Tutorial Link https://course.testpad.chitkara.edu.in/tutorials/Linux - Compressing Files/60cc8151cfaf5f6628f2d401

TUTORIAL

Linux - Compressing Files

Topics

Backing Up Data

There are many ways you can back up data or even your entire system. Basic ways to do so include the use of simple copying with cp and use of the more robust rsync.

Both can be used to synchronize entire directory trees. However, rsync is more efficient, because it checks if the file being copied already exists. If the file exists and there is no change in size or modification time, rsync will avoid an unnecessary copy and save time. Furthermore, because rsync copies only the parts of files that have actually changed, it can be very fast.

cp can only copy files to and from destinations on the local machine (unless you are copying to or from a filesystem mounted using NFS), but rsync can also be used to copy files from one machine to another. Locations are designated in the target:path form, where target can be in the form of someone@host. The someone@ part is optional and used if the remote user is different from the local user.

rsync is very efficient when recursively copying one directory tree to another, because only the differences are transmitted over the network. One often synchronizes the destination directory tree with the origin, using the -r option to recursively walk down the directory

tree copying all files and directories below the one listed as the source.

rsync:

rsync is a very powerful utility. For example, a very useful way to back up a project directory might be to use the following command:

```
$ rsync -r project-X archive-machine:archives/project-X
```

Note that rsync can be very destructive! Accidental misuse can do a lot of harm to data and programs, by inadvertently copying changes to where they are not wanted. Take care to specify the correct options and paths. It is highly recommended that you first test your rsync command using the -dry-run option to ensure that it provides the results that you want.

To use rsync at the command prompt, type rsync sourcefile destinationfile, where either file can be on the local machine or on a networked machine; The contents of sourcefile will be copied to destinationfile. A good combination of options is shown in:

```
$ rsync --progress -avrxH  --delete sourcedir destdir
```

Compressing Data

File data is often compressed to save disk space and reduce the time it takes to transmit files over networks. Linux uses a number of methods to perform this compression, including:

```
Command     Usage
gzip         The most frequently used Linux compression utility
bzip2         Produces files significantly smaller than those
produced by gzip
xz         The most space-efficient compression utility used in
Linux
zip         Is often required to examine and decompress archives
from other operating systems
```

These techniques vary in the efficiency of the compression (how much space is saved) and in how long they take to compress; generally, the more efficient techniques take longer. Decompression time does not vary as much across different methods. In addition, the tar utility is often used to group files in an archive and then compress the whole archive at once.

gzip:

gzip stands for GNU Zip. It is used to compress a file or set of files in .gz format. To decompress the file back we can use gunzip command. It will replace the original file with the zipped file. We can combine multiple files and zip them in a single file using this utility.

gzip is the most often used Linux compression utility. It compresses very well and is very fast. The following table provides some usage examples:

```
Command         Usage
gzip *             Compresses all files in the current directory;
each file is compressed and renamed with a .gz extension
gzip -r projectX     Compresses all files in the projectX
directory, along with all files in all of the directories under
projectX
gunzip foo         De-compresses foo found in the file foo.gz.
Under the hood, the gunzip command is actually the same as gzip –d
```

For example,

```
$ ls
repwords.txt  abc2.txt  abc.txt   c  file1.txt
$ gzip repwords.txt
$ ls
repwords.txt.gz  abc2.txt  abc.txt   c  file1.txt
```

The file has been compressed and placed in the same directory. We can combine multiple files in a single zipped file as below with redirection to a new file name: -

```
$ cat file1 file2 file 3 | gzip > newfile
```

We can unzip the files using gunzip command:

```
$ gunzip repwords.txt.gz
```

bzip2:

bzip2 has a syntax that is similar to gzip but it uses a different compression algorithm and produces significantly smaller files, at the price of taking a longer time to do its work. Thus, it is more likely to be used to compress larger files. Examples of common usage are also similar to gzip:

```
Command         Usage
bzip2 *         Compresses all of the files in the current
directory and replaces each file with a file renamed with a .bz2
extension
bunzip2 *.bz2         Decompresses all of the files with an
extension of .bz2 in the current directory. Under the hood,
bunzip2 is the same as calling bzip2 -d
```

xz:

xz is the most space efficient compression utility used in Linux and is now used to store archives of the Linux kernel. Once again, it trades a slower compression speed for an even higher compression ratio. Some usage examples:

```
Command     Usage
$ xz *         Compresses all of the files in the current
directory and replaces each file with one with a .xz extension
xz foo         Compresses the file foo into foo.xz using the
default compression level (-6), and removes foo if compression
succeeds
xz -dk bar.xz     Decompresses bar.xz into bar and does not
remove bar.xz even if decompression is successful
xz -dcf a.txt b.txt.xz > abcd.txt     Decompresses a mix of
compressed and uncompressed files to standard output, using a
single command
$ xz -d *.xz     Decompresses the files compressed using xz
```

Compressed files are stored with a .xz extension.

zip:

The zip program is not often used to compress files in Linux, but is often required to examine and decompress archives from other operating systems. It is only used in Linux when you get a zipped file from a Windows user. It is a legacy program.

```
Command         Usage
zip backup *         Compresses all files in the current
directory and places them in the file backup.zip
zip -r backup.zip ~     Archives your login directory (~) and all
files and directories under it in the file backup.zip
unzip backup.zip     Extracts all files in the file backup.zip
and places them in the current directory
```

tar:

Historically, tar stood for "tape archive" and was used to archive files to a magnetic tape. It allows you to create or extract files from an archive file, often called a tarball. At the same time, you can optionally compress while creating the archive, and decompress while extracting its contents. Here are some examples of the use of tar:

```
Command                Usage
$ tar xvf mydir.tar             Extract all the files in
mydir.tar into the mydir directory
$ tar zcvf mydir.tar.gz mydir         Create the archive and
compress with gzip
$ tar jcvf mydir.tar.bz2 mydir     Create the archive and
compress with bz2
$ tar Jcvf mydir.tar.xz mydir         Create the archive and
compress with xz
$ tar xvf mydir.tar.gz             Extract all the files in
mydir.tar.gz into the mydir directory. Note: You do not have to
tell tar it is in gzip format
```

You can separate out the archiving and compression stages, as in:

```
$ tar cvf mydir.tar mydir ; gzip mydir.tar
$ gunzip mydir.tar.gz ; tar xvf mydir.tar
```

but this is slower and wastes space by creating an unneeded intermediary .tar file.

gzip, bzip2 and xz:

To demonstrate the relative efficiency of gzip, bzip2, and xz, the following commands shows the results of compressing a purely text file directory tree (the include directory from the kernel source) using the three methods.

```
$ time tar zcf include.tar.gz include
$ time tar jcf include.tar.bz2 include
$ time tar Jcf include.tar.xz include
$ du -shc include include.tar.gz include.tar.bz2 include.tar.xz
```

This shows that as compression factors go up, CPU time does as well (i.e. producing smaller archives takes longer).

compress

compress command is used to reduce the file size. After compression, the file will be available with an added .Z extension. File permissions will still remain the same as before using the compress command. This command uses the adaptive Lempel-Ziv coding and it ignores the symbolic links.

If either the input and output files are not regular files, the checks for reduction in size and file overwriting are not performed, the input file is not removed, and the attributes of the input file are not retained in the output file.

Note: If no files are specified then the standard input is compressed to the standard output.

Syntax:

```
compress [OPTIONS] [FILE]
```

uncompress

On Linux operating systems, the uncompress command decompresses files that were compressed using the compress command.

The uncompress utility will restore files to their original state after they are compressed using the compress utility. If no files are specified, the standard input will be uncompressed to the standard output.

Files compressed with compress often have the extension .Z, and uncompress looks for and recognizes files with that extension as compressed files.

This utility supports the uncompressing of any files produced by compress.

Syntax:

```
uncompress [options] [files]
```