GNU tar archive tool reference by example

Table of Contents

by Yuri Slobodyanyuk https://www.linkedin.com/in/yurislobodyanyuk/

NOTE	All the examples below are for the Linux GNU tar, not for Solaris, FreeBSD, or Mac OS operating systems native versions of tar.

Archive and gzip-compress the current folder with tar

```
tar -czf gzipped-folder.tar.gz  .
```

Here:

- `c` For *create*
- `z` For *gzip* compress
- `f` Filename of the archive to create
- `.` (dot) for the current folder

The file `gzipped-folder.tar.gz` will contain all the files (including dot files) and subfolders of the current folder.

Archive and gzip-compress the current folder using maximal compression possible

There are few ways to do it. The older versions of `tar` do not accept compression level for the `gzip`, so we have to hint to the `gzip` in other way.

Set compression level as the `GZIP` environmental variable for `gzip`

Let's set the maximum compression level of 9:

```
GZIP=-9 tar -cvzf maxcompression.tar.gz .
```

NOTE	Disadvantage of this method is that it depends on the shell you are using. It works for Bash, but may fail to work in other shells.

Set compression level by piping `tar` output to the `gzip`

Most starightforward way to do it:

```
tar -cvf  -  . | gzip -9 - > maxcompression.tar.gz
```

Variation of the above:

```
tar -cvf maxcompression.tar ; gzip -9 maxcompression.tar
```

Use `-I` option for modern versions of tar

This option `I` or `--use-compress-program` appeared somewhere in version 1.22 or earlier, year of 2009. So, if your tar is newer than that (most probably is), you can change compression level:

```
tar -I 'gzip -9' -cvf maxcompression.tar.gz .
```

`I` sends its arguments in quotes as options to the compression program of choice as is.

Archive and bzip2-compress the current folder with tar

Same as the above, but use `bzip2` compression instead of the `gzip`. In the past the bzip2 compression produced smaller size archives compared to the gzip, but today they perform pretty much the same.

```
tar -cjf gzipped-folder.tar.bz2  .
```

Here:

- `c` For *create*
- `j` For *bzip2* compress
- `f` Filename of the archive to create

The file `gzipped-folder.tar.bz2` will contain all the files (including dot files) and subfolders of the current folder.

Archive the current folder but exlude specific file and/or subfolder

WARNING	Even though not explicitly mentioned in the tar's man - except for the newest versions, you HAVE to put the folder/path to work on as the LAST argument on the line, or `--exclude` will be ignored.

E.g. create an archive named `tared-folder.tar` to include all files/subfolders of the current folder except the file named `cookbook.gzip` and subfolder and its contents named `.git`:

```
tar -cvf tared-folder.tar  --exclude=cookbook.gzip --exclude=.git  .
```

`v` is for verbose output during the operation.

List contents of a tar archive (gzipped or not) without actually extracting it

Use `-t` option before the `f`:

```
tar -tf  gzipped-folder.tar.gz
```

Create a tar archive embedding the current day, month, and year in the name

When running tar as scheduled/cron-ed job, it is benefitial to include date of the archive creation in the name.

E.g.: create a tar archive named *backup-<current date>.tar* from files in the current folder ending in `*.md`:

```
tar -cf backup-`date +%d-%m-%Y`.tar   *.md
```

Result:

```
ls *.tar
backup-13-07-2021.tar
```

NOTE Look at the `man date` for more options, like hour, second etc.

Append file(s) to the existing archive

The file(s) will be appended at the end of the archive, just so you know.

E.g. let's append to the existing *backup-13-07-2021.tar* archive the file named *missfont.log*:

```
tar -rf backup-13-07-2021.tar missfont.log
```

Move the current directory and all of its contents as a whole, keeping file permissions

An old trick to compensate for various deficiencies of `cp` and `mv`.

```
tar -cf - . | (cd new-location; tar xvpf -)
```

Encrypt/Decrypt the resulting archive with OpenSSL and password

We just pipe the tar output to the OpenSSL, provided it is already installed. The password is given interactively in the CLI, so this is not very secure way to do so.

E.g. tar the current folder into tar archive and the encrypt it:

```
tar -cvf - * | openssl  enc -e -aes256 -out encrypted-dolder.tar.enc
enter aes-256-cbc encryption password:
Verifying - enter aes-256-cbc encryption password:
*** WARNING : deprecated key derivation used.
Using -iter or -pbkdf2 would be better.
```

Now, decrypt it:

```
openssl enc -d -aes256 -in  encrypted-folder.tar.enc | tar -xf  -
enter aes-256-cbc decryption password:
```

Extract only specific file(s) from the tar archive

We may specify a specific filename to extrtact or use shell globbing patterns for file name matching.

E.g.: extract only file named *README.md* from the archive tar *cookbooks.tar.bz2*:

```
tar -xjvf cookbooks.tar.bz2 ./README.md
```

E.g.: extract all Markdown files from the archive:

```
tar -xjvf cookbooks.tar.bz2 ./*.md
```

NOTE	`-j` is to extract from bzip2-compressed archive, if extracting from plain tar archive just remove -j

Archive directory on the remote server and download to the local host via SSH in one command

Task: add to tar archive and compress contents of the directory *ASM* on the remote server 19.23.55.158 and download it to the local host as file *ASM.tar.gz*

```
ssh root@19.23.55.158 'cd ASM && tar -czf - *' > ASM.tar.gz
root@19.23.55.158's password:
```

Result:

```
ls -l
-rw-r--r-- 1 root root      505 Jul 14 08:39 ASM.tar.gz
```

Here:

- `ASM` - relative path of the directory on the remote server, using absolute path is recommended.
- `tar -czf -` - creates gzip-compressed tar archive with stdout being the output device so we can redirect output on local server to the file *ASM.tar.gz*

Remove / do not preserve / anonymize username and group name of the files owner when adding files to tar archive

By default tar will add files/directories to the archive along with their owner user/group. The only reliable way to prevent this is to replace actual data with fake user/group when adding to the archive.

E.g. Add file *README.md* to the archive, but change the owner's username/group to the fictitious *Doe* with numeric id of *1002*. If we supply just username/group name, then depending on version/implementation, the tar may change them as asked but leave the real numeric IDs. To force tar not to do it, specify both - alphanumeric name and numeric ID or add beyond numeric IDs the option `--numeric-owner`, which forces tar to keep only numeric IDs.

NOTE tar does not check if the given user and group name actually exist on the system.

```
tar -cvf perms.tar README.md --owner=Doe:1002 --group=Doe:1002
```

Verify:

```
tar -vtf perms.tar
-rw-r--r-- Doe/Doe         542 2020-08-22 09:50 README.md
```

Delete only specific file(s) or folder(s) from the archive

Not really possible. There is `--delete` option that seemingly does this, but under the surface this option just combines extracting the whole archive to the temporary directory, deleting the file(s) in question, and creating the archive again from scratch into one command.

How can I run tar in parallel on multi-core CPU when creating an archive?

The short answer - you can't. The extended answer - you can't archive in parallel to the same archive (it was never the goal of `tar`, which originally wrote archives to the physical tapes that could not be accessed in parallel), but you have options (if you need at all) to parallelize compression of the archive. The options for parallel execution depend on the compressing utility used. There are `xz`, `7zip`, and `pigz` tools which can compress an archive in parallel, given the correct options. But they cannot decompress in parallel way though, only to compress.

Find all tar archives even those NOT having .tar extension

In situation where you are presented with a bunch of files with random names, finding which ones are proper tar archive can be done in few ways. The idea behind all of them is to look for the tar's **magic number** inside the file. On systems with `file` utility installed, it is really easy:

```
# file * | awk -F: '/POSIX tar archive/ {print $1}'

damaged.tar
deleteme-13-07-2021.tar
maxwithI.tar.gz
perms.tar
permstar
permstar2
```

As you can see, it found tar archives without any extension *permstar* and *permstar2*.

When the `file` tool is not available (highly unprobable), we can go more old school way looking at the magic number:

```
find .  -type f    -exec  xxd  -g 6  -s 257 -l 6 \{\} \; -print | sed -n
'/757374617220/{n;p}'
./perms.tar
./maxwithI.tar.gz
./damaged.tar
./deleteme-13-07-2021.tar
./test/deleteme-13-07-2021.tar
./permstar2
./permstar
```

Here:

- 757374617220 is the magic number for the tar filetype
- `xxd` is hex dumper to show contents of a file in hexadecimal
- `-g 6` tells xxd to group the found bytes into a group of 6 bytes (size of the magic number) when printing
- `-l 6` limits output to just 6 bytes
- `-s 257` skips first 256 bytes to start printing from byte 257 forward

tar archives symlinks instead of the objects they point to, how to fix?

Use `-h` switch to tell tar to dereference symlinks and add to archive objects (directories/files) that those symlinks point to.

```
tar -hcf .
```

This will dereference all symlinks found in the current directory.

Archive only those objects modified last 24 hours

Tar itself does not have option to search by timestamps, but `find` does.

```
find .  -mtime 0 -print0 | tar -cvf modified.tar --null -T -
```

Here:

- `-mtime` tells `find` what modification timestamps of the objects we are looking for, in days. The `0` means "0 days ago", i.e. last 24 hours. This option accepts relative values as well. E.g. `-2` means modified less than 2 days ago. And `-mtime +2` will find objects modified earlier than 2 days ago. See below for another example.

Archive only those objects modified between 24 and 48 hours ago

The extension of the above. In general, `find` is such an essential tool, that you can't do much without it in any Linux/BSD/Unix system.

```
find . -mtime 1 -print0 | tar -cvf modified.tar --null -T -
```

NOTE	To search for modified times in minute resolution, use `-mmin` instead of `-mtime`.

Verify tar archive integrity in a Bash script, i.e. non interactively

Tar itself does not calculate/save checksum in the archive it creates. The rudimentary "integrity" check can be done with `-t` switch, which produces an error and exits if the archive is severely damaged - cannot be read, headers are mangled and such. The change in the **contents** of a file this `-t` check will NOT notice. When gzip-ing tar archive, though, the CRC checksum is autosaved, but of the final tar archive, not individual files inside this archive. This way, if there is a CRC checksum mismatch on unzipping tar archive, the `gzip` will issue an error on the standard output.

So, to try and read the archive, verifying that it is readable:

```
#!/bin/bash
if ! tar tf /path/to/archive.tar &> /dev/null; then # Here we check the EXIT status of
reading a tar archive, also redirecting stdout to the /dev/null, as no need to see the
contents of archive
    do_something_if_exit_status_is_error
fi
```