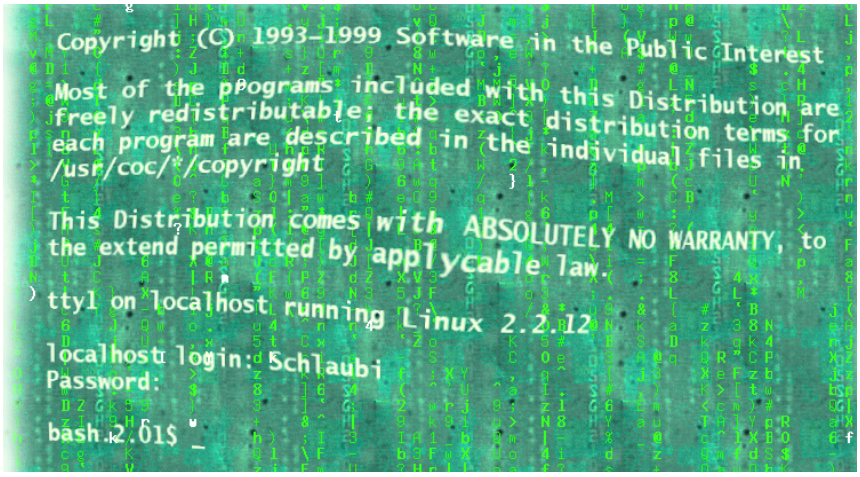
# Compression tools

# TAR AND GZIP

BY HEIKE JURZIK

**Although graphical interfaces such as KDE or GNOME are a big help, those who want to fully exploit Linux can't avoid the command line. Besides, there are many situations where it is beneficial to have a firm grip of the jungle-like syntax of the $ prompt.**




In the world of Linux, the extensions *.gz*, *.tar.gz* or *.bz2* are perhaps the commonest of all – and not just when downloading files or accessing **HOWTOS**. We discuss below these cryptic file extensions and tell you how to pack and unpack files and directories.

## Gripping

Put simply, the *gzip file(s)* command shrinks files. The resulting compressed file is called *file* .gt and retains the same access and ownership rights, along with access and modification time attributes. If the file name is too long for the file system, *gzip* truncates it – lengthy parts of the file name are shortened.

If you want to restore a compressed file you can use *gunzip* or *gzip -d* (short for *gzip --decompress*), which is actually the same program. If, on the other hand, you just want to view the packed file, you can type *zcat file.gz* (if necessary adding | *less* to the end, or you could use *zless file.gz* directly), which is the same as *gzip -c -d*. The option *-c*, incidentally, has the effect of decompressing the file to **stdout**.

The degree of compression depends on the size of the input file and the quantity of repeated character strings. That is to say, files best suited to compression are those in which similar data patterns are repeated often. For example, a 1.4 MB bitmap file might be reduced to just 709 KB after *gzip* has been applied. If you use the parameter *gzip -9*, the resulting file size is just 708 KB. By appending a digit in the range 1-9, you can decide whether you prefer compression to be faster (*gzip -1*) but not as compact, or slower but with a better degree of compression (*gzip -9*).

*gzip* also has some further options that are of interest. If, for example, a file bearing the same name already exists in the current directory, *gzip* politely asks:

```
user@host ~ > gunzip file.bmp.gz
gunzip: file.bmp already exists; do you wish ➥
to overwrite (y or n)?
```

If you want to avoid the query, use the *gzip -f* option (for *--force*). This parameter steams through packing and unpacking even if files of the same name already exist. Of interest here is the behaviour of **Symbolic links (Symlinks)**. Normally, *gzip* will decline a request to compress Symlinks by saying *gzip: link.bmp is not a directory or a regular file - ignored*. If you use the *-f* option, the file to which the link is pointing is compressed, but is given the name of the link, i.e. *link.bmp.gz*.

The *gzip* command has many more features – a quick glance at the man page will provide a good overview. If you want to have your own special *gzip* equipped with a few options as standard, you can enter this in the **environment variable** *GZIP*. For the bash shell you can define your own preferred parameters as follows, for example:

```
user@host ~ > export GZIP="-9"
user@host ~ > echo $GZIP
-9
```

## Shrinking things further?

Greater compression is possible courtesy of *bzip2*. This not only compresses better than *gzip* but is significantly faster as well. On top of this, it has a *recover* mode which means it attempts to repair possible damage to compressed files, or to unpack just the undamaged parts. Before we start to discuss this, check to see if you have *bzip2* on your system. If not you can find not only the source but also extensive information about the program at http://sourceware.cygnus.com/bzip2/index.html .

Most of the parameters work in exactly the same way as *gzip* but some differ. The extension for compressed files here is called *.bz2*. Access rights and timestamps are likewise maintained, and here, too, you're not allowed to overwrite files. If, on unpacking, an attempt is made to overwrite an existing file, no query is issued as with *gzip*. Instead you get the message: *bunzip2: Output file datei.bmp already exists, skipping*. If you want to circumvent this, then, as with *gzip*, use the option *-f* (for *--force*).

The actual differences with *bzip2* are subtle. For example, the file to be compressed need not be automatically deleted, you can keep a copy by typing *bzip2 -k* (for *--keep*). The feature *bzip2recover* has already been mentioned but what takes place when it runs is quite interesting. During compression *bzip2* decompresses files into separate blocks. Thus should a file be damaged for any reason, the data contained in the blocks that remain intact can be rescued if necessary (for more precise details of this you should refer to the man page).

## *tar* for the archive, mate!

One of the things you can do with tar combine several files into an archive - handy if you want to transfer many associated items from one computer to another. This remaining, single archive file can then be compressed more easily as a single entity. To create such an archive, enter the following:

```
user@host ~ > tar cvf archiv.tar direc➥
tory
directory/
directory/file.html
directory/test/
directory /test/file2
directory/text
```

If we break it down into the separate parameters we see the following: the option *c* stands for *--create* – that is, create a new archive. If one decides to get by without the *v* for *--verbose*, no file names are specified during the archiving process. The three letters *tar* incidentally stand for "*T*ape *AR*chiver": originally the program was intended for backing up to tape drives. That is why *f archive-name* is used here, to indicate that tar shall not write such a device but to a file on the hard disk. Following this, of course, a name has to be specified – *archive.tar*. All the directories, files and subdirectories are written to the archive file.

If you want to append further files to the existing archive you can invoke *tar rvf archiv.tar furtherfile* – where *r* stands for *--append*. To be certain that this file doesn't already exist, you can of course view the archive beforehand: *tar tvf archive.tar*, where *t* can also be replaced by the long form *--list*. If access rights and ownership are to be maintained you should use *tar pcvf archive.tar /home* (*p* stands for *--preserve-permissions*) – if the directories are unpacked again the files will be restored in their original state.

To unpack an archive once again you can use *tar xvf archive.tar*, where *x* stands for *--extract*. If individual files are to be extracted from the archive you can append their names when making this call. Finally, bear in mind that *tar* does not compress automatically. Naturally, an archive of this type can still be packed using *gzip* or *bzip2*, but you can avoid this second step and deal with everything in one go: *tar czvf archive.tar.gz directory* also zips the archive at the same time (in reality the external program *gzip* is invoked). In the same way, *z* is to be used to unpack a compressed package of this type – the command is *tar xzvf archive.tar.gz*. If you prefer to use *bzip2* instead of *gzip* you should check beforehand whether your own particular distribution provides for this (read the man page!). Invoking *tar clvf test.tar.bz2 directory* (with a capital 'i') worked on the test computer under Debian 2.1, although a second computer declined the instruction: *tar: invalid option – Try `tar --help' for more information.* (hge) ■

***HOWTOS:*** *In contrast to man pages, which are created mainly for reference, HOWTOS provide instructions on how to overcome specific problem areas and are thus much more geared towards the beginner. In current distributions they are to be found under /usr/doc/HOWTO. You can find, for example, the file Firewall-HOWTO.gz, which you can unpack and then read, or else view directly with zless or zmore.*

***stdout:*** *There are three standard channels for input and output: stdin (standard input), stdout (standard output) and stderr (standard error output). A user, for example, has the keyboard as standard input and the screen as standard output. If you decompress a file using zcat (gzip -d -c), then, providing it has not been redirected, it will be output to the screen.*

***Symbolic link:*** *A reference to another file which is handled by a certain application. If a file that a Symlink points to is deleted, the link is left with an empty pointer. Symlinks are created using the ln -s command.*

***Environment variable:*** *The shell provides the user with storage space for saving certain information which can then be accessed by programs. These environment variables each comprise the name of the variable and the value assigned to it.*