NAME

msync – synchronize memory with physical storage

SYNOPSIS

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

int msync(caddr_t *addr*, **size_t** *len*, **int** *flags***);**

MT-LEVEL

MT-Safe

DESCRIPTION

The function **msync( )** writes all modified copies of pages over the range [*addr, addr + len*) to their backing storage locations. **msync( )** optionally invalidates any copies so that further references to the pages will be obtained by the system from their backing storage locations. 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.

flags is a bit pattern built from the following values:

MS_ASYNC	perform asynchronous writes
MS_SYNC	perform synchronous writes
MS_INVALIDATE	invalidate mappings

If **MS_ASYNC** is set, **msync( )** returns immediately once all write operations are scheduled; if **MS_SYNC** is set, **msync( )** does 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.

RETURN VALUES

Upon successful completion, the function **msync( )** returns 0; otherwise, it returns –1 and sets **errno** to indicate the error.

ERRORS

EBUSY	Some or all of the addresses in the range [*addr*, *addr + len*) are locked and **MC_SYNC** with the **MS_INVALIDATE** option is specified.
EINVAL	*addr* is not a multiple of the page size as returned by **sysconf**(3C).
	flags is not some combination of **MS_ASYNC** and **MS_INVALIDATE.**
EIO	An I/O error occurred while reading from or writing to the file system.
ENOMEM	Addresses in the range [*addr, addr + len*) are outside the valid range for the address space of a process, or specify one or more pages that are not mapped.
EPERM	**MS_INVALIDATE** was specified and one or more of the pages is locked in memory.

SEE ALSO

mmap(2), **memcntl**(2), **sysconf**(3C)

NOTES

msync( ) should be used by programs that require a memory object to be in a known state, such as in building transaction facilities.