-- --

IEEE P1003.2 Draft 11.2 – September 1991

Copyright © 1991 by the
Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street
New York, NY 10017, USA
All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change. The publication, distribution, or copying of this draft, as well as all derivative works based on this draft, is expressly prohibited except as set forth below.

Permission is hereby granted for IEEE Standards Committee participants to reproduce this document for purposes of IEEE standardization activities only, and subject to the restrictions contained herein.

Permission is hereby also granted for member bodies and technical committees of ISO and IEC to reproduce this document for purposes of developing a national position, subject to the restrictions contained herein.

Permission is hereby also granted to the preceding entities to make limited copies of this document in an electronic form only for the stated activities.

The following restrictions apply to reproducing or transmitting the document in any form: 1) all copies or portions thereof must identify the document's IEEE project number and draft number, and must be accompanied by this entire notice in a prominent location; 2) no portion of this document may be redistributed in any modified or abridged form without the prior approval of the IEEE Standards Department.

Other entities seeking permission to reproduce this document, or any portion thereof, for standardization or other activities, must contact the IEEE Standards Department for the appropriate license.

Use of information contained in this unapproved draft is at your own risk.

IEEE Standards Department
Copyright and Permissions
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331, USA
+1 (908) 562-3800
+1 (908) 562-1571 [FAX]

4.42 mkfifo — Make FIFO special files

4.42.1 Synopsis

`mkfifo [−m` *mode* **]** *file* . . .

4.42.2 Description

The `mkfifo` utility shall create the FIFO special files specified by the operands, in the order specified.

For each *file* operand, the `mkfifo` utility shall perform actions equivalent to the POSIX.1 {8} *mkfifo*() function, called with the following arguments:

(1) The *file* operand is used as the *path* argument.

(2) The value of the bitwise inclusive OR of S_IRUSR, S_IWUSR, S_IRGRP, S_IWGRP, S_IROTH, and S_IWOTH is used as the *mode* argument. (If the `−m` option is specified, the *mode* option-argument overrides this default.)

4.42.3 Options

The `mkfifo` utility shall conform to the utility argument syntax guidelines described in 2.10.2.

The following option shall be supported by the implementation:

`−m` *mode* Set the file permission bits of the newly-created FIFO to the specified *mode* value. The *mode* option-argument shall be the same as the *mode* operand defined for the `chmod` utility (see 4.7). In the *symbolic_mode* strings, the *op* characters `+` and `-` shall be interpreted relative to an assumed initial mode of `a=rw`.

4.42.4 Operands

The following operand shall be supported by the implementation:

file A pathname of the FIFO special file to be created.

4.42.5 External Influences

4.42.5.1 Standard Input

None.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

4.42.5.2 Input Files

None.

4.42.5.3 Environment Variables

The following environment variables shall affect the execution of `mkfifo`:

LANG This variable shall determine the locale to use for the locale categories when both **LC_ALL** and the corresponding environment variable (beginning with **LC_**) do not specify a locale. See 2.6.

LC_ALL This variable shall determine the locale to be used to override any values for locale categories specified by the settings of **LANG** or any environment variables beginning with **LC_**.

LC_CTYPE This variable shall determine the locale for the interpretation of sequences of bytes of text data as characters (e.g., single- versus multibyte characters in arguments).

LC_MESSAGES This variable shall determine the language in which messages should be written.

4.42.5.4 Asynchronous Events

Default.

4.42.6 External Effects

4.42.6.1 Standard Output

None.

4.42.6.2 Standard Error

Used only for diagnostic messages.

4.42.6.3 Output Files

None.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

4.42.7 Extended Description

None.

4.42.8 Exit Status

The `mkfifo` utility shall exit with one of the following values:

0	All the specified FIFO special files were created successfully.
>0	An error occurred.

4.42.9 Consequences of Errors

Default.

4.42.10 Rationale. *(This subclause is not a part of P1003.2)*

Examples, Usage

None.

History of Decisions Made

This new utility was added to permit shell applications to create FIFO special files.

The `-m` option was added to control the file mode, for consistency with the similar functionality provided the `mkdir` utility.

Earlier drafts included a `-p` option similar to `mkdir`'s `-p` option that created intermediate directories leading up to the FIFO specified by the final component. This was removed because it is not commonly needed and is not common practice with similar utilities.

The functionality of `mkfifo` is described substantially through a reference to the *mkfifo* () function in POSIX. 1. For example, by default, the mode of the FIFO file is affected by the file mode creation mask in accordance with the specified behavior of POSIX .1 {8}*mkfifo* (). In this way, there is less duplication of effort required for describing details of the file creation.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.

4.43 mv — Move files

4.43.1 Synopsis

`mv [-fi]` *source_file target_file*

`mv [-fi]` *source_file ... target_dir*

4.43.2 Description

In the first synopsis form, the `mv` utility shall move the file named by the *source_file* operand to the *destination* specified by the *target_file*. This first synopsis form is assumed when the final operand does not name an existing directory.

In the second synopsis form, `mv` shall move each file named by a *source_file* operand to a *destination* file in the existing directory named by the *target_dir* operand. The *destination* path for each *source_file* shall be the concatenation of the target directory, a single slash character, and the last pathname component of the *source_file*.

If any operand specifies an existing file of a type not specified by POSIX.1 {8}, the behavior is implementation defined.

This second form is assumed when the final operand names an existing directory.

For each *source_file* the following steps shall be taken:

(1) If the destination path exists, the `-f` option is not specified, and either of the following conditions is true:

  (a) The permissions of the destination path do not permit writing and the standard input is a terminal.

  (b) The `-i` option is specified.

  the `mv` utility shall write a prompt to standard error and read a line from standard input. If the response is not affirmative, `mv` shall do nothing more with the current *source_file* and go on to any remaining *source_file* s.

(2) The `mv` utility shall perform actions equivalent to the POSIX.1 {8} *rename* () function, called with the following arguments:

  (a) The *source_file* operand is used as the *old* argument.

  (b) The destination path is used as the *new* argument.

  If this succeeds, `mv` shall do nothing more with the current *source_file* and go on to any remaining *source_file* s. If this fails for any reasons other than those described for the *errno* [EXDEV] in POSIX.1 {8}, `mv` shall write a diagnostic message to standard error, do nothing more with the current *source_file*, and go on to any remaining *source_file* s.

Copyright © 1991 IEEE. All rights reserved.
This is an unapproved IEEE Standards Draft, subject to change.