7.1 Introducción

En este capítulo, se discute la forma de gestionar los ficheros de archivado en la línea de comandos.

**Archivar el archivo** se utiliza cuando uno o más archivos necesitan ser transmitidos o almacenados de forma eficiente. Dos conceptos:

* **Archivado** - Combinación de varios archivos en uno solo, lo que elimina la sobrecarga en archivos individuales y hace que sea más fácil de transmitir.
* **Compresión** - Hacer los archivos más pequeños mediante la eliminación de información redundante.

Puede almacenar varios archivos en un único archivo y luego comprimirlo, o puede comprimir un archivo individual.

A pesar de que el espacio en disco es relativamente barato, el archivado y compresión todavía tiene valor:

* Si usted desea hacer una gran cantidad de archivos disponibles, como el código fuente de una aplicación o un conjunto de documentos, es más fácil para que la gente descargue un archivo comprimido que descargar archivos individuales.
* Los archivos de registro tienen la costumbre de llenar discos por lo que es útil para dividir por fecha y comprimir las versiones anteriores.
* Al realizar una copia de seguridad de directorios, es más fácil de mantener a todos en un archivo de lo que es mantener versiones de cada archivo.
* Algunos dispositivos de transmisión, tales como cintas funcionan mejor si usted está enviando una corriente de datos en lugar de archivos individuales.
* A menudo puede ser más rápido comprimir un archivo antes de enviarlo a una unidad de cinta o en una red más lenta y descomprimirlo en el otro extremo, de lo que sería para enviarlo sin comprimir.

Como administrador de Linux, tendrá que familiarizarse con las herramientas para archivar y comprimir archivos.

7.3 Comprimir archivos

*La compresión de* archivos hace más pequeños mediante la eliminación de la duplicación de un archivo y almacenarlo de manera que el archivo se puede restaurar. Un archivo con texto legible podría haber utilizado con frecuencia palabras sustituidas por algo más pequeño, o una imagen con un fondo sólido podría representar manchas de ese color por un código. Por lo general, no se utiliza la versión comprimida del archivo, en lugar descomprimir antes de su uso. El *algoritmo de compresión* es un procedimiento de la computadora hace para codificar el archivo original, y como resultado hacerlo más pequeño. Los informáticos investigan estos algoritmos y llegar a otros mejores que pueden trabajar más rápido o hacer que el archivo de entrada más pequeño.

Cuando se habla de compresión, hay dos tipos:

* **Sin pérdida**: No hay información se extrae desde el archivo. La compresión y descompresión de un archivo lo deja idéntico a la original.
* **Con pérdida**: La información podría ser eliminado del archivo, ya que se comprime para que descomprimir un archivo resultará en un archivo que es ligeramente diferente a la original. Por ejemplo, una imagen con dos tonos sutilmente diferentes de verde podría hacerse más pequeña mediante el tratamiento de esos dos tonos como el mismo. A menudo, el ojo no puede distinguir la diferencia de todos modos.

Generalmente los ojos y los oídos humanos no notan ligeras imperfecciones en las imágenes y el audio, especialmente a medida que se muestran en un monitor o reproducir por los altavoces. La compresión con pérdida a menudo beneficia a los medios de comunicación, ya que resulta en archivos más pequeños y la gente no nota la diferencia entre el original y la versión con los datos cambiados. Para que la información permanezca intacta, como en los documentos, registros y software, es necesaria la compresión sin pérdidas.

La mayoría de los formatos de imagen, como GIF, PNG y JPEG, implementan algún tipo de compresión con pérdida. En general, usted puede decidir la calidad que desea preservar. A los resultados más bajos de calidad en un archivo más pequeño, pero después de la descompresión puede notar cambios en bordes o decoloraciones. Si mantenemos la calidad alta se parece mucho a la imagen original, pero el tamaño del archivo será más parecido al original.

La compresión de un archivo ya comprimido no lo hacer más pequeño. Esto se olvida a menudo cuando se trata de imágenes, puesto que ya están almacenados en un formato comprimido. Con la compresión sin pérdidas, esta compresión múltiple no es un problema, pero si se comprime y descomprime un archivo varias veces utilizando un algoritmo con pérdida es muy probable que tenga pierda calidad.

Linux proporciona varias herramientas para comprimir archivos, el más común es gzip . A continuación mostramos un archivo de registro antes y después de la compresión.

**bob: tmp $** ls -l access\_log \*

-rw-r - r-- 1 sean sean 372,063 11 de octubre 21:24 access\_log

**bob: tmp $** gzip access\_log

**bob: tmp $** ls -l access\_log \*

-rw-r - r-- 1 sean sean 26080 11 de octubre 21:24 access\_log.gz

En el ejemplo anterior, existe un archivo llamado "access\_log" de 372.063 bytes. El archivo está comprimido mediante la invocación del gzip  con el nombre del archivo como único argumento. Tras completar el comando, el archivo original desaparece y se crea una versión comprimida con una extensión.gz en su lugar. El tamaño del archivo es de 26,080 bytes ahora, dando una relación de compresión de aproximadamente 14: 1, que es común con los archivos de registro.

Gzip le dará esta información si lo solicita, mediante el -l parámetro, como se muestra aquí:

**bob: tmp $** gzip -l access\_log.gz

comprimido relación uncompressed\_name sin comprimir

26080 372063 93.0% access\_log

Aquí, se puede ver que la relación de compresión es del 93%, que es la inversa de la 14: 1, es decir, 13/14. Además, cuando se descomprime el archivo se llamará access\_log.

**bob: tmp $** gunzip access\_log.gz

**bob: tmp $** ls -l access\_log \*

-rw-r - r-- 1 sean sean 372,063 11 de octubre 21:24 access\_log

Lo contrario de  gzip  es gunzip . Alternativamente,  gzip -d hace lo mismo (gunzip es sólo una secuencia de comandos que llama a gzip con los parámetros correctos).Tras ejecutar gunzip se puede ver que el archivo access\_log vuelve a su tamaño original.

Gzip también puede actuar como un filtro que significa que no lee o escribe nada en el disco, sino que recibe datos a través de un canal de entrada y la escribe a un canal de salida.

**bob: tmp $** mysqldump -A | gzip> database\_backup.gz

**bob: tmp $** gzip -l database\_backup.gz

comprimido relación uncompressed\_name sin comprimir

76866 1028003 92.5% database\_backup

El  mysqldump -A presenta el contenido de las bases de datos locales de MySQL a la consola. El carácter | (pipe) "redirige la salida de la orden anterior a la entrada de la siguiente." El programa para recibir la salida es gzip, que reconoce los nombres de archivo proporcionados por lo que debería funcionar en modo tubería. Por último, el > database\_backup.gz significa "redirigir la salida del comando anterior en un archivo llamado database\_backup.gz. Analizando este archivo con gzip -l muestra que la versión comprimida es 7.5% del tamaño del original, con la ventaja añadida de que el archivo más grande nunca tuvo que ser escrito en el disco.

Hay otro par de comandos que operan prácticamente idéntica a gzip y gunzip. Estos son bzip2 y bunzip2. Las utilidades bZIP utilizan un algoritmo de compresión diferente (llamado Burrows-Wheeler clasificación bloque, frente Lempel-Ziv codificación utilizado por gzip) que puede comprimir archivos más pequeños que gzip a expensas de más tiempo de CPU. Se puede reconocer a estos archivos porque tienen como extensión .bz o bz2 en lugar de .gz.

7.4 Almacenamiento de archivos

Si tuvieras varios archivos para enviar a alguien, podría comprimir cada uno individualmente. Usted tendría una menor cantidad de datos en total que si envía archivos sin comprimir, pero usted todavía tiene que lidiar con muchos archivos al mismo tiempo.

El archivado es la solución a este problema. La utilidad tradicional de UNIX se llama tar , que es una forma corta de archivo de cinta. Tar se utilizó para transmitir muchos archivos a una cinta para copias de seguridad o transferencia de archivos. Tar de varios archivos crea un único archivo de salida que se puede dividir de nuevo en los archivos originales en el otro extremo de la transmisión.

Tar tiene 3 modos que usted querrá estar familiarizado con:

* **Crear** : hacer un nuevo archivo de una serie de archivos
* **Extraer** : extraer los archivos de un único archivo
* **Lista** : muestrar el contenido del archivo sin extraer

Recordar los modos es clave para averiguar las opciones de la línea de comandos necesarios para hacer lo que quiera. Además del modo, usted también querrá asegurarse de especificar el nombre del archivo

Aquí, se muestra un archivo tar, también llamado un tarball, que se crea a partir de múltiples registros de acceso.

**bob: tmp $** tar -cf access\_logs.tar access\_log\*

**bob: tmp $** ls -l access\_logs.tar

-rw-rw-r-- 1 sean sean 542.720 12-oct 21:42 access\_logs.tar

La creación de un archivo requiere dos opciones. El primero, **c** , especifica el modo. El segundo, **f** , dice a tar que se asignará un nombre de archivo como argumento. El primer argumento en el ejemplo anterior crea un archivo llamado access\_logs.tar . Los argumentos restantes son los nombres de archivo de entrada, ya sea como un comodín, una lista de archivos, o ambos. En este ejemplo, utilizamos la opción de comodín para incluir todos los archivos que comienzan con access\_log .

 El tamaño final es 542,720 bytes, que es ligeramente más grande que los archivos de entrada. Tarballs se pueden comprimir para facilitar el transporte, ya sea por gzip o mediante la opción z de tar:

**bob: tmp $** tar -czf access\_logs.tar.gz access\_log \*

**bob: tmp $** ls -l access\_logs.tar.gz

-rw-rw-r-- 1 sean sean 46229 12 de octubre 21:50 access\_logs.tar.gz

**bob: tmp $** gzip -l access\_logs.tar.gz

comprimido relación uncompressed\_name sin comprimir

46229 542720 91,5% access\_logs.tar

El ejemplo anterior muestra el mismo comando que el ejemplo anterior, pero con la adición del parámetro z. La salida es mucho menor que el tarball, y el archivo resultante es compatible con gzip.

Mientras UNIX no trata las extensiones de archivo, la convención es usar .tar para archivos tar, y .tar.gz o .tgz para archivos tar comprimidos. Usted puede utilizar bzip2 en lugar de gzip sustituyendo la letra **j** por **z** y utilizando .tar.bz2, .tbz o .tbz2 como extensión de archivo (por ejemplo, alquitrán -cjf file.tbz access\_log \* ).

Dado un archivo tar, comprimido o no, usted puede ver lo que hay en ella mediante la opción **t**:

**bob: tmp $** tar -tjf access\_logs.tbz

logs /

logs / access\_log.3

logs / access\_log.1

logs / access\_log.4

logs / access\_log

logs / access\_log.2

Este ejemplo utiliza 3 opciones:

* t lista de archivos
* j : Descomprimir con bzip2 antes de leer
* f : operar en el nombre de archivo dado (access\_logs.tbz)

A continuación se muestra el contenido del archivo comprimido. Se puede ver que un directorio se antepone a los archivos. Tar es recursiva en subdirectorios de forma automática, cuando se comprime y almacena la información de la ruta dentro del archivo.

Sólo para mostrar que este fichero es todavía nada especial, vamos a enumerar los contenidos del archivo en dos pasos utilizando una tubería.

**bob: tmp $** bunzip2 -c access\_logs.tbz |tar -t

logs /

logs / access\_log.3

logs / access\_log.1

logs / access\_log.4

logs / access\_log

logs / access\_log.2

El lado izquierdo de la tubería es bunzip access\_logs.tbz -c , que descomprime el archivo, pero el ( -c opción) envía la salida a la pantalla. La salida se redirige a tar -t . Si no se especifica un archivo con -f luego alquitrán leerá de la entrada estándar.

Por último, puede extraer el archivo con la opción -x :

**bob: tmp $** tar -xjf access\_logs.tbz

**bob: tmp $** ls -l

total de 36

-rw-rw-r-- 1 sean sean 30043 14 de octubre 13:27 access\_logs.tbz

drwxrwxr-x 2 sean sean 4096 14 de octubre 13:26 registros

**bob: $ tmp** ls -l registros

total de 536

-rw-r - r-- 1 sean sean 372,063 11 de octubre 21:24 access\_log

-rw-r - r-- 1 sean sean 362 12 de octubre 21:41 access\_log.1

-rw-r - r-- 1 sean sean 153,813 12 de octubre 21:41 access\_log.2

-rw-r - r-- 1 sean sean 1136 12 de octubre 21:41 access\_log.3

-rw-r - r-- 1 sean sean 784 12 de octubre 21:41 access\_log.4

El ejemplo anterior utiliza el patrón similar al anterior, especificando la operación (extraer), la compresión (el indicador j, lo que significa bzip2), y un nombre de archivo -f (access\_logs.tbz). El archivo original es virgen y el nuevos **registros** se crea en el directorio.  Añadiendo la opción -v  y obtendrá resultados detallados de los archivos procesados. Esto es útil para que pueda ver lo que está pasando:

**bob: tmp $** tar -xjvf access\_logs.tbz

logs /

logs / access\_log.3

logs / access\_log.1

logs / access\_log.4

logs / access\_log

logs / access\_log.2

Es importante mantener el -f  al final, ya que tar asume que lo siguiente es un nombre de archivo. En el siguiente ejemplo, la f y v  fueron incorporadas, lo que lleva al tar a interpretar el comando como una operación en un archivo llamado "v" (el mensaje es relevante en cursiva.)

**bob: tmp $** tar -xjfv access\_logs.tbz

tar (niño): v: no se puede abrir: No existe el fichero o directorio

tar (niño): Error no recuperable: salir ahora

tar: Niño devolvió el código de 2

tar: El error no es recuperable: salir ahora

Si sólo quiere algunos archivos puede añadir sus nombres al final de la orden, pero por defecto, que debe coincidir con el nombre en el archivo exactamente o utilizar un patrón:

**bob: tmp $** tar -xjvf registros access\_logs.tbz /access\_log

logs /access\_log

El ejemplo anterior muestra el mismo archivo que antes, pero la extracción de sólo el archivo "logs / access\_log". La salida del comando (como se solicitó modo detallado con la bandera "v") muestra sólo el archivo de uno se ha extraído.

Tar tiene muchas más características, como la capacidad de utilizar patrones al extraer los archivos, con exclusión de ciertos archivos, o la salida de los archivos extraídos en la pantalla en lugar del disco. La documentación para el tar tiene información en profundidad.

## 7.5 archivos ZIP

La utilidad por defecto de archivado en el mundo Microsoft es el archivo ZIP. No es tan frecuente en Linux, pero se puede utilizar con los comandos zip y unzip . Las opciones tiene diferentes significados para los dos comandos diferentes.

El modo predeterminado de zip es agregar archivos a un archivo y comprimirlo.

bob: $ tmp registros logs.zip zip / \*

añadir: logs / access\_log (deflactado 93%)

y agregó: logs / access\_log.1 (deflactado 62%)

y agregó: logs / access\_log.2 (deflactado 88%)

y agregó: logs / access\_log 0.3 (deflactado 73%)

y agregó: logs / access\_log.4 (deflactado 72%)

El primer argumento en el ejemplo anterior es el nombre del archivo, en este caso, es **logs.zip** . Después de eso, se añade una lista de archivos. El resultado muestra los archivos y la relación de compresión. Cabe señalar que el tar requiere -f  para indicar se está pasando un nombre de archivo, mientras que zip y unzip requieren un nombre de archivo y por tanto no es necesario que diga explícitamente se está pasando un nombre de archivo.

Zip no es recursivo, en subdirectorios, por defecto, lo que es un comportamiento diferente al de tar. Es decir, la simple adición de "logs" en lugar de "logs / \*" sólo añadir el directorio vacío y no los archivos bajo él. Por lo que, debe utilizar la -r comando para indicar la recursividad:

**bob: tmp $** zip -r logs.zip registros

añadiendo: logs / (almacenado 0%)

y agregó: logs / access\_log.3 (deflactado 73%)

añadiendo: logs / access\_log.1 (deflactado 62%)

añadiendo: logs / access\_log. 4 (deflactado 72%)

y agregó: logs / access\_log (deflactado 93%)

y agregó: logs / access\_log.2 (deflactado 88%)

En el ejemplo anterior, se agregan todos los archivos del directorio de registros, ya que utiliza -r . La primera línea de la salida indica que un directorio se añadió al archivo, pero de lo contrario la salida es similar al ejemplo anterior.

Con la opción –l de la descompresión se listaran los archivos contenidos:

**bob: $ tmp** unzip -l logs.zip

Archivo: logs.zip

Longitud Fecha Hora Nombre

--------- ---------- ----- ----

0 10- 14-2013 14:07 logs /

1136 10/14/2013 14:07 logs / access\_log.3

362 10/14/2013 14:07 logs / access\_log.1

784 14/10/2013 14:07 logs / access\_log.4

90.703 10/14/2013 14:07 logs / access\_log

153813 10/14/2013 14:07 logs / access\_log.2

--------- -------

246798 6 archivos

Extraer los archivos es como crear el archivo, ya que el funcionamiento por defecto es extraer:

**bob: tmp $** unzip logs.zip

Archivo: logs.zip

creación: logs /

inflar: logs / access\_log.3

inflar: logs / access\_log.1

inflado: logs / access\_log.4

inflado: logs / access\_log

inflan: logs / access\_log.2

Aquí, extraemos todos los archivos en el archivo en el directorio actual. Al igual que el tar, puede pasar los nombres de archivo en la línea de comandos:

**bob: tmp $** unzip logs.zip access\_log

Archivo: logs.zip

precaución: nombre de archivo no coincide: access\_log

**bob: tmp $** troncos logs.zip descomprimir / access\_log

Archivo: logs.zip

inflado: logs / access\_log

**bob: tmp $** unzip logs.zip logs / access\_log \*.

Archivo: logs.zip

inflado: logs / access\_log.3

inflado: troncos / access\_log.1

inflado: logs / access\_log.4

inflado: logs / access\_log.2

El ejemplo anterior muestra tres intentos diferentes para extraer un archivo.

En primero, sólo el nombre del archivo se pasa sin el componente de directorio. Como el tran, el archivo no se corresponde.

El segundo intento pasa el componente directorio junto con el nombre del archivo, que extrae sólo ese archivo.

La tercera versión utiliza un comodín, que extrae los 4 archivos que coinciden con el patrón, como tar.

Las páginas de comprimir y descomprimir hombre describen las otras cosas que puedes hacer con estas herramientas, tales como reemplazar los archivos dentro del archivo, utilizan diferentes niveles de compresión, e incluso utilizan el cifrado.