NAME

uu_lock, **uu_unlock**, **uu_lockerr** - acquire and release control of a serial device

LIBRARY

System Utilities Library (libutil, -lutil)

SYNOPSIS

#include <sys/types.h>
#include <libutil.h>

int
uu_lock(*const char *ttyname*);

int
uu_lock_txfr(*const char *ttyname, pid_t pid*);

int
uu_unlock(*const char *ttyname*);

*const char **
uu_lockerr(*int uu_lockresult*);

DESCRIPTION

The **uu_lock**() function attempts to create a lock file called */var/spool/lock/LCK.*. with a suffix given by the passed *ttyname*. If the file already exists, it is expected to contain the process id of the locking program.

If the file does not already exist, or the owning process given by the process id found in the lock file is no longer running, **uu_lock**() will write its own process id into the file and return success.

uu_lock_txfr() transfers lock ownership to another process. **uu_lock**() must have previously been successful.

uu_unlock() removes the lockfile created by **uu_lock**() for the given *ttyname*. Care should be taken that **uu_lock**() was successful before calling **uu_unlock**().

uu_lockerr() returns an error string representing the error *uu_lockresult*, as returned from **uu_lock**().

RETURN VALUES

uu_unlock() returns 0 on success and -1 on failure.

uu_lock() may return any of the following values:

UU_LOCK_INUSE: The lock is in use by another process.

UU_LOCK_OK: The lock was successfully created.

UU_LOCK_OPEN_ERR: The lock file could not be opened via open(2).

UU_LOCK_READ_ERR: The lock file could not be read via read(2).

UU_LOCK_CREAT_ERR: Cannot create temporary lock file via creat(2).

UU_LOCK_WRITE_ERR: The current process id could not be written to the lock file via a call to write(2).

UU_LOCK_LINK_ERR: Cannot link temporary lock file via link(2).

UU_LOCK_TRY_ERR: Locking attempts are failed after 5 tries.

If a value of UU_LOCK_OK is passed to **uu_lockerr**(), an empty string is returned. Otherwise, a string specifying the reason for failure is returned. **uu_lockerr**() uses the current value of *errno* to determine the exact error. Care should be made not to allow *errno* to be changed between calls to **uu_lock**() and **uu_lockerr**().

uu_lock_txfr() may return any of the following values:

UU_LOCK_OK: The transfer was successful. The specified process now holds the device lock.

UU_LOCK_OWNER_ERR: The current process does not already own a lock on the specified device.

UU_LOCK_WRITE_ERR: The new process id could not be written to the lock file via a call to write(2).

ERRORS

If **uu_lock**() returns one of the error values above, the global value *errno* can be used to determine the cause. Refer to the respective manual pages for further details.

uu_unlock() will set the global variable *errno* to reflect the reason that the lock file could not be removed. Refer to the description of unlink(2) for further details.

SEE ALSO

lseek(2), open(2), read(2), write(2)

HISTORY

The functions **uu_lock**(), **uu_unlock**() and **uu_lockerr**() first appeared in FreeBSD 2.0.5.

BUGS

It is possible that a stale lock is not recognised as such if a new processes is assigned the same processes id as the program that left the stale lock.

The calling process must have write permissions to the */var/spool/lock* directory. There is no mechanism in place to ensure that the permissions of this directory are the same as those of the serial devices that might be locked.