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 'mv.1p' command

$ man mv.1p

MV(1P) POSIX Programmer's Manual MV(1P)

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

mv ? move files

SYNOPSIS

mv [-if] source_file target_file

mv [-if] source_file... target_dir

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 tar? get_file. This first synopsis form is assumed when the final operand does not name an existing directory and is not a symbolic link refer? ring to an existing directory. In this case, if source_file names a non-directory file and target_file ends with a trailing <slash> charac? ter, mv shall treat this as an error and no source_file operands will be processed.

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, or referenced if target_dir is a sym?

bolic link referring to an existing directory. The destination path for each source_file shall be the concatenation of the target directory, a single <slash> character if the target did not end in a <slash>, and the last pathname component of the source_file. This second form is assumed when the final operand names an existing directory.

If any operand specifies an existing file of a type not specified by the System Interfaces volume of POSIX.1?2017, the behavior is implemen? tation-defined.

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_files.
2. If the source_file operand and destination path resolve to either the same existing directory entry or different directory entries for the same existing file, then the destination path shall not be removed, and one of the following shall occur:
   a. No change is made to source_file, no error occurs, and no diag? nostic is issued.
   b. No change is made to source_file, a diagnostic is issued to standard error identifying the two names, and the exit status is affected.
   c. If the source_file operand and destination path name distinct directory entries, then the source_file operand is removed, no error occurs, and no diagnostic is issued.

   The mv utility shall do nothing more with the current source_file, and go on to any remaining source_files.
3. The mv utility shall perform actions equivalent to the rename()

function defined in the System Interfaces volume of POSIX.1?2017, 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_files. If this fails for any reasons other than those described for the errno [EXDEV] in the System Interfaces volume of POSIX.1?2017, mv shall write a di? agnostic message to standard error, do nothing more with the cur? rent source_file, and go on to any remaining source_files.

4. 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_files. If the destination path exists and was created by a previous step, it is unspecified whether this will treated as an error or the destination path will be overwritten.

5. 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_files.

6. The file hierarchy rooted in source_file shall be duplicated as a file hierarchy rooted in the destination path. If source_file or any of the files below it in the hierarchy are symbolic links, the links themselves shall be duplicated, including their contents, rather than any files to which they refer. The following character? istics of each file in the file hierarchy shall be duplicated:

* The time of last data modification and time of last access
* The user ID and group ID
* 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 du?

plicated.

When files are duplicated to another file system, the implementa? tion may require that the process invoking mv has read access to each file being duplicated.

If files being duplicated to another file system have hard links to other files, it is unspecified whether the files copied to the new file system have the hard links preserved or separate copies are created for the linked files.

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_files.

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

7. 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_files.

OPTIONS

The mv utility shall conform to the Base Definitions volume of POSIX.1?2017, Section 12.2, Utility Syntax Guidelines.

The following options shall be supported:

-f Do not prompt for confirmation if the destination path ex? ists. Any previous occurrence of the -i option is ignored.

-i Prompt for confirmation if the destination path exists. Any previous occurrence of the -f option is ignored.

Specifying more than one of the -f or -i options shall not be consid? ered an error. The last option specified shall determine the behavior of mv.

OPERANDS

The following operands shall be supported:

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.

STDIN

The standard input shall be used to read an input line in response to each prompt specified in the STDERR section. Otherwise, the standard input shall not be used.

INPUT FILES

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

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of mv:

LANG Provide a default value for the internationalization vari? ables that are unset or null. (See the Base Definitions vol? ume of POSIX.1?2017, Section 8.2, Internationalization Vari? ables for the precedence of internationalization variables used to determine the values of locale categories.)

LC_ALL If set to a non-empty string value, override the values of all the other internationalization variables.

LC_COLLATE

Determine the locale for the behavior of ranges, equivalence classes, and multi-character collating elements used in the extended regular expression defined for the yesexpr locale keyword in the LC_MESSAGES category.

LC_CTYPE Determine the locale for the interpretation of sequences of bytes of text data as characters (for example, single-byte as opposed to multi-byte characters in arguments and input files), the behavior of character classes used in the ex? tended regular expression defined for the yesexpr locale key? word in the LC_MESSAGES category.

LC_MESSAGES

Determine the locale used to process affirmative responses, and the locale used to affect the format and contents of di? agnostic messages and prompts written to standard error.

NLSPATH Determine the location of message catalogs for the processing of LC_MESSAGES.

ASYNCHRONOUS EVENTS

Default.

STDOUT

Not used.

STDERR

Prompts shall be written to the standard error under the conditions specified in the DESCRIPTION section. The prompts shall contain the destination pathname, but their format is otherwise unspecified. Oth? erwise, the standard error shall be used only for diagnostic messages.

OUTPUT FILES

The output files may be of any file type.

EXTENDED DESCRIPTION

None.

EXIT STATUS

The following exit values shall be returned:

0 All input files were moved successfully.

>0 An error occurred.

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.

The following sections are informative.

APPLICATION USAGE

Some implementations mark for update the last file status change time? stamp of renamed files and some do not. Applications which make use of

the last file status change timestamp may behave differently with re? spect to renamed files unless they are designed to allow for either be? havior.

The specification ensures that mv a a will not alter the contents of file a, and allows the implementation to issue an error that a file cannot be moved onto itself. Likewise, when a and b are hard links to the same file, mv a b will not alter b, but if a diagnostic is not is? sued, then it is unspecified whether a is left untouched (as it would be by the rename() function) or unlinked (reducing the link count of b).

EXAMPLES

If the current directory contains only files a (of any type defined by the System Interfaces volume of POSIX.1?2017), b (also of any type), and a directory c:

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

results with the original files a and b residing in the directory d in the current directory.

RATIONALE

Early proposals diverged from the 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 fails. This was done for compatibility with cp. The current text returns to historical practice. It should be noted that this is consistent with the rename() function defined in the System Interfaces volume of POSIX.1?2017, which does not require write permission on the target.

For absolute clarity, paragraph (1), describing the behavior of mv when prompting for confirmation, should be interpreted in the following man? ner:

```
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 4.3 BSD mv deletes all existing destination paths without prompting, even when -i is specified; this is inconsistent with the behavior of the 4.3 BSD cp utility, which always generates an error when the file is unwritable and the standard input is not a terminal. The standard developers decided that use of -i is a request for interaction, so when the destination path exists, the util?ity 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 directo?ries, but not to a different file system. The standard developers con?sidered that this was an annoying inconsistency, so this volume of POSIX.1?2017 requires directories to be able to be moved 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 spec?ify it sometimes, depending on the destination, seemed just as incon?sistent. 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 non-zero 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 imple?mentations may desire more descriptive prompts than those used on his?torical implementations. Therefore, an application not using the -f op?tion or using the -i option relies on the system to provide the most suitable dialog directly with the user, based on the behavior speci?fied.

When mv is dealing with a single file system and source_file is a sym?

bolic link, the link itself is moved as a consequence of the dependence on the rename() functionality, per the DESCRIPTION. Across file sys? tems, this has to be made explicit.

FUTURE DIRECTIONS

None.

SEE ALSO

cp, ln

The Base Definitions volume of POSIX.1?2017, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guidelines

The System Interfaces volume of POSIX.1?2017, rename()

COPYRIGHT