



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

### ***Rocky Enterprise Linux 9.2 Manual Pages on command 'flockfile.3'***

**\$ man flockfile.3**

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

#### NAME

flockfile, ftrylockfile, funlockfile - lock FILE for stdio

#### SYNOPSIS

```
#include <stdio.h>

void flockfile(FILE *filehandle);

int ftrylockfile(FILE *filehandle);

void funlockfile(FILE *filehandle);
```

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

All functions shown above:

```
/* Since glibc 2.24: */ _POSIX_C_SOURCE >= 199309L

|| /* Glibc versions <= 2.23: */ _POSIX_C_SOURCE

|| /* Glibc versions <= 2.19: */ _BSD_SOURCE || _SVID_SOURCE
```

#### DESCRIPTION

The stdio functions are thread-safe. This is achieved by assigning to each FILE object a lockcount and (if the lockcount is nonzero) an owning thread. For each library call, these functions wait until the FILE object is no longer locked by a different thread, then lock it, do the requested I/O, and unlock the object again.

(Note: this locking has nothing to do with the file locking done by functions like flock(2) and lockf(3).)

All this is invisible to the C-programmer, but there may be two reasons to wish for more detailed control. On the one hand, maybe a series of I/O actions by one thread belongs together, and should not be interrupted by the I/O of some other thread. On the other

hand, maybe the locking overhead should be avoided for greater efficiency.

To this end, a thread can explicitly lock the FILE object, then do its series of I/O actions, then unlock. This prevents other threads from coming in between. If the reason for doing this was to achieve greater efficiency, one does the I/O with the nonlocking versions of the stdio functions: with getc\_unlocked(3) and putc\_unlocked(3) instead of getc(3) and putc(3).

The flockfile() function waits for \*filehandle to be no longer locked by a different thread, then makes the current thread owner of \*filehandle, and increments the lockcount.

The funlockfile() function decrements the lock count.

The frylockfile() function is a nonblocking version of flockfile(). It does nothing in case some other thread owns \*filehandle, and it obtains ownership and increments the lockcount otherwise.

#### RETURN VALUE

The frylockfile() function returns zero for success (the lock was obtained), and nonzero for failure.

#### ERRORS

None.

#### ATTRIBUTES

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

??

?Interface                    ? Attribute   ? Value   ?

??

?flockfile(), frylockfile(), ? Thread safety ? MT-Safe ?

?funlockfile()                ?            ?            ?

??

#### CONFORMING TO

POSIX.1-2001, POSIX.1-2008.

These functions are available when \_POSIX\_THREAD\_SAFE\_FUNCTIONS is defined.

#### SEE ALSO

unlocked\_stdio(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/>.

2020-06-09

FLOCKFILE(3)