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All rights reserved as an unpublished work.

This is an unapproved and unpublished IEEE Standards Draft, subject to change.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]

4.43 `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.

(3) If the destination path exists, and it is a file of type directory and *source_file* is not a file of type directory, or it is a file not of type directory and *source_file* is a file of type directory, `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.

(4) If the destination path exists, `mv` shall attempt to remove it. If this fails for any reason, `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.

(5) The file hierarchy rooted in *source_file* shall be duplicated as a file hierarchy rooted in the destination path. The following characteristics of each file in the file hierarchy shall be duplicated:

(a) The time of last data modification and time of last access.

(b) The user ID and group ID.

(c) The file mode.

If the user ID, group ID, or file mode of a regular file cannot be duplicated, the file mode bits S_ISUID and S_ISGID shall not be duplicated.

When files are duplicated to another file system, the implementation may 1
require that the process invoking `mv` have read access to each file being 1
duplicated. 1

If the duplication of the file hierarchy fails for any reason, `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.

If the duplication of the file characteristics fails for any reason, `mv` shall write a diagnostic message to standard error, but this failure shall not cause `mv` to modify its exit status.

(6) The file hierarchy rooted in *source_file* shall be removed. If this fails for any reason, `mv` shall write a diagnostic message to the standard error, do nothing more with the current *source_file,* and go on to any remaining *source_file* s.

4.43.3 Options

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

The following options shall be supported by the implementation:

`−f` Do not prompt for confirmation if the *destination* path exists. Any previous occurrences of the `−i` option shall be ignored.

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

`-i` Prompt for confirmation if the destination path exists. Any previous occurrences of the `-f` option shall be ignored.

Specifying more than one of the `-f` or `-i` options shall not be considered an error. The last option specified shall determine `mv`'s behavior.

4.43.4 Operands

The following operands shall be supported by the implementation:

source_file A pathname of a file or directory to be moved.

target_file A new pathname for the file or directory being moved.

target_dir A pathname of an existing directory into which to move the input files.

4.43.5 External Influences

4.43.5.1 Standard Input

Used to read an input line in response to each prompt specified in Standard Error. 4.43.6.2. Otherwise, the standard input shall not be used.

4.43.5.2 Input Files

The input files specified by each *source_file* operand can be of any file type.

4.43.5.3 Environment Variables

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

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_COLLATE This variable shall determine the locale for the behavior of ranges, equivalence classes, and multicharacter collating elements used in the extended regular expression defined for the `yesexpr` locale keyword in the LC_MESSAGES category.

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

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) and the behavior of character classes within regular expressions used in the extended regular expression defined for the `yesexpr` locale keyword in the LC_MESSAGES category.

LC_MESSAGES This variable shall determine the processing of affirmative responses and the language in which messages should be written.

4.43.5.4 Asynchronous Events

Default.

4.43.6 External Effects

4.43.6.1 Standard Output

None.

4.43.6.2 Standard Error

Prompts shall be written to the standard error under the conditions specified in 4.43.2. The prompts shall contain the *destination* pathname, but their format is otherwise unspecified. Otherwise, the standard error shall be used only for diagnostic messages.

4.43.6.3 Output Files

The output files may be of any file type.

4.43.7 Extended Description

None.

4.43.8 Exit Status

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

0 All input files were moved successfully.

>0 An error occurred.

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

4.43.9 Consequences of Errors

If the copying or removal of *source_file* is prematurely terminated by a signal or error, `mv` may leave a partial copy of *source_file* at the source or destination. The `mv` utility shall not modify both *source_file* and the destination path simultaneously; termination at any point shall leave either *source_file* or the destination path complete.

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

Examples, Usage

If the current directory contains only files `a` (of any type defined by POSIX.1 {8}), `b` (also of any type), and a directory `c`:

```
mv a b c
mv c d
```

will result with the original files `a` and `b` residing in the directory `d` in the current directory.

History of Decisions Made

Previous versions of this draft diverged from *SVID* and BSD historical practice in that they required that when the destination path exists, the `-f` option is not specified, and input is not a terminal, `mv` shall fail. This was done for compatibility with `cp`. This draft returns to historical practice. It should be noted that this is consistent with the POSIX.1 {8} function *rename*(), which does not require write permission on the target.

For absolute clarity, paragraph (1), describing `mv`'s behavior when prompting for confirmation, should be interpreted in the following manner:

```
if (exists AND (NOT f_option) AND
        ((not_writable AND input_is_terminal) OR i_option))
```

The `-i` option exists on BSD systems, giving applications and users a way to avoid accidentally unlinking files when moving others. When the standard input is not a terminal, the 4.3BSD `mv` deletes all existing destination paths without prompting, even when `-i` is specified; this is inconsistent with the behavior of the 4.3BSD `cp` utility, which always generates an error when the file is unwritable and the standard input is not a terminal. The working group decided that use of `-i` is a request for interaction, so when the *destination* path exists, the utility takes instructions from whatever responds to standard input.

The *rename*() function is able to move directories within the same file system. Some historical versions of `mv` have been able to move directories, but not to a different file system. The working group felt that this was an annoying inconsistency, so the standard requires directories to be movable even across file systems. There is no `-R` option to confirm that moving a directory is actually intended, since such an option was not required for moving directories in historical practice. Requiring the application to specify it sometimes, depending on the

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

destination, seemed just as inconsistent. The semantics of the *rename* () function were preserved as much as possible. For example, `mv` is not permitted to ''rename'' files to or from directories, even though they might be empty and removable.

Historic implementations of `mv` did not exit with a nonzero exit status if they were unable to duplicate any file characteristics when moving a file across file systems, nor did they write a diagnostic message for the user. The former behavior has been preserved to prevent scripts from breaking; a diagnostic message is now required, however, so that users are alerted that the file characteristics have changed.

The exact format of the interactive prompts is unspecified. Only the general nature of the contents of prompts are specified, because implementations may desire more descriptive prompts than those used on historical implementations. Therefore, an application not using the `−f` option or using the `−i` option relies on the system to provide the most suitable dialogue directly with the user, based on the behavior specified.

4.44 nohup — Invoke a utility immune to hangups

4.44.1 Synopsis

`nohup` *utility* [*argument* . . .]

4.44.2 Description

The `nohup` utility shall invoke the utility named by the *utility* operand with arguments supplied as the *argument* operands. At the time the named *utility* is invoked, the SIGHUP signal shall be set to be ignored.

If the standard output is a terminal, all output written by the named *utility* to its standard output shall be appended to the end of the file `nohup.out` in the current directory. If `nohup.out` cannot be created or opened for appending, the output shall be appended to the end of the file `nohup.out` in the directory specified by the **HOME** environment variable. If neither file can be created or opened for appending, *utility* shall not be invoked. If a file is created, the file's permission bits shall be set to S_IRUSR | S_IWUSR instead of the default specified in 2.9.1.4.

If the standard error is a terminal, all output written by the named *utility* to its standard error shall be redirected to the same file descriptor as the standard output.

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