NAME

madvise − give advice about use of memory

SYNOPSIS

#include <sys/mman.h>

int madvise(void ****addr*, size_t** *length***, int** *advice***);**

Feature Test Macro Requirements for glibc (see **feature_test_macros**(7)):

madvise(): _BSD_SOURCE

DESCRIPTION

The **madvise**() system call advises the kernel about how to handle paging input/output in the address range beginning at address *addr* and with size *length* bytes. It allows an application to tell the kernel how it expects to use some mapped or shared memory areas, so that the kernel can choose appropriate read-ahead and caching techniques. This call does not influence the semantics of the application (except in the case of **MADV_DONTNEED**), but may influence its performance. The kernel is free to ignore the advice.

The advice is indicated in the *advice* argument which can be

MADV_NORMAL
No special treatment. This is the default.

MADV_RANDOM
Expect page references in random order. (Hence, read ahead may be less useful than normally.)

MADV_SEQUENTIAL
Expect page references in sequential order. (Hence, pages in the given range can be aggressively read ahead, and may be freed soon after they are accessed.)

MADV_WILLNEED
Expect access in the near future. (Hence, it might be a good idea to read some pages ahead.)

MADV_DONTNEED
Do not expect access in the near future. (For the time being, the application is finished with the given range, so the kernel can free resources associated with it.) Subsequent accesses of pages in this range will succeed, but will result either in re-loading of the memory contents from the underlying mapped file (see **mmap**(2)) or zero-fill-on-demand pages for mappings without an underlying file.

MADV_REMOVE (Since Linux 2.6.16)
Free up a given range of pages and its associated backing store. Currently, only shmfs/tmpfs supports this; other file systems return with the error **ENOSYS**.

MADV_DONTFORK (Since Linux 2.6.16)
Do not make the pages in this range available to the child after a **fork**(2). This is useful to prevent copy-on-write semantics from changing the physical location of a page(s) if the parent writes to it after a **fork**(2). (Such page relocations cause problems for hardware that DMAs into the page(s).)

MADV_DOFORK (Since Linux 2.6.16)
Undo the effect of **MADV_DONTFORK**, restoring the default behavior, whereby a mapping is inherited across **fork**(2).

MADV_DONTDUMP
Exclude from a core dump those pages in the range specified by *addr* and *length*. This is useful in applications that have large areas of memory that are known not to be useful in a core dump. The effect of **MADV_DONTDUMP** takes precedence over the bit mask that is set via the */proc/PID/coredump_filter* file (see **core**(5)).

MADV_DODUMP
Undo the effect of an earlier **MADV_DONTDUMP**.

RETURN VALUE

On success **madvise**() returns zero. On error, it returns −1 and *errno* is set appropriately.

ERRORS

EAGAIN
A kernel resource was temporarily unavailable.

EBADF
The map exists, but the area maps something that isn't a file.

EINVAL
The value *len* is negative, *addr* is not page-aligned, *advice* is not a valid value, or the application is attempting to release locked or shared pages (with **MADV_DONTNEED**).

EIO (for **MADV_WILLNEED**) Paging in this area would exceed the process's maximum resident set size.

ENOMEM
(for **MADV_WILLNEED**) Not enough memory: paging in failed.

ENOMEM
Addresses in the specified range are not currently mapped, or are outside the address space of the process.

CONFORMING TO

POSIX.1b. POSIX.1-2001 describes **posix_madvise**(3) with constants **POSIX_MADV_NORMAL**, etc., with a behavior close to that described here. There is a similar **posix_fadvise**(2) for file access.

MADV_REMOVE, **MADV_DONTFORK**, and **MADV_DOFORK** are Linux-specific.

NOTES

Linux Notes

The current Linux implementation (2.4.0) views this system call more as a command than as advice and hence may return an error when it cannot do what it usually would do in response to this advice. (See the ERRORS description above.) This is non-standard behavior.

The Linux implementation requires that the address *addr* be page-aligned, and allows *length* to be zero. If there are some parts of the specified address range that are not mapped, the Linux version of **madvise**() ignores them and applies the call to the rest (but returns **ENOMEM** from the system call, as it should).

SEE ALSO

getrlimit(2), **mincore**(2), **mmap**(2), **mprotect**(2), **msync**(2), **munmap**(2)

COLOPHON

This page is part of release 3.22 of the Linux *man-pages* project. A description of the project, and information about reporting bugs, can be found at http://www.kernel.org/doc/man-pages/.