



Focus on Linux

Linux / Unix Command: *zip*

[Command Library](#)

NAME

zip, zipcloak, zipnote, zipsplit - package and compress (archive) files

SYNOPSIS

```
zip [-aABcdDeEfFghjklLmoqRSTuvVwXyz!@$] [-b path] [-n suffixes] [-t mmdyyy] [-tt mmdyyy]
[ zipfile [ file1 file2 ...]] [-xi list]
```

```
zipcloak [-dhL] [-b path] zipfile
```

```
zipnote [-hwL] [-b path] zipfile
```

```
zipsplit [-hiLpst] [-n size] [-b path] zipfile
```

DESCRIPTION

zip is a compression and file packaging utility for Unix, VMS, MSDOS, OS/2, Windows NT, Minix, Atari and Macintosh, Amiga and Acorn RISC OS.

It is analogous to a combination of the UNIX commands [tar\(1\)](#) and [compress\(1\)](#) and is compatible with PKZIP (Phil Katz's ZIP for MSDOS systems).

A companion program ([unzip\(1L\)](#)), unpacks *zip* archives. The *zip* and [unzip\(1L\)](#) programs can work with archives produced by PKZIP, and PKZIP and PKUNZIP can work with archives produced by *zip*. *zip* version 2.3 is compatible with PKZIP 2.04. Note that PKUNZIP 1.10 cannot extract files produced by PKZIP 2.04 or *zip* 2.3. You must use PKUNZIP 2.04g or *unzip* 5.0p1 (or later versions) to extract them.

For a brief help on *zip* and *unzip*, run each without specifying any parameters on the command line.

The program is useful for packaging a set of files for distribution; for archiving files; and for saving disk space by temporarily compressing unused files or directories.

The *zip* program puts one or more compressed files into a single *zip* archive, along with information about the files (name, path, date, time of last modification, protection, and check information to verify file integrity). An entire directory structure can be packed into a *zip* archive with a single command. Compression ratios of 2:1 to 3:1 are common for text files. *zip* has one compression method (deflation) and can also store files without compression. *zip* automatically chooses the better of the two for each file to be compressed.

When given the name of an existing *zip* archive, *zip* will replace identically named entries in the *zip* archive or add entries for new names. For example, if *foo.zip* exists and contains *foo/file1* and *foo/file2*, and the directory *foo* contains the files *foo/file1* and *foo/file3*, then:

```
zip -r foo foo
```

will replace *foo/file1* in *foo.zip* and add *foo/file3* to *foo.zip*. After this, *foo.zip* contains *foo/file1*, *foo/file2*, and *foo/file3*, with *foo/file2* unchanged from before.

If the file list is specified as `-@`, [Not on MacOS] *zip* takes the list of input files from standard input. Under UNIX, this option can be used to powerful effect in conjunction with the [find\(1\)](#) command. For example, to archive all the C source files in the current directory and its subdirectories:

```
find . -name "*.c" -print | zip source -@
```

Most Popular

- [Linux Commands - Command Library - Man Pages](#)
- [find - Linux Command - Unix Command](#)
- [Linux Commands and Shell Commands Libraries](#)
- [Shell Commands Libraries - Linux Unix](#)
- [Linux/Unix/Computing Glossary](#)

Related Topics

- [PC Hardware / Reviews](#)
- [Focus on Windows](#)
- [Focus on Macs](#)
- [Focus on Unix](#)
- [Computer Certification](#)

(note that the pattern must be quoted to keep the shell from expanding it). *zip* will also accept a single dash ("-") as the zip file name, in which case it will write the zip file to standard output, allowing the output to be piped to another program. For example:

```
zip -r - . | dd of=/dev/nrst0 obs=16k
```

would write the zip output directly to a tape with the specified block size for the purpose of backing up the current directory.

zip also accepts a single dash ("-") as the name of a file to be compressed, in which case it will read the file from standard input, allowing zip to take input from another program. For example:

```
tar cf - . | zip backup -
```

would compress the output of the tar command for the purpose of backing up the current directory. This generally produces better compression than the previous example using the -r option, because *zip* can take advantage of redundancy between files. The backup can be restored using the command

```
unzip -p backup | tar xf -
```

When no zip file name is given and stdout is not a terminal, *zip* acts as a filter, compressing standard input to standard output. For example,

```
tar cf - . | zip | dd of=/dev/nrst0 obs=16k
```

is equivalent to

```
tar cf - . | zip - - | dd of=/dev/nrst0 obs=16k
```

zip archives created in this manner can be extracted with the program *funzip* which is provided in the *unzip* package, or by *gunzip* which is provided in the *gzip* package. For example:

```
dd if=/dev/nrst0 obs=16k | funzip | tar xvf -
```

When changing an existing *zip* archive, *zip* will write a temporary file with the new contents, and only replace the old one when the process of creating the new version has been completed without error.

If the name of the *zip* archive does not contain an extension, the extension *.zip* is added. If the name already contains an extension other than *.zip* the existing extension is kept unchanged.

OPTIONS

-a

[Systems using EBCDIC] Translate file to ASCII format.

-A

Adjust self-extracting executable archive. A self-extracting executable archive is created by prepending the SFX stub to an existing archive. The **-A** option tells *zip* to adjust the entry offsets stored in the archive to take into account this "preamble" data.

Note: self-extracting archives for the Amiga are a special case. At present, only the Amiga port of Zip is capable of adjusting or updating these without corrupting them. **-J** can be used to remove the SFX stub if other updates need to be made.

-B

[VM/CMS and MVS] force file to be read binary (default is text).

-Bn

[TANDEM] set Edit/Enscribe formatting options with n defined as
 bit 0: Don't add delimiter (Edit/Enscribe)
 bit 1: Use LF rather than CR/LF as delimiter (Edit/Enscribe)
 bit 2: Space fill record to maximum record length (Enscribe)

bit 3: Trim trailing space (Enscribe)

bit 8: Force 30K (Expand) large read for unstructured files

-b path

Use the specified *path* for the temporary *zip* archive. For example:

```
zip -b /tmp stuff *
```

will put the temporary *zip* archive in the directory */tmp*, copying over *stuff.zip* to the current directory when done. This option is only useful when updating an existing archive, and the file system containing this old archive does not have enough space to hold both old and new archives at the same time.

-c

Add one-line comments for each file. File operations (adding, updating) are done first, and the user is then prompted for a one-line comment for each file. Enter the comment followed by return, or just return for no comment.

-d

Remove (delete) entries from a *zip* archive. For example:

```
zip -d foo foo/tom/junk foo/harry/* \*.o
```

will remove the entry *foo/tom/junk*, all of the files that start with *foo/harry/*, and all of the files that end with *.o* (in any path). Note that shell pathname expansion has been inhibited with backslashes, so that *zip* can see the asterisks, enabling *zip* to match on the contents of the *zip* archive instead of the contents of the current directory.

Under MSDOS, **-d** is case sensitive when it matches names in the *zip* archive. This requires that file names be entered in upper case if they were zipped by PKZIP on an MSDOS system.

-df

[MacOS] Include only data-fork of files zipped into the archive. Good for exporting files to foreign operating-systems. Resource-forks will be ignored at all.

-D

Do not create entries in the *zip* archive for directories. Directory entries are created by default so that their attributes can be saved in the *zip* archive. The environment variable ZIPOPT can be used to change the default options. For example under Unix with sh:

```
ZIPOPT="-D"; export ZIPOPT
```

(The variable ZIPOPT can be used for any option except **-i** and **-x** and can include several options.) The option **-D** is a shorthand for **-x "*" /** but the latter cannot be set as default in the ZIPOPT environment variable.

-e

Encrypt the contents of the *zip* archive using a password which is entered on the terminal in response to a prompt (this will not be echoed; if standard error is not a tty, *zip* will exit with an error). The password prompt is repeated to save the user from typing errors.

-E

[OS/2] Use the .LONGNAME Extended Attribute (if found) as filename.

-f

Replace (freshen) an existing entry in the *zip* archive only if it has been modified more recently than the version already in the *zip* archive; unlike the update option (**-u**) this will not add files that are not already in the *zip* archive. For example:

```
zip -f foo
```

This command should be run from the same directory from which the original *zip* command was run, since paths stored in *zip* archives are always relative.

Note that the timezone environment variable TZ should be set according to the local timezone in order for the **-f**, **-u** and **-o** options to work correctly.

The reasons behind this are somewhat subtle but have to do with the differences between the Unix-format file times (always in GMT) and most of the other operating systems (always local time) and the necessity to compare the two. A typical TZ value is ``MET-1MEST" (Middle European time with automatic adjustment for ``summertime" or Daylight Savings Time).

-F

Fix the *zip* archive. This option can be used if some portions of the archive are missing. It is not guaranteed to work, so you MUST make a backup of the original archive first.

When doubled as in **-FF** the compressed sizes given inside the damaged archive are not trusted and *zip* scans for special signatures to identify the limits between the archive members. The single **-F** is more reliable if the archive is not too much damaged, for example if it has only been truncated, so try this option first.

Neither option will recover archives that have been incorrectly transferred in ascii mode instead of binary. After the repair, the **-t** option of *unzip* may show that some files have a bad CRC. Such files cannot be recovered; you can remove them from the archive using the **-d** option of *zip*.

-g

Grow (append to) the specified *zip* archive, instead of creating a new one. If this operation fails, *zip* attempts to restore the archive to its original state. If the restoration fails, the archive might become corrupted. This option is ignored when there's no existing archive or when at least one archive

member must be updated or deleted.

-h

Display the *zip* help information (this also appears if *zip* is run with no arguments).

-i files

Include only the specified files, as in:

```
zip -r foo . -i \*.c
```

which will include only the files that end in *.c* in the current directory and its subdirectories. (Note for PKZIP users: the equivalent command is

```
pkzip -rP foo *.c
```

PKZIP does not allow recursion in directories other than the current one.) The backslash avoids the shell filename substitution, so that the name matching is performed by *zip* at all directory levels.

Also possible:

```
zip -r foo . -i@include.lst
```

which will only include the files in the current directory and its subdirectories that match the patterns in the file *include.lst*.

-I

[Acorn RISC OS] Don't scan through Image files. When used, *zip* will not consider Image files (eg. DOS partitions or Spark archives when SparkFS is loaded) as directories but will store them as single files.

For example, if you have SparkFS loaded, zipping a Spark archive will result in a zipfile containing a directory (and its content) while using the 'I' option will result in a zipfile containing a Spark archive. Obviously this second case will also be obtained (without the 'I' option) if SparkFS isn't loaded.

-j

Store just the name of a saved file (junk the path), and do not store directory names. By default, *zip* will store the full path (relative to the current path).

-jj

[MacOS] record Fullpath (+ Volname). The complete path including volume will be stored. By default the relative path will be stored.

-J

Strip any prepended data (e.g. a SFX stub) from the archive.

-k

Attempt to convert the names and paths to conform to MSDOS, store only the MSDOS attribute (just the user write attribute from UNIX), and mark the entry as made under MSDOS (even though it was not); for compatibility with PKUNZIP under MSDOS which cannot handle certain names such as those with two dots.

-l

Translate the Unix end-of-line character LF into the MSDOS convention CR LF. This option should not be used on binary files. This option can be used on Unix if the zip file is intended for PKUNZIP under MSDOS. If the input files already contain CR LF, this option adds an extra CR. This ensures that *unzip -a* on Unix will get back an exact copy of the original file, to undo the effect of *zip -l*.

-ll

Translate the MSDOS end-of-line CR LF into Unix LF. This option should not be used on binary files. This option can be used on MSDOS if the zip file is intended for *unzip* under Unix.

-L

Display the *zip* license.

-m

Move the specified files into the *zip* archive; actually, this deletes the target directories/files after making the specified *zip* archive. If a directory becomes empty after removal of the files, the directory is also removed. No deletions are done until *zip* has created the archive without error. This is useful for conserving disk space, but is potentially dangerous so it is recommended to use it in combination with **-T** to test the archive before removing all input files.

-n suffixes

Do not attempt to compress files named with the given *suffixes*. Such files are simply stored (0% compression) in the output zip file, so that *zip* doesn't waste its time trying to compress them. The suffixes are separated by either colons or semicolons. For example:

```
zip -rn .Z:.zip:.tiff:.gif:.snd foo foo
```

will copy everything from *foo* into *foo.zip*, but will store any files that end in *.Z*, *.zip*, *.tiff*, *.gif*, or *.snd* without trying to compress them (image and sound files often have their own specialized compression methods). By default, *zip* does not compress files with extensions in the list *.Z:.zip:.zoo:.arc:.lzh:.arj*. Such files are stored directly in the output archive. The environment variable ZIPOPT can be used to change the default options. For example under Unix with *cs*h:

```
setenv ZIPOPT "-n .gif:.zip"
```

To attempt compression on all files, use:

```
zip -n : foo
```

The maximum compression option **-9** also attempts compression on all files regardless of extension.

On Acorn RISC OS systems the suffixes are actually filetypes (3 hex digit format). By default, zip does not compress files with filetypes in the list DDC:D96:68E (i.e. Archives, CFS files and PackDir files).

-N

[Amiga, MacOS] Save Amiga or MacOS filenotes as zipfile comments. They can be restored by using the **-N** option of unzip. If **-c** is used also, you are prompted for comments only for those files that do not have filenotes.

-o

Set the "last modified" time of the *zip* archive to the latest (oldest) "last modified" time found among the entries in the *zip* archive. This can be used without any other operations, if desired. For example:

```
zip -o foo
```

will change the last modified time of *foo.zip* to the latest time of the entries in *foo.zip*.

-P password

use *password* to encrypt zipfile entries (if any). **THIS IS INSECURE!** Many multi-user operating systems provide ways for any user to see the current command line of any other user; even on stand-alone systems there is always the threat of over-the-shoulder peeking. Storing the plaintext password as part of a command line in an automated script is even worse. Whenever possible, use the non-echoing, interactive prompt to enter passwords. (And where security is truly important, use strong encryption such as Pretty Good Privacy instead of the relatively weak encryption provided by standard zipfile utilities.)

-q

Quiet mode; eliminate informational messages and comment prompts. (Useful, for example, in shell scripts and background tasks).

-Qn

[QDOS] store information about the file in the file header with *n* defined as

bit 0: Don't add headers for any file

bit 1: Add headers for all files

bit 2: Don't wait for interactive key press on exit

-r

Travel the directory structure recursively; for example:

```
zip -r foo foo
```

In this case, all the files and directories in *foo* are saved in a *zip* archive named *foo.zip*, including files with names starting with ".", since the recursion does not use the shell's file-name substitution mechanism. If you wish to include only a specific subset of the files in directory *foo* and its subdirectories, use the **-i** option to specify the pattern of files to be included. You should not use **-r** with the name ".*", since that matches "." which will attempt to zip up the parent directory (probably not what was intended).

-R

Travel the directory structure recursively starting at the current directory; for example:

```
zip -R foo '*.c'
```

In this case, all the files matching *.c in the tree starting at the current directory are stored into a *zip* archive named *foo.zip*. Note for PKZIP users: the equivalent command is

```
pkzip -rP foo *.c
```

-S

[MSDOS, OS/2, WIN32 and ATARI] Include system and hidden files.

[MacOS] Includes finder invisible files, which are ignored otherwise.

-t mmdyyy

Do not operate on files modified prior to the specified date, where *mm* is the month (0-12), *dd* is the day of the month (1-31), and *yyyy* is the year. The *ISO 8601* date format *yyyy-mm-dd* is also accepted. For example:

```
zip -rt 12071991 infamy foo
```

```
zip -rt 1991-12-07 infamy foo
```

will add all the files in *foo* and its subdirectories that were last modified on or after 7 December 1991, to the *zip* archive *infamy.zip*.

-tt mmdyyy

Do not operate on files modified after or at the specified date, where *mm* is the month (0-12), *dd* is the day of the month (1-31), and *yyyy* is the year. The *ISO 8601* date format *yyyy-mm-dd* is also accepted. For example:

```
zip -rtt 11301995 infamy foo
```

```
zip -rtt 1995-11-30 infamy foo
```

will add all the files in *foo* and its subdirectories that were last modified before the 30 November 1995, to the *zip* archive *infamy.zip*.

-T

Test the integrity of the new zip file. If the check fails, the old zip file is unchanged and (with the **-m** option) no input files are removed.

-u

Replace (update) an existing entry in the *zip* archive only if it has been modified more recently than the version already in the *zip* archive. For example:

```
zip -u stuff *
```

will add any new files in the current directory, and update any files which have been modified since the *zip* archive *stuff.zip* was last created/modified (note that *zip* will not try to pack *stuff.zip* into itself when you do this).

Note that the **-u** option with no arguments acts like the **-f** (freshen) option.

-v

Verbose mode or print diagnostic version info.

Normally, when applied to real operations, this option enables the display of a progress indicator during compression and requests verbose diagnostic info about zipfile structure oddities.

When **-v** is the only command line argument, and stdout is not redirected to a file, a diagnostic screen is printed. In addition to the help screen header with program name, version, and release date, some pointers to the Info-ZIP home and distribution sites are given. Then, it shows information about the target environment (compiler type and version, OS version, compilation date and the enabled optional features used to create the *zip* executable).

-V

[VMS] Save VMS file attributes. *zip* archives created with this option will generally not be usable on other systems.

-w

[VMS] Append the version number of the files to the name, including multiple versions of files. (default: use only the most recent version of a specified file).

-x files

Explicitly exclude the specified files, as in:

```
zip -r foo foo -x \*.o
```

which will include the contents of *foo* in *foo.zip* while excluding all the files that end in *.o*. The backslash avoids the shell filename substitution, so that the name matching is performed by *zip* at all directory levels.

Also possible:

```
zip -r foo foo -x@exclude.lst
```

which will include the contents of *foo* in *foo.zip* while excluding all the files that match the patterns in the file *exclude.lst*.

-X

Do not save extra file attributes (Extended Attributes on OS/2, uid/gid and file times on Unix).

-y

Store symbolic links as such in the *zip* archive, instead of compressing and storing the file referred to by the link (UNIX only).

-z

Prompt for a multi-line comment for the entire *zip* archive. The comment is ended by a line containing just a period, or an end of file condition (^D on UNIX, ^Z on MSDOS, OS/2, and VAX/VMS). The comment can be taken from a file:

```
zip -z foo < foowhat
```

-#

Regulate the speed of compression using the specified digit #, where **-0** indicates no compression (store all files), **-1** indicates the fastest compression method (less compression) and **-9** indicates the slowest compression method (optimal compression, ignores the suffix list). The default compression level is **-6**.

-!

[WIN32] Use privileges (if granted) to obtain all aspects of WinNT security.

-@

Take the list of input files from standard input. Only one filename per line.

-\$

[MSDOS, OS/2, WIN32] Include the volume label for the the drive holding the first file to be compressed. If you want to include only the volume label or to force a specific drive, use the drive name as first file name, as in:

```
zip -$ foo a: c:bar
```

EXAMPLES

The simplest example:

```
zip stuff *
```

creates the archive *stuff.zip* (assuming it does not exist) and puts all the files in the current directory in it, in compressed form (the *.zip* suffix is added automatically, unless that archive name given contains a dot already; this allows the explicit specification of other suffixes).

Because of the way the shell does filename substitution, files starting with "." are not included; to include these as well:

```
zip stuff .* *
```

Even this will not include any subdirectories from the current directory.

To zip up an entire directory, the command:

```
zip -r foo foo
```

creates the archive *foo.zip*, containing all the files and directories in the directory *foo* that is contained within the current directory.

You may want to make a *zip* archive that contains the files in *foo*, without recording the directory name, *foo*. You can use the *-j* option to leave off the paths, as in:

```
zip -j foo foo/*
```

If you are short on disk space, you might not have enough room to hold both the original directory and the corresponding compressed *zip* archive. In this case, you can create the archive in steps using the *-m* option. If *foo* contains the subdirectories *tom*, *dick*, and *harry*, you can:

```
zip -rm foo foo/tom
zip -rm foo foo/dick
zip -rm foo foo/harry
```

where the first command creates *foo.zip*, and the next two add to it. At the completion of each *zip* command, the last created archive is deleted, making room for the next *zip* command to function.

PATTERN MATCHING

This section applies only to UNIX. Watch this space for details on MSDOS and VMS operation.

The UNIX shells ([sh\(1\)](#) and [csh\(1\)](#)) do filename substitution on command arguments. The special characters are:

- ?
match any single character
- *
match any number of characters (including none)
- []
match any character in the range indicated within the brackets (example: [a-f], [0-9]).

When these characters are encountered (without being escaped with a backslash or quotes), the shell will look for files relative to the current path that match the pattern, and replace the argument with a list of the names that matched.

The *zip* program can do the same matching on names that are in the *zip* archive being modified or, in the case of the *-x* (exclude) or *-i* (include) options, on the list of files to be operated on, by using backslashes or quotes to tell the shell not to do the name expansion. In general, when *zip* encounters a name in the list of files to do, it first looks for the name in the file system. If it finds it, it then adds it to the list of files to do.

If it does not find it, it looks for the name in the *zip* archive being modified (if it exists), using the pattern matching characters described above, if present. For each match, it will add that name to the list of files to be processed, unless this name matches one given with the **-x** option, or does not match any name given with the **-i** option.

The pattern matching includes the path, and so patterns like `*.o` match names that end in ".o", no matter what the path prefix is. Note that the backslash must precede every special character (i.e. `?*[]`), or the entire argument must be enclosed in double quotes ("").

In general, use backslash to make *zip* do the pattern matching with the **-f** (freshen) and **-d** (delete) options, and sometimes after the **-x** (exclude) option when used with an appropriate operation (add, **-u**, **-f**, or **-d**).

SEE ALSO

[compress\(1\)](#), [shar\(1L\)](#), [tar\(1\)](#), [unzip\(1L\)](#), [gzip\(1L\)](#)

Important: Use the *man* command (`% man`) to see how a command is used on your particular computer.

[>> Linux/Unix Command Library](#)

[>> Shell Command Library](#)



From [Juergen Haas](#),
Your Guide to [Focus on Linux](#).
FREE Newsletter. [Sign Up Now!](#)

[Topic Index](#) | [Email to a Friend](#) | [Bookmark this Site](#) | [Make this Site Your Homepage!](#) | [Print this Page](#)

[Our Story](#) | [Be a Guide](#) | [Advertising Info](#) | [Work at About](#) | [Site Map](#) | [Icons](#) | [Help](#)
[User Agreement](#) | [Ethics Policy](#) | [Patent Info.](#) | [Privacy Policy](#) | [Kids' Privacy Policy](#)

©2006 About, Inc., A part of [The New York Times Company](#). All rights reserved.