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 'freopen.3p' command

$ man freopen.3p

FREOPEN(3P) POSIX Programmer's Manual FREOPEN(3P)

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

freopen ? open a stream

SYNOPSIS

```
#include <stdio.h>
FILE *freopen(const char *restrict pathname, const char *restrict mode,
    FILE *restrict stream);
```

DESCRIPTION

The functionality described on this reference page is aligned with the ISO C standard. Any conflict between the requirements described here and the ISO C standard is unintentional. This volume of POSIX.1?2017 defers to the ISO C standard.

The freopen() function shall first attempt to flush the stream associ? ated with stream as if by a call to fflush(stream). Failure to flush the stream successfully shall be ignored. If pathname is not a null pointer, freopen() shall close any file descriptor associated with stream. Failure to close the file descriptor successfully shall be ig? nored. The error and end-of-file indicators for the stream shall be

cleared.

The freopen() function shall open the file whose pathname is the string pointed to by pathname and associate the stream pointed to by stream with it. The mode argument shall be used just as in fopen().

The original stream shall be closed regardless of whether the subse? quent open succeeds.

If pathname is a null pointer, the freopen() function shall attempt to change the mode of the stream to that specified by mode, as if the name of the file currently associated with the stream had been used. 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.

After a successful call to the freopen() function, the orientation of the stream shall be cleared, the encoding rule shall be cleared, and the associated mbstate_t object shall be set to describe an initial conversion state.

If pathname is not a null pointer, or if pathname is a null pointer and the specified mode change necessitates the file descriptor associated with the stream to be closed and reopened, the file descriptor associ? ated with the reopened stream shall be allocated and opened as if by a call to open() with the following flags:

? freopen() Mode ?	open() Flags ?
?r or rb ?	O_RDONLY ?
?w or wb ?	O_WRONLY|O_CREAT|O_TRUNC ?
?a or ab ?	O_WRONLY|O_CREAT|O_APPEND ?
?r+ or rb+ or r+b ?	O_RDWR ?
?w+ or wb+ or w+b ?	O_RDWR|O_CREAT|O_TRUNC ?
?a+ or ab+ or a+b ?	O_RDWR|O_CREAT|O_APPEND ?

RETURN VALUE

Upon successful completion, freopen() shall return the value of stream.

Otherwise, a null pointer shall be returned, and errno shall be set to indicate the error.

ERRORS

The freopen() function shall fail if:

EACCES Search permission is denied on a component of the path prefix, or the file exists and the permissions specified by mode are de? nied, or the file does not exist and write permission is denied for the parent directory of the file to be created.

EBADF The file descriptor underlying the stream is not a valid file descriptor when pathname is a null pointer.

EINTR A signal was caught during freopen().

EISDIR The named file is a directory and mode requires write access.

ELOOP A loop exists in symbolic links encountered during resolution of the path argument.

EMFILE All file descriptors available to the process are currently open.

ENAMETOOLONG

The length of a component of a pathname is longer than {NAME_MAX}.

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

ENOENT The mode string begins with 'r' and a component of pathname does not name an existing file, or mode begins with 'w' or 'a' and a component of the path prefix of pathname does not name an exist? ing file, or pathname is an empty string.

ENOENT or ENOTDIR

The pathname argument contains at least one non-<slash> charac? ter and ends with one or more trailing <slash> characters. If pathname without the trailing <slash> characters would name an existing file, an [ENOENT] error shall not occur.

ENOSPC The directory or file system that would contain the new file cannot be expanded, the file does not exist, and it was to be created.

ENOTDIR

A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory, or the pathname argument contains at least one non-<slash> character and ends with one or more trailing <slash> characters and the last pathname component names an existing file that is neither a directory nor a symbolic link to a directory.

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 mode re? quires write access.

The freopen() function may fail if:

EBADF The mode with which the file descriptor underlying the stream was opened does not support the requested mode when pathname is a null pointer.

EINVAL The value of the mode argument is not valid.

ELOOP More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the path argument.

ENAMETOOLONG

The length of a pathname exceeds {PATH_MAX}, or pathname resolu? tion of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

ENOMEM Insufficient storage space is available.

ENXIO A request was made of a nonexistent device, or the request was outside the capabilities of the device.

ETXTBSY

The file is a pure procedure (shared text) file that is being executed and mode requires write access.

The following sections are informative.

EXAMPLES

Directing Standard Output to a File

The following example logs all standard output to the /tmp/logfile file.

```
#include <stdio.h>
...
FILE *fp;
...
fp = freopen ("/tmp/logfile", "a+", stdout);
...
```

APPLICATION USAGE

The freopen() function is typically used to attach the pre-opened streams associated with stdin, stdout, and stderr to other files.

Since implementations are not required to support any stream mode changes when the pathname argument is NULL, portable applications can? not rely on the use of freopen() to change the stream mode, and use of this feature is discouraged. The feature was originally added to the ISO C standard in order to facilitate changing stdin and stdout to bi? nary mode. Since a 'b' character in the mode has no effect on POSIX systems, this use of the feature is unnecessary in POSIX applications.

However, even though the 'b' is ignored, a successful call to fre? open(NULL, "wb", stdout) does have an effect. In particular, for regu? lar files it truncates the file and sets the file-position indicator for the stream to the start of the file. It is possible that these side-effects are an unintended consequence of the way the feature is specified in the ISO/IEC 9899:1999 standard, but unless or until the ISO C standard is changed, applications which successfully call fre? open(NULL, "wb", stdout) will behave in unexpected ways on conforming systems in situations such as:

```
{ appl file1; appl file2; } > file3
```

which will result in file3 containing only the output from the second invocation of appl.

RATIONALE

None.

FUTURE DIRECTIONS

None.

SEE ALSO

Section 2.5, Standard I/O Streams, fclose(), fdopen(), fflush(), fmemo?

pen(), fopen(), mbsinit(), open(), open_memstream()

The Base Definitions volume of POSIX.1?2017, <stdio.h>

COPYRIGHT