NAME

madvise, **posix_madvise** - give advice about use of memory

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/mman.h>

int
madvise(*void *addr, size_t len, int behav*);

int
posix_madvise(*void *addr, size_t len, int behav*);

DESCRIPTION

The **madvise**() system call allows a process that has knowledge of its memory behavior to describe it to the system. The **posix_madvise**() interface is identical, except it returns an error number on error and does not modify *errno*, and is provided for standards conformance.

The known behaviors are:

MADV_NORMAL Tells the system to revert to the default paging behavior.

MADV_RANDOM Is a hint that pages will be accessed randomly, and prefetching is likely not advantageous.

MADV_SEQUENTIAL Causes the VM system to depress the priority of pages immediately preceding a given page when it is faulted in.

MADV_WILLNEED Causes pages that are in a given virtual address range to temporarily have higher priority, and if they are in memory, decrease the likelihood of them being freed. Additionally, the pages that are already in memory will be immediately mapped into the process, thereby eliminating unnecessary overhead of going through the entire process of faulting the pages in. This WILL NOT fault pages in from backing store, but quickly map the pages already in memory into the calling process.

MADV_DONTNEED Allows the VM system to decrease the in-memory priority of pages in the specified address range. Consequently, future references to this address range

are more likely to incur a page fault.

MADV_FREE Gives the VM system the freedom to free pages, and tells the system that information in the specified page range is no longer important. This is an efficient way of allowing malloc(3) to free pages anywhere in the address space, while keeping the address space valid. The next time that the page is referenced, the page might be demand zeroed, or might contain the data that was there before the MADV_FREE call. References made to that address space range will not make the VM system page the information back in from backing store until the page is modified again.

MADV_NOSYNC Request that the system not flush the data associated with this map to physical backing store unless it needs to. Typically this prevents the file system update daemon from gratuitously writing pages dirtied by the VM system to physical disk. Note that VM/file system coherency is always maintained, this feature simply ensures that the mapped data is only flush when it needs to be, usually by the system pager.

This feature is typically used when you want to use a file-backed shared memory area to communicate between processes (IPC) and do not particularly need the data being stored in that area to be physically written to disk. With this feature you get the equivalent performance with mmap that you would expect to get with SysV shared memory calls, but in a more controllable and less restrictive manner. However, note that this feature is not portable across UNIX platforms (though some may do the right thing by default). For more information see the MAP_NOSYNC section of mmap(2)

MADV_AUTOSYNC Undoes the effects of MADV_NOSYNC for any future pages dirtied within the address range. The effect on pages already dirtied is indeterminate - they may or may not be reverted. You can guarantee reversion by using the msync(2) or fsync(2) system calls.

MADV_NOCORE Region is not included in a core file.

MADV_CORE Include region in a core file.

MADV_PROTECT Informs the VM system this process should not be killed when the swap space is exhausted. The process must have superuser privileges. This should be used judiciously in processes that must remain running for the system to properly function.

Portable programs that call the **posix_madvise**() interface should use the aliases POSIX_MADV_NORMAL, POSIX_MADV_SEQUENTIAL, POSIX_MADV_RANDOM, POSIX_MADV_WILLNEED, and POSIX_MADV_DONTNEED rather than the flags described above.

RETURN VALUES

The **madvise**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

ERRORS

The **madvise**() system call will fail if:

[EINVAL]	The *behav* argument is not valid.
[ENOMEM]	The virtual address range specified by the *addr* and *len* arguments is not valid.
[EPERM]	MADV_PROTECT was specified and the process does not have superuser privileges.

SEE ALSO

mincore(2), mprotect(2), msync(2), munmap(2), posix_fadvise(2)

STANDARDS

The **posix_madvise**() interface conforms to IEEE Std 1003.1-2001 ("POSIX.1").

HISTORY

The **madvise**() system call first appeared in 4.4BSD.