NAME

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

SYNOPSIS

/usr/ucb/cc [*flag* . . .] *file* . . .

#include <sys/file.h>

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

DESCRIPTION

flock( ) applies or removes an *advisory* lock on the file associated with the file descriptor *fd*. The compatibility version of **flock( )** has been implemented on top of **fcntl**(2) locking. It does not provide complete binary compatibility.

Advisory locks allow cooperating processes to perform consistent operations on files, but do not guarantee exclusive access (that is, 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. More than one process may hold a shared lock for a file at any given time, but multiple exclusive, or both shared and exclusive, locks may not exist simultaneously on a file.

A lock is applied by specifying an *operation* parameter **LOCK_SH** for a shared lock or **LOCK_EX** for an exclusive lock. The *operation* paramerer may be ORed with **LOCK_NB** to make the operation non-blocking. To unlock an existing lock, the *operation* should be **LOCK_UN**.

Read permission is required on a file to obtain a shared lock, and write permission is required to obtain an exclusive lock. Locking a segment that is already locked by the calling process causes the old lock type to be removed and the new lock type to take effect.

Requesting a lock on an object that is already locked normally causes the caller to block 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.

RETURN VALUES

flock( ) returns:

0 on success.

–1 on failure and sets **errno** to indicate the error.

ERRORS

EBADF	The argument *fd* is an invalid descriptor.
EINVAL	*operation* is not a valid argument.
EOPNOTSUPP	The argument *fd* refers to an object other than a file.
EWOULDBLOCK	The file is locked and the **LOCK_NB** option was specified.

SEE ALSO

lockd(1M), **chmod**(2), **close**(2), **dup**(2), **exec**(2), **fcntl**(2), **fork**(2), **open**(2), **lockf**(3C)

NOTES

Use of these interfaces should be restricted to only applications written on BSD platforms. Use of these interfaces with any of the system libraries or in multi-thread applications is unsupported.

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. Locks are not inherited by a child process.

Processes blocked awaiting a lock may be awakened by signals.

Mandatory locking may occur, depending on the mode bits of the file. See **chmod**(2).

Locks obtained through the **flock( )** mechanism under SunOS 4.1 were known only within the system on which they were placed. This is no longer true.