Administración de archivos desde la línea de comandos

RH124

Capitulo 2

Descripción general	
Meta	Copiar, mover, crear, eliminar y organizar archivos mientras se trabaja desde el aviso de la shell Bash.
Objetivos	• Identificar el objetivo de directorios importantes en un sistema Linux. • Especificar archivos usando nombres de rutas absolutas y relativas. • Crear, copiar, mover y quitar archivos y directorios usando utilidades de la línea de comandos. • Hacer coincidir uno o más nombres de archivo con expansión de shell como argumentos de comandos de la shell.
Secciones	• Jerarquía del sistema de archivos Linux (y práctica) • Búsqueda de archivos por nombre (y práctica) • Administración de archivos con las herramientas de línea de comandos (y práctica) • Coincidencia de nombres de archivo mediante el uso de expansión de nombre de ruta (y práctica)
Trabajo de laboratorio	• Administración de archivos con expansión de shell

Jerarquía del sistema de archivos Linux

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían entender el diseño y la organización fundamentales del sistema de archivos, y la ubicación de los tipos de archivo clave.

Jerarquía del sistema de archivos

Todos los archivos de un sistema Linux se guardan en sistemas de archivos que están organizados en un árbol de directorios *invertido* individual conocido como *jerarquía de sistema de archivos*. Este árbol está invertido porque se dice que la raíz del árbol está en la parte *superior* de la jerarquía y las ramas de los directorios y subdirectorios se extienden debajo de *root*.

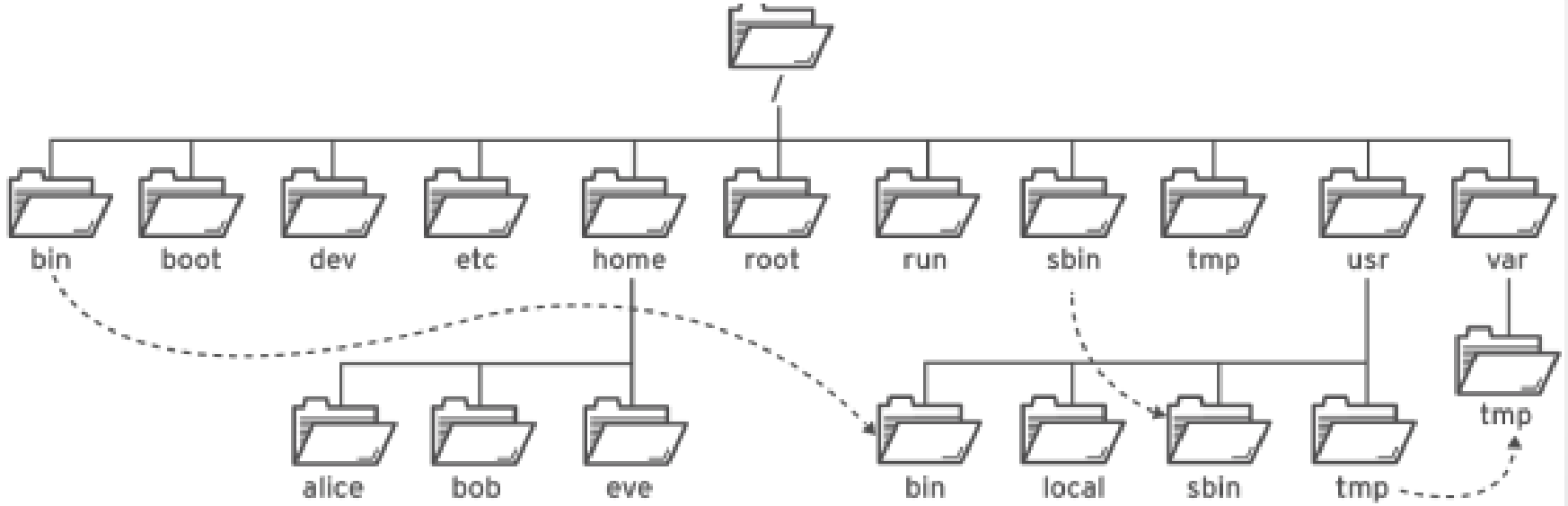

Figura 2.1: Directorios del sistema de archivos importantes en Red Hat Enterprise Linux 7

Directorios Red Hat Enterprise Linux importantes

Ubicación	Propósito
/usr	Software instalado, bibliotecas compartidas, incluye archivos y datos de programa estáticos de solo lectura. Los subdirectorios importantes incluyen: - **/usr/bin**: *Comandos del usuario*. - **/usr/sbin**: *Comandos de administración del sistema*. - **/usr/local**: *Software personalizado en forma local*.
/etc	Archivos de configuración específicos para este sistema.
/var	Datos variables específicos de este sistema que deberían conservarse entre los arranques. Los archivos que cambian en forma dinámica (por ejemplo, bases de datos, directorios caché, archivos de registro, documentos en cola de impresión y contenido de sitio web) pueden encontrarse en **/var**.
/run	Datos de tiempo de ejecución para procesos que se iniciaron desde el último arranque. Esto incluye archivos de ID de proceso y archivos de bloqueo, entre otros elementos. El contenido de este directorio se vuelve a crear en el arranque nuevo. (Este directorio consolida **/var/run** y **/var/lock** de versiones anteriores de Red Hat Enterprise Linux).
/home	*Los directorios de inicio* son aquellos donde los usuarios habituales guardan sus datos personales y los archivos de configuración.
/root	Es el directorio de inicio para el superusuario administrativo, root.
/tmp	Es un espacio con capacidad de escritura para archivos temporales. Los archivos a los que no se haya accedido, y que no se hayan cambiado ni modificado durante 10 días se eliminan de este directorio automáticamente. Existe otro directorio temporal, **/var/tmp**, en el que los archivos que no tuvieron acceso, cambios ni modificaciones durante más de 30 días se eliminan automáticamente.
/boot	Son los archivos necesarios para iniciar el proceso de arranque.

Ubicación	Propósito
/dev	Contiene *archivos de dispositivo* especiales que son usados por el sistema para acceder al hardware.

Importante

En Red Hat Enterprise Linux 7, cuatro directorios antiguos en **/** ahora tienen contenido idéntico al de sus equivalentes que están en **/usr**:
- **/bin** y **/usr/bin**.
- **/sbin** y **/usr/sbin**.
- **/lib** y **/usr/lib**.
- **/lib64** y **/usr/lib64**.

En versiones anteriores de Red Hat Enterprise Linux, estos eran directorios distintos que contenían diferentes conjuntos de archivos. En RHEL 7, los directorios de **/** son enlaces simbólicos para los directorios coincidentes de **/usr**.

Referencias

Página del manual: **hier**(7)

> Estándar de jerarquía del sistema de archivos
> http://www.pathname.com/fhs

Práctica: Jerarquía de sistemas de archivos

Ubicación de archivos por nombre

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder usar en forma correcta los nombres de ruta, cambiar el directorio que está en uso y utilizar comandos para determinar los contenidos y las ubicaciones del directorio.

Rutas absolutas y relativas

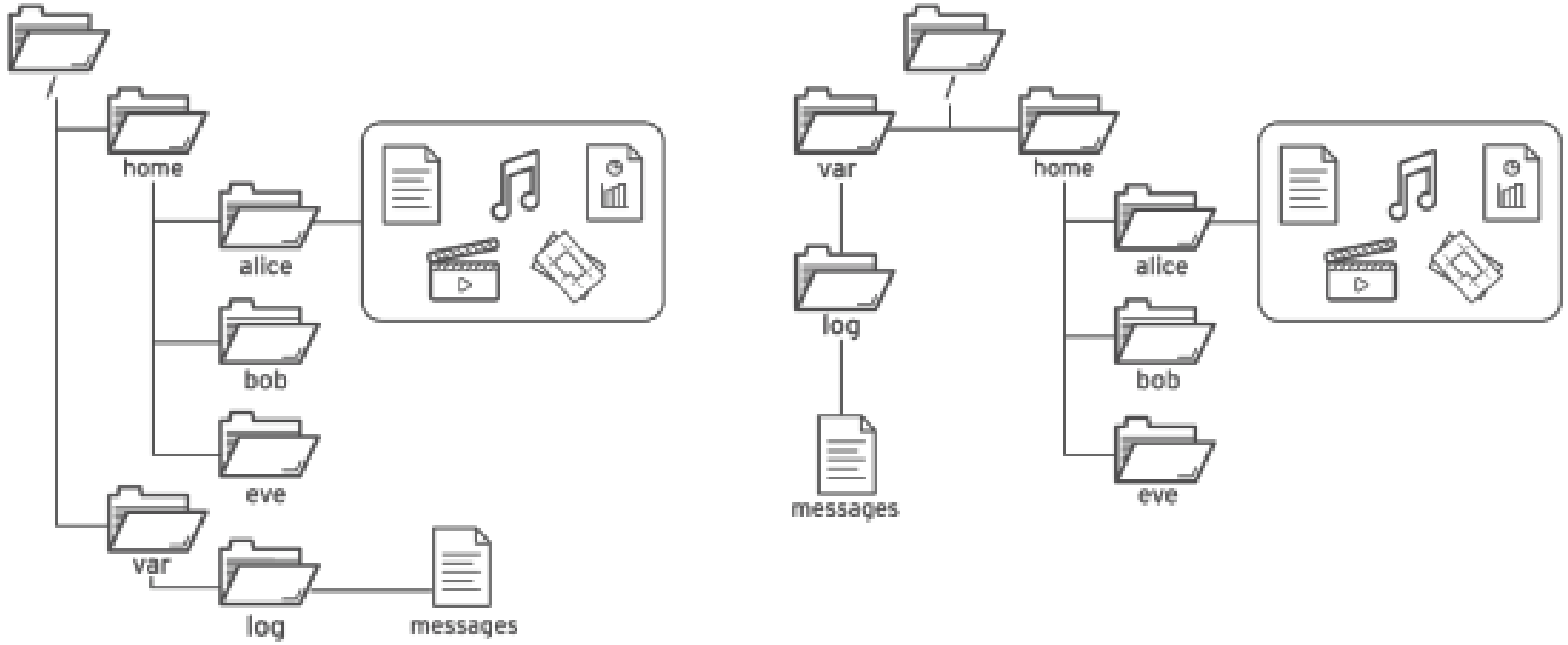

Figura 2.2: La vista del explorador de archivos habitual (izquierda) equivale a la vista descendente (derecha).

Rutas absolutas

Una *ruta absoluta* es un nombre *completamente calificado* que comienza en el directorio (`/`) raíz y especifica cada subdirectorio que se atraviesa para llegar y que representa en forma exclusiva un solo archivo. Cada archivo del sistema de archivos tiene un único nombre de ruta absoluta, reconocido con una regla simple: un nombre de archivo con una barra (`/`) como primer carácter es el nombre de la ruta absoluta. Por ejemplo, el nombre de ruta absoluta para el archivo de registro de mensajes del sistema es **`/var/log/messages`**. Los nombres de rutas absolutas pueden ser extensos para escribir; en consecuencia, los archivos también pueden ubicarse *en forma relativa*.

Rutas relativas

Al igual que una ruta absoluta, una *ruta relativa* identifica un archivo único y especifica solo la ruta necesaria para llegar al archivo desde el directorio de trabajo. Para reconocer nombres de ruta relativos, se sigue una regla simple: un nombre de ruta que no tenga *otro carácter más que una barra* (`/`) como primer carácter es un nombre de ruta relativo. Un usuario en el directorio **`/var`** podría referirse en forma relativa al archivo de registro del mensaje como **`log/messages`**.

Rutas de navegación

El comando **pwd** muestra el nombre de ruta completo de la ubicación actual, que ayuda a determinar la sintaxis adecuada para llegar a los archivos con los nombres de ruta relativos. El comando **ls** enumera el contenido del directorio para el directorio especificado o, si no se indica un directorio, el directorio actual.

```
[student@desktopX ~]$ pwd
/home/student
[student@desktopX ~]$ ls
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos
[student@desktopX ~]$
```

Use el comando **cd** para cambiar los directorios. En caso del directorio de trabajo de **/home/student**, la sintaxis de ruta relativa es más corta para llegar al subdirectorio **Videos**. A continuación, se llega al subdirectorio **Documents** con la sintaxis de ruta absoluta.

```
[student@desktopX ~]$ cd Videos
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd /home/student/Documents
```

```
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ cd
[student@desktopX ~]$ pwd
/home/student
[student@desktopX ~]$
```

Por lo general, el comando **touch** actualiza la marca de tiempo de un archivo con respecto a la fecha y hora actual sin modificarlo. Esto es útil para crear archivos vacíos, que pueden usarse para practicar, ya que si se "toca" el nombre de un archivo que no existe, se produce la creación del archivo. Si se usa **touch**, se crean archivos de práctica en los subdirectorios **Documents** y **Videos**.

```
[student@desktopX ~]$ touch Videos/blockbuster1.ogg
[student@desktopX ~]$ touch Videos/blockbuster2.ogg
[student@desktopX ~]$ touch Documents/thesis_chapter1.odf
[student@desktopX ~]$ touch Documents/thesis_chapter2.odf
[student@desktopX ~]$
```

El comando **ls** tiene varias opciones para mostrar los atributos en los archivos. Los más comunes y útiles son **-l** (formato de lista extenso), **-a** (todos los archivos, incluidos los archivos *ocultos*) y **-R** (recursivos, incluido el contenido de todos los subdirectorios).

```
[student@desktopX ~]$ ls -l
total 15
drwxr-xr-x.  2 student student 4096 Feb  7 14:02 Desktop
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Documents
drwxr-xr-x.  3 student student 4096 Jan  9 15:00 Downloads
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Music
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Pictures
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Public
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Templates
drwxr-xr-x.  2 student student 4096 Jan  9 15:00 Videos
[student@desktopX ~]$ ls -la
total 15
drwx------. 16 student student   4096 Feb  8 16:15 .
drwxr-xr-x.  6 root    root      4096 Feb  8 16:13 ..
-rw-------.  1 student student  22664 Feb  8 00:37 .bash_history
-rw-r--r--.  1 student student     18 Jul  9  2013 .bash_logout
-rw-r--r--.  1 student student    176 Jul  9  2013 .bash_profile
-rw-r--r--.  1 student student    124 Jul  9  2013 .bashrc
drwxr-xr-x.  4 student student   4096 Jan 20 14:02 .cache
drwxr-xr-x.  8 student student   4096 Feb  5 11:45 .config
drwxr-xr-x.  2 student student   4096 Feb  7 14:02 Desktop
drwxr-xr-x.  2 student student   4096 Jan  9 15:00 Documents
drwxr-xr-x.  3 student student   4096 Jan 25 20:48 Downloads
drwxr-xr-x. 11 student student   4096 Feb  6 13:07 .gnome2
drwx------.  2 student student   4096 Jan 20 14:02 .gnome2_private
-rw-------.  1 student student  15190 Feb  8 09:49 .ICEauthority
drwxr-xr-x.  3 student student   4096 Jan  9 15:00 .local
drwxr-xr-x.  2 student student   4096 Jan  9 15:00 Music
drwxr-xr-x.  2 student student   4096 Jan  9 15:00 Pictures
drwxr-xr-x.  2 student student   4096 Jan  9 15:00 Public
drwxr-xr-x.  2 student student   4096 Jan  9 15:00 Templates
drwxr-xr-x.  2 student student   4096 Jan  9 15:00 Videos
[student@desktopX ~]$
```

Importante

Los nombres de archivo que comienzan con un punto (**.**) indican archivos *ocultos* de la vista normal con **ls** y otros comandos. Esta *no* es una característica de seguridad. Los archivos ocultos evitan que los archivos de configuración necesarios del usuario llenen los directorios principales. Existen muchos comandos que procesan archivos ocultos solo con opciones de línea de comandos específicas y previenen que la configuración de un usuario se copie por accidente en otros directorios o usuarios.

Para proteger el *contenido* de un archivo y que no sea visualizado en forma inadecuada se necesitan *permisos de archivos*.

```
[student@desktopX ~]$ ls -R
.:
Desktop  Documents  Downloads  Music  Pictures  Public  Templates  Videos

./Desktop:

./Documents:
thesis_chapter1.odf  thesis_chapter2.odf

./Downloads:

./Music:

./Pictures:

./Public:

./Templates:

./Videos:
blockbuster1.ogg  blockbuster2.ogg
[student@desktopX ~]$
```

El comando **cd** tiene muchas opciones. Muy pocas son tan útiles como para justificar que se practiquen por anticipado y se usen con cierta frecuencia. El comando **cd -** cambia el directorio al directorio donde estaba el usuario *antes* de estar en el directorio actual. Observe cómo este usuario aprovecha este comportamiento para alternar entre dos directorios; esta opción es práctica cuando se procesa una serie de tareas similares.

```
[student@desktopX ~]$ cd Videos
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd /home/student/Documents
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ cd -
```

```
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd -
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ cd -
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd
[student@desktopX ~]$
```

```
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd .
[student@desktopX Videos]$ pwd
/home/student/Videos
[student@desktopX Videos]$ cd ..
[student@desktopX ~]$ pwd
/home/student
[student@desktopX ~]$ cd ..
[student@desktopX home]$ pwd
/home
[student@desktopX home]$ cd ..
[student@desktopX /]$ pwd
/
[student@desktopX /]$ cd
[student@desktopX ~]$ pwd
/home/student
[student@desktopX ~]$
```

Referencias

`info libc 'file name resolution'` (*Manual de referencia de la biblioteca GNU C*)

- Sección 11.2.2: Resolución de nombre de archivo

Páginas del manual: **bash**(7), **cd**(1), **ls**(1), **pwd**(1), **unicode**(1) y **utf-8**(7).

> UTF-8 y Unicode
> http://www.utf-8.com/

Práctica: Ubicación de archivos y directorios

Administración de archivos con las herramientas de línea de comandos

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder crear, copiar, vincular, desplazar y eliminar archivos y subdirectorios en varios directorios.

Administración de archivos de la línea de comandos

La administración de archivos implica la creación, la eliminación, el copiado y el desplazamiento de archivos. Además, los directorios pueden crearse, eliminarse, copiarse y desplazarse para organizar los archivos en forma lógica. Cuando se trabaja en la línea de comandos, la administración de archivos requiere el conocimiento del directorio de trabajo actual para elegir una sintaxis de ruta absoluta o relativa como la opción más eficiente para la tarea inmediata.

Tabla 2.2. Comandos de administración de archivos

Actividad	Fuente única (nota)	Varias fuentes (nota)
Copiar archivo	cp file1 file2	cp file1 file2 file3 dir (4)
Desplazar archivos	mv file1 file2 (1)	mv file1 file2 file3 dir (4)
Eliminar archivo	rm file1	rm -f file1 file2 file3 (5)
Crear directorio	mkdir dir	mkdir -p par1/par2/dir (6)
Copiar directorio	cp -r dir1 dir2 (2)	cp -r dir1 dir2 dir3 dir4 (4)
Desplazar directorio	mv dir1 dir2 (3)	mv dir1 dir2 dir3 dir4 (4)
Eliminar directorio	rm -r dir1 (2)	rm -rf dir1 dir2 dir3 (5)

Nota:

(1) El resultado es un nombre nuevo.

(2) La opción "recursive" se requiere para procesar un directorio fuente.

(3) Si dir2 existe, el resultado es un movimiento. Si dir2 no existe, el resultado es un nombre nuevo.

(4) El último argumento debe ser un directorio.

(5) Tenga precaución con la opción "force"; no se le pedirá que confirme su acción.

(6) Tenga precaución con la opción "create parent"; no se detectan errores de escritura.

Creación de directorios

El comando **mkdir** crea uno o más directorios o subdirectorios y genera errores si ya existe el nombre del archivo o cuando se intenta crear un directorio en un directorio de inicio que no existe. La opción **-p** *parent* crea directorios de inicio faltantes para el destino solicitado. Tenga cuidado cuando use **mkdir -p** ya que los errores de ortografía accidentales generan directorios involuntarios sin activar mensajes de error.

En el siguiente ejemplo, un usuario intenta usar **mkdir** para crear un subdirectorio denominado **Watched** en el directorio existente **Videos**, pero escribe mal el nombre del directorio.

```
[student@desktopX ~]$ mkdir Video/Watched
mkdir: cannot create directory `Video/Watched': No such file or directory
```

mkdir generó un error porque **Videos** se escribió mal y el directorio **Video** no existe. Si el usuario utilizó **mkdir** con la opción **-p**, no habría ningún error y el usuario tendría dos directorios, **Videos** y **Video**, y el subdirectorio **Watched** se crearía en el lugar equivocado.

```
[student@desktopX ~]$ mkdir Videos/Watched
[student@desktopX ~]$ cd Documents
[student@desktopX Documents]$ mkdir ProjectX ProjectY
[student@desktopX Documents]$ mkdir -p Thesis/Chapter1 Thesis/Chapter2 Thesis/Chapter3
[student@desktopX Documents]$ cd
[student@desktopX ~]$ ls -R Videos Documents
Documents:
ProjectX  ProjectY  Thesis  thesis_chapter1.odf  thesis_chapter2.odf

Documents/ProjectX:

Documents/ProjectY:

Documents/Thesis:
Chapter1  Chapter2  Chapter3

Documents/Thesis/Chapter1:

Documents/Thesis/Chapter2:

Documents/Thesis/Chapter3:

Videos:
blockbuster1.ogg  blockbuster2.ogg  Watched

Videos/Watched:

[student@desktopX ~]$
```

El último **mkdir** creó tres subdirectorios de Chapter*N* con un comando. La opción **-p** *parent* creó el directorio principal faltante **Thesis**.

Copia de archivos

El comando **cp** copia uno o más archivos para que se conviertan en archivos nuevos e independientes. La sintaxis permite copiar un archivo existente en un archivo nuevo en un directorio actual o en otro directorio, o copiar varios archivos en otro directorio. En cualquier destino, los nombres de los archivos nuevos deben ser únicos. Si el nombre del archivo nuevo no es único, el comando de copia sobrescribirá el archivo existente.

```
[student@desktopX ~]$ cd Videos
[student@desktopX Videos]$ cp blockbuster1.ogg blockbuster3.ogg
[student@desktopX Videos]$ ls -l
total 0
-rw-rw-r--. 1 student student    0 Feb  8 16:23 blockbuster1.ogg
-rw-rw-r--. 1 student student    0 Feb  8 16:24 blockbuster2.ogg
-rw-rw-r--. 1 student student    0 Feb  8 19:02 blockbuster3.ogg
drwxrwxr-x. 2 student student 4096 Feb  8 23:35 Watched
```

```
[student@desktopX Videos]$
```

Cuando se copian varios archivos con un comando, el último argumento debe ser un directorio. Los archivos copiados conservan su nombre original en el directorio nuevo.
Es probable que se sobrescriban los nombres de archivo con conflicto que existan en un destino. Para evitar que los usuarios sobrescriban directorios con contenido, existen varios comandos **cp** de archivo que omiten directorios especificados como origen. Para poder copiar directorios que no están vacíos, es decir, con contenido, se requiere la opción **-r** *recursive*.

```
[student@desktopX Videos]$ cd ../Documents
[student@desktopX Documents]$ cp thesis_chapter1.odf thesis_chapter2.odf Thesis ProjectX
cp: omitting directory `Thesis'
[student@desktopX Documents]$ cp -r Thesis ProjectX
[student@desktopX Documents]$ cp thesis_chapter2.odf Thesis/Chapter2/
[student@desktopX Documents]$ ls -R
.:
ProjectX  ProjectY  Thesis  thesis_chapter1.odf  thesis_chapter2.odf

./ProjectX:
Thesis  thesis_chapter1.odf  thesis_chapter2.odf

./ProjectX/Thesis:

./ProjectY:

./Thesis:
Chapter1  Chapter2  Chapter3

./Thesis/Chapter1:

./Thesis/Chapter2:
thesis_chapter2.odf

./Thesis/Chapter3:
[student@desktopX Documents]$
```

En el primer comando **cp**, **Thesis** no pudo copiar, pero sí lo hicieron **thesis_chapter1.odf** y **thesis_chapter2.odf**. Con la opción **-r** *recursive*, se pudo copiar **Thesis**.

Desplazamiento de archivos

El comando **mv** cambia el nombre a los archivos en el mismo directorio o reubica archivos en un directorio nuevo. El contenido del archivo se conserva sin modificaciones. Los archivos que se desplazan hacia un sistema de archivos diferente requieren de la creación de un archivo nuevo mediante la copia del archivo de origen y, a continuación, la eliminación de dicho archivo. A pesar de que, por lo general, los archivos grandes son transparentes para el usuario, pueden demorar mucho en desplazarse.

```
[student@desktopX Videos]$ cd ../Documents
[student@desktopX Documents]$ ls -l
total 0
-rw-rw-r--. 1 student student    0 Feb  8 16:24 thesis_chapter1.odf
-rw-rw-r--. 1 student student    0 Feb  8 16:24 thesis_chapter2.odf
[student@desktopX Documents]$ mv thesis_chapter2.odf thesis_chapter2_reviewed.odf
[student@desktopX Documents]$ mv thesis_chapter1.odf Thesis/Chapter1
[student@desktopX Documents]$ ls -lR
.:
total 16
drwxrwxr-x. 2 student student 4096 Feb 11 11:58 ProjectX
drwxrwxr-x. 2 student student 4096 Feb 11 11:55 ProjectY
drwxrwxr-x. 5 student student 4096 Feb 11 11:56 Thesis
-rw-rw-r--. 1 student student    0 Feb 11 11:54 thesis_chapter2_reviewed.odf

./ProjectX:
total 0
-rw-rw-r--. 1 student student 0 Feb 11 11:58 thesis_chapter1.odf
-rw-rw-r--. 1 student student 0 Feb 11 11:58 thesis_chapter2.odf

./ProjectX/Thesis:
total 0

./ProjectY:
total 0

./Thesis:
total 12
drwxrwxr-x. 2 student student 4096 Feb 11 11:59 Chapter1
drwxrwxr-x. 2 student student 4096 Feb 11 11:56 Chapter2
drwxrwxr-x. 2 student student 4096 Feb 11 11:56 Chapter3

./Thesis/Chapter1:
total 0
-rw-rw-r--. 1 student student 0 Feb 11 11:54 thesis_chapter1.odf

./Thesis/Chapter2:
total 0
-rw-rw-r--. 1 student student 0 Feb 11 11:54 thesis_chapter2.odf

./Thesis/Chapter3:
total 0
[student@desktopX Documents]$
```

El primer comando **mv** es un ejemplo de cómo cambiarle el nombre a un archivo. El segundo provoca que el archivo sea reubicado en otro directorio.

Eliminación de archivos y directorios

La sintaxis predeterminada para **rm** elimina archivos, pero no directorios. La eliminación de un directorio y, potencialmente, de muchos subdirectorios y archivos que estén en él, requiere la opción **-r** *recursive*. No existe una función de deshacer la eliminación de línea de comandos; ni tampoco una papelera de reciclaje desde donde se pueda restaurar la eliminación.

```
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ rm thesis_chapter2_reviewed.odf
[student@desktopX Documents]$ rm Thesis/Chapter1
rm: cannot remove `Thesis/Chapter1': Is a directory
[student@desktopX Documents]$ rm -r Thesis/Chapter1
[student@desktopX Documents]$ ls -l Thesis
total 8
drwxrwxr-x. 2 student student 4096 Feb 11 12:47 Chapter2
drwxrwxr-x. 2 student student 4096 Feb 11 12:48 Chapter3
[student@desktopX Documents]$ rm -ri Thesis
rm: descend into directory `Thesis'? y
rm: descend into directory `Thesis/Chapter2'? y
rm: remove regular empty file `Thesis/Chapter2/thesis_chapter2.odf'? y
rm: remove directory `Thesis/Chapter2'? y
rm: remove directory `Thesis/Chapter3'? y
rm: remove directory `Thesis'? y
[student@desktopX Documents]$
```

Después de que **rm** no pudo eliminar el directorio **Chapter1**, la opción **-r** *recursive* pudo hacerlo en forma correcta. El último comando **rm** analizó primero cada subdirectorio y eliminó en forma individual los archivos que contenía antes de eliminar cada directorio que ahora está vacío. El uso de **-i** pide la confirmación para cada eliminación de forma interactiva. Esto es básicamente lo opuesto de **-f**, que fuerza la eliminación sin solicitar confirmación al usuario.

El comando **rmdir** elimina directorios solo si están vacíos. Los directorios eliminados no pueden recuperarse.

```
[student@desktopX Documents]$ pwd
/home/student/Documents
[student@desktopX Documents]$ rmdir ProjectY
[student@desktopX Documents]$ rmdir ProjectX
rmdir: failed to remove `ProjectX': Directory not empty
[student@desktopX Documents]$ rm -r ProjectX
[student@desktopX Documents]$ ls -lR
.:
total 0
[student@desktopX Documents]$
```

El comando **rmdir** no pudo eliminar **ProjectX** que no estaba vacío, pero **rm -r** pudo hacerlo en forma correcta.

Referencias

Páginas del manual **cp**(1), **mkdir**(1), **mv**(1), **rm**(1) y **rmdir**(1)

Práctica: Administración de archivo de línea de comandos

Coincidencia de nombres de archivo mediante el uso de expansión de nombre de ruta

RH124

Objetivos

Tras finalizar esta sección, los estudiantes deberían poder usar metacaracteres y técnicas de expansión para mejorar la eficiencia de procesamiento de administración de archivos.

Globbing de archivos: expansión de nombre de ruta

La shell Bash tiene una capacidad de coincidencia de nombre de ruta que, históricamente, se llamó *globbing*, que es la forma abreviada del programa de expansión de ruta de archivo del "comando global" de las primeras versiones de UNIX. La función globbing de Bash, que comúnmente se denomina *coincidencia de patrón* o "comodines", facilita la administración de grandes cantidades de archivos. Con *metacaracteres* que se "expanden" para establecer una coincidencia entre los archivos y los nombres de ruta que se buscan, los comandos trabajan en un conjunto de archivos orientado al mismo tiempo.

Coincidencia del patrón

Globbing es una operación de análisis de comandos de la shell que expande un patrón de comodín en una lista de nombres de ruta coincidentes. Los metacaracteres de la línea de comandos se reemplazan por la lista de coincidencia antes de la ejecución del comando. Los patrones, en especial las clases de carácter entre corchetes, que no ofrecen coincidencias, muestran la solicitud de patrón original como texto literal. Los siguientes son metacaracteres y clases de patrón de uso frecuente.

Patrón	Coincidencias
*	Cualquier secuencia de cero o más caracteres.
?	Cualquier carácter individual.
~	El directorio de inicio del usuario actual.
~*username*	Directorio de inicio del *username* del usuario.
~+	El directorio de trabajo actual.
~-	El directorio de trabajo anterior.
[*abc...*]	Cualquier carácter en la clase incluida.
[!*abc...*]	Cualquier carácter que *no* esté incluido en la clase.
[^*abc...*]	Cualquier carácter que *no* esté incluido en la clase.
[[:alpha:]]	Cualquier carácter alfabético.[1]
[[:lower:]]	Cualquier carácter en minúsculas.[1]
[[:upper:]]	Cualquier carácter en mayúscula.[1]
[[:alnum:]]	Cualquier dígito o carácter alfabético.[1]
[[:punct:]]	Cualquier carácter imprimible que no sea un espacio o alfanumérico.[1]
[[:digit:]]	Cualquier dígito, **0-9**.[1]

Patrón	Coincidencias
[[:space:]]	Cualquier carácter de espacio en blanco; puede incluir tabulaciones, renglón nuevo, retornos de carro y avances de página, así como el espacio.[1]
Nota	[1]clase de carácter POSIX establecido previamente; se adapta a la región actual.

Un conjunto de archivos de muestra sirve para demostrar la expansión.

```
[student@desktopX ~]$ mkdir glob; cd glob
[student@desktopX glob]$ touch alfa bravo charlie delta echo able baker cast dog easy
[student@desktopX glob]$ ls
able  alfa  baker  bravo  cast  charlie  delta  dog  easy  echo
[student@desktopX glob]$
```

Primero, coincidencias de patrón simple que usan ***** y **?**.

```
[student@desktopX glob]$ ls a*
able  alfa
[student@desktopX glob]$ ls *a*
able  alfa  baker  bravo  cast  charlie  delta  easy
[student@desktopX glob]$ ls [ac]*
able  alfa  cast  charlie
[student@desktopX glob]$ ls ????
able  alfa  cast  easy  echo
[student@desktopX glob]$ ls ?????
baker  bravo  delta
[student@desktopX glob]$
```

Expansión de tilde

El carácter del tilde (~), cuando está seguido de una barra separadora, coincide con el directorio principal del usuario actual. Si está seguido por una secuencia de caracteres hasta la barra, se interpretará como un nombre de usuario, en caso de que uno coincida. Si ningún nombre de usuario coincide, aparecerá el propio tilde seguido de la secuencia de caracteres.

Sustitución de comandos

La sustitución de comandos permite obtener un comando para reemplazar el comando mismo. La sustitución de comandos se produce cuando un comando está encerrado entre un signo de dólar al principio y paréntesis, **$(*command*)**, o con acento grave, **`*command*`**. La forma con acento grave es más antigua y tiene dos desventajas: 1) el acento grave puede confundirse fácilmente a la vista con las comillas simples, y 2) el acento grave no puede anidarse dentro de los acentos graves. La forma **$(*command*)** puede anidar varias expansiones de comando dentro de cada una.

```
[student@desktopX glob]$ echo Today is `date +%A`.
Today is Wednesday.
[student@desktopX glob]$ echo The time is $(date +%M) minutes past $(date +%l%p).
The time is 26 minutes past 11AM.
[student@desktopX glob]$
```

Evitar la expansión de argumentos

Muchos caracteres tienen un significado especial en la shell Bash. Para ignorar los significados especiales del metacaracter, se usa la *cita* y el *escape* para evitar la expansión de la shell. La barra invertida (\) es un carácter de escape en Bash y protege al carácter individual siguiente de que se interprete de manera especial. Para proteger las secuencias de carácter más extensas, se usan comillas simples (') o dobles (") para encerrar las secuencias.

Use comillas dobles para suprimir el globbing y la expansión de la shell, pero permita la sustitución de comandos y variables. A nivel conceptual, la sustitución de variables es idéntica a la sustitución de comandos, pero puede usar sintaxis de llaves opcional.

```
[student@desktopX glob]$ host=$(hostname -s); echo $host
desktopX
[student@desktopX glob]$ echo "***** hostname is ${host} *****"
***** hostname is desktopX *****
[student@desktopX glob]$ echo Your username variable is \$USER.
Your username variable is $USER.
[student@desktopX glob]$
```

Use comillas simples para interpretar *todo* el texto literalmente. Observe la diferencia, tanto en la pantalla como en el teclado, entre las comillas simples (') y el acento grave de sustitución de comando (`). Además de suprimir el globbing y la expansión de la shell, las comillas dirigen a la shell para que también suprima la sustitución de comandos y variables. El signo de interrogación es el metacaracter que también necesita protección para evitar la expansión.

```
[student@desktopX glob]$ echo "Will variable $host evaluate to $(hostname -s)?"
Will variable desktopX evaluate to desktopX?
[student@desktopX glob]$ echo 'Will variable $host evaluate to $(hostname -s)?'
Will variable $host evaluate to $(hostname -s)?
[student@desktopX glob]$
```

Práctica: Expansión del nombre de ruta