NAME

flockfile, funlockfile, ftrylockfile – acquire and release stream lock

SYNOPSIS

#include <stdio.h>

void flockfile(FILE **stream***);**

void funlockfile(FILE **stream***);**

POSIX

cc [*flag* **...]** *file* **... –D_POSIX_PTHREAD_SEMANTICS [** *library* **...]**

int ftrylockfile(FILE **stream***);**

MT-LEVEL

MT-Safe

DESCRIPTION

flockfile( ) acquires an internal lock of a stream *stream*. If the lock is already acquired by another thread, the thread calling **flockfile( )** is suspended until it can acquire the lock. In the case that the stream lock is available, **flockfile( )** not only acquires the lock, but keeps track of the number of times it is being called by the current thread. This implies that the stream lock can be acquired more than once by the same thread.

funlockfile( ) releases the lock being held by the current thread. In the case of recursive locking, this function must be called the same number of times **flockfile( )** was called. After the number of **funlockfile( )** calls is equal to the number of **flockfile( )** calls, the stream lock is available for other threads to acquire.

ftrylockfile( ) acquires an internal lock of a stream *stream*, only if that object is available. In essence **ftrylockfile( )** is a non-blocking version of **flockfile( )**.

RETURN VALUES

ftrylockfile( ) returns zero on success or non-zero to indicate a lock cannot be acquired.

EXAMPLES

For example:

```
FILE iop;
 . . .
flockfile(iop);
fprintf(iop, "hello ");
fprintf(iop, "world0);
fputc(iop, 'a');
funlockfile(iop);
```

will print everything out together, blocking other threads that might want to write to the same file between **fprintf( )**s.

An unlocked interface is available in case performance is an issue. For example:

```
flockfile(iop);
while (!feof(iop)) {
        *c++ = getc_unlocked(iop);
}
funlockfile(iop);
```

SEE ALSO

intro(3), **ferror**(3S), **getc**(3S), **putc**(3S), **stdio**(3S), **ungetc**(3S)

NOTES

When compiling multi-thread applications, the **_REENTRANT** flag must be defined on the compile line. This flag should only be used in multi-thread applications.

The interfaces on this page are as specified in POSIX 1003.1c Draft #10.