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 'chmod.1p' command

$ man chmod.1p

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

chmod ? change the file modes

SYNOPSIS

chmod [-R] mode file...

DESCRIPTION

The chmod utility shall change any or all of the file mode bits of the file named by each file operand in the way specified by the mode oper? and.

It is implementation-defined whether and how the chmod utility affects any alternate or additional file access control mechanism (see the Base Definitions volume of POSIX.1?2017, Section 4.5, File Access Permis? sions) being used for the specified file.

Only a process whose effective user ID matches the user ID of the file, or a process with appropriate privileges, shall be permitted to change the file mode bits of a file.

Upon successfully changing the file mode bits of a file, the chmod utility shall mark for update the last file status change timestamp of

the file.

OPTIONS

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

The following option shall be supported:

-R Recursively change file mode bits. For each file operand that names a directory, chmod shall change the file mode bits of the directory and all files in the file hierarchy below it.

OPERANDS

The following operands shall be supported:

mode Represents the change to be made to the file mode bits of each file named by one of the file operands; see the EXTENDED DESCRIPTION section.

file A pathname of a file whose file mode bits shall be modified.

STDIN

Not used.

INPUT FILES

None.

ENVIRONMENT VARIABLES

The following environment variables shall affect the execution of chmod:

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_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).

LC_MESSAGES

Determine the locale that should be used to affect the format

and contents of diagnostic messages written to standard er?
ror.

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

ASYNCHRONOUS EVENTS

Default.

STDOUT

Not used.

STDERR

The standard error shall be used only for diagnostic messages.

OUTPUT FILES

None.

EXTENDED DESCRIPTION

The mode operand shall be either a symbolic_mode expression or a non-negative octal integer. The symbolic_mode form is described by the grammar later in this section.

Each clause shall specify an operation to be performed on the current file mode bits of each file. The operations shall be performed on each file in the order in which the clauses are specified.

The who symbols u, g, and o shall specify the user, group, and other parts of the file mode bits, respectively. A who consisting of the sym? bol a shall be equivalent to ugo.

The perm symbols r, w, and x represent the read, write, and exe? cute/search portions of file mode bits, respectively. The perm symbol s shall represent the set-user-ID-on-execution (when who contains or im? plies u) and set-group-ID-on-execution (when who contains or implies g) bits.

The perm symbol X shall represent the execute/search portion of the file mode bits if the file is a directory or if the current (unmodi? fied) file mode bits have at least one of the execute bits (S_IXUSR, S_IXGRP, or S_IXOTH) set. It shall be ignored if the file is not a di? rectory and none of the execute bits are set in the current file mode bits.

The permcopy symbols u, g, and o shall represent the current permis? sions associated with the user, group, and other parts of the file mode bits, respectively. For the remainder of this section, perm refers to the non-terminals perm and permcopy in the grammar.

If multiple actionlists are grouped with a single wholist in the gram? mar, each actionlist shall be applied in the order specified with that wholist. The op symbols shall represent the operation performed, as follows:

+ If perm is not specified, the '+' operation shall not change the file mode bits.

  If who is not specified, the file mode bits represented by perm for the owner, group, and other permissions, except for those with corresponding bits in the file mode creation mask of the in? voking process, shall be set.

  Otherwise, the file mode bits represented by the specified who and perm values shall be set.

- If perm is not specified, the '-' operation shall not change the file mode bits.

  If who is not specified, the file mode bits represented by perm for the owner, group, and other permissions, except for those with corresponding bits in the file mode creation mask of the in? voking process, shall be cleared.

  Otherwise, the file mode bits represented by the specified who and perm values shall be cleared.

= Clear the file mode bits specified by the who value, or, if no who value is specified, all of the file mode bits specified in this volume of POSIX.1?2017.

  If perm is not specified, the '=' operation shall make no further modifications to the file mode bits.

  If who is not specified, the file mode bits represented by perm for the owner, group, and other permissions, except for those with corresponding bits in the file mode creation mask of the in? voking process, shall be set.

Otherwise, the file mode bits represented by the specified who and perm values shall be set.

When using the symbolic mode form on a regular file, it is implementation-defined whether or not:

* Requests to set the set-user-ID-on-execution or set-group-ID-on-execution bit when all execute bits are currently clear and none are being set are ignored.
* Requests to clear all execute bits also clear the set-user-ID-on-execution and set-group-ID-on-execution bits.
* Requests to clear the set-user-ID-on-execution or set-group-ID-on-execution bits when all execute bits are currently clear are ignored. However, if the command ls -l file writes an s in the position indicating that the set-user-ID-on-execution or set-group-ID-on-execution is set, the commands chmod u-s file or chmod g-s file, respectively, shall not be ignored.

When using the symbolic mode form on other file types, it is implementation-defined whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execution bits are honored.

If the who symbol o is used in conjunction with the perm symbol s with no other who symbols being specified, the set-user-ID-on-execution and set-group-ID-on-execution bits shall not be modified. It shall not be an error to specify the who symbol o in conjunction with the perm symbol s.

The perm symbol t shall specify the S_ISVTX bit. When used with a file of type directory, it can be used with the who symbol a, or with no who symbol. It shall not be an error to specify a who symbol of u, g, or o in conjunction with the perm symbol t, but the meaning of these combinations is unspecified. The effect when using the perm symbol t with any file type other than directory is unspecified.

For an octal integer mode operand, the file mode bits shall be set absolutely.

For each bit set in the octal number, the corresponding file permission bit shown in the following table shall be set; all other file permis?

sion bits shall be cleared. For regular files, for each bit set in the octal number corresponding to the set-user-ID-on-execution or the set-group-ID-on-execution, bits shown in the following table shall be set; if these bits are not set in the octal number, they are cleared. For other file types, it is implementation-defined whether or not requests to set or clear the set-user-ID-on-execution or set-group-ID-on-execu? tion bits are honored.

Octal	Mode Bit	Octal	Mode Bit	Octal	Mode Bit	Octal	Mode Bit
4000	S_ISUID	0400	S_IRUSR	0040	S_IRGRP	0004	S_IROTH
2000	S_ISGID	0200	S_IWUSR	0020	S_IWGRP	0002	S_IWOTH
1000	S_ISVTX	0100	S_IXUSR	0010	S_IXGRP	0001	S_IXOTH

When bits are set in the octal number other than those listed in the table above, the behavior is unspecified.

Grammar for chmod

The grammar and lexical conventions in this section describe the syntax for the symbolic_mode operand. The general conventions for this style of grammar are described in Section 1.3, Grammar Conventions. A valid symbolic_mode can be represented as the non-terminal symbol sym? bolic_mode in the grammar. This formal syntax shall take precedence over the preceding text syntax description.

The lexical processing is based entirely on single characters. Imple? mentations need not allow <blank> characters within the single argument being processed.

```
%start    symbolic_mode
%%
symbolic_mode    : clause
                 | symbolic_mode ',' clause
                 ;
```

```
clause          : actionlist
                | wholist actionlist
                ;
wholist         : who
                | wholist who
                ;
who             : 'u' | 'g' | 'o' | 'a'
                ;
actionlist      : action
                | actionlist action
                ;
action          : op
                | op permlist
                | op permcopy
                ;
permcopy        : 'u' | 'g' | 'o'
                ;
op              : '+' | '-' | '='
                ;
permlist        : perm
                | perm permlist
                ;
perm            : 'r' | 'w' | 'x' | 'X' | 's' | 't'
                ;
```

EXIT STATUS

The following exit values shall be returned:

0 The utility executed successfully and all requested changes were made.

>0 An error occurred.

CONSEQUENCES OF ERRORS

Default.

The following sections are informative.

APPLICATION USAGE

Some implementations of the chmod utility change the mode of a direc?
tory before the files in the directory when performing a recursive (-R
option) change; others change the directory mode after the files in the
directory. If an application tries to remove read or search permission
for a file hierarchy, the removal attempt fails if the directory is
changed first; on the other hand, trying to re-enable permissions to a
restricted hierarchy fails if directories are changed last. Users
should not try to make a hierarchy inaccessible to themselves.

Some implementations of chmod never used the umask of the process when
changing modes; systems conformant with this volume of POSIX.1?2017 do
so when who is not specified. Note the difference between:

```
chmod a-w file
```

which removes all write permissions, and:

```
chmod -- -w file
```

which removes write permissions that would be allowed if file was cre?
ated with the same umask.

Conforming applications should never assume that they know how the set-
user-ID and set-group-ID bits on directories are interpreted.

EXAMPLES

```
??????????????????????????????????????????
?Mode  ?          Results           ?
??????????????????????????????????????????
?a+=   ? Equivalent to a+,a=; clears    ?
?      ? all file mode bits.          ?
?go+-w ? Equivalent to go+,go-w; clears ?
?      ? group and other write bits.   ?
?g=o-w ? Equivalent to g=o,g-w; sets    ?
?      ? group bit to match other bits  ?
?      ? and then clears group write    ?
?      ? bit.                          ?
?g-r+w ? Equivalent to g-r,g+w; clears  ?
?      ? group read bit and sets group  ?
?      ? write bit.                    ?
```

?uo=g ? Sets owner bits to match group ?

? ? bits and sets other bits to ?

? ? match group bits. ?

??

RATIONALE

The functionality of chmod is described substantially through refer? ences to concepts defined in the System Interfaces volume of POSIX.1?2017. In this way, there is less duplication of effort required for describing the interactions of permissions. However, the behavior of this utility is not described in terms of the chmod() function from the System Interfaces volume of POSIX.1?2017 because that specification requires certain side-effects upon alternate file access control mecha? nisms that might not be appropriate, depending on the implementation.

Implementations that support mandatory file and record locking as spec? ified by the 1984 /usr/group standard historically used the combination of set-group-ID bit set and group execute bit clear to indicate manda? tory locking. This condition is usually set or cleared with the sym? bolic mode perm symbol l instead of the perm symbols s and x so that the mandatory locking mode is not changed without explicit indication that that was what the user intended. Therefore, the details on how the implementation treats these conditions must be defined in the documen? tation. This volume of POSIX.1?2017 does not require mandatory locking (nor does the System Interfaces volume of POSIX.1?2017), but does allow it as an extension. However, this volume of POSIX.1?2017 does require that the ls and chmod utilities work consistently in this area. If ls -l file indicates that the set-group-ID bit is set, chmod g-s file must clear it (assuming appropriate privileges exist to change modes).

The System V and BSD versions use different exit status codes. Some im? plementations used the exit status as a count of the number of errors that occurred; this practice is unworkable since it can overflow the range of valid exit status values. This problem is avoided here by specifying only 0 and >0 as exit values.

The System Interfaces volume of POSIX.1?2017 indicates that implementa?

tion-defined restrictions may cause the S_ISUID and S_ISGID bits to be ignored. This volume of POSIX.1?2017 allows the chmod utility to choose to modify these bits before calling chmod() (or some function providing equivalent capabilities) for non-regular files. Among other things, this allows implementations that use the set-user-ID and set-group-ID bits on directories to enable extended features to handle these exten? sions in an intelligent manner.

The X perm symbol was adopted from BSD-based systems because it pro? vides commonly desired functionality when doing recursive (-R option) modifications. Similar functionality is not provided by the find util? ity. Historical BSD versions of chmod, however, only supported X with op+; it has been extended in this volume of POSIX.1?2017 because it is also useful with op=. (It has also been added for op- even though it duplicates x, in this case, because it is intuitive and easier to ex? plain.)

The grammar was extended with the permcopy non-terminal to allow his? torical-practice forms of symbolic modes like o=u -g (that is, set the ``other'' permissions to the permissions of ``owner'' minus the permis? sions of ``group'').

FUTURE DIRECTIONS

None.

SEE ALSO

ls, umask

The Base Definitions volume of POSIX.1?2017, Section 4.5, File Access Permissions, Chapter 8, Environment Variables, Section 12.2, Utility Syntax Guidelines

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

COPYRIGHT

event of any discrepancy between this version and the original IEEE and The Open Group Standard, the original IEEE and The Open Group Standard is the referee document. The original Standard can be obtained online at http://www.opengroup.org/unix/online.html .

Any typographical or formatting errors that appear in this page are most likely to have been introduced during the conversion of the source files to man page format. To report such errors, see https://www.ker?nel.org/doc/man-pages/reporting_bugs.html .