## NAME

**mlockall**, **munlockall** - lock (unlock) the address space of a process

## LIBRARY

Standard C Library (libc, -lc)

## SYNOPSIS

**#include <sys/mman.h>**

*int*
**mlockall**(*int flags*);

*int*
**munlockall**(*void*);

## DESCRIPTION

The **mlockall**() system call locks into memory the physical pages associated with the address space of a process until the address space is unlocked, the process exits, or execs another program image.

The following flags affect the behavior of **mlockall**():

MCL_CURRENT Lock all pages currently mapped into the process's address space.

MCL_FUTURE Lock all pages mapped into the process's address space in the future, at the time the mapping is established. Note that this may cause future mappings to fail if those mappings cause resource limits to be exceeded.

Since physical memory is a potentially scarce resource, processes are limited in how much they can lock down. A single process can lock the minimum of a system-wide "wired pages" limit *vm.max_user_wired* and the per-process RLIMIT_MEMLOCK resource limit.

If *security.bsd.unprivileged_mlock* is set to 0 these calls are only available to the super-user. If *vm.old_mlock* is set to 1 the per-process RLIMIT_MEMLOCK resource limit will not be applied for **mlockall**() calls.

The **munlockall**() call unlocks any locked memory regions in the process address space. Any regions mapped after an **munlockall**() call will not be locked.

## RETURN VALUES

A return value of 0 indicates that the call succeeded and all pages in the range have either been locked or

unlocked. A return value of -1 indicates an error occurred and the locked status of all pages in the range remains unchanged. In this case, the global location *errno* is set to indicate the error.

## ERRORS

**mlockall**() will fail if:

| | |
|---|---|
| [EINVAL] | The *flags* argument is zero, or includes unimplemented flags. |
| [ENOMEM] | Locking the indicated range would exceed either the system or per-process limit for locked memory. |
| [EAGAIN] | Some or all of the memory mapped into the process’s address space could not be locked when the call was made. |
| [EPERM] | The calling process does not have the appropriate privilege to perform the requested operation. |

## SEE ALSO

mincore(2), mlock(2), mmap(2), munmap(2), setrlimit(2)

## STANDARDS

The **mlockall**() and **munlockall**() functions are believed to conform to IEEE Std 1003.1-2001 ("POSIX.1").

## HISTORY

The **mlockall**() and **munlockall**() functions first appeared in FreeBSD 5.1.

## BUGS

The per-process and system-wide resource limits of locked memory apply to the amount of virtual memory locked, not the amount of locked physical pages. Hence two distinct locked mappings of the same physical page counts as 2 pages aginst the system limit, and also against the per-process limit if both mappings belong to the same physical map.