEARCH` may have equal values. The values shall be suitable for use in `#if` preprocessing directives.

`O_EXEC` Open for execute only (non-directory files). The result is unspecified if this flag is applied to a directory.

`O_RDONLY` Open for reading only.

`O_RDWR` Open for reading and writing.

`O_SEARCH` Open directory for search only. The result is unspecified if this flag is applied to a non-directory file.

`O_WRONLY` Open for writing only.

The `<fcntl.h>` header shall define the symbolic constants for file modes for use as values of `mode_t` as described in `<sys/stat.h>`.

The `<fcntl.h>` header shall define the following symbolic constant as a special value used in place of a file descriptor for the `*at()` func?

tions which take a directory file descriptor as a parameter:

AT_FDCWD Use the current working directory to determine the target of relative file paths.

The `<fcntl.h>` header shall define the following symbolic constant as a value for the flag used by `faccessat()`:

AT_EACCESS Check access using effective user and group ID.

The `<fcntl.h>` header shall define the following symbolic constant as a value for the flag used by `fstatat()`, `fchmodat()`, `fchownat()`, and `utimensat()`:

AT_SYMLINK_NOFOLLOW

Do not follow symbolic links.

The `<fcntl.h>` header shall define the following symbolic constant as a value for the flag used by `linkat()`:

AT_SYMLINK_FOLLOW

Follow symbolic link.

The `<fcntl.h>` header shall define the following symbolic constant as a value for the flag used by `unlinkat()`:

AT_REMOVEDIR

Remove directory instead of file.

The `<fcntl.h>` header shall define the following symbolic constants for the advice argument used by `posix_fadvise()`:

POSIX_FADV_DONTNEED

The application expects that it will not access the specified data in the near future.

POSIX_FADV_NOREUSE

The application expects to access the specified data once and then not reuse it thereafter.

POSIX_FADV_NORMAL

The application has no advice to give on its behavior with respect to the specified data. It is the default characteristic if no advice is given for an open file.

POSIX_FADV_RANDOM

The application expects to access the specified data in a random

order.

POSIX_FADV_SEQUENTIAL

The application expects to access the specified data sequentially from lower offsets to higher offsets.

POSIX_FADV_WILLNEED

The application expects to access the specified data in the near future.

The `<fcntl.h>` header shall define the flock structure describing a file lock. It shall include the following members:

short l_type Type of lock; F_RDLCK, F_WRLCK, F_UNLCK.

short l_whence Flag for starting offset.

off_t l_start Relative offset in bytes.

off_t l_len Size; if 0 then until EOF.

pid_t l_pid Process ID of the process holding the lock; returned with F_GETLK.

The `<fcntl.h>` header shall define the mode_t, off_t, and pid_t types as described in `<sys/types.h>`.

The following shall be declared as functions and may also be defined as macros. Function prototypes shall be provided.

```
int creat(const char *, mode_t);
```

```
int fcntl(int, int, ...);
```

```
int open(const char *, int, ...);
```

```
int openat(int, const char *, int, ...);
```

```
int posix_fadvise(int, off_t, off_t, int);
```

```
int posix_fallocate(int, off_t, off_t);
```

Inclusion of the `<fcntl.h>` header may also make visible all symbols from `<sys/stat.h>` and `<unistd.h>`.

The following sections are informative.

APPLICATION USAGE

Although no existing implementation defines `AT_SYMLINK_FOLLOW` and `AT_SYMLINK_NOFOLLOW` as the same numeric value, POSIX.1?2008 does not prohibit that as the two constants are not used with the same interfaces.

While many of the symbolic constants introduced in the `<fcntl.h>` header do not strictly need to be used in `#if` preprocessor directives, widespread historic practice has defined them as macros that are usable in such constructs, and examination of existing applications has shown that they are occasionally used in such a way. Therefore it was decided to retain this requirement on an implementation in POSIX.1-2008.

FUTURE DIRECTIONS

None.

SEE ALSO

`<stdio.h>`, `<sys_stat.h>`, `<sys_types.h>`, `<unistd.h>`

The System Interfaces volume of POSIX.1-2017, `creat()`, `exec`, `fcntl()`, `futimens()`, `open()`, `posix_fadvise()`, `posix_fallocate()`, `posix_madvise()`

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.