NAME

posix_fadvise - predeclare an access pattern for file data

SYNOPSIS

#include <fcntl.h>

int posix_fadvise(int *fd***, off_t** *offset***, off_t** *len***, int** *advice***);**

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

posix_fadvise():
_XOPEN_SOURCE >= 600 || _POSIX_C_SOURCE >= 200112L

DESCRIPTION

Programs can use **posix_fadvise**() to announce an intention to access file data in a specific pattern in the future, thus allowing the kernel to perform appropriate optimizations.

The *advice* applies to a (not necessarily existent) region starting at *offset* and extending for *len* bytes (or until the end of the file if *len* is 0) within the file referred to by *fd*. The*advice* is not binding; it merely constitutes an expectation on behalf of the application.

Permissible values for *advice* include:

POSIX_FADV_NORMAL
Indicates that the application has no advice to give about its access pattern for the specified data. If no advice is given for an open file, this is the default assumption.

POSIX_FADV_SEQUENTIAL
The application expects to access the specified data sequentially (with lower offsets read before higher ones).

POSIX_FADV_RANDOM
The specified data will be accessed in random order.

POSIX_FADV_NOREUSE
The specified data will be accessed only once.

POSIX_FADV_WILLNEED
The specified data will be accessed in the near future.

POSIX_FADV_DONTNEED
The specified data will not be accessed in the near future.

RETURN VALUE

On success, zero is returned. On error, an error number is returned.

ERRORS

EBADF
The *fd* argument was not a valid file descriptor.

EINVAL
An invalid value was specified for *advice*.

ESPIPE
The specified file descriptor refers to a pipe or FIFO. (Linux actually returns **EINVAL** in this case.)

VERSIONS

Kernel support first appeared in Linux 2.5.60; the underlying system call is called **fadvise64**(). Library support has been provided since glibc version 2.2, via the wrapper function **posix_fadvise**().

CONFORMING TO

POSIX.1-2001. Note that the type of the *len* argument was changed from *size_t* to *off_t* in POSIX.1-2003 TC1.

NOTES

Under Linux, **POSIX_FADV_NORMAL** sets the readahead window to the default size for the backing device; **POSIX_FADV_SEQUENTIAL** doubles this size, and **POSIX_FADV_RANDOM** disables file readahead entirely. These changes affect the entire file, not just the specified region (but other open file handles to the same file are unaffected).

POSIX_FADV_WILLNEED initiates a nonblocking read of the specified region into the page cache. The amount of data read may be decreased by the kernel depending on virtual memory load. (A few megabytes will usually be fully satisfied, and more is rarely useful.)

In kernels before 2.6.18, **POSIX_FADV_NOREUSE** had the same semantics as **POSIX_FADV_WILLNEED**. This was probably a bug; since kernel 2.6.18, this flag is a no-op.

POSIX_FADV_DONTNEED attempts to free cached pages associated with the specified region. This is useful, for example, while streaming large files. A program may periodically request the kernel to free cached data that has already been used, so that more useful cached pages are not discarded instead.

Pages that have not yet been written out will be unaffected, so if the application wishes to guarantee that pages will be released, it should call fsync(2) or fdatasync(2) first.

Architecture-specific variants

Some architectures require 64-bit arguments to be aligned in a suitable pair of registers (see syscall(2) for further detail). On such architectures, the call signature of **posix_fadvise**() shown in the SYNOPSIS would force a register to be wasted as padding between the *fd* and *offset* arguments. Therefore, these architectures define a version of the system call that orders the arguments suitably, but otherwise is otherwise exactly the same as **posix_fadvise**().

For example, since Linux 2.6.14, ARM has the following system call:

long arm_fadvise64_64(int *fd***, int** *advice***,**
loff_t *offset***, loff_t** *len***);**

These architecture-specific details are generally hidden from applications by the glibc **posix_fadvise**() wrapper function, which invokes the appropriate architecture-specific system call.

BUGS

In kernels before 2.6.6, if *len* was specified as 0, then this was interpreted literally as "zero bytes", rather than as meaning "all bytes through to the end of the file".

SEE ALSO

readahead(2), sync_file_range(2), posix_fallocate(3), **posix_madvise(3)**

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/.