



**Full credit is given to the above companies including the Operating System (OS) that this PDF file was generated!**

### ***Rocky Enterprise Linux 9.2 Manual Pages on command 'mkfifo.3'***

**\$ man mkfifo.3**

MKFIFO(3)                      Linux Programmer's Manual                      MKFIFO(3)

NAME

mkfifo, mkfifoat - make a FIFO special file (a named pipe)

SYNOPSIS

```
#include <sys/types.h>
#include <sys/stat.h>
int mkfifo(const char *pathname, mode_t mode);
#include <fcntl.h>        /* Definition of AT_* constants */
#include <sys/stat.h>
int mkfifoat(int dirfd, const char *pathname, mode_t mode);
```

Feature Test Macro Requirements for glibc (see feature\_test\_macros(7)):

mkfifoat():

Since glibc 2.10:

```
_POSIX_C_SOURCE >= 200809L
```

Before glibc 2.10:

```
_ATFILE_SOURCE
```

DESCRIPTION

mkfifo() makes a FIFO special file with name pathname. mode specifies the FIFO's permissions. It is modified by the process's umask in the usual way: the permissions of the created file are (mode & ~umask).

A FIFO special file is similar to a pipe, except that it is created in a different way.

Instead of being an anonymous communications channel, a FIFO special file is entered into the filesystem by calling mkfifo().

Once you have created a FIFO special file in this way, any process can open it for reading or writing, in the same way as an ordinary file. However, it has to be open at both ends simultaneously before you can proceed to do any input or output operations on it. Opening a FIFO for reading normally blocks until some other process opens the same FIFO for writing, and vice versa. See `fifo(7)` for nonblocking handling of FIFO special files.

#### `mkfifoat()`

The `mkfifoat()` function operates in exactly the same way as `mkfifo()`, except for the differences described here.

If the pathname given in `pathname` is relative, then it is interpreted relative to the directory referred to by the file descriptor `dirfd` (rather than relative to the current working directory of the calling process, as is done by `mkfifo()` for a relative pathname).

If `pathname` is relative and `dirfd` is the special value `AT_FDCWD`, then `pathname` is interpreted relative to the current working directory of the calling process (like `mkfifo()`).

If `pathname` is absolute, then `dirfd` is ignored.

#### RETURN VALUE

On success `mkfifo()` and `mkfifoat()` return 0. In the case of an error, -1 is returned (in which case, `errno` is set appropriately).

#### ERRORS

**EACCES** One of the directories in `pathname` did not allow search (execute) permission.

**EDQUOT** The user's quota of disk blocks or inodes on the filesystem has been exhausted.

**EEXIST** `pathname` already exists. This includes the case where `pathname` is a symbolic link, dangling or not.

#### **ENAMETOOLONG**

Either the total length of `pathname` is greater than `PATH_MAX`, or an individual filename component has a length greater than `NAME_MAX`. In the GNU system, there is no imposed limit on overall filename length, but some filesystems may place limits on the length of a component.

**ENOENT** A directory component in `pathname` does not exist or is a dangling symbolic link.

**ENOSPC** The directory or filesystem has no room for the new file.

#### **ENOTDIR**

A component used as a directory in `pathname` is not, in fact, a directory.

**EROFS** `pathname` refers to a read-only filesystem.

The following additional errors can occur for `mkfifoat()`:

EBADF dirfd is not a valid file descriptor.

### ENOTDIR

pathname is a relative path and dirfd is a file descriptor referring to a file other than a directory.

### VERSIONS

mkfifoat() was added to glibc in version 2.4. It is implemented using mknodat(2), available on Linux since kernel 2.6.16.

### ATTRIBUTES

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

??

?Interface ? Attribute ? Value ?

??

?mkfifo(), mkfifoat() ? Thread safety ? MT-Safe ?

??

### CONFORMING TO

mkfifo(): POSIX.1-2001, POSIX.1-2008.

mkfifoat(): POSIX.1-2008.

### SEE ALSO

mkfifo(1), close(2), open(2), read(2), stat(2), umask(2), write(2), fifo(7)

### COLOPHON

This page is part of release 5.10 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/>.