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 reloading 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 filesystems 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_HWPOISON (since Linux 2.6.32)
Poison a page and handle it like a hardware memory corruption. This operation is available only for privileged (**CAP_SYS_ADMIN**) processes. This operation may result in the calling process receiving a **SIGBUS** and the page being unmapped. This feature is intended for testing of memory error-handling code; it is available only if the kernel was

configured with **CONFIG_MEMORY_FAILURE**.

MADV_SOFT_OFFLINE (since Linux 2.6.33)
Soft offline the pages in the range specified by *addr* and *length*. The memory of each page in the specified range is preserved (i.e., when next accessed, the same content will be visible, but in a new physical page frame), and the original page is offlined (i.e., no longer used, and taken out of normal memory management). The effect of the **MADV_SOFT_OFFLINE** operation is invisible to (i.e., does not change the semantics of) the calling process. This feature is intended for testing of memory error-handling code; it is available only if the kernel was configured with **CONFIG_MEMORY_FAILURE**.

MADV_MERGEABLE (since Linux 2.6.32)
Enable Kernel Samepage Merging (KSM) for the pages in the range specified by *addr* and *length*. The kernel regularly scans those areas of user memory that have been marked as mergeable, looking for pages with identical content. These are replaced by a single write-protected page (which is automatically copied if a process later wants to update the content of the page). KSM merges only private anonymous pages (see mmap(2)). The KSM feature is intended for applications that generate many instances of the same data (e.g., virtualization systems such as KVM). It can consume a lot of processing power; use with care. See the Linux kernel source file *Documentation/vm/ksm.txt* for more details. The **MADV_MERGEABLE** and **MADV_UNMERGEABLE** operations are available only if the kernel was configured with **CONFIG_KSM**.

MADV_UNMERGEABLE (since Linux 2.6.32)
Undo the effect of an earlier **MADV_MERGEABLE** operation on the specified address range; KSM unmerges whatever pages it had merged in the address range specified by *addr* and *length*.

MADV_HUGEPAGE (since Linux 2.6.38)
Enables Transparent Huge Pages (THP) for pages in the range specified by *addr* and *length*. Currently, Transparent Huge Pages work only with private anonymous pages (see mmap(2)). The kernel will regularly scan the areas marked as huge page candidates to replace them with huge pages. The kernel will also allocate huge pages directly when the region is naturally aligned to the huge page size (see **posix_memalign(2)).** This feature is primarily aimed at applications that use large mappings of data and access large regions of that memory at a time (e.g., virtualization systems such as QEMU). It can very easily waste memory (e.g., a 2MB mapping that only ever accesses 1 byte will result in 2MB of wired memory instead of one 4KB page). See the Linux kernel source file *Documentation/vm/transhuge.txt* for more details. The **MADV_HUGEPAGE** and **MADV_NOHUGEPAGE** operations are available only if the kernel was configured with **CONFIG_TRANSPARENT_HUGEPAGE**.

MADV_NOHUGEPAGE (since Linux 2.6.38)
Ensures that memory in the address range specified by *addr* and *length* will not be collapsed into huge pages.

MADV_DONTDUMP (since Linux 3.4)
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 (since Linux 3.4)
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
This error can occur for the following reasons:

* The value *len* is negative.
* *addr* is not page-aligned.
* *advice* is not a valid value
* The application is attempting to release locked or shared pages (with **MADV_DONTNEED**).
* **MADV_MERGEABLE** or **MADV_UNMERGEABLE** was specified in *advice*, but the kernel was not configured with **CONFIG_KSM**.

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**, **POSIX_MADV_RANDOM**, and so on, with a behavior close to that described here. There is a similar posix_fadvise(2) for file access.

MADV_REMOVE, **MADV_DONTFORK**, **MADV_DOFORK**, **MADV_HWPOISON**, **MADV_MERGEABLE**, and **MADV_UNMERGEABLE** 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 nonstandard 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), prctl(2), core(5)

COLOPHON

This page is part of release 3.74 of the Linux *man-pages* project. A description of the project, information about reporting bugs, and the latest version of this page, can be found at http://www.kernel.org/doc/man-pages/.