ARCHIVAR Y COPIAR ARCHIVOS ENTRE SISTEMAS

RH124

Capitulo 12

Visión general:	
Meta	Archivar y copiar archivos de un sistema a otro.
Objetivos	• Usar TAR para crear documentos de archivos comprimidos nuevos y extraer documentos desde documentos de archivos existentes. • Copiar archivos en forma segura desde o hacia un sistema remoto que ejecuta sshd. • Sincronizar en forma segura el contenido de un archivo o directorio local con una copia remota.
Secciones	• Administración de archivos TAR comprimidos (y práctica) • Copia de archivos entre sistemas en forma segura (y práctica) • Sincronización de archivos entre sistemas en forma segura (y práctica)
Trabajo de laboratorio	• Archivar y copiar archivos entre sistemas

Administración de archivos tar comprimidos

Objetivo

Tras finalizar esta sección, los estudiantes deberían poder usar tar para crear ficheros de archivos comprimidos nuevos y extraer ficheros desde ficheros de archivos existentes.

¿Qué es tar?

El archivado y la compresión de archivos es útil cuando se realizan copias de seguridad y se transfieren datos a través de una red. Uno de los comandos más antiguos y más usados para crear y trabajar con archivos de copias de seguridad es el comando **`tar`**.

Con el comando **`tar`**, los usuarios pueden reunir grandes conjuntos de ficheros en un solo fichero (archivo). El archivo puede comprimirse con **`gzip`**, **`bzip2`** o **`xz`**.

El comando tar también puede enumerar el contenido de los archivos o extraer sus ficheros a su sistema actual. En esta sección, se incluyen ejemplos de cómo usar el comando **`tar`**.

Uso del comando `tar`

Para usar el comando **`tar`**, es necesario realizar una de las tres acciones que se indican a continuación:

- **`c`** (cree un archivo)
- **`t`** (enumere el contenido de un archivo)
- **`x`** (extraiga un archivo)

Las opciones más usadas son:

- **`f`** ***`file name`*** (nombre del fichero del archivo en el que se trabajará)
- **`v`** (exceso de palabras; se utiliza para ver qué ficheros se agregan o extraen del archivo)

nota

Un **`-`** inicial no es necesario para las opciones de tar.

Archivar ficheros y directorios con `tar`

Antes de crear un archivo tar, verifique que no haya otro archivo en el directorio con el mismo nombre del archivo que se creará. El comando **`tar`** sobrescribirá el archivo existente sin ningún comentario.

La primera opción para usar cuando se crea un archivo nuevo es la **`c`**, seguida de la opción **`f`**, luego un solo espacio, el nombre del fichero del archivo que se creará y, por último, la lista de los ficheros y directorios que deben agregarse al archivo. El archivo se crea en el directorio actual, a menos que se especifique lo contrario.

En el siguiente ejemplo, se crea un archivo con el nombre **`archive.tar`** que contiene fichero1, fichero2 y fichero3 en el directorio de inicio del usuario.

```
[user@host ~]# tar cf archive.tar file1 file2 file3
[user@host ~]# ls archive.tar
archive.tar
```

nota

Cuando se archivan ficheros con nombres de ruta absolutos, el **/** inicial de la ruta se elimina del nombre del archivo de manera predeterminada. Esta acción ayuda a evitar errores que pueden causar que se sobrescriban ficheros importantes. Por lo general, los ficheros se extraen según el directorio de trabajo actual del comando tar.

Para que **tar** pueda archivar los ficheros seleccionados, es obligatorio que el usuario que ejecute el comando **tar** pueda leer los ficheros. Por ejemplo, la creación de un archivo nuevo de la carpeta **/etc** y todo su contenido requiere privilegios de usuario raíz porque solo este tipo de usuario tiene permitido leer todos los archivos que están en esta carpeta. Un usuario sin privilegios podría crear un archivo de la carpeta /etc, pero el archivo omitiría los ficheros que no incluyan el permiso de lectura para el usuario y omitiría los directorios que no incluyan permiso de lectura y ejecución para el usuario.

no incluyan permiso de lectura y ejecución para el usuario.

Como usuario raíz, cree el archivo tar **/root/etc.tar** con el directorio **/etc** como contenido:

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

Importante

Si bien **tar** almacena la propiedad y los permisos de los archivos, existen otros atributos que no se almacenan en el archivo **tar** de manera predeterminada, como el contexto SELinux y las ACL. Para almacenar esos atributos ampliados en el archivo **tar**, se necesita la opción **--xattrs** cuando se crea un archivo.

Enumeración del contenido de un archivo tar

Para enumerar el contenido de un archivo, se necesitan las opciones **t** y **f**, además del archivo en que se trabajará.

Enumere el contenido del archivo **/root/etc.tar**:

```
[root@host ~]# tar tf /root/etc.tar
etc/
etc/fstab
etc/crypttab
etc/mtab
...
```

Extracción de un archivo creado con `tar`

Por lo general, un archivo tar debería extraerse en un directorio vacío para garantizar que no sobrescriba ningún archivo existente. Si los archivos son extraídos por el usuario root, **`tar`** intenta conservar el usuario original y la propiedad del grupo de los archivos. Si un usuario habitual extrae los archivos con **`tar`**, los archivos extraídos son propiedad de ese usuario.

Extraiga el archivo **`/root/etc.tar`** en el directorio **`/root/etcbackup`**:

```
[root@host ~]# mkdir /root/etcbackup
[root@host ~]# cd /root/etcbackup
[root@host etcbackup]# tar xf /root/etc.tar
```

De manera predeterminada, cuando se extraen ficheros de un archivo, el desenmascaramiento se elimina de los permisos de contenido del archivo. Esta es una medida de seguridad y evita que los archivos que se extraen con más frecuencia reciban permisos de ejecución de manera predeterminada. Para proteger los permisos de un fichero archivado, se usará la opción **p** cuando se extraiga un archivo.

Extraiga el archivo **/root/myscripts.tar** en el directorio **/root/scripts** y conserve los permisos de los archivos extraídos:

```
[root@host ~]# mkdir /root/scripts
[root@host ~]# cd /root/scripts
[root@host scripts]# tar xpf /root/myscripts.tar
```

Creación de un archivo tar comprimido

Existen tres métodos de compresión admitidos por el comando **tar**. La compresión gzip es la más rápida y antigua, y la que tiene mayor disponibilidad. Por lo general, la compresión bzip2 genera ficheros de archivo más pequeños comparados con gzip y tiene menor disponibilidad que gzip, mientras que el método de compresión xz es relativamente nuevo, pero en general ofrece la mejor relación de compresión de los métodos disponibles.

nota

La efectividad del algoritmo de compresión depende de la naturaleza exacta de los datos que se comprimen. Los ficheros de datos que ya están comprimidos, como los formatos de imagen comprimidos o archivos rpm, generalmente generan una relación de compresión baja.

Una práctica adecuada es usar un solo directorio de nivel superior, que puede contener otros directorios y ficheros, para simplificar la extracción de los ficheros de manera organizada.

Para crear un archivo comprimido, puede especificarse una de las siguientes opciones de **`tar`**:

- **`z`** para la compresión de gzip (filename.tar.gz o filename.tgz)
- **`j`** para la compresión de bzip2 (filename.tar.bz2)
- **`J`** para la compresión de xz (filename.tar.xz)

Extraiga el contenido (opción x) de un archivo tar (opción z) comprimido con gzip con el nombre **/root/etcbackup.tar.gz** en el directorio **/tmp/etcbackup**:

```
[root@serverX ~]$ mkdir /tmp/etcbackup
[root@serverX ~]$ cd /tmp/etcbackup
[root@serverX etcbackup]$ tar xzf /root/etcbackup.tar.gz
```

Extraiga el contenido (opción x) de un archivo tar comprimido con bzip2 (opción j) con el nombre **/root/logbackup.tar.bz2** en el directorio **/tmp/logbackup**:

```
[root@serverX ~]$ mkdir /tmp/logbackup
[root@serverX ~]$ cd /tmp/logbackup
[root@serverX logbackup]# tar xjf /root/logbackup.tar.bz2
```

Extraiga el contenido (opción x) de un archivo tar comprimido con xz (opción J) con el nombre **/root/sshbackup.tar.xz** en el directorio **/tmp/sshbackup**:

```
[root@serverX ~]$ mkdir /tmp/sshbackup
[root@serverX ~]$ cd /tmp/sshbackup
[root@serverX sshbackup]# tar xJf /root/sshbackup.tar.xz
```

nota

La enumeración de un archivo **`tar`** comprimido funciona de la misma manera que la enumeración de un archivo **`tar`** sin comprimir.

nota

Además, **`gzip`**, **`bzip2`** y **`xz`** se pueden usar de manera independiente para comprimir archivos individuales. Por ejemplo, **`gzip etc.tar`** genera el archivo comprimido **`etc.tar.gz`**, mientras que **`bzip2 abc.tar`** genera el archivo comprimido **`abc.tar.bz2`** y **`xz myarchive.tar`** genera el archivo comprimido **`myarchive.tar.xz`**.

Los comandos de descompresión correspondientes son **`gunzip`**, **`bunzip2`** y **`unxz`**. Por ejemplo, **`gunzip /tmp/etc.tar.gz`** genera el archivo tar sin comprimir **`etc.tar`**, mientras que **`bunzip2 abc.tar.bz2`** genera el archivo tar sin comprimir **`abc.tar`** y **`unxz myarchive.tar.xz`** genera el archivo tar sin comprimir **`myarchive.tar`**.

Descripción general de las opciones de `tar`

El comando **`tar`** tiene varias opciones que puede usar. La siguiente tabla enumera algunas de las opciones más usadas y sus significados.

Descripción general de las opciones de `tar`

Opción	Significado
`c`	Cree un archivo nuevo.
`x`	Extráigalo de un fichero existente.
`t`	Enumere el contenido de un archivo.
`v`	Exceso de palabras: muestra cuáles son los archivos que se archivan o extraen.
`f`	Nombre del archivo: esta opción tiene que estar seguida del nombre del fichero del archivo que se usará o creará.
`p`	Conserve los permisos de los ficheros y directorios cuando se extrae un archivo sin eliminar el desenmascaramiento.
`z`	Usa compresión **`gzip`** (**`.tar.gz`**).
`j`	Use la compresión **`bzip2`** (**`.tar.bz2`**). Generalmente, **`bzip2`** logra una mejor relación de compresión que **`gzip`**.
`J`	Use la compresión **`xz`** (**`.tar.xz`**). Generalmente, **`xz`** logra una mejor relación de compresión que **`bzip2`**.

Referencias

Páginas del manual: **tar** (1), **gzip** (1), **gunzip** (1), **bzip2** (1), **bunzip2** (1), **xz** (1) y **unxz** (1)

Práctica: Copia de seguridad y restauración de archivos a partir de un archivo tar

Copia segura de archivos entre sistemas

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder copiar archivos de manera segura desde un sistema remoto que ejecute el servidor sshd y hasta él.

Copiar archivos desde una ubicación remota y hasta ella con scp.

El comando **ssh** es útil para ejecutar los comandos de shell en sistemas remotos en forma segura. También se puede utilizar para copiar archivos de una máquina a otra en forma segura. El comando **scp** transfiere archivos desde un host remoto hasta el sistema local o desde el sistema local hasta un host remoto. Utiliza el servidor SSH para la autenticación y la transferencia de datos cifrados.

Las ubicaciones remotas de sistemas de archivos siempre se especifican con el formato **[user@]host:/path** tanto para la ubicación de origen como para la de destino de los archivos que se transferirán. La parte **user@** es opcional y, si falta, se utiliza el usuario local actual que invoca el comando **scp**. Antes de que se inicie la transferencia, el usuario debe autenticarse con el servidor SSH con contraseña o claves de SSH.

El siguiente es un ejemplo de cómo copiar los archivos locales que se encuentran en desktopX, **/etc/yum.conf** y **/etc/hosts** de manera segura en la cuenta student del sistema remoto serverX en el directorio **/home/student/**:

```
[student@desktopX ~]$ scp /etc/yum.conf /etc/hosts serverX:/home/student
student@serverX's password: student
yum.conf                                      100%  813     0.8KB/s   00:00
hosts                                         100%  227     0.2KB/s   00:00
```

Un usuario puede copiar un archivo desde una cuenta remota de una máquina remota en un sistema de archivos local con **scp**. En este ejemplo, copie el archivo **/etc/hostname** desde la cuentastudent de la máquina serverX; en el directorio local **/home/student/**.

```
[student@desktopX ~]$ scp serverX:/etc/hostname /home/student/
student@serverX's password: student
hostname                                      100%   22     0.0KB/s   00:00
```

Para copiar un árbol de directorios completo de manera recursiva, se encuentra disponible la opción **-r**. En el siguiente ejemplo, el directorio remoto **/var/log** en serverX se copia de manera recursiva en el directorio local **/tmp/** en desktopX. Para poder leer todos los archivos que se copiaron en el directorio **/tmp**, el usuario debe conectarse a la ubicación remota como **root**.

```
[student@desktopX ~]$ scp -r root@serverX:/var/log /tmp
root@serverX's password: redhat
...
```

Transferencia de archivos remota con sftp

Si se prefiere una herramienta interactiva para la carga de archivos en un servidor SSH o su descarga, puede utilizarse el comando **sftp**. Una sesión con **sftp** es similar a una sesión FTP clásica, solo que emplea el mecanismo de autenticación segura y la transferencia de datos cifrados del servidor SSH.

Para iniciar una sesión **sftp**, **sftp** espera una ubicación remota con el formato **[user@]host**, en el que la parte **user@** es opcional y, si falta, se utiliza el usuario que invoca el comando **sftp**. Para establecer la sesión **sftp**, es necesario realizar la autenticación con cualquiera de los métodos que acepta el servidor SSH.

```
[student@desktopX ~]$ sftp serverX
student@serverX's password: student
Connected to serverX.
sftp>
```

La sesión **sftp** acepta diversos comandos que funcionan de la misma manera en el sistema de archivos remoto que en el sistemas de archivos local, como **ls**, **cd**, **mkdir**, **rmdir** y **pwd**. Además, existen los comandos **put** y **get** para la carga y descarga de archivos. El comando **exit** finaliza la sesión **sftp**.

Cargue el archivo local **/etc/hosts** en el directorio recientemente creado **/home/student/hostbackup** en el host remoto serverX. La sesión **sftp** siempre supone que el comando put es seguido de un archivo en el sistema de archivos local y comienza en el directorio de inicio del usuario conectado; en este caso, **/home/student**:

```
sftp> mkdir hostbackup
sftp> cd hostbackup
sftp> put /etc/hosts
Uploading /etc/hosts to /home/student/hostbackup/hosts
/etc/hosts                                    100%  227     0.2KB/s   00:00
sftp>
```

Para descargar el archivo remoto **/etc/yum.conf** del host remoto en el directorio actual del sistema de archivos local, ejecute el comando **get /etc/yum.conf** y finalice la sesión **sftp** con el comando **exit**.

```
sftp> get /etc/yum.conf
Fetching /etc/yum.conf to yum.conf
/etc/yum.conf                                 100%  813     0.8KB/s   00:00
sftp> exit
[student@desktopX ~]$
```

Referencias

Páginas del manual: **scp**(1), **sftp**(1)

Práctica: Copia de archivos por medio de la red con scp

Sincronización de archivos entre sistemas en forma segura

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder sincronizar en forma eficiente y segura el contenido de un archivo o directorio local con una copia remota.

Sincronizar archivos y carpetas con rsync

Con la herramienta **rsync** también se pueden copiar archivos en forma segura de un sistema a otro. Se diferencia de scp en que si dos archivos o directorios son similares entre dos sistemas, rsync solo necesita copiar las diferencias entre los sistemas, mientras que scp necesita copiar todo.

Una de las ventajas de **rsync** es que puede copiar archivos entre un sistema local y un sistema remoto en forma segura y eficiente. Cuando la sincronización inicial de un directorio demora prácticamente el mismo tiempo que el copiado, cualquier sincronización posterior solo requerirá que se copien las diferencias mediante la red.

Una de las opciones más importantes de **rsync** es la opción **-n** para realizar un simulacro. Un simulacro es una imitación de lo que sucede cuando el comando se ejecuta de verdad. Mostrará los cambios que realizará cuando se ejecute el comando sin la opción de simulacro. Se recomienda realizar un simulacro de cualquier operación de **rsync** para garantizar que no se sobrescriba ni elimine ningún archivo.

Las dos opciones más usadas cuando se sincronizan archivos y carpetas con **rsync** son **-a** y **-v**. Mientras la opción **-v** agrega un exceso de palabras al resultado a medida que continúa la sincronización, la opción **-a** implica un "modo archivo" y habilita las siguientes opciones, todas al mismo tiempo:

- **-r**, sincronizar en forma recurrente todo el árbol de directorio
- **-l**, sincronizar los enlaces simbólicos
- **-p**, mantener los permisos
- **-t**, conservar las marcas de tiempo
- **-g**, conservar la propiedad del grupo
- **-o**, conservar al propietario de los archivos
- **-D**, sincronizar los archivos de dispositivo

Si bien la opción **-a** ya sincroniza enlaces simbólicos, existen otras opciones necesarias para conservar los enlaces duros ya que, de lo contrario, son tratados como archivos separados. La opción **-H** habilita la manipulación de enlaces duros; por lo tanto, el comando **rsync** identificará los enlaces físicos que están en la carpeta de origen y vinculará los archivos en consecuencia en la carpeta de destino, en lugar de solo copiarlos como archivos separados.

nota

La opción **-a** no sincroniza los permisos de archivo avanzados, como los contextos de archivos ACL o SELinux. Para habilitar la sincronización de ACL, se requiere la opción **-A** además de la opción **-a**, mientras que para sincronizar los contextos de SELinux de archivos de origen a archivos de destino, es necesario agregar la opción **-X**.

La manera básica de usar **rsync** es sincronizar dos carpetas locales. En el siguiente ejemplo, el directorio **/var/log** obtiene una copia sincronizada en la carpeta **/tmp**. El directorio **log** junto con su contenido se crea en el directorio **/tmp**.

```
[student@desktopX ~]$ su -
Password: redhat
[root@desktopX ~]# rsync -av /var/log /tmp
...
```

Para sincronizar solo el contenido de una carpeta sin crearla en el directorio de destino, se necesita agregar una barra al final del directorio de origen. En este ejemplo, el directorio **log** no se crea en la carpeta **/tmp**. Solo se sincroniza el contenido del directorio **/var/log/** en la carpeta **/tmp**.

```
[root@desktopX ~]# rsync -av /var/log/ /tmp
...
```

Importante

Cuando se ingrese el directorio de origen para rsync, es muy importante recordar si está presente la barra final en el nombre del directorio. Esto determinará si el *directorio* o solo el *contenido del directorio* se sincroniza en el destino. Nota: La terminación del tabulador agregará automáticamente una barra al final de los nombres de directorios.

Al igual que con **scp**, el comando **rsync** espera a que se especifiquen las ubicaciones del sistema de archivos remoto en el formato **[user@]host:/path**. En caso de que falte la parte **user@** opcional, se usa el usuario que invoca el comando **rsync** para conectarse a la ubicación remota. Se puede usar una ubicación remota como origen o destino. En el siguiente ejemplo, la carpeta local **/var/log** obtiene una copia sincronizada en el directorio **/tmp**, en la máquina serverX. Para que **rsync** sincronice la propiedad de los archivos transferidos, la ubicación de destino debe estar escrita como usuario root; en consecuencia, conéctese a serverX del sistema remoto como usuario root. El usuario root que se conecta debe autenticarse con el servidor SSH mediante cualquiera de los métodos aceptados; por ejemplo, contraseña o claves de SSH.

```
[root@desktopX ~]$ rsync -av /var/log serverX:/tmp
root@serverX's password: redhat
...
```

De la misma manera, la carpeta remota **/var/log** en serverX puede sincronizarse en el directorio local **/tmp** en desktopX:

```
[root@desktopX ~]$ rsync -av serverX:/var/log /tmp
root@serverX's password: redhat
...
```

Referencias

Página del manual (1)**rsync**

Práctica: Sincronización segura de dos directorios con rsync