NAME

flock - apply or remove an advisory lock on an open file

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

```
#include <sys/file.h>

#define   LOCK_SH       0x01      /* shared file lock */
#define   LOCK_EX       0x02      /* exclusive file lock */
#define   LOCK_NB       0x04      /* do not block when locking */
#define   LOCK_UN       0x08      /* unlock file */

int
flock(int fd, int operation);
```

DESCRIPTION

The **flock**() system call applies or removes an *advisory* lock on the file associated with the file descriptor *fd*. A lock is applied by specifying an *operation* argument that is one of LOCK_SH or LOCK_EX with the optional addition of LOCK_NB. To unlock an existing lock operation should be LOCK_UN.

Advisory locks allow cooperating processes to perform consistent operations on files, but do not guarantee consistency (i.e., processes may still access files without using advisory locks possibly resulting in inconsistencies).

The locking mechanism allows two types of locks: *shared* locks and *exclusive* locks. At any time multiple shared locks may be applied to a file, but at no time are multiple exclusive, or both shared and exclusive, locks allowed simultaneously on a file.

A shared lock may be *upgraded* to an exclusive lock, and vice versa, simply by specifying the appropriate lock type; this results in the previous lock being released and the new lock applied (possibly after other processes have gained and released the lock).

Requesting a lock on an object that is already locked normally causes the caller to be blocked until the lock may be acquired. If LOCK_NB is included in *operation*, then this will not happen; instead the call will fail and the error EWOULDBLOCK will be returned.

NOTES

Locks are on files, not file descriptors. That is, file descriptors duplicated through dup(2) or fork(2) do not result in multiple instances of a lock, but rather multiple references to a single lock. If a process holding a lock on a file forks and the child explicitly unlocks the file, the parent will lose its lock.

The **flock**(), fcntl(2), and lockf(3) locks are compatible. Processes using different locking interfaces can cooperate over the same file safely. However, only one of such interfaces should be used within the same process. If a file is locked by a process through **flock**(), any record within the file will be seen as locked from the viewpoint of another process using fcntl(2) or lockf(3), and vice versa.

Processes blocked awaiting a lock may be awakened by signals.

RETURN VALUES

The **flock**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **flock**() system call fails if:

[EWOULDBLOCK]	The file is locked and the LOCK_NB option was specified.
[EBADF]	The argument *fd* is an invalid descriptor.
[EINVAL]	The argument *fd* refers to an object other than a file.
[EOPNOTSUPP]	The argument *fd* refers to an object that does not support file locking.
[ENOLCK]	A lock was requested, but no locks are available.

SEE ALSO

close(2), dup(2), execve(2), fcntl(2), fork(2), open(2), flopen(3), lockf(3)

HISTORY

The **flock**() system call appeared in 4.2BSD.