## NAME

**mkdir**, **mkdirat** - make a directory file

## LIBRARY

Standard C Library (libc, -lc)

## SYNOPSIS

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

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

*int*
**mkdirat**(*int fd*, *const char *path*, *mode_t mode*);

## DESCRIPTION

The directory *path* is created with the access permissions specified by *mode* and restricted by the umask(2) of the calling process.

The directory's owner ID is set to the process's effective user ID. The directory's group ID is set to that of the parent directory in which it is created.

The **mkdirat**() system call is equivalent to **mkdir**() except in the case where *path* specifies a relative path. In this case the newly created directory is created relative to the directory associated with the file descriptor *fd* instead of the current working directory. If **mkdirat**() is passed the special value AT_FDCWD in the *fd* parameter, the current working directory is used and the behavior is identical to a call to **mkdir**().

## RETURN VALUES

The **mkdir**() function returns the value 0 if successful; otherwise the value -1 is returned and the global variable *errno* is set to indicate the error.

## ERRORS

The **mkdir**() system call will fail and no directory will be created if:

[ENOTDIR] A component of the path prefix is not a directory.

[ENAMETOOLONG]
A component of a pathname exceeded 255 characters, or an entire path name exceeded 1023 characters.

[ENOENT] A component of the path prefix does not exist.

[EACCES] Search permission is denied for a component of the path prefix, or write permission is denied on the parent directory of the directory to be created.

[ELOOP] Too many symbolic links were encountered in translating the pathname.

[EPERM] The parent directory of the directory to be created has its immutable flag set, see the chflags(2) manual page for more information.

[EROFS] The named directory would reside on a read-only file system.

[EMLINK] The new directory cannot be created because the parent directory contains too many subdirectories.

[EEXIST] The named file exists.

[ENOSPC] The new directory cannot be created because there is no space left on the file system that will contain the directory.

[ENOSPC] There are no free inodes on the file system on which the directory is being created.

[EDQUOT] The new directory cannot be created because the user’s quota of disk blocks on the file system that will contain the directory has been exhausted.

[EDQUOT] The user’s quota of inodes on the file system on which the directory is being created has been exhausted.

[EIO] An I/O error occurred while making the directory entry or allocating the inode.

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

[EINTEGRITY] Corrupted data was detected while reading from the file system.

[EFAULT] The *path* argument points outside the process’s allocated address space.

In addition to the errors returned by the **mkdir**(), the **mkdirat**() may fail if:

[EBADF] The *path* argument does not specify an absolute path and the *fd* argument is

neither AT_FDCWD nor a valid file descriptor open for searching.

[ENOTDIR] The *path* argument is not an absolute path and *fd* is neither AT_FDCWD nor a file descriptor associated with a directory.

**SEE ALSO**

chflags(2), chmod(2), stat(2), umask(2)

**STANDARDS**

The **mkdir**() system call is expected to conform to IEEE Std 1003.1-1990 ("POSIX.1"). The **mkdirat**() system call follows The Open Group Extended API Set 2 specification.

**HISTORY**

The **mkdirat**() system call appeared in FreeBSD 8.0. The **mkdir**() system call appeared in Version 1 AT&T UNIX.