NAME

posix_fallocate - allocate file space

SYNOPSIS

#include <fcntl.h>

int posix_fallocate(int *fd***, off_t** *offset***, off_t** *len***);**

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

posix_fallocate():
    _POSIX_C_SOURCE >= 200112L

DESCRIPTION

The function **posix_fallocate**() ensures that disk space is allocated for the file referred to by the file descriptor *fd* for the bytes in the range starting at *offset* and continuing for *len* bytes. After a successful call to **posix_fallocate**(), subsequent writes to bytes in the specified range are guaranteed not to fail because of lack of disk space.

If the size of the file is less than *offset*+*len*, then the file is increased to this size; otherwise the file size is left unchanged.

RETURN VALUE

posix_fallocate() returns zero on success, or an error number on failure. Note that*errno* is not set.

ERRORS

EBADF
fd is not a valid file descriptor, or is not opened for writing.

EFBIG
offset+*len* exceeds the maximum file size.

EINTR
A signal was caught during execution.

EINVAL
offset was less than 0, or *len* was less than or equal to 0, or the underlying filesystem does not support the operation.

ENODEV
fd does not refer to a regular file.

ENOSPC
There is not enough space left on the device containing the file referred to by *fd*.

ESPIPE
fd refers to a pipe.

VERSIONS

posix_fallocate() is available since glibc 2.1.94.

ATTRIBUTES

For an explanation of the terms used in this section, see attributes(7).

Interface	Attribute	Value
posix_fallocate()	Thread safety	MT-Safe (but see NOTES)

CONFORMING TO

POSIX.1-2001.

POSIX.1-2008 says that an implementation *shall* give the **EINVAL** error if *len* was 0, or *offset* was less than 0. POSIX.1-2001 says that an implementation *shall* give the **EINVAL** error if *len* is less than 0, or *offset* was less than 0, and *may* give the error if *len* equals zero.

NOTES

In the glibc implementation, **posix_fallocate**() is implemented using the fallocate(2) system call, which is MT-safe. If the underlying filesystem does not support fallocate(2), then the operation is emulated with the following caveats:

* The emulation is inefficient.
* There is a race condition where concurrent writes from another thread or process could be overwritten with null bytes.
* There is a race condition where concurrent file size increases by another thread or process could result in a file whose size is smaller than expected.
* If *fd* has been opened with the **O_APPEND** or **O_WRONLY** flags, the function will fail with the error **EBADF.**

In general, the emulation is not MT-safe. On Linux, applications may use fallocate(2) if they cannot tolerate the emulation caveats. In general, this is only recommended if the application plans to terminate the operation if **EOPNOTSUPP** is returned, otherwise the application itself will need to implement a fallback with all the same problems as the emulation provided by glibc.

SEE ALSO

fallocate(1), fallocate(2), lseek(2), posix_fadvise(2)

COLOPHON

This page is part of release 4.10 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 https://www.kernel.org/doc/man-pages/.