NAME

memcntl – memory management control

SYNOPSIS

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

int memcntl(caddr_t *addr*, **size_t** *len*, **int** *cmd*, **caddr_t** *arg*, **int** *attr*, **int** *mask***);**

MT-LEVEL

MT-Safe

DESCRIPTION

The function **memcntl( )** allows the calling process to apply a variety of control operations over the address space identified by the mappings established for the address range [*addr*, *addr* + *len*).

addr must be a multiple of the pagesize as returned by **sysconf**(3C). The scope of the control operations can be further defined with additional selection criteria (in the form of attributes) according to the bit pattern contained in *attr*.

The following attributes specify page mapping selection criteria:

SHARED Page is mapped shared.

PRIVATE Page is mapped private.

The following attributes specify page protection selection criteria:

PROT_READ Page can be read.

PROT_WRITE Page can be written.

PROT_EXEC Page can be executed.

The selection criteria are constructed by an OR of the attribute bits and must match exactly.

In addition, the following criteria may be specified:

PROC_TEXT Process text.

PROC_DATA Process data.

where **PROC_TEXT** specifies all privately mapped segments with read and execute permission, and **PROC_DATA** specifies all privately mapped segments with write permission.

Selection criteria can be used to describe various abstract memory objects within the address space on which to operate. If an operation shall not be constrained by the selection criteria, *attr* must have the value **0**.

The operation to be performed is identified by the argument *cmd*. The symbolic names for the operations are defined in **<sys/mman.h>** as follows:

MC_LOCK Lock in memory all pages in the range with attributes *attr*. A given page may be locked multiple times through different mappings; however, within a given mapping, page locks do not nest. Multiple lock operations on the same address in the same process will all be removed with a single unlock operation. A page locked in one process and mapped in another (or visible through a different mapping in the locking process) is locked in memory as long as the locking process does neither an implicit nor explicit unlock operation. If a locked mapping is removed, or a page is deleted through file removal or truncation, an unlock operation is implicitly performed. If a writable **MAP_PRIVATE** page in the address range is changed, the lock will be transferred to the private page.

At present *arg* is unused, but must be **0** to ensure compatibility with potential future enhancements.

MC_LOCKAS Lock in memory all pages mapped by the address space with attributes *attr*. At present *addr* and *len* are unused, but must be **NULL** and **0** respectively, to ensure compatibility with potential future enhancements. *arg* is a bit pattern built from the flags:

MCL_CURRENT Lock current mappings

MCL_FUTURE Lock future mappings

The value of *arg* determines whether the pages to be locked are those currently mapped by the address space, those that will be mapped in the future, or both. If **MCL_FUTURE** is specified, then all mappings subsequently added to the address space will be locked, provided sufficient memory is available.

MC_SYNC Write to their backing storage locations all modified pages in the range with attributes *attr*. Optionally, invalidate cache copies. The backing storage for a modified **MAP_SHARED** mapping is the file the page is mapped to; the backing storage for a modified **MAP_PRIVATE** mapping is its swap area. *arg* is a bit pattern built from the flags used to control the behavior of the operation:

MS_ASYNC perform asynchronous writes

MS_SYNC perform synchronous writes

MS_INVALIDATE invalidate mappings

MS_ASYNC returns immediately once all write operations are scheduled; with **MS_SYNC** the function will not return until all write operations are completed.

MS_INVALIDATE invalidates all cached copies of data in memory, so that further references to the pages will be obtained by the system from their backing storage locations. This operation should be used by applications that require a memory object to be in a known state.

MC_UNLOCK Unlock all pages in the range with attributes *attr*. At present *arg* is unused, but must be **0** to ensure compatibility with potential future enhancements.

MC_UNLOCKAS
Remove address space memory locks, and locks on all pages in the address space with attributes *attr*. At present *addr*, *len*, and *arg* are unused, but must be **NULL**, **0** and **0** respectively, to ensure compatibility with potential future enhancements.

The *mask* argument must be zero; it is reserved for future use.

Locks established with the lock operations are not inherited by a child process after **fork( )**. **memcntl( )** fails if it attempts to lock more memory than a system-specific limit.

Due to the potential impact on system resources, all operations, with the exception of **MC_SYNC**, are restricted to processes with super-user effective user ID . The **memcntl( )** function subsumes the operations of **plock** and **mctl**.

RETURN VALUES

Upon successful completion, the function **memcntl( )** returns a value of **0**; otherwise, it returns a value of −**1** and sets **errno** to indicate an error.

ERRORS

Under the following conditions, the function **memcntl( )** fails and sets **errno** to:

EAGAIN if some or all of the memory identified by the operation could not be locked when **MC_LOCK** or **MC_LOCKAS** is specified.

EBUSY if some or all the addresses in the range [*addr*, *addr* + *len*) are locked and **MC_SYNC** with **MS_INVALIDATE** option is specified.

EINVAL if *addr* is not a multiple of the page size as returned by **sysconf**.

EINVAL if *addr* and/or *len* do not have the value **0** when **MC_LOCKAS** or **MC_UNLOCKAS** is specified.

EINVAL if *arg* is not valid for the function specified.

EINVAL if invalid selection criteria are specified in *attr*.

ENOMEM if some or all the addresses in the range [*addr*, *addr* + *len*) are invalid for the address space of the process or pages not mapped are specified.

EPERM if the process's effective user ID is not super-user and one of **MC_LOCK**, **MC_LOCKAS**, **MC_UNLOCK**, **MC_UNLOCKAS** was specified.

SEE ALSO

mmap(2), **mprotect**(2), **plock**(3C), **mlock**(3C), **mlockall**(3C), **msync**(3C), **sysconf**(3C)