NAME

fallocate − manipulate file space

SYNOPSIS

```
#define _GNU_SOURCE
#include <fcntl.h>

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

DESCRIPTION

This is a non-portable, Linux-specific system call. For the portable, POSIX.1-specified method of ensuring that space is allocated for a file, see **posix_fallocate**().

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

Allocating disk space

The default operation (i.e., *mode* is zero) of **fallocate**() allocates and initializes to zero 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. 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 manner is useful for optimizing append workloads.

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* deallocates 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 successful call, subsequent reads from this range will return zeroes.

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.

RETURN VALUE

fallocate() returns zero on success, and -1 on failure.

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.

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

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
The file system containing the file referred to by *fd* does not support this operation.

EOPNOTSUPP
The *mode* is not supported by the file system containing the file referred to by *fd*.

VERSIONS

fallocate() is available on Linux since kernel 2.6.23. Support is provided by glibc since version 2.10.

CONFORMING TO

fallocate() is Linux-specific.

SEE ALSO

ftruncate(2), **posix_fadvise**(3), **posix_fallocate**(3)

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/.