NAME

mknod − create a special or ordinary file

SYNOPSIS

#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <unistd.h>

int mknod(const char * *pathname***, mode_t** *mode***, dev_t** *dev***);**

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

mknod(): _BSD_SOURCE || _SVID_SOURCE || _XOPEN_SOURCE >= 500

DESCRIPTION

The system call **mknod**() creates a file system node (file, device special file or named pipe) named *pathname*, with attributes specified by *mode* and *dev*.

The *mode* argument specifies both the permissions to use and the type of node to be created. It should be a combination (using bitwise OR) of one of the file types listed below and the permissions for the new node.

The permissions are modified by the process's *umask* in the usual way: the permissions of the created node are *(mode & ˜umask)*.

The file type must be one of **S_IFREG**, **S_IFCHR**, **S_IFBLK**, **S_IFIFO** or **S_IFSOCK** to specify a regular file (which will be created empty), character special file, block special file, FIFO (named pipe), or Unix domain socket, respectively. (Zero file type is equivalent to type **S_IFREG**.)

If the file type is **S_IFCHR** or **S_IFBLK** then *dev* specifies the major and minor numbers of the newly created device special file (**makedev**(3) may be useful to build the value for *dev*); otherwise it is ignored.

If *pathname* already exists, or is a symbolic link, this call fails with an **EEXIST** error.

The newly created node will be owned by the effective user ID of the process. If the directory containing the node has the set-group-ID bit set, or if the file system is mounted with BSD group semantics, the new node will inherit the group ownership from its parent directory; otherwise it will be owned by the effective group ID of the process.

RETURN VALUE

mknod() returns zero on success, or −1 if an error occurred (in which case, *errno* is set appropriately).

ERRORS

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

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

EFAULT
pathname points outside your accessible address space.

EINVAL
mode requested creation of something other than a regular file, device special file, FIFO or socket.

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

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 node.

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

EPERM
mode requested creation of something other than a regular file, FIFO (named pipe), or Unix domain socket, and the caller is not privileged (Linux: does not have the **CAP_MKNOD** capability); also returned if the file system containing *pathname* does not support the type of node requested.

EROFS
pathname refers to a file on a read-only file system.

CONFORMING TO
SVr4, 4.4BSD, POSIX.1-2001 (but see below).

NOTES
POSIX.1-2001 says: "The only portable use of **mknod**() is to create a FIFO-special file. If *mode* is not **S_IFIFO** or *dev* is not 0, the behavior of **mknod**() is unspecified." However, nowadays one should never use **mknod**() for this purpose; one should use **mkfifo**(3), a function especially defined for this purpose.

Under Linux, this call cannot be used to create directories. One should make directories with **mkdir**(2).

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

SEE ALSO
chmod(2), **chown**(2), **fcntl**(2), **mkdir**(2), **mknodat**(2), **mount**(2), **socket**(2), **stat**(2), **umask**(2), **unlink**(2), **makedev**(3), **mkfifo**(3), **path_resolution**(7)

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/.