mkfifo() — Make a FIFO Special File

Standards

Standards / Extensions	C or C++	Dependencies
POSIX.1 XPG4 XPG4.2 Single UNIX Specification, Version 3	both	

Format

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

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

General Description

Sets the access, change, and modification times for the new file. It also sets the change and modification times for the directory that contains the new file.

mkfifo() creates a new FIFO special file, *pathname*. The file permission bits in *mode* are changed by the file creation mask of the process, and then used to set the file permission bits of the FIFO file being created. If *pathname* contains a symbolic link, mkfifo() fails. For more information on the file creation mask, see umask() — Set and Retrieve File Creation Mask; for information about the file permission bits, see chmod() — Change the Mode of a File or Directory.

The owner ID of the FIFO file is set to the effective user ID of the process. The group ID of the FIFO file is set to the group ID of the owning directory. *pathname* cannot end in a symbolic link.

Returned Value

If successful, mkfifo() returns 0.

If unsuccessful, mkfifo() does not create a FIFO file, returns -1, and sets errno to one of the following values:

Error Code
Description

EACCES
: The process does not have search permission on some component of *pathname*, or does not have write permission on the parent directory of the file to be created.

EEXIST
: Either the named file refers to a symbolic link, or there is already a file or directory with the given *pathname*.

EINTR
: A signal is received while this open is blocked waiting for an open() for read (if O_WRONLY was specified) or for an open() for write (if O_RDONLY was specified).

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 *pathname*.

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 using pathconf().

ENOENT
: Some component of *pathname* does not exist, or *pathname* is an empty string.

ENOSPC
: The file system does not have enough space to contain a new file, or the parent directory cannot be extended.

ENOTDIR
: A component of the *pathname* prefix is not a directory.

EROFS
: The parent directory of the FIFO file is on a read-only file system.