NAME

mknod, **mknodat** - make a special file node

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

#include <sys/stat.h>

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

int
mknodat(*int fd, const char *path, mode_t mode, dev_t dev*);

DESCRIPTION

The file system node *path* is created with the file type and access permissions specified in *mode*. The access permissions are modified by the process's umask value.

If *mode* indicates a block or character special file, *dev* is a configuration dependent specification denoting a particular device on the system. Otherwise, *dev* is ignored.

The **mknod**() system call requires super-user privileges.

The **mknodat**() system call is equivalent to **mknod**() except in the case where *path* specifies a relative path. In this case the newly created device node is created relative to the directory associated with the file descriptor *fd* instead of the current working directory. If **mknodat**() 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 **mknod**().

RETURN VALUES

The **mknod**() 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 **mknod**() system call will fail and the file will be not 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.

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

[EPERM] The process’s effective user ID is not super-user.

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

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

[ENOSPC] The directory in which the entry for the new node is being placed cannot be extended because there is no space left on the file system containing the directory.

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

[EDQUOT] The directory in which the entry for the new node is being placed cannot be extended because the user’s quota of disk blocks on the file system containing the directory has been exhausted.

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

[EROFS] The named file resides on a read-only file system.

[EEXIST] The named file exists.

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

[EINVAL] Creating anything else than a block or character special file (or a *whiteout*) is not supported.

In addition to the errors returned by the **mknod**(), the **mknodat**() 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

chmod(2), mkfifo(2), stat(2), umask(2)

STANDARDS

The **mknodat**() system call follows The Open Group Extended API Set 2 specification.

HISTORY

The **mknod**() function appeared in Version 4 AT&T UNIX. The **mknodat**() system call appeared in FreeBSD 8.0.