Rocky Enterprise Linux 9.2 Manual Pages on command 'fallocate.2'

C:\>man fallocate.2

FALLOCATE(2) Linux Programmer's Manual FALLOCATE(2)

NAME

fallocate - manipulate file space

SYNOPSIS

#define _GNU_SOURCE /* See feature_test_macros(7) */

#include <fcntl.h>

int fallocate(int fd, int mode, off_t offset, off_t len);

DESCRIPTION

This is a nonportable, Linux-specific system call. For the portable, POSIX.1-spec?

ified method of ensuring that space is allocated for a file, see posix_fallo?

cate(3).

fallocate() allows the caller to directly manipulate the allocated disk space for

the file referred to by fd for the byte range starting at offset and continuing for

len bytes.

The mode argument determines the operation to be performed on the given range. De?

tails of the supported operations are given in the subsections below.

Allocating disk space

The default operation (i.e., mode is zero) of fallocate() allocates the disk space

within the range specified by offset and len. The file size (as reported by

stat(2)) will be changed if offset+len is greater than the file size. Any subre?

gion within the range specified by offset and len that did not contain data before

the call will be initialized to zero. This default behavior closely resembles the

behavior of the posix_fallocate(3) library function, and is intended as a method of optimally implementing that function.

After a successful call, subsequent writes into the range specified by offset and len are guaranteed not to fail because of lack of disk space.

If the FALLOC_FL_KEEP_SIZE flag is specified in mode, the behavior of the call is similar, but the file size will not be changed even if offset+len is greater than the file size. Preallocating zeroed blocks beyond the end of the file in this man? ner is useful for optimizing append workloads.

If the FALLOC_FL_UNSHARE flag is specified in mode, shared file data extents will be made private to the file to guarantee that a subsequent write will not fail due to lack of space. Typically, this will be done by performing a copy-on-write oper? ation on all shared data in the file. This flag may not be supported by all filesystems.

Because allocation is done in block size chunks, fallocate() may allocate a larger range of disk space than was specified.

Deallocating file space

Specifying the FALLOC_FL_PUNCH_HOLE flag (available since Linux 2.6.38) in mode de? allocates space (i.e., creates a hole) in the byte range starting at offset and continuing for len bytes. Within the specified range, partial filesystem blocks are zeroed, and whole filesystem blocks are removed from the file. After a suc? cessful call, subsequent reads from this range will return zeros.

The FALLOC_FL_PUNCH_HOLE flag must be ORed with FALLOC_FL_KEEP_SIZE in mode; in other words, even when punching off the end of the file, the file size (as reported by stat(2)) does not change.

Not all filesystems support FALLOC_FL_PUNCH_HOLE; if a filesystem doesn't support the operation, an error is returned. The operation is supported on at least the following filesystems:

* XFS (since Linux 2.6.38)
* ext4 (since Linux 3.0)
* Btrfs (since Linux 3.7)
* tmpfs(5) (since Linux 3.5)
* gfs2(5) (since Linux 4.16)

Collapsing file space

Specifying the FALLOC_FL_COLLAPSE_RANGE flag (available since Linux 3.15) in mode removes a byte range from a file, without leaving a hole. The byte range to be collapsed starts at offset and continues for len bytes. At the completion of the operation, the contents of the file starting at the location offset+len will be ap? pended at the location offset, and the file will be len bytes smaller.

A filesystem may place limitations on the granularity of the operation, in order to ensure efficient implementation. Typically, offset and len must be a multiple of the filesystem logical block size, which varies according to the filesystem type and configuration. If a filesystem has such a requirement, fallocate() fails with the error EINVAL if this requirement is violated.

If the region specified by offset plus len reaches or passes the end of file, an error is returned; instead, use ftruncate(2) to truncate a file.

No other flags may be specified in mode in conjunction with FALLOC_FL_COL? LAPSE_RANGE.

As at Linux 3.15, FALLOC_FL_COLLAPSE_RANGE is supported by ext4 (only for extent-based files) and XFS.

Zeroing file space

Specifying the FALLOC_FL_ZERO_RANGE flag (available since Linux 3.15) in mode zeros space in the byte range starting at offset and continuing for len bytes. Within the specified range, blocks are preallocated for the regions that span the holes in the file. After a successful call, subsequent reads from this range will return zeros.

Zeroing is done within the filesystem preferably by converting the range into un? written extents. This approach means that the specified range will not be physi? cally zeroed out on the device (except for partial blocks at the either end of the range), and I/O is (otherwise) required only to update metadata.

If the FALLOC_FL_KEEP_SIZE flag is additionally specified in mode, the behavior of the call is similar, but the file size will not be changed even if offset+len is greater than the file size. This behavior is the same as when preallocating space with FALLOC_FL_KEEP_SIZE specified.

Not all filesystems support FALLOC_FL_ZERO_RANGE; if a filesystem doesn't support the operation, an error is returned. The operation is supported on at least the following filesystems:

* XFS (since Linux 3.15)

* ext4, for extent-based files (since Linux 3.15)

* SMB3 (since Linux 3.17)

* Btrfs (since Linux 4.16)

Increasing file space

Specifying the FALLOC_FL_INSERT_RANGE flag (available since Linux 4.1) in mode in? creases the file space by inserting a hole within the file size without overwriting any existing data. The hole will start at offset and continue for len bytes. When inserting the hole inside file, the contents of the file starting at offset will be shifted upward (i.e., to a higher file offset) by len bytes. Inserting a hole in? side a file increases the file size by len bytes.

This mode has the same limitations as FALLOC_FL_COLLAPSE_RANGE regarding the granu? larity of the operation. If the granularity requirements are not met, fallocate() fails with the error EINVAL. If the offset is equal to or greater than the end of file, an error is returned. For such operations (i.e., inserting a hole at the end of file), ftruncate(2) should be used.

No other flags may be specified in mode in conjunction with FALLOC_FL_INSERT_RANGE.

FALLOC_FL_INSERT_RANGE requires filesystem support. Filesystems that support this operation include XFS (since Linux 4.1) and ext4 (since Linux 4.2).

RETURN VALUE

On success, fallocate() returns zero. On error, -1 is returned and errno is set to indicate the error.

ERRORS

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

EFBIG offset+len exceeds the maximum file size.

EFBIG mode is FALLOC_FL_INSERT_RANGE, and the current file size+len exceeds the maximum file size.

EINTR A signal was caught during execution; see signal(7).

EINVAL offset was less than 0, or len was less than or equal to 0.

EINVAL mode is FALLOC_FL_COLLAPSE_RANGE and the range specified by offset plus len reaches or passes the end of the file.

EINVAL mode is FALLOC_FL_INSERT_RANGE and the range specified by offset reaches or passes the end of the file.

EINVAL mode is FALLOC_FL_COLLAPSE_RANGE or FALLOC_FL_INSERT_RANGE, but either off?
set or len is not a multiple of the filesystem block size.

EINVAL mode contains one of FALLOC_FL_COLLAPSE_RANGE or FALLOC_FL_INSERT_RANGE and also other flags; no other flags are permitted with FALLOC_FL_COLLAPSE_RANGE or FALLOC_FL_INSERT_RANGE.

EINVAL mode is FALLOC_FL_COLLAPSE_RANGE or FALLOC_FL_ZERO_RANGE or FALLOC_FL_IN?
SERT_RANGE, but the file referred to by fd is not a regular file.

EIO An I/O error occurred while reading from or writing to a filesystem.

ENODEV fd does not refer to a regular file or a directory. (If fd is a pipe or FIFO, a different error results.)

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

ENOSYS This kernel does not implement fallocate().

EOPNOTSUPP
The filesystem containing the file referred to by fd does not support this operation; or the mode is not supported by the filesystem containing the file referred to by fd.

EPERM The file referred to by fd is marked immutable (see chattr(1)).

EPERM mode specifies FALLOC_FL_PUNCH_HOLE or FALLOC_FL_COLLAPSE_RANGE or FAL?
LOC_FL_INSERT_RANGE and the file referred to by fd is marked append-only (see chattr(1)).

EPERM The operation was prevented by a file seal; see fcntl(2).

ESPIPE fd refers to a pipe or FIFO.

ETXTBSY
mode specifies FALLOC_FL_COLLAPSE_RANGE or FALLOC_FL_INSERT_RANGE, but the file referred to by fd is currently being executed.

VERSIONS

fallocate() is available on Linux since kernel 2.6.23. Support is provided by glibc since version 2.10. The FALLOC_FL_* flags are defined in glibc headers only since version 2.18.

CONFORMING TO

fallocate() is Linux-specific.

SEE ALSO

fallocate(1), ftruncate(2), posix_fadvise(3), posix_fallocate(3)

COLOPHON

This page is part of release 5.05 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/.