NAME

open − open a file for reading or writing, or create a new file

SYNOPSIS

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

int open(const char ****path*, int** *flags* [**, mode_t** *mode*]**)**

DESCRIPTION

Open opens the file *path* for reading and/or writing, as specified by the *flags* argument and returns a descriptor for that file. The *flags* argument may indicate the file is to be created if it does not already exist (by specifying the O_CREAT flag), in which case the file is created with mode *mode* as described in **chmod**(2) and modified by the process' umask value (see **umask**(2)).

Path is the address of a string of ASCII characters representing a path name, terminated by a null character. The flags specified are formed by *or*'ing the following values

O_RDONLY	open for reading only
O_WRONLY	open for writing only
O_RDWR	open for reading and writing
O_NONBLOCK	do not block on open
O_APPEND	append on each write
O_CREAT	create file if it does not exist
O_TRUNC	truncate size to 0
O_EXCL	error if create and file exists

Opening a file with O_APPEND set causes each write on the file to be appended to the end. If O_TRUNC is specified and the file exists, the file is truncated to zero length. If O_EXCL is set with O_CREAT, then if the file already exists, the open returns an error. This can be used to implement a simple exclusive access locking mechanism. If O_EXCL is set and the last component of the pathname is a symbolic link, the open will fail even if the symbolic link points to a non-existent name. If the O_NONBLOCK flag is specified and the open call would result in the process being blocked for some reason, the open returns immediately.

Upon successful completion a non-negative integer termed a file descriptor is returned. The file pointer used to mark the current position within the file is set to the beginning of the file.

The new descriptor is set to remain open across **execve** system calls; see **close**(2).

The system imposes a limit on the number of descriptors open simultaneously by one process.

ERRORS

The named file is opened unless one or more of the following are true:

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

[ENAMETOOLONG]
The path name exceeds PATH_MAX characters.

[ENOENT] O_CREAT is not set and the named file does not exist.

[ENOENT] A component of the path name that must exist does not exist.

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

[EACCES] The required permissions (for reading and/or writing) are denied for the named file.

[EACCES] O_CREAT is specified, the file does not exist, and the directory in which it is to be created does not permit writing.

[EACCES] A device to be opened for writing is physically write protected.

[ELOOP] Too many symbolic links were encountered in translating the pathname. (Minix-vmd)

[EISDIR] The named file is a directory, and the arguments specify it is to be opened for writing.

[EROFS] The named file resides on a read-only file system, and the file is to be modified.

[EMFILE] The system limit for open file descriptors per process has already been reached.

[ENFILE] The system file table is full.

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

[ENOSPC] O_CREAT is specified, the file does not exist, and the directory in which the entry for the new file is being placed cannot be extended because there is no space left on the file system containing the directory.

[ENOSPC] O_CREAT is specified, the file does not exist, and there are no free inodes on the file system on which the file is being created.

[EIO] An I/O error occurred while making the directory entry or allocating the inode for O_CREAT.

[EFAULT] *Path* points outside the process's allocated address space.

[EEXIST] O_CREAT and O_EXCL were specified and the file exists.

SEE ALSO

chmod(2), **close**(2), **dup**(2), **fcntl**(2), **lseek**(2), **read**(2), **write**(2), **umask**(2).