NAME

open - open a file

SYNOPSIS

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

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

DESCRIPTION

The *open()* function establishes the connection between a file and a file descriptor. It creates an open file description that refers to a file and a file descriptor that refers to that open file description. The file descriptor is used by other I/O functions to refer to that file. The *path* argument points to a pathname naming the file.

The *open()* function will return a file descriptor for the named file that is the lowest file descriptor not currently open for that process. The open file description is new, and therefore the file descriptor does not share it with any other process in the system. The FD_CLOEXEC file descriptor flag associated with the new file descriptor will be cleared.

The file offset used to mark the current position within the file is set to the beginning of the file.

The file status flags and file access modes of the open file description will be set according to the value of *oflag*.

Values for *oflag* are constructed by a bitwise-inclusive-OR of flags from the following list, defined in *<fcntl.h>*. Applications must specify exactly one of the first three values (file access modes) below in the value of *oflag*:

O_RDONLY
: Open for reading only.

O_WRONLY
: Open for writing only.

O_RDWR
: Open for reading and writing. The result is undefined if this flag is applied to a FIFO.

Any combination of the following may be used:

O_APPEND
: If set, the file offset will be set to the end of the file prior to each write.

O_CREAT
: If the file exists, this flag has no effect except as noted under O_EXCL below. Otherwise, the file is created; the user ID of the file is set to the effective user ID of the process; the group ID of the file is set to the group ID of the file's parent directory or to the effective group ID of the process; and the access permission bits (see *<sys/stat.h>*) of the file mode are set to the value of the third argument taken as type **mode_t** modified as follows: a bitwise-AND is performed on the file-mode bits and the corresponding bits in the complement of the process' file mode creation mask. Thus, all bits in the file mode whose

corresponding bit in the file mode creation mask is set are cleared. When bits other than the file permission bits are set, the effect is unspecified. The third argument does not affect whether the file is open for reading, writing or for both.

O_DSYNC
: Write I/O operations on the file descriptor complete as defined by synchronised I/O data integrity completion

O_EXCL
: If O_CREAT and O_EXCL are set, *open()* will fail if the file exists. The check for the existence of the file and the creation of the file if it does not exist will be atomic with respect to other processes executing *open()* naming the same filename in the same directory with O_EXCL and O_CREAT set. If O_CREAT is not set, the effect is undefined.

O_NOCTTY
: If set and *path* identifies a terminal device, *open()* will not cause the terminal device to become the controlling terminal for the process.

O_NONBLOCK
: When opening a FIFO with O_RDONLY or O_WRONLY set: If O_NONBLOCK is set:

  An *open()* for reading only will return without delay. An *open()* for writing only will return an error if no process currently has the file open for reading.

  If O_NONBLOCK is clear:

  An *open()* for reading only will block the calling thread until a thread opens the file for writing. An *open()* for writing only will block the calling thread until a thread opens the file for reading.

  When opening a block special or character special file that supports non-blocking opens:

  If O_NONBLOCK is set:

  The *open()* function will return without blocking for the device to be ready or available. Subsequent behaviour of the device is device-specific.

  If O_NONBLOCK is clear:

  The *open()* function will block the calling thread until the device is ready or available before returning.

  Otherwise, the behaviour of O_NONBLOCK is unspecified.

O_RSYNC
: Read I/O operations on the file descriptor complete at the same level of integrity as specified by the O_DSYNC and O_SYNC flags. If both O_DSYNC and O_RSYNC are set in *oflag*, all I/O operations on the file descriptor complete as defined by synchronised I/O data integrity completion. If both O_SYNC and O_RSYNC are set in flags, all I/O operations on the file descriptor complete as defined by synchronised I/O file integrity completion.

O_SYNC
: Write I/O operations on the file descriptor complete as defined by synchronised I/O file integrity completion.

O_TRUNC
: If the file exists and is a regular file, and the file is successfully opened O_RDWR or O_WRONLY, its length is truncated to 0 and the mode and owner are unchanged. It will have no effect on FIFO special files or terminal device files. Its effect on other file types is implementation-dependent. The result of using O_TRUNC with O_RDONLY is undefined.

If O_CREAT is set and the file did not previously exist, upon successful completion, *open()* will mark for update the *st_atime*, *st_ctime* and *st_mtime* fields of the file and the *st_ctime* and *st_mtime* fields of the parent directory.

If O_TRUNC is set and the file did previously exist, upon successful completion, *open()* will mark for update the *st_ctime* and *st_mtime* fields of the file.

If both the O_SYNC and O_DSYNC flags are set, the effect is as if only the O_SYNC flag was set.

If *path* refers to a STREAMS file, *oflag* may be constructed from O_NONBLOCK OR-ed with either O_RDONLY, O_WRONLY or O_RDWR. Other flag values are not applicable to STREAMS devices and have no effect on them. The value O_NONBLOCK affects the operation of STREAMS drivers and certain functions applied to file descriptors associated with STREAMS files. For STREAMS drivers, the implementation of O_NONBLOCK is device-specific.

If *path* names the master side of a pseudo-terminal device, then it is unspecified whether *open()* locks the slave side so that it cannot be opened. Portable applications must call *unlockpt()* before opening the slave side.

The largest value that can be represented correctly in an object of type **off_t** will be established as the offset maximum in the open file description.

RETURN VALUE

Upon successful completion, the function will open the file and return a non-negative integer representing the lowest numbered unused file descriptor. Otherwise, -1 is returned and *errno* is set to indicate the error. No files will be created or modified if the function returns -1.

ERRORS

The *open()* function will fail if:

[EACCES]
: Search permission is denied on a component of the path prefix, or the file exists and the permissions specified by *oflag* are denied, or the file does not exist and write permission is denied for the parent directory of the file to be created, or O_TRUNC is specified and write permission is denied.

[EEXIST]
: O_CREAT and O_EXCL are set, and the named file exists.

[EINTR]
: A signal was caught during *open()*.

[EINVAL]
: The implementation does not support synchronised I/O for this file.

[EIO]
: The *path* argument names a STREAMS file and a hangup or error occurred during the *open()*.

[EISDIR]
: The named file is a directory and *oflag* includes O_WRONLY or O_RDWR.

[ELOOP]
: Too many symbolic links were encountered in resolving *path*.

[EMFILE]
: {OPEN_MAX} file descriptors are currently open in the calling process.

[ENAMETOOLONG]
: The length of the *path* argument exceeds {PATH_MAX} or a pathname component is longer than {NAME_MAX}.

[ENFILE]

The maximum allowable number of files is currently open in the system.

[ENOENT]
O_CREAT is not set and the named file does not exist; or O_CREAT is set and either the path prefix does not exist or the *path* argument points to an empty string.

[ENOSR]
The *path* argument names a STREAMS-based file and the system is unable to allocate a STREAM.

[ENOSPC]
The directory or file system that would contain the new file cannot be expanded, the file does not exist, and O_CREAT is specified.

[ENOTDIR]
A component of the path prefix is not a directory.

[ENXIO]
O_NONBLOCK is set, the named file is a FIFO, O_WRONLY is set and no process has the file open for reading.

[ENXIO]
The named file is a character special or block special file, and the device associated with this special file does not exist.

[EOVERFLOW]
The named file is a regular file and the size of the file cannot be represented correctly in an object of type **off_t**.

[EROFS]
The named file resides on a read-only file system and either O_WRONLY, O_RDWR, O_CREAT (if file does not exist) or O_TRUNC is set in the *oflag* argument.

The *open()* function may fail if:

[EAGAIN]
The *path* argument names the slave side of a pseudo-terminal device that is locked.

[EINVAL]
The value of the *oflag* argument is not valid.

[ENAMETOOLONG]
Pathname resolution of a symbolic link produced an intermediate result whose length exceeds {PATH_MAX}.

[ENOMEM]
The *path* argument names a STREAMS file and the system is unable to allocate resources.

[ETXTBSY]
The file is a pure procedure (shared text) file that is being executed and *oflag* is O_WRONLY or O_RDWR.

EXAMPLES

None.

APPLICATION USAGE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

chmod(), *close()*, *creat()*, *dup()*, *fcntl()*, *lseek()*, *read()*, *umask()*, *unlockpt()*, *write()*, *<fcntl.h>*, *<sys/stat.h>*, *<sys/types.h>*.

DERIVATION

Derived from Issue 1 of the SVID.

[Main Index | XSH | XCU | XBD | XCURSES | XNS]