mknod() — Make a Device file or Directory or File

Standards

Standards / Extensions	C or C++	Dependencies
XPG4.2 Single UNIX Specification, Version 3 z/OS UNIX	both	

Format

_OPEN_SYS

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

int mknod(const char *path, mode_t mode, rdev_t dev_identifier);
```

_XOPEN_SOURCE_EXTENDED 1

```
#define _XOPEN_SOURCE_EXTENDED 1
#include <sys/stat.h>

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

General Description

Creates a new directory, regular file, character special file, or FIFO special file (named pipe), with the pathname specified in the *path* argument.

The first byte of the *mode* argument determines the file type of the file:

S_IFCHR
: Character special file

S_IFFIFO
: FIFO special file

S_IFREG
: Regular file

S_IFDIR
: Directory file

The file permission bits of the new file are initialized with the remaining bits in *mode* and changed by the file creation mask of the process. For more information on these symbols, refer to chmod() — Change the Mode of a File or Directory.

dev_identifier applies only to a character special file. It is ignored for the other file types. *dev_identifier*contains a value representing the device major and device minor numbers. The major number is contained in the high-order 16 bits; it identifies a device driver supporting a class of devices, such as interactive terminals. The minor number is contained in the low-order 16 bits of *dev_identifier*; it identifies a specific device within the class referred to by the device major number. With z/OS UNIX services, the device major numbers are:

1
: Master pseudoterminal

2
: Slave pseudoterminal

3
: /dev/tty

4
: /dev/null

5
: /dev/fdn

6
: Sockets

7
: OCSRTY

8
OCSADMIN
9
"/dev/console"

Device major numbers 1,2 and 7: The device minor numbers range between 0 and one less than the maximum number of pseudoterminal pairs defined by the installation.

Device major numbers 3,4,6,8 and 9: The device minor number is ignored.

Device major number 5: The device minor number value represents the file descriptor to be referred to. For example, device minor 0 refers to file descriptor 0.

When it completes successfully, mknod() marks for update the following fields of the file: `st_atime`,`st_ctime`, and `st_mtime`. It also marks for update the `st_ctime` and `st_mtime` fields of the directory that contains the new file.

Returned Value

If successful, mknod() returns 0.

If unsuccessful, mknod() returns -1 and sets errno to one of the following values:

Error Code
Description
EACCES
Search permission is denied on a component of *path*, or write permission is denied on the parent directory of the file to be created.
EEXIST
A file by that name already exists.
ELOOP
A loop exists in symbolic links. This error is issued if more than POSIX_SYMLOOP (defined in the limits.h header file) symbolic links are detected in the resolution of *path*.
EMLINK
The link count of the parent directory has already reached the maximum defined for the system.
ENAMETOOLONG
pathname is longer than **PATH_MAX** characters, or some component of *pathname* is longer than**NAME_MAX** characters while _POSIX_NO_TRUNC is in effect. For symbolic links, the length of the pathname string substituted for a symbolic link exceeds **PATH_MAX**. The **PATH_MAX** and**NAME_MAX** values can be determined through pathconf().
ENOENT
A component of *path* was not found, or no *path* was specified.
ENOSPC
The file system does not have enough space to contain a new directory, or the parent directory cannot be extended.
ENOTDIR
A component of *path* is not a directory
EPERM
Added for XPG4.2: The invoking process does not have appropriate privileges and the file type is not FIFO-special.
EROFS
The file named in *path* cannot be created, because it would reside on a read-only file system.