NAME

mkfifo, **mkfifoat** - make a fifo file

LIBRARY

Standard C Library (libc, -lc)

SYNOPSIS

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

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

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

DESCRIPTION

The **mkfifo**() system call creates a new fifo file with name *path*. The access permissions are specified by *mode* and restricted by the umask(2) of the calling process.

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

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

RETURN VALUES

The **mkfifo**() 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 **mkfifo**() system call will fail and no fifo will be created if:

[ENOTSUP] The kernel has not been configured to support fifo's.

[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] A component of the path prefix denies search permission, or write permission is denied on the parent directory of the fifo to be created.

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

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

[EEXIST] The named file exists.

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

[ENOSPC] The directory in which the entry for the new fifo 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 fifo is being created.

[EDQUOT] The directory in which the entry for the new fifo 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 fifo is being created has been exhausted.

[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.

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

In addition to the errors returned by the **mkfifo**(), the **mkfifoat**() 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), mknod(2), stat(2), umask(2)

STANDARDS

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

HISTORY

The **mkfifoat**() system call appeared in FreeBSD 8.0.