**Mac/Linux Commands**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pwd | print working director |  |  |  |  |  |
| hostname | my computer’s network name |  |  |  |  |  |
| mkdir | make directory |  |  |  |  |  |
| cd | change directory |  |  |  |  |  |
| ls | list directory |  |  |  |  |  |
| rmdir | remove directory |  |  |  |  |  |
| pushd | push directory |  |  |  |  |  |
| popd | pop directory |  |  |  |  |  |
| cp | copy a file or directory |  |  |  |  |  |
| mv | move a file or directory |  |  |  |  |  |
| less | page through a file |  |  |  |  |  |
| cat | print the whole file |  |  |  |  |  |
| xargs | execute arguments |  |  |  |  |  |
| find | find files |  |  |  |  |  |
| grep | find things inside files |  |  |  |  |  |
| man | read a manual page |  |  |  |  |  |
| apropos | find what man page is appropriate |  |  |  |  |  |
| env | look at your environment |  |  |  |  |  |
| echo | print some arguments |  |  |  |  |  |
| export | export/set a new environment variable |  |  |  |  |  |
| exit | exit the shell |  |  |  |  |  |
| sudo | DANGER! become super user root DANGER! |  |  |  |  |  |

**Windows Commands**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| pwd | print working director |  |  |  |  |  |
| hostname | my computer’s network name |  |  |  |  |  |
| mkdir | make directory |  |  |  |  |  |
| cd | change directory |  |  |  |  |  |
| ls | list directory |  |  |  |  |  |
| rmdir | remove directory |  |  |  |  |  |
| pushd | push directory |  |  |  |  |  |
| popd | pop directory |  |  |  |  |  |
| cp | copy a file or directory |  |  |  |  |  |
| mv | move a file or directory |  |  |  |  |  |
| more | page through a file |  |  |  |  |  |
| type | print the whole file |  |  |  |  |  |
| forfiles | run a command on lots of files |  |  |  |  |  |
| dir –r | find files |  |  |  |  |  |
| select-string | find things inside files |  |  |  |  |  |
| help | read a manual page |  |  |  |  |  |
| helpctr | find what man page is appropriate |  |  |  |  |  |
| echo | print some arguments |  |  |  |  |  |
| set | export/set a new environment variable |  |  |  |  |  |
| exit | exit the shell |  |  |  |  |  |
| runas | DANGER! become super user root DANGER! |  |  |  |  |  |

# Notes on Command Line Crash Course (+Macworld: Master the command line series)

This appendix is a quick super fast course in using the command line. It is intended to be done rapidly in about a day or two, and not meant to teach you advanced shell usage.

* [Introduction](http://learnpythonthehardway.org/book/appendix-a-cli/introduction.html)
* [Exercise 1: The Setup](http://learnpythonthehardway.org/book/appendix-a-cli/ex1.html)
* [Exercise 2: Paths, Folders, Directories (pwd)](http://learnpythonthehardway.org/book/appendix-a-cli/ex2.html)
* [Exercise 3: If You Get Lost](http://learnpythonthehardway.org/book/appendix-a-cli/ex3.html)
* [Exercise 4: Make A Directory (mkdir)](http://learnpythonthehardway.org/book/appendix-a-cli/ex4.html)
* [Exercise 5: Change Directory (cd)](http://learnpythonthehardway.org/book/appendix-a-cli/ex5.html)
* [Exercise 6: List Directory (ls)](http://learnpythonthehardway.org/book/appendix-a-cli/ex6.html)
* [Exercise 7: Remove Directory (rmdir)](http://learnpythonthehardway.org/book/appendix-a-cli/ex7.html)
* [Exercise 8: Moving Around (pushd, popd)](http://learnpythonthehardway.org/book/appendix-a-cli/ex8.html)
* [Exercise 9: Making Empty Files (Touch, New-Item)](http://learnpythonthehardway.org/book/appendix-a-cli/ex9.html)
* [Exercise 10: Copy a File (cp)](http://learnpythonthehardway.org/book/appendix-a-cli/ex10.html)
* [Exercise 11: Moving a File (mv)](http://learnpythonthehardway.org/book/appendix-a-cli/ex11.html)
* [Exercise 12: View a File (less, MORE)](http://learnpythonthehardway.org/book/appendix-a-cli/ex12.html)
* [Exercise 13: Stream a File (cat)](http://learnpythonthehardway.org/book/appendix-a-cli/ex13.html)
* [Exercise 14: Removing a File (rm)](http://learnpythonthehardway.org/book/appendix-a-cli/ex14.html)
* [Exercise 15: Exiting Your Terminal (exit)](http://learnpythonthehardway.org/book/appendix-a-cli/ex15.html)
* [Next Steps](http://learnpythonthehardway.org/book/appendix-a-cli/next.html)

A very important part of learning to use the command line interface (CLI) on a computer with a graphical user interface (GUI) is figuring out how they work together. When I started using computers there was no "GUI" and you did everything with the DOS prompt (the CLI). Later, when computers became powerful enough that everyone could have graphics, it was simple for me to match CLI directories with GUI windows and folders.

Most people today, however, have no comprehension of the CLI, paths, and directories. In fact, it's very difficult to teach it to them and the only way to learn about the connection is for you to constantly work with the CLI until one day it clicks that things you do in the GUI will show up in the CLI.

Tilde (~) is a shortcut for home folder, useful for path directories. This command takes the file at the precise path you specify as the source argument, and moves it to the directory (folder) which is the destination. Note that if there’s no file there, or if you type the name incorrectly, Terminal will give you a "No such file or directory" error.

Cd change directory to move around, directory is Unix speak for folder

Cd temp/stuff/things/”AWESOME”/\_cool\_ to inner directory with one command.

\*

cd .. goes to directory above the one you’re currently in

Cd ../../../../ back to temp with one command, but not further above that.

Cd or cd~ to your "home directory" with one command.

cd / you’ll go to the root level of your startup disk.

cd – brings you back to the directory you were in before the last time you issued the cd command.

ls –lr lists all the details of items in a folder (dir –r) in windows

ls –l does the same

ls –a shows invisible files (files with . in front of their name)

rm -rf <directory name> deletes it when rmdir is unable to

mkdir make directory command

\*Can make multiple directories all at once e.g. mkdir MyDirectory1 MyDirectory2 MyDirectory3

mkdir -p <directory name> will make an entire path even if all the directories don't exist e.g. mkdir –p temp/super/awesome/cool/punch

The pushd command takes your current directory and "pushes" it into a list for later, then it changes to another directory. It's like saying, "Save where I am, then go here."

The popd command takes the last directory you pushed and "pops" it off, taking you back there.

Touch on Unix makes an empty file, it also changes the times on the file. On Windows you don't have this command, so you learned how to use the New-Item command, which does the same thing but can also make new directories.

Cp <item in existing directory> <new item will appear in same directory>

\*note if second item you type doesn’t exist already, it will join as a new item in same directory as first item in command; however if you type a folder name with no .txt it will still be an empty file created, NOT a directory

\*note cp command overwrites files that already exist so be careful when copying

Cp <item in existing directory> <directory where item will be copied>

Cp –r <directory in existing directory> <directory where directory will be copied> copies directories including all the files they contain

Cp <file to be copied> <source destination>

\*e.g. cp ~/Desktop/MyFile.rtf ~/Documents

-R is the recursive flag or option, which tells the cp command to copy every item in the folder: every sub-folder, every file and folder in every sub-folder, and so one, all the way down, to the new location.

\*When you use options with commands, this additional letter—always preceded by a hyphen (-)—tells the command to do something a bit differently.

\*Copy a directory from your Desktop to your Documents folder like this: cp -R ~/Desktop/MyFolder /Documents

\*Can also copy a file with cp and change its name. In this case, need to specify not just a destination directory, but also a name for the file: cp ~/Desktop/MyFile.rtf ~/Documents/MyFile1.rtf

Mv <file OR directory to be copied> <source destination>

\*Same syntax as cp, except it can move directories without the –r flag

\*Just as in the Finder, copying a file to a different directory won’t delete the original, whereas moving will.

\*Asterisk (\*) wildcard allows you to simplify commands for moving multiple files, e.g. cp ~/Desktop/\*.rtf ~/Documents moves all the .rtf files

\*Mv command has two functions: moving files and renaming files

\*To quickly rename, what you do is essentially move a file to the same location, but change its name. If you specify a name for its destination, the mv command changes the file’s name when it moves the file. E.g. mv ~/Desktop/MyFile.rtf ~/Desktop/MyFile-old.rtf

\*The mv command is a valuable tool for troubleshooting; use this to create a backup copy of a preference file in case you need it again.

/ at the end of your directory ensures the file is really a directory, so if the directory doesn't exist you get an error.

~/desktop to get to desktop where ~/ represents going up a level in the directory and back down another branch

Less command opens text in pager, press q to quit

\*On Unix you use the spacebar and w to go down and up. Arrow keys also work.

Cat command just spews the whole file to the screen with no paging or stopping

\*cat test.txt test2.txt just spews out each file one after another with no spacing

More command does the same as above, shows text after command, rather than open new window (this is supposed to be the Windows equivalent of the less command, but works on Unix for some reason)

Pager: When you look at a man page, you do so in Terminal through another command, called a pager; by default, this is the less command. What a pager does is allow you to view content in Terminal page by page, or line by line.

\*Return moves page down one line

\*Spacebar, the page will scroll one page (the number of lines visible in your Terminal)

\*: on bottom of page shows there is more to read

\*Depending on settings, you can use cmd+B to go back, cmd+F to go forward

\*Press q to quit the less command

rm removes files

rmdir removes directory

rm –rf recursively delete all a directory and all of its contents, because you cannot remove a directory with something in it otherwise without running into errors

\* be careful when running recursive remove on files.

Exit exits the terminal

Man command opens a manual help page about any command

Info command does the same thing

Man –k gives you a short version without opening a new pager

\*Just Google man and the name of a command and you’ll get plenty of hits

\*Apple has a documentation repository organized alphabetically (useful because a popup menu near the top of the page lets you choose an OS X version) [https://developer.apple.com/library/mac/documentation/darwin/reference/manpages/#](https://developer.apple.com/library/mac/documentation/darwin/reference/manpages/)

\*ManOpen app lets you view pages in a more attractive way than Terminal <http://www.clindberg.org/projects/ManOpen.html>

Do More

Here's the list for Unix:

* xargs
* sudo
* chmod
* chown

For Windows look up these things:

* forfiles
* runas
* attrib
* icacls

Yay! I am not a barely capable shell user.

Bash is the shell we’ve been using in Unix. Not the greatest shell but it's everywhere and has a lot of features so it's a good start. Here's a short list of links about Bash to read:

**Bash Cheat Sheet**

<http://cli.learncodethehardway.org/bash_cheat_sheet.pdf> created by [Raphael](http://freeworld.posterous.com/65140847) and CC licensed.

**Reference Manual**

<http://www.gnu.org/software/bash/manual/bashref.html>

Snippet 1:

$ ninas-air:temp blanks$ mv awesome.txt uncool.txt

$ ninas-air:temp blanks$ ls

anything iamcool.txt nothing something

iamawesome.txt neat.txt oldplace uncool.txt

$ ninas-air:temp blanks$ mv neat.txt newplace

$ ninas-air:temp blanks$ ls

anything iamcool.txt nothing something

iamawesome.txt newplace oldplace uncool.txt

$ ninas-air:temp blanks$ cd newplace

-bash: cd: newplace: Not a directory

$ ninas-air:temp blanks$ mkdir newplace

mkdir: newplace: File exists

$ ninas-air:temp blanks$ rmdir newplace

rmdir: newplace: Not a directory

$ ninas-air:temp blanks$ rm newplace

$ ninas-air:temp blanks$ mv newplace oldplace

mv: rename newplace to oldplace/newplace: No such file or directory

$ ninas-air:temp blanks$ cd oldplace

$ ninas-air:oldplace blanks$ cd ..

$ ninas-air:temp blanks$ ls

anything iamcool.txt oldplace uncool.txt

iamawesome.txt nothing something

$ ninas-air:temp blanks$ ls nothing

nothing

$ ninas-air:temp blanks$ ls something

awesome.txt

$ ninas-air:temp blanks$ ls nothing/

ls: nothing/: Not a directory

$ ninas-air:temp blanks$ ls something/

awesome.txt

$ ninas-air:temp blanks$ ls oldplace/

awesome.txt something

$ ninas-air:temp blanks$ mv iamcool.txt oldplace/

$ ninas-air:temp blanks$ ls oldplace/

awesome.txt iamcool.txt something

$ ninas-air:temp blanks$ cd oldplace

$ ninas-air:oldplace blanks$ mv iamcool.txt ~/temp

$ ninas-air:oldplace blanks$ cd ..

$ ninas-air:temp blanks$ ls

anything iamcool.txt oldplace uncool.txt

iamawesome.txt nothing something

$ ninas-air:temp blanks$ mv iamcool.txt desktop/

mv: rename iamcool.txt to desktop/: No such file or directory

$ ninas-air:temp blanks$ mv iamcool.txt ~/desktop/

$ ninas-air:temp blanks$ ls ~/desktop/

Launcher.app iamcool.txt

$ ninas-air:temp blanks$ cd ~/desktop

$ ninas-air:desktop blanks$ mv iamcool.txt ~/temp/

$ ninas-air:desktop blanks$ ls ~/temp/

anything iamcool.txt oldplace uncool.txt

iamawesome.txt nothing something

$ ninas-air:desktop blanks$ cd ~/temp

$ ninas-air:temp blanks$ pwd

/Users/blanks/temp

Snippet 2:

ninas-air:temp blanks$ less test.txt

ninas-air:temp blanks$ more test.txt

Exercise 12: View a File (less, MORE) To do this exercise you're going to do some work using the commands you know so far. You'll also need a text editor that can make plain text (.txt) files. Here's what you do: Open your text editor and type some stuff into a new file. On OSX this could be TextWrangler. On Windows this might be Notepad++. On Linux this could be gedit. Any editor will work. Save that file to your desktop and name it test.txt. In your shell use the commands you know to copy this file to your temp directory that you've been working with. Once you've done that, complete this exercise.

Snippet 3:

$ ninas-air:temp blanks$ cp -r test2.txt something

$ ninas-air:temp blanks$ ls something

awesome.txt test2.txt

$ ninas-air:temp blanks$ ls

something test2.txt

$ ninas-air:temp blanks$ rmdir test2.txt

rmdir: test2.txt: Not a directory

$ ninas-air:temp blanks$ rm test2.txt

$ ninas-air:temp blanks$ rm -rf something/

Snippet 4:

$ ninas-air:~ blanks$ help

GNU bash, version 3.2.51(1)-release (x86\_64-apple-darwin13)

These shell commands are defined internally. Type `help' to see this list.

Type `help name' to find out more about the function `name'.

Use `info bash' to find out more about the shell in general.

Use `man -k' or `info' to find out more about commands not in this list.

A star (\*) next to a name means that the command is disabled.

JOB\_SPEC [&] (( expression ))

. filename [arguments] :

[ arg... ] [[ expression ]]

alias [-p] [name[=value] ... ] bg [job\_spec ...]

bind [-lpvsPVS] [-m keymap] [-f fi break [n]

builtin [shell-builtin [arg ...]] caller [EXPR]

case WORD in [PATTERN [| PATTERN]. cd [-L|-P] [dir]

command [-pVv] command [arg ...] compgen [-abcdefgjksuv] [-o option

complete [-abcdefgjksuv] [-pr] [-o continue [n]

declare [-afFirtx] [-p] [name[=val dirs [-clpv] [+N] [-N]

disown [-h] [-ar] [jobspec ...] echo [-neE] [arg ...]

enable [-pnds] [-a] [-f filename] eval [arg ...]

exec [-cl] [-a name] file [redirec exit [n]

export [-nf] [name[=value] ...] or false

fc [-e ename] [-nlr] [first] [last fg [job\_spec]

for NAME [in WORDS ... ;] do COMMA for (( exp1; exp2; exp3 )); do COM

function NAME { COMMANDS ; } or NA getopts optstring name [arg]

hash [-lr] [-p pathname] [-dt] [na help [-s] [pattern ...]

history [-c] [-d offset] [n] or hi if COMMANDS; then COMMANDS; [ elif

jobs [-lnprs] [jobspec ...] or job kill [-s sigspec | -n signum | -si

let arg [arg ...] local name[=value] ...

logout popd [+N | -N] [-n]

printf [-v var] format [arguments] pushd [dir | +N | -N] [-n]

pwd [-LP] read [-ers] [-u fd] [-t timeout] [

readonly [-af] [name[=value] ...] return [n]

select NAME [in WORDS ... ;] do CO set [--abefhkmnptuvxBCHP] [-o opti

shift [n] shopt [-pqsu] [-o long-option] opt

source filename [arguments] suspend [-f]

test [expr] time [-p] PIPELINE

times trap [-lp] [arg signal\_spec ...]

true type [-afptP] name [name ...]

typeset [-afFirtx] [-p] name[=valu ulimit [-SHacdfilmnpqstuvx] [limit

umask [-p] [-S] [mode] unalias [-a] name [name ...]

unset [-f] [-v] [name ...] until COMMANDS; do COMMANDS; done

variables - Some variable names an wait [n]

while COMMANDS; do COMMANDS; done { COMMANDS ; }

$ ninas-air:~ blanks$ help name xargs

-bash: help: no help topics match `xargs'. Try `help help' or `man -k xargs' or `info xargs'.

$ ninas-air:~ blanks$ help help

help: help [-s] [pattern ...]

Display helpful information about builtin commands. If PATTERN is

specified, gives detailed help on all commands matching PATTERN,

otherwise a list of the builtins is printed. The -s option

restricts the output for each builtin command matching PATTERN to

a short usage synopsis.

$ ninas-air:~ blanks$ man -k xargs

xargs(1) - construct argument list(s) and execute utility

xargs(1) - construct argument list(s) and execute utility

$ ninas-air:~ blanks$ man -k sudo

sudo(8) - execute a command as another user

sudoers(5) - list of which users may execute what

visudo(8) - edit the sudoers file

sudo(8) - execute a command as another user

sudoers(5) - list of which users may execute what

visudo(8) - edit the sudoers file

$ ninas-air:~ blanks$ man -k chmod

chmod(1) - change file modes or Access Control Lists

chmod(1) - change file modes or Access Control Lists

chmod(2), fchmod(2) - change mode of file

lchmod(3) - change mode of file

$ ninas-air:~ blanks$ man -k chown

chown(8) - change file owner and group

chown(2), fchown(2), lchown(2) - change owner and group of a file

chown(8) - change file owner and group

Snippet 5:

$ ninas-air:~ blanks$ info xargs

$ ninas-air:~ blanks$ info sudo

$ ninas-air:~ blanks$ info chmod

$ ninas-air:~ blanks$ info chown

**File: \*manpages\*, Node: xargs, Up: (dir)**

XARGS(1) BSD General Commands Manual XARGS(1)

NAME

xargs -- construct argument list(s) and execute utility

SYNOPSIS

xargs [-0opt] [-E eofstr] [-I replstr [-R replacements]] [-J replstr]

[-L number] [-n number [-x]] [-P maxprocs] [-s size]

[utility [argument ...]]

DESCRIPTION

The xargs utility reads space, tab, newline and end-of-file delimited

strings from the standard input and executes utility with the strings as

arguments.

Any arguments specified on the command line are given to utility upon

each invocation, followed by some number of the arguments read from the

standard input of xargs. The utility is repeatedly executed until stan-

dard input is exhausted.

Spaces, tabs and newlines may be embedded in arguments using single

(`` ' '') or double (``"'') quotes or backslashes (``\''). Single quotes

escape all non-single quote characters, excluding newlines, up to the

matching single quote. Double quotes escape all non-double quote charac-

ters, excluding newlines, up to the matching double quote. Any single

character, including newlines, may be escaped by a backslash.

The options are as follows:

-0 Change xargs to expect NUL (``\0'') characters as separators,

instead of spaces and newlines. This is expected to be used in

concert with the -print0 function in find(1).

-E eofstr

Use eofstr as a logical EOF marker.

-I replstr

Execute utility for each input line, replacing one or more occur-

rences of replstr in up to replacements (or 5 if no -R flag is

specified) arguments to utility with the entire line of input.

The resulting arguments, after replacement is done, will not be

allowed to grow beyond 255 bytes; this is implemented by concate-

nating as much of the argument containing replstr as possible, to

the constructed arguments to utility, up to 255 bytes. The 255

byte limit does not apply to arguments to utility which do not

contain replstr, and furthermore, no replacement will be done on

utility itself. Implies -x.

-J replstr

If this option is specified, xargs will use the data read from

standard input to replace the first occurrence of replstr instead

of appending that data after all other arguments. This option

will not affect how many arguments will be read from input (-n),

or the size of the command(s) xargs will generate (-s). The

option just moves where those arguments will be placed in the

command(s) that are executed. The replstr must show up as a dis-

tinct argument to xargs. It will not be recognized if, for

instance, it is in the middle of a quoted string. Furthermore,

only the first occurrence of the replstr will be replaced. For

example, the following command will copy the list of files and

directories which start with an uppercase letter in the current

directory to destdir:

/bin/ls -1d [A-Z]\* | xargs -J % cp -rp % destdir

-L number

Call utility for every number non-empty lines read. A line end-

ing with a space continues to the next non-empty line. If EOF is

reached and fewer lines have been read than number then utility

will be called with the available lines. The -L and -n options

are mutually-exclusive; the last one given will be used.

-n number

Set the maximum number of arguments taken from standard input for

each invocation of utility. An invocation of utility will use

less than number standard input arguments if the number of bytes

accumulated (see the -s option) exceeds the specified size or

there are fewer than number arguments remaining for the last

invocation of utility. The current default value for number is

5000.

-o Reopen stdin as /dev/tty in the child process before executing

the command. This is useful if you want xargs to run an interac-

tive application.

-P maxprocs

Parallel mode: run at most maxprocs invocations of utility at

once.

-p Echo each command to be executed and ask the user whether it

should be executed. An affirmative response, `y' in the POSIX

locale, causes the command to be executed, any other response

causes it to be skipped. No commands are executed if the process

is not attached to a terminal.

-R replacements

Specify the maximum number of arguments that -I will do replace-

ment in. If replacements is negative, the number of arguments in

which to replace is unbounded.

-s size

Set the maximum number of bytes for the command line length pro-

vided to utility. The sum of the length of the utility name, the

arguments passed to utility (including NULL terminators) and the

current environment will be less than or equal to this number.

The current default value for size is ARG\_MAX - 4096.

-t Echo the command to be executed to standard error immediately

before it is executed.

-x Force xargs to terminate immediately if a command line containing

number arguments will not fit in the specified (or default) com-

mand line length.

If utility is omitted, echo(1) is used.

Undefined behavior may occur if utility reads from the standard input.

The xargs utility exits immediately (without processing any further

input) if a command line cannot be assembled, utility cannot be invoked,

an invocation of utility is terminated by a signal, or an invocation of

utility exits with a value of 255.

LEGACY DESCRIPTION

In legacy mode, the -L option treats all newlines as end-of-line, regard-

less of whether the line is empty or ends with a space. In addition, the

-L and -n options are not mutually-exclusive.

For more information about legacy mode, see compat(5).

EXIT STATUS

The xargs utility exits with a value of 0 if no error occurs. If utility

cannot be found, xargs exits with a value of 127, otherwise if utility

cannot be executed, xargs exits with a value of 126. If any other error

occurs, xargs exits with a value of 1.

SEE ALSO

echo(1), find(1), execvp(3), compat(5)

STANDARDS

The xargs utility is expected to be IEEE Std 1003.2 (``POSIX.2'') compli-

ant. The -J, -o, -P and -R options are non-standard FreeBSD extensions

which may not be available on other operating systems.

HISTORY

The xargs command appeared in PWB UNIX.

BUGS

If utility attempts to invoke another command such that the number of

arguments or the size of the environment is increased, it risks execvp(3)

failing with E2BIG.

The xargs utility does not take multibyte characters into account when

performing string comparisons for the -I and -J options, which may lead

to incorrect results in some locales.

BSD August 2, 2004 BSD

-----Info: (\*manpages\*)xargs, 160 lines --Top-----------------------------------

Welcome to Info version 4.8. Type ? for help, m for menu item.

**File: \*manpages\*, Node: sudo, Up: (dir)**

SUDO(8) MAINTENANCE COMMANDS SUDO(8)

NAME

sudo - execute a command as another user

SYNOPSIS

sudo -h | -K | -k | -L | -V

sudo -v [-AknS] [-g group name|#gid] [-p prompt] [-u username|#uid]

sudo -l[l] [-AknS] [-g group name|#gid] [-p prompt] [-U user name]

[-u user name|#uid] [command]

sudo [-AbEHnPS] [-C fd] [-g group name|#gid] [-p prompt]

[-u user name|#uid] [VAR=value] [-i | -s] [command]

sudoedit [-AnS] [-C fd] [-g group name|#gid] [-p prompt]

[-u user name|#uid] file ...

DESCRIPTION

-----Info: (\*sudo allows a permitted user to execute a command as the superuser or

another user, as specified in the sudoers file. The real and effective

uid and gid are set to match those of the target user as specified in

the passwd file and the group vector is initialized based on the group

file (unless the -P option was specified). If the invoking user is

root or if the target user is the same as the invoking user, no

password is required. Otherwise, sudo requires that users authenticate

themselves with a password by default (NOTE: in the default

configuration this is the user's password, not the root password).

Once a user has been authenticated, a time stamp is updated and the

user may then use sudo without a password for a short period of time (5

minutes unless overridden in sudoers).

When invoked as sudoedit, the -e option (described below), is implied.

sudo determines who is an authorized user by consulting the file

/etc/sudoers. By running sudo with the -v option, a user can update

the time stamp without running a command. If a password is required,

sudo will exit if the user's password is not entered within a

configurable time limit. The default password prompt timeout is 5

minutes.

manpageIf a user who is not listed in the sudoers file tries to run a command

via sudo, mail is sent to the proper authorities, as defined at

configure time or in the sudoers file (defaults to root). Note that

the mail will not be sent if an unauthorized user tries to run sudo

with the -l or -v option. This allows users to determine for

themselves whether or not they are allowed to use sudo.

If sudo is run by root and the SUDO\_USER environment variable is set,

sudo will use this value to determine who the actual user is. This can

be used by a user to log commands through sudo even when a root shell

has been invoked. It also allows the -e option to remain useful even

when being run via a sudo-run script or program. Note however, that

the sudoers lookup is still done for root, not the user specified by

SUDO\_USER.

sudo can log both successful and unsuccessful attempts (as well as

errors) to syslog(3), a log file, or both. By default sudo will log

via syslog(3) but this is changeable at configure time or via the

sudoers file.

OPTIONS

sudo accepts the following command line options:

-A Normally, if sudo requires a password, it will read it from

the current terminal. If the -A (askpass) option is

specified, a (possibly graphical) helper program is

executed to read the user's password and output the

password to the standard output. If the SUDO\_ASKPASS

environment variable is set, it specifies the path to the

helper program. Otherwise, the value specified by the

askpass option in sudoers(5) is used.

-b The -b (background) option tells sudo to run the given

command in the background. Note that if you use the -b

option you cannot use shell job control to manipulate the

process.

-C fd Normally, sudo will close all open file descriptors other

than standard input, standard output and standard error.

The -C (close from) option allows the user to specify a

s\*)sudo, 520 lines starting point above the standard error (file descriptor

three). Values less than three are not permitted. This

option is only available if the administrator has enabled

the closefrom\_override option in sudoers(5).

-E The -E (preserve environment) option will override the

env\_reset option in sudoers(5)). It is only available when

either the matching command has the SETENV tag or the

setenv option is set in sudoers(5).

-e The -e (edit) option indicates that, instead of running a

command, the user wishes to edit one or more files. In

lieu of a command, the string "sudoedit" is used when

consulting the sudoers file. If the user is authorized by

sudoers the following steps are taken:

1. Temporary copies are made of the files to be edited

with the owner set to the invoking user.

2. The editor specified by the SUDO\_EDITOR, VISUAL or

EDITOR environment variables is run to edit the

temporary files. If none of SUDO\_EDITOR, VISUAL or

EDITOR are set, the first program listed in the editor

sudoers variable is used.

3. If they have been modified, the temporary files are

copied back to their original location and the

temporary versions are removed.

If the specified file does not exist, it will be created.

Note that unlike most commands run by sudo, the editor is

run with the invoking user's environment unmodified. If,

for some reason, sudo is unable to update a file with its

edited version, the user will receive a warning and the

edited copy will remain in a temporary file.

-g group Normally, sudo sets the primary group to the one specified

by the passwd database for the user the command is being

run as (by default, root). The -g (group) option causes

sudo to run the specified command with the primary group

set to group. To specify a gid instead of a group name,--Top------------------------------------ use #gid. When running commands as a gid, many shells

require that the '#' be escaped with a backslash ('\'). If

no -u option is specified, the command will be run as the

invoking user (not root). In either case, the primary

group will be set to group.

-H The -H (HOME) option sets the HOME environment variable to

the homedir of the target user (root by default) as

specified in passwd(5). The default handling of the HOME

environment variable depends on sudoers(5) settings. By

default, sudo will set HOME if env\_reset or always\_set\_home

are set, or if set\_home is set and the -s option is

specified on the command line.

-h The -h (help) option causes sudo to print a usage message

and exit.

-i [command]

The -i (simulate initial login) option runs the shell

specified in the passwd(5) entry of the target user as a

login shell. This means that login-specific resource files

such as .profile or .login will be read by the shell. If a

command is specified, it is passed to the shell for

execution. Otherwise, an interactive shell is executed.

sudo attempts to change to that user's home directory

before running the shell. It also initializes the

environment, leaving DISPLAY and TERM unchanged, setting

HOME, MAIL, SHELL, USER, LOGNAME, and PATH, as well as the

contents of /etc/environment on Linux and AIX systems. All

other environment variables are removed.

-K The -K (sure kill) option is like -k except that it removes

the user's time stamp entirely and may not be used in

conjunction with a command or other option. This option

does not require a password.

-k When used by itself, the -k (kill) option to sudo

invalidates the user's time stamp by setting the time on it

to the Epoch. The next time sudo is run a password will be

required. This option does not require a password and was

added to allow a user to revoke sudo permissions from a

.logout file.

When used in conjunction with a command or an option that

may require a password, the -k option will cause sudo to

ignore the user's time stamp file. As a result, sudo will

prompt for a password (if one is required by sudoers) and

will not update the user's time stamp file.

-L The -L (list defaults) option will list the parameters that

may be set in a Defaults line along with a short

description for each. This option will be removed from a

future version of sudo.

-l[l] [command]

If no command is specified, the -l (list) option will list

the allowed (and forbidden) commands for the invoking user

(or the user specified by the -U option) on the current

host. If a command is specified and is permitted by

sudoers, the fully-qualified path to the command is

displayed along with any command line arguments. If

command is specified but not allowed, sudo will exit with a

status value of 1. If the -l option is specified with an l

argument (i.e. -ll), or if -l is specified multiple times,

a longer list format is used.

-n The -n (non-interactive) option prevents sudo from

prompting the user for a password. If a password is

required for the command to run, sudo will display an error

messages and exit.

-P The -P (preserve group vector) option causes sudo to

preserve the invoking user's group vector unaltered. By

default, sudo will initialize the group vector to the list

of groups the target user is in. The real and effective

group IDs, however, are still set to match the target user.

-p prompt The -p (prompt) option allows you to override the default

password prompt and use a custom one. The following

percent (`%') escapes are supported:

%H expanded to the local host name including the domain

%H expanded to the local host name including the domain

name (on if the machine's host name is fully qualified

or the fqdn sudoers option is set)

%h expanded to the local host name without the domain name

%p expanded to the user whose password is being asked for

(respects the rootpw, targetpw and runaspw flags in

sudoers)

%U expanded to the login name of the user the command will

be run as (defaults to root)

%u expanded to the invoking user's login name

%% two consecutive % characters are collapsed into a

single % character

The prompt specified by the -p option will override the

system password prompt on systems that support PAM unless

the passprompt\_override flag is disabled in sudoers.

-S The -S (stdin) option causes sudo to read the password from

the standard input instead of the terminal device. The

password must be followed by a newline character.

-s [command]

The -s (shell) option runs the shell specified by the SHELL

environment variable if it is set or the shell as specified

in passwd(5). If a command is specified, it is passed to

the shell for execution. Otherwise, an interactive shell

is executed.

-U user The -U (other user) option is used in conjunction with the

-l option to specify the user whose privileges should be

listed. Only root or a user with sudo ALL on the current

host may use this option.

-u user The -u (user) option causes sudo to run the specified

command as a user other than root. To specify a uid

instead of a user name, use #uid. When running commands as

a uid, many shells require that the '#' be escaped with a

backslash ('\'). Note that if the targetpw Defaults option

is set (see sudoers(5)) it is not possible to run commands

with a uid not listed in the password database.

-V The -V (version) option causes sudo to print the version

number and exit. If the invoking user is already root the

-V option will print out a list of the defaults sudo was

compiled with as well as the machine's local network

addresses.

-v If given the -v (validate) option, sudo will update the

user's time stamp, prompting for the user's password if

necessary. This extends the sudo timeout for another 5

minutes (or whatever the timeout is set to in sudoers) but

does not run a command.

-- The -- option indicates that sudo should stop processing

command line arguments.

Environment variables to be set for the command may also be passed on

the command line in the form of VAR=value, e.g.

LD\_LIBRARY\_PATH=/usr/local/pkg/lib. Variables passed on the command

line are subject to the same restrictions as normal environment

variables with one important exception. If the setenv option is set in

sudoers, the command to be run has the SETENV tag set or the command

matched is ALL, the user may set variables that would overwise be

forbidden. See sudoers(5) for more information.

RETURN VALUES

Upon successful execution of a program, the exit status from sudo will

simply be the exit status of the program that was executed.

Otherwise, sudo quits with an exit value of 1 if there is a

configuration/permission problem or if sudo cannot execute the given

command. In the latter case the error string is printed to stderr. If

sudo cannot stat(2) one or more entries in the user's PATH an error is

printed on stderr. (If the directory does not exist or if it is not

really a directory, the entry is ignored and no error is printed.)

This should not happen under normal circumstances. The most common

reason for stat(2) to return "permission denied" is if you are running

an automounter and one of the directories in your PATH is on a machine

that is currently unreachable.

SECURITY NOTES

sudo tries to be safe when executing external commands.

There are two distinct ways to deal with environment variables. By

default, the env\_reset sudoers option is enabled. This causes commands

to be executed with a minimal environment containing TERM, PATH, HOME,

SHELL, LOGNAME, USER and USERNAME in addition to variables from the

invoking process permitted by the env\_check and env\_keep sudoers

options. There is effectively a whitelist for environment variables.

If, however, the env\_reset option is disabled in sudoers, any variables

not explicitly denied by the env\_check and env\_delete options are

inherited from the invoking process. In this case, env\_check and

env\_delete behave like a blacklist. Since it is not possible to

blacklist all potentially dangerous environment variables, use of the

default env\_reset behavior is encouraged.

In all cases, environment variables with a value beginning with () are

removed as they could be interpreted as bash functions. The list of

environment variables that sudo allows or denies is contained in the

output of sudo -V when run as root. This list reflects the built-in

defaults, which may be overridden in sudoers.

On Mac OS X, sudoers has been configured to only whitelist a small set

of environment variables by default. See the sudoers file for more

information.

Note that the dynamic linker on most operating systems will remove

variables that can control dynamic linking from the environment of

setuid executables, including sudo. Depending on the operating system

this may include \_RLD\*, DYLD\_\*, LD\_\*, LDR\_\*, LIBPATH, SHLIB\_PATH, and

others. These type of variables are removed from the environment

before sudo even begins execution and, as such, it is not possible for

sudo to preserve them.

To prevent command spoofing, sudo checks "." and "" (both denoting

current directory) last when searching for a command in the user's PATH

(if one or both are in the PATH). Note, however, that the actual PATH

environment variable is not modified and is passed unchanged to the

program that sudo executes.

sudo will check the ownership of its time stamp directory (/var/db/sudo

by default) and ignore the directory's contents if it is not owned by

root or if it is writable by a user other than root. On systems that

allow non-root users to give away files via chown(2), if the time stamp

directory is located in a directory writable by anyone (e.g., /tmp), it

is possible for a user to create the time stamp directory before sudo

is run. However, because sudo checks the ownership and mode of the

directory and its contents, the only damage that can be done is to

"hide" files by putting them in the time stamp dir. This is unlikely

to happen since once the time stamp dir is owned by root and

inaccessible by any other user, the user placing files there would be

unable to get them back out. To get around this issue you can use a

directory that is not world-writable for the time stamps (/var/adm/sudo

for instance) or create /var/db/sudo with the appropriate owner (root)

and permissions (0700) in the system startup files.

sudo will not honor time stamps set far in the future. Timestamps with

a date greater than current\_time + 2 \* TIMEOUT will be ignored and sudo

will log and complain. This is done to keep a user from creating

his/her own time stamp with a bogus date on systems that allow users to

give away files.

On systems where the boot time is available, sudo will also not honor

time stamps from before the machine booted.

Since time stamp files live in the file system, they can outlive a

user's login session. As a result, a user may be able to login, run a

command with sudo after authenticating, logout, login again, and run

sudo without authenticating so long as the time stamp file's

modification time is within 5 minutes (or whatever the timeout is set

to in sudoers). When the tty\_tickets option is enabled in sudoers, the

time stamp has per-tty granularity but still may outlive the user's

session. On Linux systems where the devpts filesystem is used, Solaris

systems with the devices filesystem, as well as other systems that

utilize a devfs filesystem that monotonically increase the inode number

of devices as they are created (such as Mac OS X), sudo is able to

determine when a tty-based time stamp file is stale and will ignore it.

Administrators should not rely on this feature as it is not universally

available.

Please note that sudo will normally only log the command it explicitly

runs. If a user runs a command such as sudo su or sudo sh, subsequent

commands run from that shell will not be logged, nor will sudo's access

control affect them. The same is true for commands that offer shell

escapes (including most editors). Because of this, care must be taken

when giving users access to commands via sudo to verify that the

command does not inadvertently give the user an effective root shell.

For more information, please see the PREVENTING SHELL ESCAPES section

in sudoers(5).

ENVIRONMENT

sudo utilizes the following environment variables:

EDITOR Default editor to use in -e (sudoedit) mode if neither

SUDO\_EDITOR nor VISUAL is set

MAIL In -i mode or when env\_reset is enabled in sudoers, set

to the mail spool of the target user

HOME Set to the home directory of the target user if -i or

-H are specified, env\_reset or always\_set\_home are set

in sudoers, or when the -s option is specified and

set\_home is set in sudoers

PATH Set to a sane value if the secure\_path sudoers option

is set.

SHELL Used to determine shell to run with -s option

SUDO\_ASKPASS Specifies the path to a helper program used to read the

password if no terminal is available or if the -A

option is specified.

SUDO\_COMMAND Set to the command run by sudo

SUDO\_EDITOR Default editor to use in -e (sudoedit) mode

SUDO\_GID Set to the group ID of the user who invoked sudo

SUDO\_PROMPT Used as the default password prompt

SUDO\_PS1 If set, PS1 will be set to its value for the program

being run

SUDO\_UID Set to the user ID of the user who invoked sudo

SUDO\_USER Set to the login of the user who invoked sudo

USER Set to the target user (root unless the -u option is

specified)

VISUAL Default editor to use in -e (sudoedit) mode if

SUDO\_EDITOR is not set

FILES

/etc/sudoers List of who can run what

/var/db/sudo Directory containing time stamps

/etc/environment Initial environment for -i mode on Linux and

AIX

EXAMPLES

Note: the following examples assume suitable sudoers(5) entries.

To get a file listing of an unreadable directory:

$ sudo ls /usr/local/protected

To list the home directory of user yaz on a machine where the file

system holding ~yaz is not exported as root:

$ sudo -u yaz ls ~yaz

To edit the index.html file as user www:

$ sudo -u www vi ~www/htdocs/index.html

To view system logs only accessible to root and users in the adm group:

$ sudo -g adm view /var/log/syslog

To run an editor as jim with a different primary group:

$ sudo -u jim -g audio vi ~jim/sound.txt

To shutdown a machine:

$ sudo shutdown -r +15 "quick reboot"

To make a usage listing of the directories in the /home partition.

Note that this runs the commands in a sub-shell to make the cd and file

redirection work.

$ sudo sh -c "cd /home ; du -s \* | sort -rn > USAGE"

SEE ALSO

grep(1), su(1), stat(2), passwd(5), sudoers(5), visudo(8)

AUTHORS

Many people have worked on sudo over the years; this version consists

of code written primarily by:

Todd C. Miller

See the HISTORY file in the sudo distribution or visit

http://www.sudo.ws/sudo/history.html for a short history of sudo.

CAVEATS

There is no easy way to prevent a user from gaining a root shell if

that user is allowed to run arbitrary commands via sudo. Also, many

programs (such as editors) allow the user to run commands via shell

escapes, thus avoiding sudo's checks. However, on most systems it is

possible to prevent shell escapes with sudo's noexec functionality.

See the sudoers(5) manual for details.

It is not meaningful to run the cd command directly via sudo, e.g.,

$ sudo cd /usr/local/protected

since when the command exits the parent process (your shell) will still

be the same. Please see the EXAMPLES section for more information.

If users have sudo ALL there is nothing to prevent them from creating

their own program that gives them a root shell regardless of any '!'

elements in the user specification.

Running shell scripts via sudo can expose the same kernel bugs that

make setuid shell scripts unsafe on some operating systems (if your OS

has a /dev/fd/ directory, setuid shell scripts are generally safe).

BUGS

If you feel you have found a bug in sudo, please submit a bug report at

http://www.sudo.ws/sudo/bugs/

SUPPORT

Limited free support is available via the sudo-users mailing list, see

http://www.sudo.ws/mailman/listinfo/sudo-users to subscribe or search

the archives.

DISCLAIMER

sudo is provided ``AS IS'' and any express or implied warranties,

including, but not limited to, the implied warranties of

merchantability and fitness for a particular purpose are disclaimed.

See the LICENSE file distributed with sudo or

http://www.sudo.ws/sudo/license.html for complete details.

1.7.4 July 19, 2010 SUDO(8)

Welcome to Info version 4.8. Type ? for help, m for menu item.

**File: \*manpages\*, Node: chmod, Up: (dir)**

CHMOD(1) BSD General Commands Manual CHMOD(1)

NAME

chmod -- change file modes or Access Control Lists

SYNOPSIS

chmod [-fv] [-R [-H | -L | -P]] mode file ...

chmod [-fv] [-R [-H | -L | -P]] [-a | +a | =a] ACE file ...

chmod [-fhv] [-R [-H | -L | -P]] [-E] file ...

chmod [-fhv] [-R [-H | -L | -P]] [-C] file ...

chmod [-fhv] [-R [-H | -L | -P]] [-N] file ...

DESCRIPTION

The chmod utility modifies the file mode bits of the listed files as

specified by the mode operand. It may also be used to modify the Access

Control Lists (ACLs) associated with the listed files.

The generic options are as follows:

-f Do not display a diagnostic message if chmod could not modify the

mode for file.

-H If the -R option is specified, symbolic links on the command line

are followed. (Symbolic links encountered in the tree traversal

are not followed by default.)

-h If the file is a symbolic link, change the mode of the link

itself rather than the file that the link points to.

-L If the -R option is specified, all symbolic links are followed.

-P If the -R option is specified, no symbolic links are followed.

This is the default.

-R Change the modes of the file hierarchies rooted in the files

instead of just the files themselves.

-v Cause chmod to be verbose, showing filenames as the mode is modi-

fied. If the -v flag is specified more than once, the old and

new modes of the file will also be printed, in both octal and

symbolic notation.

The -H, -L and -P options are ignored unless the -R option is specified.

In addition, these options override each other and the command's actions

are determined by the last one specified.

Only the owner of a file or the super-user is permitted to change the

mode of a file.

DIAGNOSTICS

The chmod utility exits 0 on success, and >0 if an error occurs.

MODES

Modes may be absolute or symbolic. An absolute mode is an octal number

constructed from the sum of one or more of the following values:

4000 (the set-user-ID-on-execution bit) Executable files with

this bit set will run with effective uid set to the uid of

the file owner. Directories with the set-user-id bit set

will force all files and sub-directories created in them to

be owned by the directory owner and not by the uid of the

creating process, if the underlying file system supports

this feature: see chmod(2) and the suiddir option to

mount(8).

2000 (the set-group-ID-on-execution bit) Executable files with

this bit set will run with effective gid set to the gid of

the file owner.

1000 (the sticky bit) See chmod(2) and sticky(8).

0400 Allow read by owner.

0200 Allow write by owner.

0100 For files, allow execution by owner. For directories,

allow the owner to search in the directory.

0040 Allow read by group members.

0020 Allow write by group members.

0010 For files, allow execution by group members. For directo-

ries, allow group members to search in the directory.

0004 Allow read by others.

0002 Allow write by others.

0001 For files, allow execution by others. For directories

allow others to search in the directory.

For example, the absolute mode that permits read, write and execute by

the owner, read and execute by group members, read and execute by others,

and no set-uid or set-gid behaviour is 755 (400+200+100+040+010+004+001).

The symbolic mode is described by the following grammar:

mode ::= clause [, clause ...]

clause ::= [who ...] [action ...] action

action ::= op [perm ...]

who ::= a | u | g | o

op ::= + | - | =

perm ::= r | s | t | w | x | X | u | g | o

The who symbols ``u'', ``g'', and ``o'' specify the user, group, and

other parts of the mode bits, respectively. The who symbol ``a'' is

equivalent to ``ugo''.

The perm symbols represent the portions of the mode bits as follows:

r The read bits.

s The set-user-ID-on-execution and set-group-ID-on-execution

bits.

t The sticky bit.

w The write bits.

x The execute/search bits.

X The execute/search bits if the file is a directory or any

of the execute/search bits are set in the original (unmodi-

fied) mode. Operations with the perm symbol ``X'' are only

meaningful in conjunction with the op symbol ``+'', and are

ignored in all other cases.

u The user permission bits in the original mode of the file.

g The group permission bits in the original mode of the file.

o The other permission bits in the original mode of the file.

The op symbols represent the operation performed, as follows:

+ If no value is supplied for perm, the ``+'' operation has no

effect. If no value is supplied for who, each permission bit spec-

ified in perm, for which the corresponding bit in the file mode

creation mask is clear, is set. Otherwise, the mode bits repre-

sented by the specified who and perm values are set.

- If no value is supplied for perm, the ``-'' operation has no

effect. If no value is supplied for who, each permission bit spec-

ified in perm, for which the corresponding bit in the file mode

creation mask is clear, is cleared. Otherwise, the mode bits rep-

resented by the specified who and perm values are cleared.

= The mode bits specified by the who value are cleared, or, if no who

value is specified, the owner, group and other mode bits are

cleared. Then, if no value is supplied for who, each permission

bit specified in perm, for which the corresponding bit in the file

mode creation mask is clear, is set. Otherwise, the mode bits rep-

resented by the specified who and perm values are set.

Each clause specifies one or more operations to be performed on the mode

bits, and each operation is applied to the mode bits in the order speci-

fied.

Operations upon the other permissions only (specified by the symbol ``o''

by itself), in combination with the perm symbols ``s'' or ``t'', are

ignored.

EXAMPLES OF VALID MODES

644 make a file readable by anyone and writable by the owner

only.

go-w deny write permission to group and others.

=rw,+X set the read and write permissions to the usual defaults,

but retain any execute permissions that are currently set.

+X make a directory or file searchable/executable by everyone

if it is already searchable/executable by anyone.

755

u=rwx,go=rx

u=rwx,go=u-w make a file readable/executable by everyone and writable by

the owner only.

go= clear all mode bits for group and others.

g=u-w set the group bits equal to the user bits, but clear the

group write bit.

ACL MANIPULATION OPTIONS

ACLs are manipulated using extensions to the symbolic mode grammar. Each

file has one ACL, containing an ordered list of entries. Each entry

refers to a user or group, and grants or denies a set of permissions. In

cases where a user and a group exist with the same name, the user/group

name can be prefixed with "user:" or "group:" in order to specify the

type of name.

If the user or group name contains spaces you can use ':' as the delim-

iter between name and permission.

The following permissions are applicable to all filesystem objects:

delete Delete the item. Deletion may be granted by either this

permission on an object or the delete\_child right on the

containing directory.

readattr

Read an objects basic attributes. This is implicitly

granted if the object can be looked up and not explicitly

denied.

writeattr

Write an object's basic attributes.

readextattr

Read extended attributes.

writeextattr

Write extended attributes.

readsecurity

Read an object's extended security information (ACL).

writesecurity

Write an object's security information (ownership, mode,

ACL).

chown Change an object's ownership.

The following permissions are applicable to directories:

list List entries.

search Look up files by name.

add\_file

Add a file.

add\_subdirectory

Add a subdirectory.

delete\_child

Delete a contained object. See the file delete permission

above.

The following permissions are applicable to non-directory filesystem

objects:

read Open for reading.

write Open for writing.

append Open for writing, but in a fashion that only allows writes

into areas of the file not previously written.

execute

Execute the file as a script or program.

ACL inheritance is controlled with the following permissions words, which

may only be applied to directories:

file\_inherit

Inherit to files.

directory\_inherit

Inherit to directories.

limit\_inherit

This flag is only relevant to entries inherited by subdi-

rectories; it causes the directory\_inherit flag to be

cleared in the entry that is inherited, preventing further

nested subdirectories from also inheriting the entry.

only\_inherit

The entry is inherited by created items but not considered

when processing the ACL.

The ACL manipulation options are as follows:

+a The +a mode parses a new ACL entry from the next argument on the

commandline and inserts it into the canonical location in the

ACL. If the supplied entry refers to an identity already listed,

the two entries are combined.

Examples

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

# chmod +a "admin allow write" file1

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: admin allow write

# chmod +a "guest deny read" file1

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: guest deny read

2: admin allow write

# chmod +a "admin allow delete" file1

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: guest deny read

2: admin allow write,delete

# chmod +a "User 1:allow:read" file

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: guest deny read

2: User 1 allow read

3: admin allow write,delete

The +a mode strives to maintain correct canonical form for the

ACL.

local deny

local allow

inherited deny

inherited allow

By default, chmod adds entries to the top of the local deny and

local allow lists. Inherited entries are added by using the +ai

mode.

Examples

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: guest deny read

2: admin allow write,delete

3: juser inherited deny delete

4: admin inherited allow delete

5: backup inherited deny read

6: admin inherited allow write-security

# chmod +ai "others allow read" file1

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: guest deny read

2: admin allow write,delete

3: juser inherited deny delete

4: others inherited allow read

5: admin inherited allow delete

6: backup inherited deny read

7: admin inherited allow write-security

+a# When a specific ordering is required, the exact location at which

an entry will be inserted is specified with the +a# mode.

Examples

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: guest deny read

2: admin allow write

# chmod +a# 2 "others deny read" file1

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: guest deny read

2: others deny read

3: admin allow write

The +ai# mode may be used to insert inherited entries at a spe-

cific location. Note that these modes allow non-canonical ACL

ordering to be constructed.

-a The -a mode is used to delete ACL entries. All entries exactly

matching the supplied entry will be deleted. If the entry lists a

subset of rights granted by an entry, only the rights listed are

removed. Entries may also be deleted by index using the -a# mode.

Examples

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: guest deny read

2: admin allow write,delete

# chmod -a# 1 file1

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: admin allow write,delete

# chmod -a "admin allow write" file1

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: admin allow delete

Inheritance is not considered when processing the -a mode; rights

and entries will be removed regardless of their inherited state.

If the user or group name contains spaces you can use ':' as the

delimiter

Example

# chmod +a "User 1:allow:read" file

=a# Individual entries are rewritten using the =a# mode.

Examples

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: admin allow delete

# chmod =a# 1 "admin allow write,chown"

# ls -le

-rw-r--r--+ 1 juser wheel 0 Apr 28 14:06 file1

owner: juser

1: admin allow write,chown

This mode may not be used to add new entries.

-E Reads the ACL information from stdin, as a sequential list of

ACEs, separated by newlines. If the information parses cor-

rectly, the existing information is replaced.

-C Returns false if any of the named files have ACLs in non-canoni-

cal order.

-i Removes the 'inherited' bit from all entries in the named file(s)

ACLs.

-I Removes all inherited entries from the named file(s) ACL(s).

-N Removes the ACL from the named file(s).

COMPATIBILITY

The -v option is non-standard and its use in scripts is not recommended.

SEE ALSO

chflags(1), fsaclctl(1), install(1), chmod(2), stat(2), umask(2), fts(3),

setmode(3), symlink(7), chown(8), mount(8), sticky(8)

STANDARDS

The chmod utility is expected to be IEEE Std 1003.2 (``POSIX.2'') compat-

ible with the exception of the perm symbol ``t'' which is not included in

that standard.

HISTORY

A chmod command appeared in Version 1 AT&T UNIX.

BSD July 08, 2004 BSD

-----Info: (\*manpages\*)chmod, 401 lines --Top-----------------------------------

Welcome to Info version 4.8. Type ? for help, m for menu item.

**File: \*manpages\*, Node: chown, Up: (dir)**

CHOWN(8) BSD System Manager's Manual CHOWN(8)

NAME

chown -- change file owner and group

SYNOPSIS

chown [-fhv] [-R [-H | -L | -P]] owner[:group] file ...

chown [-fhv] [-R [-H | -L | -P]] :group file ...

DESCRIPTION

The chown utility changes the user ID and/or the group ID of the speci-

fied files. Symbolic links named by arguments are silently left

unchanged unless -h is used.

The options are as follows:

-f Don't report any failure to change file owner or group, nor mod-

ify the exit status to reflect such failures.

-H If the -R option is specified, symbolic links on the command line

are followed. (Symbolic links encountered in the tree traversal

are not followed.)

-h If the file is a symbolic link, change the user ID and/or the

group ID of the link itself.

-L If the -R option is specified, all symbolic links are followed.

-P If the -R option is specified, no symbolic links are followed.

Instead, the user and/or group ID of the link itself are modi-

fied. This is the default. Use -h to change the user ID and/or

the group of symbolic links.

-R Change the user ID and/or the group ID for the file hierarchies

rooted in the files instead of just the files themselves.

-v Cause chown to be verbose, showing files as the owner is modi-

fied.

The -H, -L and -P options are ignored unless the -R option is specified.

In addition, these options override each other and the command's actions

are determined by the last one specified.

The owner and group operands are both optional; however, at least one

must be specified. If the group operand is specified, it must be pre-

ceded by a colon (``:'') character.

The owner may be either a numeric user ID or a user name. If a user name

is also a numeric user ID, the operand is used as a user name. The group

may be either a numeric group ID or a group name. If a group name is

also a numeric group ID, the operand is used as a group name.

For obvious security reasons, the ownership of a file may only be altered

by a super-user. Similarly, only a member of a group can change a file's

group ID to that group.

DIAGNOSTICS

The chown utility exits 0 on success, and >0 if an error occurs.

COMPATIBILITY

Previous versions of the chown utility used the dot (``.'') character to

distinguish the group name. This has been changed to be a colon (``:'')

character, so that user and group names may contain the dot character.

On previous versions of this system, symbolic links did not have owners.

The -v option is non-standard and its use in scripts is not recommended.

LEGACY DESCRIPTION

In legacy mode, the -R and -RP options do not change the user ID or the

group ID of symbolic links.

SEE ALSO

chgrp(1), find(1), chown(2), fts(3), compat(5), symlink(7)

STANDARDS

The chown utility is expected to be IEEE Std 1003.2 (``POSIX.2'') compli-

ant.

HISTORY

A chown utility appeared in Version 1 AT&T UNIX.

BSD March 31, 1994 BSD

-----Info: (\*manpages\*)chown, 86 lines --Top------------------------------------

Welcome to Info version 4.8. Type ? for help, m for menu item.