



## ***Rocky Enterprise Linux 9.2 Manual Pages on command 'mkdirat.2'***

**C:\>man mkdirat.2**

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

### NAME

mkdir, mkdirat - create a directory

### SYNOPSIS

```
#include <sys/stat.h>

#include <sys/types.h>

int mkdir(const char *pathname, mode_t mode);

#include <fcntl.h>            /* Definition of AT_* constants */

#include <sys/stat.h>

int mkdirat(int dirfd, const char *pathname, mode_t mode);
```

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

mkdirat():

Since glibc 2.10:

```
  _POSIX_C_SOURCE >= 200809L
```

Before glibc 2.10:

```
  _ATFILE_SOURCE
```

### DESCRIPTION

mkdir() attempts to create a directory named `pathname`.

The argument `mode` specifies the mode for the new directory (see `inode(7)`). It is modified by the process's `umask` in the usual way: in the absence of a default `ACL`, the `mode` of the created directory is `(mode & ~umask & 0777)`. Whether other mode bits are honored for the created directory depends on the operating system. For

Linux, see NOTES below.

The newly created directory will be owned by the effective user ID of the process.

If the directory containing the file has the set-group-ID bit set, or if the filesystem is mounted with BSD group semantics (mount -o bsdgroups or, synonymously mount -o grp), the new directory will inherit the group ownership from its parent; otherwise it will be owned by the effective group ID of the process.

If the parent directory has the set-group-ID bit set, then so will the newly created directory.

#### mkdirat()

The mkdirat() system call operates in exactly the same way as mkdir(), 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 mkdir() 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 mkdir()).

If pathname is absolute, then dirfd is ignored.

See openat(2) for an explanation of the need for mkdirat().

#### RETURN VALUE

mkdir() and mkdirat() return zero on success, or -1 if an error occurred (in which case, errno is set appropriately).

#### ERRORS

**EACCESS** The parent directory does not allow write permission to the process, or one of the directories in pathname did not allow search permission. (See also path\_resolution(7).)

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

**EEXIST** pathname already exists (not necessarily as a directory). This includes the case where pathname is a symbolic link, dangling or not.

**EFAULT** pathname points outside your accessible address space.

**EINVAL** The final component ("basename") of the new directory's pathname is invalid

(e.g., it contains characters not permitted by the underlying filesystem).

**ELOOP** Too many symbolic links were encountered in resolving pathname.

**EMLINK** The number of links to the parent directory would exceed `LINK_MAX`.

**ENAMETOOLONG**

pathname was too long.

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

**ENOMEM** Insufficient kernel memory was available.

**ENOSPC** The device containing pathname has no room for the new directory.

**ENOSPC** The new directory cannot be created because the user's disk quota is exhausted.

**ENOTDIR**

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

**EPERM** The filesystem containing pathname does not support the creation of directories.

**EROFS** pathname refers to a file on a read-only filesystem.

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

**EBADF** `dirfd` is not a valid file descriptor.

**ENOTDIR**

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

**VERSIONS**

`mkdirat()` was added to Linux in kernel 2.6.16; library support was added to glibc in version 2.4.

**CONFORMING TO**

`mkdir()`: SVr4, BSD, POSIX.1-2001, POSIX.1-2008.

`mkdirat()`: POSIX.1-2008.

**NOTES**

Under Linux, apart from the permission bits, the `S_ISVTX` mode bit is also honored.

There are many infelicities in the protocol underlying NFS. Some of these affect `mkdir()`.

**Glibc notes**

On older kernels where `mkdirat()` is unavailable, the glibc wrapper function falls

back to the use of `mkdir()`. When `pathname` is a relative pathname, `glibc` constructs a pathname based on the symbolic link in `/proc/self/fd` that corresponds to the `dirfd` argument.

#### SEE ALSO

`mkdir(1)`, `chmod(2)`, `chown(2)`, `mknod(2)`, `mount(2)`, `rmdir(2)`, `stat(2)`, `umask(2)`, `un?`  
`link(2)`, `acl(5)` `path_resolution(7)`

#### 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/>.

Linux

2017-09-15

MKDIR(2)