NAME

mlock, munlock – lock (or unlock) pages in memory

SYNOPSIS

#include <sys/types.h>

int mlock(caddr_t *addr***, size_t** *len***);**

int munlock(caddr_t *addr***, size_t** *len***);**

MT-LEVEL

MT-Safe

DESCRIPTION

The function **mlock( )** uses the mappings established for the address range [*addr*, *addr* + *len*) to identify pages to be locked in memory. If the page identified by a mapping changes, such as occurs when a copy of a writable **MAP_PRIVATE** page is made upon the first store, the lock will be transferred to the newly copied private page.

munlock( ) removes locks established with **mlock( )**.

A given page may be locked multiple times by executing an **mlock( )** through different mappings. That is, if two different processes lock the same page, then the page will remain locked until both processes remove their locks. However, within a given mapping, page locks do not nest – multiple **mlock( )** operations on the same address in the same process will all be removed with a single **munlock( )**. Of course, a page locked in one process and mapped in another (or visible through a different mapping in the locking process) is still locked in memory. This fact can be used to create applications that do nothing other than lock important data in memory, thereby avoiding page I/O faults on references from other processes in the system.

If the mapping through which an **mlock( )** has been performed is removed, an **munlock( )** is implicitly performed. An **munlock( )** is also performed implicitly when a page is deleted through file removal or truncation.

Locks established with **mlock( )** are not inherited by a child process after a **fork( )** and are not nested.

Because of the impact on system resources, the use of **mlock( )** and **munlock( )** is restricted to the super-user.

Attempts to **mlock( )** more memory than a system-specific limit will fail.

RETURN VALUES

Upon successful completion, the functions **mlock( )** and **munlock( )** return 0; otherwise, they return −1 and set **errno** to indicate the error.

ERRORS

EAGAIN — **mlock( )** only. Some or all of the memory identified by the range [*addr*, *addr* + *len*) could not be locked because of insufficient system resources.

EINVAL — *addr* is not a multiple of the page size as returned by **sysconf**(3C).

ENOMEM — Addresses in the range [*addr*, *addr* + *len*) are invalid for the address space of a process, or specify one or more pages which are not mapped.

EPERM — The process's effective user ID is not super-user.

SEE ALSO

fork(2), **memcntl**(2), **mmap**(2), **plock**(3C), **mlockall**(3C), **sysconf**(3C)

NOTES

mlock and **munlock** require super-user privileges.