Chapter 7. Archiving Files

Goal

Create compressed archives of files so that they can be backed up and transferred to other systems.

Sections

- Managing Compressed Tar Archives (and Guided Exercise)

7.1. Managing Compressed Tar Archives

Objectives

- Describe compressing tar archives to list the files and directories and extract them.

Create Archives from the Command Line

An *archive* is a single regular file or device file that contains multiple files. The device file could be a tape drive, flash drive, or other removable media. When using a regular file, archiving is analogous to the `zip` utility and similar variations that are popular on most operating systems.

Archive files are used to create manageable personal backups, or to simplify transferring a set of files across a network when other methods, such as the `rsync` utility, are unavailable or might be more complex. Archive files can be created with or without using compression to reduce the archive file size.

Note

The original, ubiquitous `zip` compression and file packaging utility uses the PKZIP (Phil Katz's ZIP for MS-DOS systems) algorithm, and is supported on RHEL with the `zip` and `unzip` commands. Many other compression algorithms have been developed since `zip` was introduced, and each one has its advantages in speed or compression ratio. For creating compressed archives for general use, any `tar`-supported compression algorithm is acceptable.

On Linux, the `tar` utility is the common command to create, manage, and extract archives. Use the `tar` command to gather multiple files into a single archive file. A *tar archive* is a structured sequence of file metadata and data with an index so you can extract individual files.

Files can be compressed during creation by using one of the supported compression algorithms. The `tar` command can list the contents of an archive without extracting, and can extract original files directly from both compressed and uncompressed archives.

Options of the Tar Utility

One of the following `tar` command actions is required to perform a `tar` operation:

- `-c` or **`--create`** : Create an archive file.
- `-t` or **`--list`** : List the contents of an archive.
- `-x` or **`--extract`** : Extract an archive.

The following `tar` command general options are often included:

- `-v` or **`--verbose`** : Show the files that are being archived or extracted during the `tar` operation.
- `-f` or **`--file`** : Follow this option with the archive file name to create or open.
- `-p` or **`--preserve-permissions`** : Preserve the original file permissions when extracting.

- **`--xattrs`** : Enable extended attribute support, and store extended file attributes.
- **`--selinux`** : Enable SELinux context support, and store SELinux file contexts.

The following `tar` command compression options are used to select an algorithm:

- **`-a`** or **`--auto-compress`** : Use the archive's suffix to determine the algorithm to use.
- **`-z`** or **`--gzip`** : Use the `gzip` compression algorithm, which results in a `.tar.gz` suffix.
- **`-j`** or **`--bzip2`** : Use the `bzip2` compression algorithm, which results in a `.tar.bz2` suffix.
- **`-J`** or **`--xz`** : Use the `xz` compression algorithm, which results in a `.tar.xz` suffix.

Note

The `tar` command still supports the legacy option style that does not use a dash (`-`) character. You might find this syntax in legacy scripts or documentation, and the behavior is essentially the same. For command consistency, Red Hat recommends using the short- or long-option styles instead.

Create an Archive

To create an archive with the `tar` command, use the `-c` and `-f` options with the archive file name as the first argument, followed by a list of files and directories to include in the archive.

The `tar` command recognizes absolute and relative file name syntax. By default, tar removes the leading forward slash (`/`) character from absolute file names, so that files are stored internally with relative path names. This technique is safer, because extracting absolute path names always overwrites existing files. With files that are archived with relative path names, files can be extracted to a new directory without overwriting existing files.

The following command creates the `mybackup.tar` archive to contain the `myapp1.log`, `myapp2.log`, and `myapp3.log` files from the user's home directory. If a file with the same name as the requested archive exists in the target directory, then the `tar` command overwrites the file.

```
user@host:~$ tar -cf mybackup.tar myapp1.log myapp2.log myapp3.log
```

A user must have read permission on the target files that are being archived. For example, creating an archive in the `/etc` directory requires `root` privileges, because only privileged users can read all `/etc` files. An unprivileged user can create an archive of the `/etc` directory, but the archive excludes files that the user cannot read, and excludes directories for which the user lacks the read and execute permissions.

In this example, the `root` user creates the `/root/etc-backup.tar` archive of the `/etc` directory.

```
root@host:~# tar -cf /root/etc-backup.tar /etc
tar: Removing leading `/' from member names
```

Important

Extended file attributes, such as access control lists (ACL) and SELinux file contexts, are not preserved by default in an archive. Use the `--acls` option to include POSIX ACLs; use the `--selinux` option to include SELinux file contexts; or use the `--xattrs` option to include other extended attributes.

Although extended attributes are outside the scope of this lesson, you can find more information in the references section.

List Archive Contents

Use the `tar` command `-t` option to list the file names from within the archive that is specified with the `-f` option. The files are listed with relative name syntax, because the leading forward slash was removed during archive creation.

```
root@host:~# tar -tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

Extract Archive Contents

Extract a tar archive to an empty directory to avoid overwriting existing files. When the `root` user extracts an archive, the extracted files preserve the original user and group ownership. If a regular user extracts files, then the user becomes the owner of the extracted files.

Create the `/root/etcbackup` directory to extract the tar archive.

```
root@host:~# mkdir /root/etcbackup
```

List the file names in the `/root/etc.tar` archive.

```
root@host:~# tar -tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

Extract the archive to the `/root/etcbackup` directory:

```
root@host:~/etcbackup# tar -xf /root/etc.tar
```

When you extract files from an archive, the current `umask` is used to modify each extracted file's permissions. Instead, use the `tar` command `-p` option to preserve the original archived permissions for extracted files. The `-p` option is enabled by default for a superuser.

```
user@host:~/scripts$ tar -xpf /home/user/myscripts.tar
```

Create a Compressed Archive

The `tar` command supports these compression methods, and others:

- The **gzip** compression is the earlier, fastest method, and is widely available across platforms.
- The **bzip2** compression creates smaller archives but is less widely available than `gzip`.
- The **xz** compression is later, and offers the best compression ratio of the available methods.

The effectiveness of any compression algorithm depends on the type of data that is compressed. Previously compressed data files, such as picture formats or RPM files, typically do not significantly compress further, regardless of the compression algorithm that is used.

Create the `/root/etcbackup.tar.gz` archive with `gzip` compression from the contents of the `/etc` directory:

```
root@host:~# tar -czf /root/etcbackup.tar.gz /etc
tar: Removing leading `/' from member names
```

Create the `/root/logbackup.tar.bz2` archive with `bzip2` compression from the contents of the `/var/log` directory:

```
root@host:~# tar -cjf /root/logbackup.tar.bz2 /var/log
tar: Removing leading `/' from member names
```

Create the `/root/sshconfig.tar.xz` archive with `xz` compression from the contents of the `/etc/ssh` directory:

```
root@host:~# tar -cJf /root/sshconfig.tar.xz /etc/ssh
tar: Removing leading `/' from member names
```

After creating a compressed archive, you can still verify its table of contents with the `tar` command `-tf` options. It is not necessary to specify the compression option when listing a compressed archive file, because the compression type is read from the archive's header.

List the archived content in the `/root/etcbackup.tar.gz` file, which uses the `gzip` compression:

```
root@host:~# tar -tf /root/etcbackup.tar.gz
etc/
etc/fstab
etc/crypttab
etc/mtab
...output omitted...
```

Extract Compressed Archive Contents

The `tar` command can automatically determine which compression was used, so it is not necessary to specify the compression option. If you include an incorrect compression type, then the `tar` command reports that the specified compression type does not match the file's type. Listing a compressed `tar` archive works in the same way as listing an uncompressed `tar` archive.

In the following example, the `tar` command uses the `-z` option, which indicates `gzip` compression, but the file name extension is `.xz`, which indicates `xz` compression:

```
root@host:~# tar -xzf /root/etcbackup.tar.xz

gzip: stdin: not in gzip format
tar: Child returned status 1
tar: Error is not recoverable: exiting now
```

The `gzip`, `bzip2`, and `xz` algorithms are also implemented as stand-alone commands for compressing individual files without creating an archive. With these commands, you cannot create a single compressed file of multiple files, such as a directory. As previously discussed, to create a compressed archive of multiple files, use the `tar` command with your preferred compression option. To uncompress a single compressed file or a compressed archive file without extracting its contents, use the `gunzip`, `bunzip2`, or `unxz` stand-alone commands.

The `gzip` and `xz` commands provide the `-l` option to view the uncompressed size of a compressed single or archive file. Use this option to verify that enough space is available before uncompressing or extracting a file.

Use the `gzip` command to view the uncompressed size of the `file.tar.gz` archive.

```
user@host:~$ gzip -l file.tar.gz
         compressed        uncompressed  ratio uncompressed_name
          221603125           303841280  27.1% file.tar
```

Use the `xz` command to view the uncompressed size of the `file.tar.gz` archive.

```
user@host:~$ xz -l file.tar.xz
Strms  Blocks   Compressed Uncompressed  Ratio  Check   Filename
    1       1    195.7 MiB    289.8 MiB  0.675  CRC64   file.xz
```

References

tar(1), gzip(1), gunzip(1), bzip2(1), bunzip2(1), xz(1), unxz(1), xattr(7) and acl(5) man pages

7.2. Guided Exercise
Manage Compressed Tar Archives

Create a compressed tar archive, list the files and directories that are stored in it, and extract them.

Outcomes

- Archive a directory tree and extract the archive content to another location.

Prerequisites

As the `student` user on the `workstation` machine, run the following `lab` command to prepare your environment for this exercise, and to ensure that all required resources are available:

```
student@workstation:~$ lab start archive-manage
```

Instructions

1. From the `workstation` machine, log in to the `servera` machine as the `student` user and switch to the `root` user. Use `student` as the password.

```
student@workstation:~$ ssh student@servera
...output omitted...
[student@servera ~]$ sudo -i
[sudo] password for student: student
[root@servera ~]#
```

2. Create archives of the `/etc` directory, both without compression, and with the `gzip`, `bzip2`, and `xz` compression algorithms. Compare the sizes of the resulting archive files.

2.1. Create a `tar` archive of the `/etc` directory. Save the archive file as the `/tmp/etc.tar` archive.

```
[root@servera ~]# tar -cf /tmp/etc.tar /etc
...output omitted...
```

2.2. Create a `tar` archive of the `/etc` directory that is compressed by the `gzip` compression algorithm. Save the archive file as the `/tmp/etc.tar.gz` archive.

```
[root@servera ~]# tar -czf /tmp/etc.tar.gz /etc
...output omitted...
```

2.3. Create a `tar` archive of the `/etc` directory that is compressed by the `bzip2` compression algorithm. Save the archive file as the `/tmp/etc.tar.bz2` archive.

```
[root@servera ~]# tar -cjf /tmp/etc.tar.bz2 /etc
...output omitted...
```

2.4. Create a `tar` archive of the `/etc` directory that is compressed by the `xz` compression algorithm. Save the archive file as the `/tmp/etc.tar.xz` archive.

```
[root@servera ~]# tar -cJf /tmp/etc.tar.xz /etc
...output omitted...
```

2.5. Use the `ls -lh` command to compare the sizes of the resulting archive files.

```
[root@servera ~]# ls -lh /tmp/etc.tar*
-rw-r--r--. 1 root root  22M Jun  4 11:13 /tmp/etc.tar
-rw-r--r--. 1 root root 4.7M Jun  4 11:54 /tmp/etc.tar.bz2
-rw-r--r--. 1 root root 5.5M Jun  4 11:30 /tmp/etc.tar.gz
-rw-r--r--. 1 root root 4.1M Jun  4 11:56 /tmp/etc.tar.xz
```

3. Verify that the `etc.tar.gz` archive contains the files from the `/etc` directory.

```
[root@servera ~]# tar -tzf /tmp/etc.tar.gz
etc/
etc/xdg/
etc/xdg/autostart/
etc/xdg/systemd/
etc/xdg/systemd/user
...output omitted...
```

4. Create the `/backuptest` directory. Verify that the `etc.tar.gz` backup file is a valid archive by decompressing the file to the `/backuptest` directory.

4.1. Create the `/backuptest` directory and change to that directory.

```
[root@servera ~]# mkdir /backuptest
[root@servera ~]# cd /backuptest
[root@servera backuptest]#
```

4.2. Extract the `/tmp/etc.tar.gz` archive to the `/backuptest` directory.

```
[root@servera backuptest]# tar -xzf /tmp/etc.tar.gz
```

4.3. List the contents of the `/backuptest` directory.

```
[root@servera backuptest]# ls -l
total 12
drwxr-xr-x. 112 root root 8192 Jun  4 12:30 etc
```

4.4. Verify that the directory contains the `/etc` directory backup files.

```
[root@servera backuptest]# ls -l etc
total 1228
-rw-r--r--.  1 root root   5923 Nov 26  2024 DIR_COLORS
-rw-r--r--.  1 root root   6005 Nov 26  2024 DIR_COLORS.lightbgcolor
-rw-r--r--.  1 root root     94 Oct 29  2024 GREP_COLORS
drwxr-xr-x.  7 root root    134 Apr 23 14:35 NetworkManager
drwxr-xr-x.  2 root root     48 Apr 23 14:35 PackageKit
drwxr-xr-x.  5 root root     51 Apr 23 14:35 X11
-rw-r--r--.  1 root root     12 Feb 13 00:00 adjtime
-rw-r--r--.  1 root root   1529 Nov 29  2023 aliases
...output omitted...
```

5. Return to the `workstation` machine as the `student` user.

```
[root@servera backuptest]# exit
logout
[student@servera ~]$ exit
logout
Connection to servera closed.
student@workstation:~$
```

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
student@workstation:~$ lab finish archive-manage
```

7.3. Summary

In this lesson, you learned about tools to archive and compress files and directories.

The `tar` utility is a versatile command to create, list, and extract archive files. It can handle both single files and entire directories, and can preserve metadata and special attributes. The `tar` utility can also compress archives by using the `gzip`, `bzip2`, and `xz` compression algorithms.

The `gzip`, `bzip2`, and `xz` compression tools can also be used as stand-alone commands. Each of these tools has its own compression algorithm, with varying levels of compression and speed that depend on the data that is compressed.