NAME

fopen, fdopen, freopen – stream open functions

SYNOPSIS

#include <stdio.h>

FILE *fopen(const char * *path***, const char *** *mode***);**

FILE *fdopen(int *fd***, const char *** *mode***);**

FILE *freopen(const char * *path***, const char *** *mode***, FILE *** *stream***);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

fdopen(): _POSIX_C_SOURCE >= 1 || _XOPEN_SOURCE || _POSIX_SOURCE

DESCRIPTION

The **fopen**() function opens the file whose name is the string pointed to by *path* and associates a stream with it.

The argument *mode* points to a string beginning with one of the following sequences (Additional characters may follow these sequences.):

r Open text file for reading. The stream is positioned at the beginning of the file.

r+ Open for reading and writing. The stream is positioned at the beginning of the file.

w Truncate file to zero length or create text file for writing. The stream is positioned at the beginning of the file.

w+ Open for reading and writing. The file is created if it does not exist, otherwise it is truncated. The stream is positioned at the beginning of the file.

a Open for appending (writing at end of file). The file is created if it does not exist. The stream is positioned at the end of the file.

a+ Open for reading and appending (writing at end of file). The file is created if it does not exist. The initial file position for reading is at the beginning of the file, but output is always appended to the end of the file.

The *mode* string can also include the letter 'b' either as a last character or as a character between the characters in any of the two-character strings described above. This is strictly for compatibility with C89 and has no effect; the 'b' is ignored on all POSIX conforming systems, including Linux. (Other systems may treat text files and binary files differently, and adding the 'b' may be a good idea if you do I/O to a binary file and expect that your program may be ported to non-Unix environments.)

See NOTES below for details of glibc extensions for *mode*.

Any created files will have mode **S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP | S_IROTH | S_IWOTH** (0666), as modified by the process's umask value (see **umask**(2)).

Reads and writes may be intermixed on read/write streams in any order. Note that ANSI C requires that a file positioning function intervene between output and input, unless an input operation encounters end-of-file. (If this condition is not met, then a read is allowed to return the result of writes other than the most recent.) Therefore it is good practice (and indeed sometimes necessary under Linux) to put an **fseek**(3) or **fgetpos**(3) operation between write and read operations on such a stream. This operation may be an apparent no-op (as in *fseek(..., 0L, SEEK_CUR)* called for its synchronizing side effect.

Opening a file in append mode (**a** as the first character of *mode*) causes all subsequent write operations to this stream to occur at end-of-file, as if preceded by an

```
fseek(stream,0,SEEK_END);
```

call.

The **fdopen**() function associates a stream with the existing file descriptor, *fd*. The *mode* of the stream (one of the values "r", "r+", "w", "w+", "a", "a+") must be compatible with the mode of the file descriptor. The file position indicator of the new stream is set to that belonging to *fd*, and the error and end-of-file indicators are cleared. Modes "w" or "w+" do not cause truncation of the file. The file descriptor is not dup'ed, and will be closed when the stream created by **fdopen**() is closed. The result of applying **fdopen**() to a shared memory object is undefined.

The **freopen**() function opens the file whose name is the string pointed to by *path* and associates the stream pointed to by *stream* with it. The original stream (if it exists) is closed. The *mode* argument is used just as in the **fopen**() function. The primary use of the **freopen**() function is to change the file associated with a standard text stream (*stderr*, *stdin*, or *stdout*).

RETURN VALUE

Upon successful completion **fopen**(), **fdopen**() and **freopen**() return a *FILE* pointer. Otherwise, NULL is returned and *errno* is set to indicate the error.

ERRORS

EINVAL

The *mode* provided to **fopen**(), **fdopen**(), or **freopen**() was invalid.

The **fopen**(), **fdopen**() and **freopen**() functions may also fail and set *errno* for any of the errors specified for the routine **malloc**(3).

The **fopen**() function may also fail and set *errno* for any of the errors specified for the routine **open**(2).

The **fdopen**() function may also fail and set *errno* for any of the errors specified for the routine **fcntl**(2).

The **freopen**() function may also fail and set *errno* for any of the errors specified for the routines **open**(2), **fclose**(3) and **fflush**(3).

CONFORMING TO

The **fopen**() and **freopen**() functions conform to C89. The **fdopen**() function conforms to POSIX.1-1990.

NOTES

Glibc Notes

The GNU C library allows the following extensions for the string specified in *mode*:

c (since glibc 2.3.3)

Do not make the open operation, or subsequent read and write operations, thread cancellation points.

e (since glibc 2.7)

Open the file with the **O_CLOEXEC** flag. See **open**(2) for more information.

m (since glibc 2.3)

Attempt to access the file using **mmap**(2), rather than I/O system calls (**read**(2), **write**(2)). Currently, use of **mmap**(2) is only attempted for a file opened for reading.

x Open the file exclusively (like the **O_EXCL** flag of **open**(2)). If the file already exists, **fopen**() fails, and sets *errno* to **EEXIST**. This flag is ignored for **fdopen**().

SEE ALSO

open(2), **fclose**(3), **fileno**(3), **fmemopen**(3), **fopencookie**(3)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.