Rocky Enterprise Linux 9.2 Manual Pages on command 'freopen.3'

$ man freopen.3

FOPEN(3)   Linux Programmer's Manual   FOPEN(3)

NAME

    fopen, fdopen, freopen - stream open functions

SYNOPSIS

    #include <stdio.h>

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

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

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

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

    fdopen(): _POSIX_C_SOURCE

DESCRIPTION

    The fopen() function opens the file whose name is the string pointed to by pathname and

    associates a stream with it.

    The argument mode points to a string beginning with one of the following sequences (possi?

    bly followed by additional characters, as described below):

    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 posi?

          tioned 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. Output is always appended to the end of the file. POSIX is silent
on what the initial read position is when using this mode. For glibc, the initial
file position for reading is at the beginning of the file, but for An?
droid/BSD/MacOS, the initial file position for reading is at the end of the file.

The mode string can also include the letter 'b' either as a last character or as a charac?
ter 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 ex?
pect that your program may be ported to non-UNIX environments.)

See NOTES below for details of glibc extensions for mode.

Any created file will have the 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 al?
lowed 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) op?
eration between write and read operations on such a stream. This operation may be an ap?
parent 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 the call:

   fseek(stream, 0, SEEK_END);

The file descriptor associated with the stream is opened as if by a call to open(2) with
the following flags:

   ???
   ?fopen() mode ? open() flags   ?
   ???
   ?   r   ? O_RDONLY   ?
   ???
   ?   w   ? O_WRONLY | O_CREAT | O_TRUNC ?

```
??????????????????????????????????????????????????
?    a    ? O_WRONLY | O_CREAT | O_APPEND ?
??????????????????????????????????????????????????
?    r+   ? O_RDWR                ?
??????????????????????????????????????????????????
?    w+   ? O_RDWR | O_CREAT | O_TRUNC   ?
??????????????????????????????????????????????????
?    a+   ? O_RDWR | O_CREAT | O_APPEND  ?
??????????????????????????????????????????????????
```

fdopen()

   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.

freopen()

   The freopen() function opens the file whose name is the string pointed to by pathname 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.

   If the pathname argument is a null pointer, freopen() changes the mode of the stream to
   that specified in mode; that is, freopen() reopens the pathname that is associated with
   the stream.   The specification for this behavior was added in the C99 standard, which
   says:

       In this case, the file descriptor associated with the stream need not be closed if
       the call to freopen() succeeds. It is implementation-defined which changes of mode
       are permitted (if any), and under what circumstances.

   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. Other?
   wise, 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).

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

??

?Interface   ? Attribute   ? Value   ?

??

?fopen(), fdopen(), freopen() ? Thread safety ? MT-Safe ?

??

CONFORMING TO

fopen(), freopen(): POSIX.1-2001, POSIX.1-2008, C89, C99.

fdopen(): POSIX.1-2001, POSIX.1-2008.

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. This flag is ignored for fdopen().

e (since glibc 2.7)

Open the file with the O_CLOEXEC flag. See open(2) for more information. This flag is ignored for fdopen().

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 attempted only for a file opened for read‐ ing.

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().

In addition to the above characters, fopen() and freopen() support the following syntax in

mode:

,ccs=string

The given string is taken as the name of a coded character set and the stream is marked as

wide-oriented. Thereafter, internal conversion functions convert I/O to and from the

character set string. If the ,ccs=string syntax is not specified, then the wide-orienta?

tion of the stream is determined by the first file operation.   If that operation is a

wide-character operation, the stream is marked wide-oriented, and functions to convert to

the coded character set are loaded.

BUGS

When parsing for individual flag characters in mode (i.e., the characters preceding the

"ccs" specification), the glibc implementation of fopen() and freopen() limits the number

of characters examined in mode to 7 (or, in glibc versions before 2.14, to 6, which was

not enough to include possible specifications such as "rb+cmxe"). The current implementa?

tion of fdopen() parses at most 5 characters in mode.

SEE ALSO

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

COLOPHON

This page is part of release 5.10 of the Linux man-pages project. A description of the

project, information about reporting bugs, and the latest version of this page, can be

found at https://www.kernel.org/doc/man-pages/.

GNU   2020-12-21   FOPEN(3)