mmv

Mass Move and rename - Move, copy, append or link Multiple **files** using wildcard patterns.

Syntax

```
mmv [Source_Option] [-h] [-d|p] [-g|t] [-v|n] [--] [from to]
```

Options:

```
-d   Delete
-p   Protect (don't delete or overwrite)
-g   Go
-t   Terminate
-n   no-execute mode (display messages about what would have been done)
-v   verbose mode
-h   help
```

Source_Options:

```
-m   Move source file to target name.
     Both must be on the same device. Will not move directories.
     If the source file is a symbolic link, moves the link without checking
     if the link's target from the new directory is different than the old.

-x   same as -m, except cross-device moves are done by copying, then deleting source.
     When copying, sets the permission bits and file modification time of the target
     file to that of the source file.

-r   Rename source file or directory to target name.
     The target name must not include a path: the file remains in the same directory
     in all cases. This option is the only way of renaming directories under mmv.

-c   Copy source file to target name.
     Sets the file modification time and permission bits of the target file to that
     of the source file, regardless of whether the target file already exists.
     Chains and cycles (to be explained below) are not allowed.

-o   Overwrite target name with source file.
     If target file exists, it is overwritten, keeping its original owner and
     permission bits. If it does not exist, it is created, with read-write permission
     bits set according to umask, and the execute permission bits copied from the
     source file. In either case, the file modification time is set to the current time.

-a   Append contents of source file to target name.
     Target file modification time is set to the current time. If target file does not
     exist, it is created with permission bits set as under -o.
     Unlike all other options, -a allows multiple source files to have the same target name,
     e.g. "mmv -a \*.c big" will append all ".c" files to "big".
     Chains and cycles are also allowed, so "mmv -a f f" will double up "f".

-l   Link target name to source file.
     Both must be on the same device, and the source must not be a directory.
     Chains and cycles are not allowed.

-s   Same as -l, but use symbolic links instead of hard links.
     For the resulting link to aim back at the source, either the source name
     must begin with a '/', or the target must reside in either the current or the
     source directory. If none of these conditions are met, the link is refused.
     However, source and target can reside on different devices, and the source can be a directory.

If no source_option is specified, the task is given by the command name under which mmv was invoked (argv[

command_name   default task

mmv            -x (Move except cross device)
mcp            -c (Copy)
mad            -a (Append)
mln            -l (Link)
```

Options need not be given separately, i.e. "mmv -mk" is allowed.

Mmv moves (or copies, appends, or links, as specified) each source file matching a **from** pattern to the target name specified by the **to** pattern.

This multiple action is performed safely, i.e. without any unexpected deletion of files due to collisions of target names with existing filenames or with other target names. Furthermore, before doing anything, mmv attempts to detect any errors that would result from the entire set of actions specified and gives the user the choice of either proceeding by avoiding the offending parts or cancelling.

mmv does support large files (LFS) but it does *NOT* support sparse files (i.e. it explodes them).

Multiple Pattern Pairs

Multiple from -- to pattern pairs can be specified by omitting the pattern pair on the command line, and entering them on the standard input, one pair per line. (If a pattern pair is given on the command line, the standard input is not read.) Thus,

```
mmv
a b
c d
```

would rename "a" to "b" and "c" to "d".

If a file can be matched to several of the given from patterns, the to pattern of the first matching pair is used. Thus,

```
mmv
a b
a c
```

would give the error message "a -> c : no match" because file "a" (even if it exists) was already matched by the first pattern pair.

The From Pattern

The from pattern is a filename with embedded wildcards: '*', '?', '['...']', and ';'. The first three have their usual sh(1) meanings of, respectively, matching any string of characters, matching any single character, and matching any one of a set of characters.

Between the '[' and ']', a range from character 'a' through character 'z' is specified with "a-z". The set of matching characters can be negated by inserting a '^' after the '['. Thus, "[^b-e2-5_]" will match any character but 'b' through 'e', '2' through '5', and '_'.

Note that paths are allowed in the patterns, and wildcards can be intermingled with slashes arbitrarily. The ';' wildcard is useful for matching files at any depth in the directory tree. It matches the same as "*/" repeated any number of times, including zero, and can only occur either at the beginning of the pattern or following a '/'. Thus ";*.c" will match all ".c" files in or below the current directory, while "/;*.c" will match them anywhere on the file system.

In addition, if the from pattern (or the to pattern) begins with "~/", the '~' is replaced with the home directory name. (Note that the "~user" feature of csh(1) is not implemented.) However, the '~' is not treated as a wildcard, in the sense that it is not assigned a wildcard index (see below).

Since matching a directory under a task option other than -r or -s would result in an error, tasks other than -r and -s match directories only against completely explicit from patterns (i.e. not containing wildcards). Under -r and -s, this applies only to "." and "..".

Files beginning with '.' are only matched against from patterns that begin with an explicit '.'. However, if -h is specified, they are matched normally.

Warning: since the shell normally expands wildcards before passing the command-line arguments to mmv, it is usually necessary to enclose the command-line from and to patterns in quotes.

The To Pattern

The to pattern is a filename with embedded wildcard indexes, where an index consists of the character '#' followed by a string of digits.

When a source file matches a from pattern, a target name for the file is constructed out of the to pattern by replacing the wildcard indexes by the actual characters that matched the referenced wildcards in the source name. Thus, if the from pattern is "abc*.*" and the to pattern is "xyz#2.#1", then "abc.txt" is targeted to "xyztxt.". (The first '*' matched "", and the second matched "txt".) Similarly, for the pattern pair ";*.[clp]" -> "#1#3/#2", "foo1/foo2/prog.c" is targeted to "foo1/foo2/c/prog". Note that there is no '/' following the "#1" in the to pattern, since the string matched by any ';' is always either empty or ends in a '/'. In this case, it matches "foo1/foo2/".

To convert the string matched by a wildcard to either lowercase or uppercase before embedding it in the target name, insert 'l' or 'u', respectively, between the '#' and the string of digits.

The to pattern, like the from pattern, can begin with a "~/" (see above). This does not necessitate enclosing the to pattern in quotes on the command line since csh(1) expands the '~' in the exact same manner as mmv (or, in the case of sh(1), does not expand it at all).

For all task options other than -r, if the target name is a directory, the real target name is formed by appending a '/' followed by the last component of the source file name. For example, "mmv dir1/a dir2" will, if "dir2" is indeed a directory, actually move "dir1/a" to "dir2/a". However, if "dir2/a" already exists and is itself a directory, this is considered an error.

To strip any character (e.g. '*', '?', or '#') of its special meaning to mmv, as when the actual replacement name must

contain the character '#', precede the special character with a '\' (and enclose the argument in quotes because of the shell). This also works to terminate a wildcard index when it has to be followed by a digit in the filename, e.g. "a#1\1".

Chains and Cycles

A chain is a sequence of specified actions where the target name of one action refers to the source file of another action. For example,

```
mmv
a b
b c
```

specifies the chain "a" -> "b" -> "c". A cycle is a chain where the last target name refers back to the first source file, e.g. "mmv a a".

Mmv detects chains and cycles regardless of the order in which their constituent actions are actually given. Where allowed, i.e. in moving, renaming, and appending files, chains and cycles are handled gracefully, by performing them in the proper order. Cycles are broken by first renaming one of the files to a temporary name (or just remembering its original size when doing appends).

Collisions and Deletions

When any two or more matching files would have to be moved, copied, or linked to the same target filename, mmv detects the condition as an error before performing any actions. Furthermore, mmv checks if any of its actions will result in the destruction of existing files.

If the -d (delete) option is specified, all file deletions or overwrites are done silently.

Under -p (protect), all deletions or overwrites (except those specified with "(*)" on the standard input, see below) are treated as errors. And if neither option is specified, the user is queried about each deletion or overwrite separately. (A new stream to "/dev/tty" is used for all interactive queries, not the standard input.)

Error Handling

Whenever any error in the user's action specifications is detected, an error message is given on the standard output, and mmv proceeds to check the rest of the specified actions. Once all errors are detected, mmv queries the user whether he wishes to continue by avoiding the erroneous actions or to exit altogether. This and all other queries may be avoided by specifying either the -g (go) or -t (terminate)
option. The former will resolve all difficulties by avoiding the erroneous actions; the latter will exit mmv if any errors are detected.

Specifying either of them defaults mmv to -p, unless -d is specified (see above). Thus, -g and -t are most useful when running mmv in the background or in a shell script, when interactive queries are undesirable.

Reports

Once the actions to be performed are determined, mmv performs them silently, unless either the -v (verbose) or -n (no-execute) option is specified. The former causes mmv to report each performed action on the standard output as

```
a -> b : done.
```

Here, "a" and "b" would be replaced by the source and target names, respectively. If the action deletes the old target, a "(*)" is inserted after the the target name. Also, the "->" symbol is modified when a cycle has to be broken: the '>' is changed to a '^' on the action prior to which the old target is renamed to a temporary, and the'-' is changed to a '=' on the action where the temporary is used.

Under -n, none of the actions are performed, but messages like the above are printed on the standard output with the ": done." omitted.

The output generated by -n can (after editing, if desired) be fed back to mmv on the standard input (by omitting the from -- to pair on the mmv command line). To facilitate this, mmv ignores lines on the standard input that look like its own error and "done" messages, as well as all lines beginning with white space, and will accept pattern pairs with or without the intervening "->" (or "-^", "=>", or "=^"). Lines with "(*)" after the target pattern have the effect of enabling -d for the files matching this pattern only, so that such deletions are done silently. When feeding mmv its own output, one must remember to specify again the task option (if any) originally used to generate it.

Although mmv attempts to predict all mishaps prior to performing any specified actions, accidents can happen. For example, mmv does not check for adequate free space when copying. Thus, despite all efforts, it is still possible for an action to fail after some others have already been done. To make recovery as easy as possible, mmv reports which actions have already been done and which are still to be performed after such a failure occurs. It then exits, not attempting to do anything else. Once the user has cleared up the problem, he can feed this report back to mmv on the standard input to have it complete the task. (The user is queried for a file name to dump this report if the standard output has not been redirected.)

Exit

Mmv exits with status 1 if it exits before doing anything, with status 2 if it exits due to failure after completing some of the actions, and with status 0 otherwise.

Bugs

If the search pattern is not quoted, the shell expands the wildcards. Mmv then (usually) gives some error message, but can not determine that the lack of quotes is the cause.

To avoid difficulties in semantics and error checking, mmv refuses to move or create directories.

If the mmv tool is not installed on your distro, get it with: `apt-get install mmv`

Examples

Rename the file extension of all .csv files in the current directory to .xls

```
mmv "*.csv" "#1.xls"
```

Copy all the file names that start with `bicyclerace`, and rename to `bicyclerace-paris` preserving the original file extention:

```
mmv "bicyclerace.*" "bicyclerace-paris.#1"
```

Copy `report6part4.txt` to `./french/rapport6partie4.txt` along with all similarly named files:

```
mmv -c "report*part*.txt" "./french/rapport#1partie#2.txt"
```

Append the contents of all .txt files into one file:

```
mmv -a "*.txt" "all.txt"
```

"All things change, nothing is extinguished. There is nothing in the whole world which is permanent. Everything flows onward; all things are brought into being with a changing nature; the ages themselves glide by in constant movement" ~ Ovid

Related linux commands

mv - Move or rename files or directories.
cp - Copy one or more files to another location.
ln - Make links between files.
umask - Users file creation mask.