

**NAME**

**tar** — manipulate tape archives

**SYNOPSIS**

```
tar [bundled-flags <args>] [<file> | <pattern> ...]
tar {-c} [options] [files | directories]
tar {-r | -u} -f archive-file [options] [files | directories]
tar {-t | -x} [options] [patterns]
```

**DESCRIPTION**

**tar** creates and manipulates streaming archive files. This implementation can extract from tar, pax, cpio, zip, jar, ar, xar, rar, rpm, 7-zip, and ISO 9660 cdrom images and can create tar, pax, cpio, ar, zip, 7-zip, and shar archives.

The first synopsis form shows a “bundled” option word. This usage is provided for compatibility with historical implementations. See “COMPATIBILITY” below for details.

The other synopsis forms show the preferred usage. The first option to **tar** is a mode indicator from the following list:

- c Create a new archive containing the specified items. The long option form is `--create`.
- r Like `-c`, but new entries are appended to the archive. Note that this only works on uncompressed archives stored in regular files. The `-f` option is required. The long option form is `--append`.
- t List archive contents to stdout. The long option form is `--list`.
- u Like `-r`, but new entries are added only if they have a modification date newer than the corresponding entry in the archive. Note that this only works on uncompressed archives stored in regular files. The `-f` option is required. The long form is `--update`.
- x Extract to disk from the archive. If a file with the same name appears more than once in the archive, each copy will be extracted, with later copies overwriting (replacing) earlier copies. The long option form is `--extract`.

In `-c`, `-r`, or `-u` mode, each specified file or directory is added to the archive in the order specified on the command line. By default, the contents of each directory are also archived.

In extract or list mode, the entire command line is read and parsed before the archive is opened. The pathnames or patterns on the command line indicate which items in the archive should be processed. Patterns are shell-style globbing patterns as documented in *tcsh*(1).

**OPTIONS**

Unless specifically stated otherwise, options are applicable in all operating modes.

**@archive** (c and r modes only) The specified archive is opened and the entries in it will be appended to the current archive. As a simple example,

```
tar -c -f - newfile @original.tar
```

writes a new archive to standard output containing a file *newfile* and all of the entries from *original.tar*. In contrast,

```
tar -c -f - newfile original.tar
```

creates a new archive with only two entries. Similarly,

```
tar -czf - --format pax @-
```

reads an archive from standard input (whose format will be determined automatically) and converts it into a gzip-compressed pax-format archive on stdout. In this way, **tar** can be used to convert archives from one format to another.

-a, `--auto-compress`

(c mode only) Use the archive suffix to decide a set of the format and the compressions. As a simple example,

```
tar -a -cf archive.tgz source.c source.h
```

creates a new archive with restricted pax format and gzip compression,

```
tar -a -cf archive.tar.bz2.uu source.c source.h
```

creates a new archive with restricted pax format and bzip2 compression and uuencode compression,

```
tar -a -cf archive.zip source.c source.h
```

creates a new archive with zip format,

```
tar -a -jcf archive.tgz source.c source.h
```

ignores the “-j” option, and creates a new archive with restricted pax format and gzip compression,

```
tar -a -jcf archive.xxx source.c source.h
```

if it is unknown suffix or no suffix, creates a new archive with restricted pax format and bzip2 compression.

**--acls**

(c, r, u, x modes only) Archive or extract POSIX.1e or NFSv4 ACLs. This is the reverse of **--no-acls** and the default behavior in c, r, and u modes (except on Mac OS X) or if **tar** is run in x mode as root. On Mac OS X this option translates extended ACLs to NFSv4 ACLs. To store extended ACLs the **--mac-metadata** option is preferred.

**-B, --read-full-blocks**

Ignored for compatibility with other *tar*(1) implementations.

**-b *blocksize*, --block-size *blocksize***

Specify the block size, in 512-byte records, for tape drive I/O. As a rule, this argument is only needed when reading from or writing to tape drives, and usually not even then as the default block size of 20 records (10240 bytes) is very common.

**-C *directory*, --cd *directory*, --directory *directory***

In c and r mode, this changes the directory before adding the following files. In x mode, change directories after opening the archive but before extracting entries from the archive.

**--chroot**

(x mode only) **chroot()** to the current directory after processing any **-C** options and before extracting any files.

**--clamp-mtime**

(use with **--mtime**) Only set the modification time if the file is newer than the date specified in **--mtime**.

**--clear-nochange-fflags**

(x mode only) Before removing file system objects to replace them, clear platform-specific file attributes or file flags that might prevent removal.

**--exclude *pattern***

Do not process files or directories that match the specified pattern. Note that exclusions take precedence over patterns or filenames specified on the command line.

**--exclude-vcs**

Do not process files or directories internally used by the version control systems ‘Arch’, ‘Bazaar’, ‘CVS’, ‘Darcs’, ‘Mercurial’, ‘RCS’, ‘SCCS’, ‘SVN’ and ‘git’.

**--fflags**

(c, r, u, x modes only) Archive or extract platform-specific file attributes or file flags. This is the reverse of **--no-fflags** and the default behavior in c, r, and u modes or if **tar** is run in x mode as root.

- `--format format`  
(c, r, u mode only) Use the specified format for the created archive. Supported formats include “cpio”, “pax”, “shar”, and “ustar”. Other formats may also be supported; see *libarchive-formats(5)* for more information about currently-supported formats. In r and u modes, when extending an existing archive, the format specified here must be compatible with the format of the existing archive on disk.
- `-f file, --file file`  
Read the archive from or write the archive to the specified file. The filename can be - for standard input or standard output. The default varies by system; on FreeBSD, the default is */dev/sa0*; on Linux, the default is */dev/st0*.
- `--gid id`  
Use the provided group id number. On extract, this overrides the group id in the archive; the group name in the archive will be ignored. On create, this overrides the group id read from disk; if `--gname` is not also specified, the group name will be set to match the group id.
- `--gname name`  
Use the provided group name. On extract, this overrides the group name in the archive; if the provided group name does not exist on the system, the group id (from the archive or from the `--gid` option) will be used instead. On create, this sets the group name that will be stored in the archive; the name will not be verified against the system group database.
- `--group name[:gid]`  
Use the provided group, if *gid* is not provided, *name* can be either a group name or numeric id. See the `--gname` option for details.
- `-H` (c and r modes only) Symbolic links named on the command line will be followed; the target of the link will be archived, not the link itself.
- `-h` (c and r modes only) Synonym for `-L`.
- `-I` Synonym for `-T`.
- `--help`  
Show usage.
- `--hfsCompression`  
(x mode only) Mac OS X specific (v10.6 or later). Compress extracted regular files with HFS+ compression.
- `--ignore-zeros`  
An alias of `--options read_concatenated_archives` for compatibility with GNU tar.
- `--include pattern`  
Process only files or directories that match the specified pattern. Note that exclusions specified with `--exclude` take precedence over inclusions. If no inclusions are explicitly specified, all entries are processed by default. The `--include` option is especially useful when filtering archives. For example, the command  

```
tar -c -f new.tar --include='*foo*' @old.tgz
```

creates a new archive *new.tar* containing only the entries from *old.tgz* containing the string ‘foo’.
- `-J, --xz`  
(c mode only) Compress the resulting archive with xz(1). In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes XZ compression automatically when reading archives.

- j, --bzip, --bzip2, --bunzip2  
(c mode only) Compress the resulting archive with *bzip2*(1). In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes bzip2 compression automatically when reading archives.
- k, --keep-old-files  
(x mode only) Do not overwrite existing files. In particular, if a file appears more than once in an archive, later copies will not overwrite earlier copies.
- keep-newer-files  
(x mode only) Do not overwrite existing files that are newer than the versions appearing in the archive being extracted.
- L, --dereference  
(c and r modes only) All symbolic links will be followed. Normally, symbolic links are archived as such. With this option, the target of the link will be archived instead.
- l, --check-links  
(c and r modes only) Issue a warning message unless all links to each file are archived.
- lrzip  
(c mode only) Compress the resulting archive with *lrzip*(1). In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes lrzip compression automatically when reading archives.
- lz4 (c mode only) Compress the archive with lz4-compatible compression before writing it. In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes lz4 compression automatically when reading archives.
- zstd  
(c mode only) Compress the archive with zstd-compatible compression before writing it. In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes zstd compression automatically when reading archives.
- lzma  
(c mode only) Compress the resulting archive with the original LZMA algorithm. In extract or list modes, this option is ignored. Use of this option is discouraged and new archives should be created with --xz instead. Note that this **tar** implementation recognizes LZMA compression automatically when reading archives.
- lzop  
(c mode only) Compress the resulting archive with *lzop*(1). In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes LZO compression automatically when reading archives.
- m, --modification-time  
(x mode only) Do not extract modification time. By default, the modification time is set to the time stored in the archive.
- mac-metadata  
(c, r, u and x mode only) Mac OS X specific. Archive or extract extended ACLs and extended file attributes using *copyfile*(3) in AppleDouble format. This is the reverse of --no-mac-metadata. and the default behavior in c, r, and u modes or if **tar** is run in x mode as root. Currently supported only for pax formats (including pax restricted, the default tar format for **bsdtar**)
- mtime *date*  
(c, r, u modes only) Set the modification times of added files to the specified date.

- n, --norecurse, --no-recursion  
Do not operate recursively on the content of directories.
- newer *date*  
(c, r, u modes only) Only include files and directories newer than the specified date. This compares ctime entries.
- newer-mtime *date*  
(c, r, u modes only) Like --newer, except it compares mtime entries instead of ctime entries.
- newer-than *file*  
(c, r, u modes only) Only include files and directories newer than the specified file. This compares ctime entries.
- newer-mtime-than *file*  
(c, r, u modes only) Like --newer-than, except it compares mtime entries instead of ctime entries.
- nodump  
(c and r modes only) Honor the nodump file flag by skipping this file.
- nopreserveHFSCompression  
(x mode only) Mac OS X specific (v10.6 or later). Do not compress extracted regular files which were compressed with HFS+ compression before archived. By default, compress the regular files again with HFS+ compression.
- null  
(use with -I or -T) Filenames or patterns are separated by null characters, not by newlines. This is often used to read filenames output by the -print0 option to *find(1)*.
- no-acls  
(c, r, u, x modes only) Do not archive or extract POSIX.1e or NFSv4 ACLs. This is the reverse of --acls and the default behavior if **tar** is run as non-root in x mode (on Mac OS X as any user in c, r, u and x modes).
- no-fflags  
(c, r, u, x modes only) Do not archive or extract file attributes or file flags. This is the reverse of --fflags and the default behavior if **tar** is run as non-root in x mode.
- no-mac-metadata  
(c, r, u and x mode only) Mac OS X specific. Do not archive or extract ACLs and extended file attributes using *copyfile(3)* in AppleDouble format. This is the reverse of --mac-metadata and the default behavior if **tar** is run as non-root in x mode.
- no-read-sparse  
(c, r, u modes only) Do not read sparse file information from disk. This is the reverse of --read-sparse.
- no-safe-writes  
(x mode only) Do not create temporary files and use *rename(2)* to replace the original ones. This is the reverse of --safe-writes.
- no-same-owner  
(x mode only) Do not extract owner and group IDs. This is the reverse of --same-owner and the default behavior if **tar** is run as non-root.
- no-same-permissions  
(x mode only) Do not extract full permissions (SGID, SUID, sticky bit, file attributes or file flags, extended file attributes and ACLs). This is the reverse of -p and the default behavior if **tar** is run as non-root.

- `--no-xattrs`  
(c, r, u, x modes only) Do not archive or extract extended file attributes. This is the reverse of `--xattrs` and the default behavior if **tar** is run as non-root in x mode.
- `--numeric-owner`  
This is equivalent to `--uname "" --gname ""`. On extract, it causes user and group names in the archive to be ignored in favor of the numeric user and group ids. On create, it causes user and group names to not be stored in the archive.
- `-O, --to-stdout`  
(x, t modes only) In extract (-x) mode, files will be written to standard out rather than being extracted to disk. In list (-t) mode, the file listing will be written to stderr rather than the usual stdout.
- `-o`  
(x mode) Use the user and group of the user running the program rather than those specified in the archive. Note that this has no significance unless `-p` is specified, and the program is being run by the root user. In this case, the file modes and flags from the archive will be restored, but ACLs or owner information in the archive will be discarded.
- `-o`  
(c, r, u mode) A synonym for `--format ustar`
- `--older date`  
(c, r, u modes only) Only include files and directories older than the specified date. This compares ctime entries.
- `--older-mtime date`  
(c, r, u modes only) Like `--older`, except it compares mtime entries instead of ctime entries.
- `--older-than file`  
(c, r, u modes only) Only include files and directories older than the specified file. This compares ctime entries.
- `--older-mtime-than file`  
(c, r, u modes only) Like `--older-than`, except it compares mtime entries instead of ctime entries.
- `--one-file-system`  
(c, r, and u modes) Do not cross mount points.
- `--options options`  
Select optional behaviors for particular modules. The argument is a text string containing comma-separated keywords and values. These are passed to the modules that handle particular formats to control how those formats will behave. Each option has one of the following forms:  
  - `key=value`  
The key will be set to the specified value in every module that supports it. Modules that do not support this key will ignore it.
  - `key`  
The key will be enabled in every module that supports it. This is equivalent to `key=1`.
  - `!key`  
The key will be disabled in every module that supports it.
  - `module:key=value, module:key, module:!key`  
As above, but the corresponding key and value will be provided only to modules whose name matches *module*.

The complete list of supported modules and keys for create and append modes is in *archive\_write\_set\_options(3)* and for extract and list modes in *archive\_read\_set\_options(3)*.

Examples of supported options:

  - `iso9660:joliet`  
Support Joliet extensions. This is enabled by default, use `!joliet` or `iso9660:!joliet` to disable.

`iso9660:rockridge`  
 Support Rock Ridge extensions. This is enabled by default, use `!rockridge` or `iso9660:!rockridge` to disable.

`gzip:compression-level`  
 A decimal integer from 1 to 9 specifying the gzip compression level.

`gzip:timestamp`  
 Store timestamp. This is enabled by default, use `!timestamp` or `gzip:!timestamp` to disable.

`lrzip:compression=type`  
 Use *type* as compression method. Supported values are `bzip2`, `gzip`, `lzo` (ultra fast), and `zpaq` (best, extremely slow).

`lrzip:compression-level`  
 A decimal integer from 1 to 9 specifying the lrzip compression level.

`lz4:compression-level`  
 A decimal integer from 1 to 9 specifying the lzop compression level.

`lz4:stream-checksum`  
 Enable stream checksum. This is by default, use `lz4:!stream-checksum` to disable.

`lz4:block-checksum`  
 Enable block checksum (Disabled by default).

`lz4:block-size`  
 A decimal integer from 4 to 7 specifying the lz4 compression block size (7 is set by default).

`lz4:block-dependence`  
 Use the previous block of the block being compressed for a compression dictionary to improve compression ratio.

`zstd:compression-level=N`  
 A decimal integer specifying the zstd compression level. Supported values depend on the library version, common values are from 1 to 22.

`zstd:threads=N`  
 Specify the number of worker threads to use, or 0 to use as many threads as there are CPU cores in the system.

`zstd:frame-per-file`  
 Start a new compression frame at the beginning of each file in the archive.

`zstd:min-frame-in=N`  
 In combination with `zstd:frame-per-file`, do not start a new compression frame unless the uncompressed size of the current frame is at least *N* bytes. The number may be followed by `k / kB`, `M / MB`, or `G / GB` to indicate kilobytes, megabytes or gigabytes respectively.

`zstd:min-frame-out=N`, `zstd:min-frame-size=N`  
 In combination with `zstd:frame-per-file`, do not start a new compression frame unless the compressed size of the current frame is at least *N* bytes. The number may be followed by `k / kB`, `M / MB`, or `G / GB` to indicate kilobytes, megabytes or gigabytes respectively.

`zstd:max-frame-in=N`, `zstd:max-frame-size=N`  
 Start a new compression frame as soon as possible after the uncompressed size of the current frame exceeds *N* bytes. The number may be followed by `k / kB`, `M / MB`, or `G / GB` to indicate kilobytes, megabytes or gigabytes respectively. Values less than 1,024 will be rejected.

`zstd:max-frame-out=N`  
 Start a new compression frame as soon as possible after the compressed size of the current frame exceeds *N* bytes. The number may be followed by `k / kB`, `M / MB`, or `G / GB` to indicate kilobytes, megabytes or gigabytes respectively. Values less than 1,024 will be rejected.

`lzop:compression-level`

A decimal integer from 1 to 9 specifying the lzop compression level.

`xz:compression-level`

A decimal integer from 0 to 9 specifying the xz compression level.

`xz:threads`

Specify the number of worker threads to use. Setting threads to a special value 0 makes `xz(1)` use as many threads as there are CPU cores on the system.

`mtree:keyword`

The mtree writer module allows you to specify which mtree keywords will be included in the output. Supported keywords include: cksum, device, flags, gid, gname, indent, link, md5, mode, nlink, rmd160, sha1, sha256, sha384, sha512, size, time, uid, uname. The default is equivalent to: “device, flags, gid, gname, link, mode, nlink, size, time, type, uid, uname”.

`mtree:all`

Enables all of the above keywords. You can also use `mtree:!all` to disable all keywords.

`mtree:use-set`

Enable generation of `/set` lines in the output.

`mtree:indent`

Produce human-readable output by indenting options and splitting lines to fit into 80 columns.

`zip:compression=type`

Use *type* as compression method. Supported values are store (uncompressed) and deflate (gzip algorithm).

`zip:encryption`

Enable encryption using traditional zip encryption.

`zip:encryption=type`

Use *type* as encryption type. Supported values are zipcrypt (traditional zip encryption), aes128 (WinZip AES-128 encryption) and aes256 (WinZip AES-256 encryption).

`read_concatenated_archives`

Ignore zeroed blocks in the archive, which occurs when multiple tar archives have been concatenated together. Without this option, only the contents of the first concatenated archive would be read. This option is comparable to the `-i`, `--ignore-zeros` option of GNU tar.

If a provided option is not supported by any module, that is a fatal error.

`-P, --absolute-paths`

Preserve pathnames. By default, absolute pathnames (those that begin with a `/` character) have the leading slash removed both when creating archives and extracting from them. Also, **tar** will refuse to extract archive entries whose pathnames contain `..` or whose target directory would be altered by a symlink. This option suppresses these behaviors.

`-p, --insecure, --preserve-permissions`

(x mode only) Preserve file permissions. Attempt to restore the full permissions, including file modes, file attributes or file flags, extended file attributes and ACLs, if available, for each item extracted from the archive. This is the reverse of `--no-same-permissions` and the default if **tar** is being run as root. It can be partially overridden by also specifying `--no-acls`, `--no-fflags`, `--no-mac-metadata` or `--no-xattrs`.

`--passphrase passphrase`

The *passphrase* is used to extract or create an encrypted archive. Currently, zip is the only supported format that supports encryption. You shouldn't use this option unless you realize how insecure use of this option is.



- `--posix`  
(c, r, u mode only) Synonym for `--format pax`
- `-q, --fast-read`  
(x and t mode only) Extract or list only the first archive entry that matches each pattern or file-name operand. Exit as soon as each specified pattern or filename has been matched. By default, the archive is always read to the very end, since there can be multiple entries with the same name and, by convention, later entries overwrite earlier entries. This option is provided as a performance optimization.
- `--read-sparse`  
(c, r, u modes only) Read sparse file information from disk. This is the reverse of `--no-read-sparse` and the default behavior.
- `-S`  
(x mode only) Extract files as sparse files. For every block on disk, check first if it contains only NULL bytes and seek over it otherwise. This works similar to the `conv=sparse` option of `dd`.
- `-s pattern`  
Modify file or archive member names according to *pattern*. The pattern has the format */old/new/[bghHprRsS]* where *old* is a basic regular expression, *new* is the replacement string of the matched part, and the optional trailing letters modify how the replacement is handled. If *old* is not matched, the pattern is skipped. Within *new*, `~` is substituted with the match, `\1` to `\9` with the content of the corresponding captured group. The optional trailing *g* specifies that matching should continue after the matched part and stop on the first unmatched pattern. The optional trailing *s* specifies that the pattern applies to the value of symbolic links. The optional trailing *p* specifies that after a successful substitution the original path name and the new path name should be printed to standard error. The optional trailing *b* specifies that the substitution should be matched from the beginning of the string rather than from right after the position at which the previous matching substitution ended. Optional trailing *H*, *R*, or *S* characters suppress substitutions for hardlink targets, regular filenames, or symlink targets, respectively. Optional trailing *h*, *r*, or *s* characters enable substitutions for hardlink targets, regular filenames, or symlink targets, respectively. The default is *hrs* which applies substitutions to all names. In particular, it is never necessary to specify *h*, *r*, or *s*.
- `--safe-writes`  
(x mode only) Extract files atomically. By default **tar** unlinks the original file with the same name as the extracted file (if it exists), and then creates it immediately under the same name and writes to it. For a short period of time, applications trying to access the file might not find it, or see incomplete results. If `--safe-writes` is enabled, **tar** first creates a unique temporary file, then writes the new contents to the temporary file, and finally renames the temporary file to its final name atomically using `rename(2)`. This guarantees that an application accessing the file, will either see the old contents or the new contents at all times.
- `--same-owner`  
(x mode only) Extract owner and group IDs. This is the reverse of `--no-same-owner` and the default behavior if **tar** is run as root.
- `--strip-components count`  
Remove the specified number of leading path elements. Pathnames with fewer elements will be silently skipped. Note that the pathname is edited after checking inclusion/exclusion patterns but before security checks.
- `-T filename, --files-from filename`  
In x or t mode, **tar** will read the list of names to be extracted from *filename*. In c mode, **tar** will read names to be archived from *filename*. The special name `"-C"` on a line by itself will cause the current directory to be changed to the directory specified on the following line. Names are terminated by newlines unless `--null` is specified. Note that `--null` also disables the special handling of lines containing `"-C"`. Note: If you are generating lists of files using `find(1)`,

you probably want to use `-n` as well.

- `--totals`  
(c, r, u modes only) After archiving all files, print a summary to stderr.
- `-U, --unlink, --unlink-first`  
(x mode only) Unlink files before creating them. This can be a minor performance optimization if most files already exist, but can make things slower if most files do not already exist. This flag also causes **tar** to remove intervening directory symlinks instead of reporting an error. See the “SECURITY” section below for more details.
- `--uid id`  
Use the provided user id number and ignore the user name from the archive. On create, if `--uname` is not also specified, the user name will be set to match the user id.
- `--uname name`  
Use the provided user name. On extract, this overrides the user name in the archive; if the provided user name does not exist on the system, it will be ignored and the user id (from the archive or from the `--uid` option) will be used instead. On create, this sets the user name that will be stored in the archive; the name is not verified against the system user database.
- `--use-compress-program program`  
Pipe the input (in x or t mode) or the output (in c mode) through *program* instead of using the builtin compression support.
- `--owner name[:uid]`  
Use the provided user, if *uid* is not provided, *name* can be either a username or numeric id. See the `--uname` option for details.
- `-v, --verbose`  
Produce verbose output. In create and extract modes, **tar** will list each file name as it is read from or written to the archive. In list mode, **tar** will produce output similar to that of `ls(1)`. An additional `-v` option will also provide ls-like details in create and extract mode.
- `--version`  
Print version of **tar** and **libarchive**, and exit.
- `-w, --confirmation, --interactive`  
Ask for confirmation for every action.
- `-X filename, --exclude-from filename`  
Read a list of exclusion patterns from the specified file. See `--exclude` for more information about the handling of exclusions.
- `--xattrs`  
(c, r, u, x modes only) Archive or extract extended file attributes. This is the reverse of `--no-xattrs` and the default behavior in c, r, and u modes or if **tar** is run in x mode as root.
- `-y`  
(c mode only) Compress the resulting archive with `bzip2(1)`. In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes bzip2 compression automatically when reading archives.
- `-Z, --compress, --uncompress`  
(c mode only) Compress the resulting archive with `compress(1)`. In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes compress compression automatically when reading archives.
- `-z, --gunzip, --gzip`  
(c mode only) Compress the resulting archive with `gzip(1)`. In extract or list modes, this option is ignored. Note that this **tar** implementation recognizes gzip compression automatically when reading archives.

## ENVIRONMENT

The following environment variables affect the execution of **tar**:

### TAR\_READER\_OPTIONS

The default options for format readers and compression readers. The `--options` option overrides this.

### TAR\_WRITER\_OPTIONS

The default options for format writers and compression writers. The `--options` option overrides this.

**LANG** The locale to use. See *environ(7)* for more information.

**TAPE** The default device. The `-f` option overrides this. Please see the description of the `-f` option above for more details.

**TZ** The timezone to use when displaying dates. See *environ(7)* for more information.

## EXIT STATUS

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

## EXAMPLES

The following creates a new archive called *file.tar.gz* that contains two files *source.c* and *source.h*:

```
tar -czf file.tar.gz source.c source.h
```

To view a detailed table of contents for this archive:

```
tar -tvf file.tar.gz
```

To extract all entries from the archive on the default tape drive:

```
tar -x
```

To examine the contents of an ISO 9660 cdrom image:

```
tar -tf image.iso
```

To move file hierarchies, invoke **tar** as

```
tar -cf - -C srcdir | tar -xpf - -C destdir
```

or more traditionally

```
cd srcdir ; tar -cf - | (cd destdir ; tar -xpf -)
```

In create mode, the list of files and directories to be archived can also include directory change instructions of the form `-Cfoo/baz` and archive inclusions of the form `@archive-file`. For example, the command line

```
tar -c -f new.tar foo1 @old.tgz -C/tmp foo2
```

will create a new archive *new.tar*. **tar** will read the file *foo1* from the current directory and add it to the output archive. It will then read each entry from *old.tgz* and add those entries to the output archive. Finally, it will switch to the */tmp* directory and add *foo2* to the output archive.

An input file in *mtree(5)* format can be used to create an output archive with arbitrary ownership, permissions, or names that differ from existing data on disk:

```
$ cat input.mtree
#mtree
usr/bin uid=0 gid=0 mode=0755 type=dir
usr/bin/ls uid=0 gid=0 mode=0755 type=file content=myls
$ tar -cvf output.tar @input.mtree
```

The `--newer` and `--newer-mtime` switches accept a variety of common date and time specifications, including “12 Mar 2005 7:14:29pm”, “2005-03-12 19:14”, “5 minutes ago”, and “19:14 PST May 1”.

The `--options` argument can be used to control various details of archive generation or reading. For example, you can generate mtree output which only contains `type`, `time`, and `uid` keywords:

```
tar -cf file.tar --format=mtree --options='!all,type,time,uid' dir
```

or you can set the compression level used by gzip or xz compression:

```
tar -czf file.tar --options='compression-level=9'.
```

For more details, see the explanation of the `archive_read_set_options()` and `archive_write_set_options()` API calls that are described in `archive_read(3)` and `archive_write(3)`.

## COMPATIBILITY

The bundled-arguments format is supported for compatibility with historic implementations. It consists of an initial word (with no leading - character) in which each character indicates an option. Arguments follow as separate words. The order of the arguments must match the order of the corresponding characters in the bundled command word. For example,

```
tar tbf 32 file.tar
```

specifies three flags `t`, `b`, and `f`. The `b` and `f` flags both require arguments, so there must be two additional items on the command line. The `32` is the argument to the `b` flag, and *file.tar* is the argument to the `f` flag.

The mode options `c`, `r`, `t`, `u`, and `x` and the options `b`, `f`, `l`, `m`, `o`, `v`, and `w` comply with SUSv2.

For maximum portability, scripts that invoke **tar** should use the bundled-argument format above, should limit themselves to the `c`, `t`, and `x` modes, and the `b`, `f`, `m`, `v`, and `w` options.

Additional long options are provided to improve compatibility with other tar implementations.

## SECURITY

Certain security issues are common to many archiving programs, including **tar**. In particular, carefully-crafted archives can request that **tar** extract files to locations outside of the target directory. This can potentially be used to cause unwitting users to overwrite files they did not intend to overwrite. If the archive is being extracted by the superuser, any file on the system can potentially be overwritten. There are three ways this can happen. Although **tar** has mechanisms to protect against each one, savvy users should be aware of the implications:

- Archive entries can have absolute pathnames. By default, **tar** removes the leading `/` character from filenames before restoring them to guard against this problem.
- Archive entries can have pathnames that include `..` components. By default, **tar** will not extract files containing `..` components in their pathname.
- Archive entries can exploit symbolic links to restore files to other directories. An archive can restore a symbolic link to another directory, then use that link to restore a file into that directory. To guard against this, **tar** checks each extracted path for symlinks. If the final path element is a symlink, it will be removed and replaced with the archive entry. If `-U` is specified, any intermediate symlink will also be unconditionally removed. If neither `-U` nor `-P` is specified, **tar** will refuse to extract the entry.

To protect yourself, you should be wary of any archives that come from untrusted sources. You should examine the contents of an archive with

```
tar -tf filename
```

before extraction. You should use the `-k` option to ensure that **tar** will not overwrite any existing files or the `-U` option to remove any pre-existing files. You should generally not extract archives while running with super-user privileges. Note that the `-P` option to **tar** disables the security checks above and allows you to extract an archive while preserving any absolute pathnames, `..` components, or symlinks to other directories.

**SEE ALSO**

*bzip2(1)*, *compress(1)*, *cpio(1)*, *gzip(1)*, *mt(1)*, *pax(1)*, *shar(1)*, *xz(1)*, *libarchive(3)*, *libarchive-formats(5)*, *tar(5)*

**STANDARDS**

There is no current POSIX standard for the `tar` command; it appeared in ISO/IEC 9945-1:1996 (“POSIX.1”) but was dropped from IEEE Std 1003.1-2001 (“POSIX.1”). The options supported by this implementation were developed by surveying a number of existing tar implementations as well as the old POSIX specification for tar and the current POSIX specification for pax.

The `ustar` and `pax` interchange file formats are defined by IEEE Std 1003.1-2001 (“POSIX.1”) for the `pax` command.

**HISTORY**

A `tar` command appeared in Seventh Edition Unix, which was released in January, 1979. There have been numerous other implementations, many of which extended the file format. John Gilmore’s `pdstar` public-domain implementation (circa November, 1987) was quite influential, and formed the basis of GNU tar. GNU tar was included as the standard system tar in FreeBSD beginning with FreeBSD 1.0.

This is a complete re-implementation based on the *libarchive(3)* library. It was first released with FreeBSD 5.4 in May, 2005.

**BUGS**

This program follows ISO/IEC 9945-1:1996 (“POSIX.1”) for the definition of the `-l` option. Note that GNU tar prior to version 1.15 treated `-l` as a synonym for the `--one-file-system` option.

The `-C dir` option may differ from historic implementations.

All archive output is written in correctly-sized blocks, even if the output is being compressed. Whether or not the last output block is padded to a full block size varies depending on the format and the output device. For tar and cpio formats, the last block of output is padded to a full block size if the output is being written to standard output or to a character or block device such as a tape drive. If the output is being written to a regular file, the last block will not be padded. Many compressors, including *gzip(1)* and *bzip2(1)*, complain about the null padding when decompressing an archive created by `tar`, although they still extract it correctly.

The compression and decompression is implemented internally, so there may be insignificant differences between the compressed output generated by

```
tar -czf -file
```

and that generated by

```
tar -cf -file | gzip
```

The default should be to read and write archives to the standard I/O paths, but tradition (and POSIX) dictates otherwise.

The `r` and `u` modes require that the archive be uncompressed and located in a regular file on disk. Other archives can be modified using `c` mode with the `@archive-file` extension.

To archive a file called `@foo` or `-foo` you must specify it as `./@foo` or `./-foo`, respectively.

In create mode, a leading `./` is always removed. A leading `/` is stripped unless the `-P` option is specified.

There needs to be better support for file selection on both create and extract.

There is not yet any support for multi-volume archives.

Converting between dissimilar archive formats (such as tar and cpio) using the `@-` convention can cause hard link information to be lost. This is a consequence of the incompatible ways that different archive formats store hardlink information.