Full credit is given to the above companies including the OS that this PDF file was generated!

Red Hat Enterprise Linux Release 9.2 Manual Pages on 'mkdir.3p' command

$ man mkdir.3p

MKDIR(3P) POSIX Programmer's Manual MKDIR(3P)

PROLOG

This manual page is part of the POSIX Programmer's Manual. The Linux implementation of this interface may differ (consult the corresponding Linux manual page for details of Linux behavior), or the interface may not be implemented on Linux.

NAME

mkdir, mkdirat ? make a directory

SYNOPSIS

```
#include <sys/stat.h>
int mkdir(const char *path, mode_t mode);
#include <fcntl.h>
int mkdirat(int fd, const char *path, mode_t mode);
```

DESCRIPTION

The mkdir() function shall create a new directory with name path. The file permission bits of the new directory shall be initialized from mode. These file permission bits of the mode argument shall be modi? fied by the process' file creation mask.

When bits in mode other than the file permission bits are set, the meaning of these additional bits is implementation-defined.

The directory's user ID shall be set to the process' effective user ID.

The directory's group ID shall be set to the group ID of the parent di? rectory or to the effective group ID of the process. Implementations

shall provide a way to initialize the directory's group ID to the group ID of the parent directory. Implementations may, but need not, provide an implementation-defined way to initialize the directory's group ID to the effective group ID of the calling process.

The newly created directory shall be an empty directory.

If path names a symbolic link, mkdir() shall fail and set errno to [EEXIST].

Upon successful completion, mkdir() shall mark for update the last data access, last data modification, and last file status change timestamps of the directory. Also, the last data modification and last file status change timestamps of the directory that contains the new entry shall be marked for update.

The mkdirat() function shall be equivalent to the mkdir() function ex? cept in the case where path specifies a relative path. In this case the newly created directory is created relative to the directory associated with the file descriptor fd instead of the current working directory.

If the access mode of the open file description associated with the file descriptor is not O_SEARCH, the function shall check whether di? rectory searches are permitted using the current permissions of the di? rectory underlying the file descriptor. If the access mode is O_SEARCH, the function shall not perform the check.

If mkdirat() is passed the special value AT_FDCWD in the fd parameter, the current working directory shall be used and the behavior shall be identical to a call to mkdir().

RETURN VALUE

Upon successful completion, these functions shall return 0. Otherwise, these functions shall return -1 and set errno to indicate the error. If -1 is returned, no directory shall be created.

ERRORS

These functions shall fail if:

EACCES Search permission is denied on a component of the path prefix, or write permission is denied on the parent directory of the di? rectory to be created.

EEXIST The named file exists.

ELOOP A loop exists in symbolic links encountered during resolution of the path argument.

EMLINK The link count of the parent directory would exceed {LINK_MAX}.

ENAMETOOLONG

The length of a component of a pathname is longer than {NAME_MAX}.

ENOENT A component of the path prefix specified by path does not name an existing directory or path is an empty string.

ENOSPC The file system does not contain enough space to hold the con? tents of the new directory or to extend the parent directory of the new directory.

ENOTDIR

A component of the path prefix names an existing file that is neither a directory nor a symbolic link to a directory.

EROFS The parent directory resides on a read-only file system.

In addition, the mkdirat() function shall fail if:

EACCES The access mode of the open file description associated with fd is not O_SEARCH and the permissions of the directory underlying fd do not permit directory searches.

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 reading or searching.

ENOTDIR

The path argument is not an absolute path and fd is a file de? scriptor associated with a non-directory file.

These functions may fail if:

ELOOP More than {SYMLOOP_MAX} symbolic links were encountered during resolution of the path argument.

ENAMETOOLONG

The length of a pathname exceeds {PATH_MAX}, or pathname resolu? tion of a symbolic link produced an intermediate result with a length that exceeds {PATH_MAX}.

The following sections are informative.

EXAMPLES

Creating a Directory

The following example shows how to create a directory named /home/cnd/mod1, with read/write/search permissions for owner and group, and with read/search permissions for others.

```
#include <sys/types.h>
#include <sys/stat.h>
int status;
...
status = mkdir("/home/cnd/mod1", S_IRWXU | S_IRWXG | S_IROTH | S_IXOTH);
```

APPLICATION USAGE

None.

RATIONALE

The mkdir() function originated in 4.2 BSD and was added to System V in Release 3.0.

4.3 BSD detects [ENAMETOOLONG].

The POSIX.1?1990 standard required that the group ID of a newly created directory be set to the group ID of its parent directory or to the ef? fective group ID of the creating process. FIPS 151?2 required that im? plementations provide a way to have the group ID be set to the group ID of the containing directory, but did not prohibit implementations also supporting a way to set the group ID to the effective group ID of the creating process. Conforming applications should not assume which group ID will be used. If it matters, an application can use chown() to set the group ID after the directory is created, or determine under what conditions the implementation will set the desired group ID.

The purpose of the mkdirat() function is to create a directory in di? rectories other than the current working directory without exposure to race conditions. Any part of the path of a file could be changed in parallel to the call to mkdir(), resulting in unspecified behavior. By opening a file descriptor for the target directory and using the mkdi? rat() function it can be guaranteed that the newly created directory is

located relative to the desired directory.

FUTURE DIRECTIONS

None.

SEE ALSO

chmod(), mkdtemp(), mknod(), umask()

The Base Definitions volume of POSIX.1?2017, <fcntl.h>, <sys_stat.h>, <sys_types.h>

COPYRIGHT