NAME

chmod – change the permissions mode of a file

SYNOPSIS

chmod [**–fR**] **<***absolute-mode***>** *file*. . .
chmod [**–fR**] **<***symbolic-mode-list***>** *file*. . .

AVAILABILITY

SUNWcsu

DESCRIPTION

chmod changes or assigns the mode of a file. The mode of a file specifies its permissions and other attributes. The mode may be absolute or symbolic.

Absolute *mode*

An absolute *mode* is specified using octal numbers:

chmod *nnnn file* . . .

where:

n a number from **0** to **7**. An absolute mode is constructed from the OR of any of the following modes:

4000	Set user ID on execution.
20#0	Set group ID on execution if # is **7**, **5**, **3**, or **1**. Enable mandatory locking if # is **6**, **4**, **2**, or **0**. For directories, files are created with BSD semantics for propagation of the group ID. With this option, files and subdirectories created in the directory inherit the group ID of the directory, rather than of the current process. It may be cleared only by using symbolic mode.
1000	Turn on sticky bit. See **chmod**(2).
0400	Allow read by owner.
0200	Allow write by owner.
0100	Allow execute (search in directory) by owner.
0700	Allow read, write, and execute (search) by owner.
0040	Allow read by group.
0020	Allow write by group.
0010	Allow execute (search in directory) by group.
0070	Allow read, write, and execute (search) by group.
0004	Allow read by others.
0002	Allow write by others.
0001	Allow execute (search in directory) by others.
0007	Allow read, write, and execute (search) by others.

Note that the **setgid** bit cannot be set (or cleared) in absolute mode; it must be set (or cleared) in symbolic mode using **g+s** (or **g-s**).

Symbolic *mode*

A symbolic *mode* specification has the following format:

chmod <*symbolic-mode-list*> *file* . . .

where: <*symbolic-mode-list*> is a comma-separated list (with no intervening whitespace) of symbolic mode expressions of the form:

[*who*] *operator* [*permissions*]

Operations are performed in the order given. Multiple *permissions* letters following a single operator cause the corresponding operations to be performed simultaneously.

who zero or more of the characters **u**, **g**, **o**, and **a** specifying whose permissions are to be changed or assigned:

u	user's permissions
g	group's permissions
o	others' permissions
a	all permissions (user, group, and other)

If *who* is omitted, it defaults to **a**, but the setting of the file mode creation mask (see **umask** in **sh**(1) or **csh**(1) for more information) is taken into account. When *who* is omitted, **chmod** will not override the restrictions of your user mask.

operator either +, –, or =, signifying how permissions are to be changed:

+ Add permissions.

If *permissions* is omitted, nothing is added.

If *who* is omitted, add the file mode bits represented by *permissions*, *except* for the those with corresponding bits in the file mode creation mask.

If *who* is present, add the file mode bits represented by the *permissions*.

– Take away permissions.

If *permissions* is omitted, do nothing.

If *who* is omitted, clear the file mode bits represented by *permissions*, *except* for those with corresponding bits in the file mode creation mask.

If *who* is present, clear the file mode bits represented by *permissions*.

= Assign permissions absolutely.

If *who* is omitted, clear all file mode bits; if *who* is present, clear the file mode bits represented by *who*.

If *permissions* is omitted, do nothing else.

If *who* is omitted, add the file mode bits represented by *permissions*, *except* for the those with corresponding bits in the file mode creation mask.

If *who* is present, add the file mode bits represented by *permissions*.

Unlike other symbolic operations, **=** has an absolute effect in that it resets all other bits represented by *who*. Omitting *permissions* is useful only with **=** to take away all permissions.

permission any compatible combination of the following letters:

r	read permission
w	write permission
x	execute permission
l	mandatory locking
s	user or group set-ID
t	sticky bit
u,**g**,**o**	indicate that *permission* is to be taken from the current user, group or other mode respectively.

Permissions to a file may vary depending on your user identification number (UID) or group identification number (GID). Permissions are described in three sequences each having three characters:

User	Group	Other
rwx	**rwx**	**rwx**

This example (user, group, and others all have permission to read, write, and execute a given file) demonstrates two categories for granting permissions: the access class and the permissions themselves.

The letter **s** is only meaningful with **u** or **g**, and **t** only works with **u**.

Mandatory file and record locking (**l**) refers to a file's ability to have its reading or writing permissions locked while a program is accessing that file.

In a directory which has the set-group-ID bit set (reflected as either **-----s---** or **-----l---** in the output of **'ls -ld'**), files and subdirectories are created with the group-ID of the parent directory—not that of current process.

It is not possible to permit group execution and enable a file to be locked on execution at the same time. In addition, it is not possible to turn on the set-group-ID bit and enable a file to be locked on execution at the same time. The following examples, therefore, are invalid and elicit error messages:

chmod g+x,+l *file*
chmod g+s,+l *file*

Only the owner of a file or directory (or the super-user) may change that file's or directory's mode. Only the super-user may set the sticky bit on a non-directory file. If you are not super-user, **chmod** will mask the sticky-bit but will not return an error. In order to turn on a file's set-group-ID bit, your own group ID must correspond to the file's and group execution must be set.

OPTIONS

The following options are supported:

–f Force. **chmod** will not complain if it fails to change the mode of a file.

–R Recursively descend through directory arguments, setting the mode for each file as described above. When symbolic links are encountered, the mode of the target file is changed, but no recursion takes place.

OPERANDS

The following operands are supported:

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

file A path name of a file whose file mode bits are to be modified.

EXAMPLES

Deny execute permission to everyone:

example% chmod a–x *file*

Allow only read permission to everyone:

example% chmod 444 *file*

Make a file readable and writable by the group and others:

example% chmod go+rw *file*
example% chmod 066 *file*

Cause a file to be locked during access:

example% chmod +l *file*

Allow everyone to read, write, and execute the file and turn on the set group-ID.

example% chmod a=rwx,g+s *file*
example% chmod 2777 *file*

ENVIRONMENT

See **environ**(5) for descriptions of the following environment variables that affect the execution of **chmod**: **LC_CTYPE**, **LC_MESSAGES**, and **NLSPATH**.

EXIT STATUS

The following exit values are returned:

0 Successful completion.
>0 An error occurred.

SEE ALSO

ls(1), **chmod**(2), **environ**(5)

NOTES

Absolute changes don't work for the set-group-ID bit of a directory. You must use **g+s** or **g-s**.

chmod permits you to produce useless modes so long as they are not illegal (for instance, making a text file executable). **chmod** does not check the file type to see if mandatory locking is meaningful.

If the filesystem is mounted with the *nosuid* option, *setuid* execution is not allowed.