
BASH Unidad 2

Navegación y manipulación del sistema de archivos

Tabla de contenidos

Tarjetas didácticas

Para ayudarte a recordar los comandos de esta unidad y la anterior, su uso y las opciones más comunes, puedes apoyarte en estas tarjetas didácticas. La aplicación está disponible tanto en web como para móvil o tablet.

Navegar por el sistema de archivos

A continuación vamos a introducir los comandos:

- pwd — mostrar el directorio de trabajo.
- cd — cambiar el directorio de trabajo.
- ls — listar archivos y directorios.

Organización del sistema de archivos

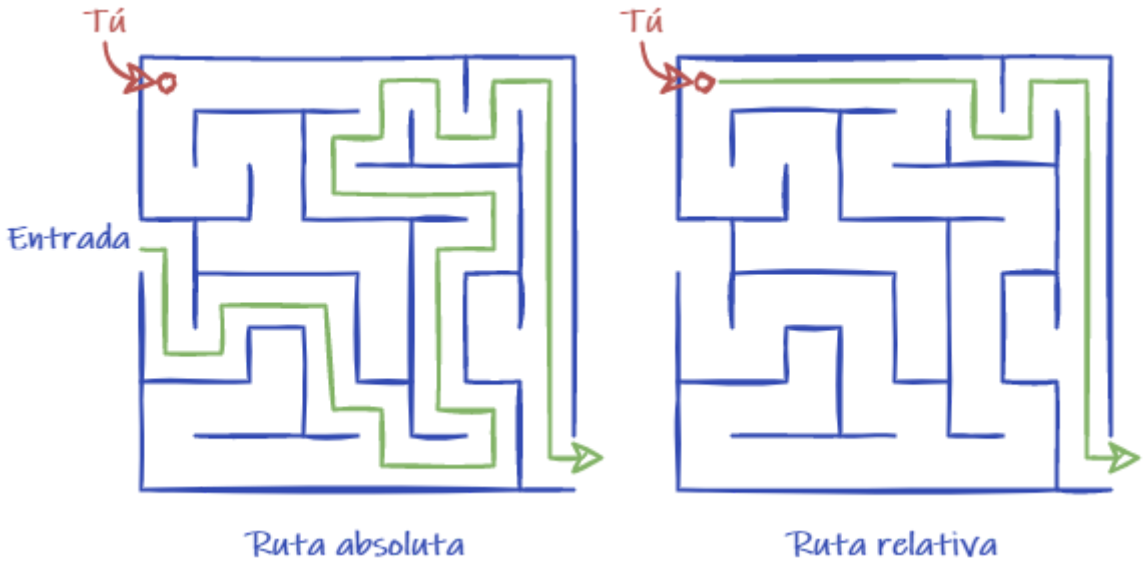
Los archivos en un sistema Linux se organizan en un árbol de directorios —también llamados carpetas, en otros sistemas operativos— cada uno de los cuales puede contener archivos y otros directorios —llamados subdirectorios—.

En todos los sistemas estilo UNIX como Linux:

- **El primer directorio del sistema de archivos es el directorio raíz**.
- **No se emplea el concepto de letras de unidad**. Mientras que en otros sistemas operativos, el sistema de archivos se divide en una serie de árboles de directorio independientes —uno para cada unidad de almacenamiento identificada por una letra diferente— en Linux hay un único árbol. A diferentes unidades de almacenamiento les puede corresponder distintas ramas del árbol principal, pero únicamente existe un árbol.

Puesto que en la shell no podemos disponer de una imagen gráfica del sistema de archivos, necesitamos una forma diferente de representarlo:

1. Piensa en el árbol del sistema de archivos como un laberinto en el que te has despertado.
2. En cualquier momento puedes estar en una única habitación del laberinto —un directorio—.
3. Dentro de esa sala puedes ver su contenido —los archivos— y puertas de salida hacia otras salas —el directorio padre y los subdirectorios del directorio—.
4. En cualquier momento puedes describir la ruta para llegar a una sala trazándola desde la entrada o —el directorio raíz— o desde la sala donde estás actualmente —al que llamamos directorio de trabajo—.



Directorio de trabajo

Como hemos comentado, el directorio en el que estás en un momento dado es el directorio de trabajo. Para descubrir su ruta desde la raíz solo tienes que usar el comando pwd.

```
yo@mihost:~$ pwd
/home/yo
```

Ahora cambiamos de directorio usando el comando cd y volvemos a probar pwd:

```
yo@mihost:~$ cd /var/lib
yo@mihost:/var/lib$ pwd
/var/lib
```

Luego podemos cambiar al directorio padre del directorio de trabajo actual usando el comando cd y volvemos a probar pwd:

```
yo@mihost:~$ cd ..
yo@mihost:/var$ pwd
/var
```

El directorio de trabajo se hereda. Es decir, si cambiamos en la shell de directorio de trabajo y luego ejecutamos un comando, este comando se ejecuta en el nuevo directorio de trabajo.

Directorio personal

El directorio de trabajo inicial, nada más abrir una sesión, es tu directorio o carpeta personal:

- Es el lugar donde pondrás tus propios archivos.

- En la mayor parte de los sistemas dicho directorio será `/home/tu_nombre_de_usuario` pero realmente la elección final está en manos del administrador del equipo.

Rutas absolutas

Las rutas absolutas comienzan en el directorio raíz y siguen el árbol rama a rama hasta alcanzar el directorio o archivo deseado.

Por ejemplo, hay un directorio del sistema donde se instalan la mayor parte de los programas. La ruta a ese directorio es /usr/bin. Eso significa que:

1. Partiendo del directorio raíz —denotando por la primera barra "/" de la ruta—…
2. ...hay un directorio llamado "usr"...
3. ...que a su vez contiene un directorio llamado "bin".

Para seguir un camino en el laberinto y cambiar el directorio de trabajo, solo tenemos que usar el comando cd. Para hacerlo, escribe `cd` seguido de la ruta del directorio de trabajo deseado. Vemos que ocurre:

```
yo@mihost:~$ cd /usr/bin
yo@mihost:/usr/bin$ pwd
/usr/bin
yo@mihost:/usr/bin$ ls
[                       lwp-request
2to3                    lwp-rget
2to3-2.6                lxterm
a2p                     lz
aalib-config            lzcat
aconnect                lzma
acpi_fakekey            lzmadec
acpi_listen             lzmainfo
add-apt-repository      m17n-db
addpart                 magnifier
...
```

Obsérvese que como el *prompt* ha cambiado de `yo@mihost:~$` a `yo@mihost:/usr/bin$` para ayudar a saber en todo momento cuál es nuestro directorio de trabajo, sin usar pwd.

También hemos aprovechado para usar el comando ls para listar los archivos y directorios en el directorio de trabajo.

```
yo@mihost:~$ ls
bin Descargas  Desktop  Documentos  Dropbox  Imágenes  Insync
Música Vídeos  VirtualBox VMs  Workspace
```

Rutas relativas

Donde una ruta relativa comienza desde el directorio raíz y avanza hasta el directorio o archivo de destino, una ruta relativa comienza en el directorio de trabajo. Para que esto sea posible se utiliza notación especial para representar ciertas posiciones relativas dentro del sistema de archivos:

- "." — hace referencia al directorio de trabajo.
- ".." — hace referencia al directorio padre del directorio de trabajo.

Veamos cómo funciona. Primero vayamos de nuevo a /usr/bin:

```
yo@mihost:~$ cd /usr/bin
yo@mihost:/usr/bin$ pwd
/usr/bin
```

Ahora supongamos que queremos ir al padre de /usr/bin, que es /usr. Eso lo podríamos hacer usando una ruta absoluta:

```
yo@mihost:/usr/bin$ cd /usr
yo@mihost:/usr$ pwd
/usr
```

o una ruta relativa:

```
yo@mihost:/usr/bin$ cd ..
yo@mihost:/usr$ pwd
/usr
```

Ahora podemos volver de /usr a /usr/bin, también de dos maneras diferentes. Usando una ruta absoluta:

```
yo@mihost:/usr$ cd /usr/bin
yo@mihost:/usr/bin$ pwd
/usr/bin
```

o una ruta relativa:

```
yo@mihost:/usr$ cd ./bin
yo@mihost:/usr/bin$ pwd
/usr/bin
```

Ahora bien, en casi todos los casos se puede omitir el "`./`", ya que se entiende que es implícito si la ruta no es absoluta —si una ruta no empieza por `/` es que es relativa—:

```
yo@mihost:/usr$ cd bin
yo@mihost:/usr/bin$ pwd
/usr/bin
```

Otros atajos

- Si usas el comando cd seguido de nada, el directorio de trabajo cambiará a tu directorio personal. Igual que si usas `cd ~`.
- Si usas el comando `cd ~usuario`, el directorio de trabajo cambiará al directorio personal del usuario especificado.
- Si usas `cd -`, el directorio de trabajo cambiará al directorio de trabajo previo.

Algunos detalles sobre los nombres de archivo

- **Los nombres de archivo en Linux y otros sistemas POSIX distinguen entre mayúsculas y minúsculas**. Es decir, que los nombres "File1" y "file1" hacen referencia a archivos diferentes.
- **Los archivos cuyo nombre comienza por "." son archivos ocultos**. Esto solo significa que ls no los mostrará, pero `ls -a` sí lo hará. Generalmente, se utilizan para almacenar la configuración de los diferentes programas.
- **Linux y otros sistemas POSIX no entienden el concepto de extensión**. Los archivos se pueden nombrar como se quiera, aunque algunas aplicaciones si esperan que usemos extensiones.
- **Aunque Linux permite nombres con espacios y signos de puntuación, es mejor limitarse a puntos ".', guiones "-' y guiones bajos "_"**. Es especialmente importante evitar el uso de espacios, usando guiones bajos en su lugar.

Tour guiado por el sistema de archivos

La tabla de abajo muestra una lista de sitios del sistema de archivos interesantes para explorar. Obviamente, no se trata de una lista completa, pero sin duda se incluyen algunos de los más interesantes.

Directorio	Descripción
/	Directorio raíz. Es donde comienza el sistema de archivos. En la mayor parte de los casos solo contiene subdirectorios.
/boot	Aquí residen los archivos del núcleo de Linux y del cargador de arranque. El núcleo está en un archivo llamado vmlinuz.
/etc	Contiene los archivos de configuración del sistema. La mayor parte de ellos están en formato texto. Algunos de los más interesantes son: **/etc/passwd** — contiene la información esencial de cada usuario. Aquí es donde se definen los usuarios del sistema.

	/etc/groups — contiene la información esencial de cada grupo de usuarios. Aquí es donde se definen los grupos de usuario del sistema. **/etc/fstab** — contiene una tabla de dispositivos de almacenamiento (por lo general, discos duros) que serán montados (conectados al sistema de archivos) durante el arranque del sistema. **/etc/hosts** — lista de los nombres de red y sus IP que el sistema conoce sin depender de otros servicios, como el DNS. **/etc/init.d** — tradicionalmente, contiene los scripts que inician los distintos servicios, generalmente durante el arranque. Pero está en desuso ahora que se está tendiendo a gestionar los servicios mediante `systemd`.
/bin, /usr/bin	Contienen la mayor parte de los programas del sistema. El directorio /bin tiene aquellos que son esenciales para que el sistema funcione. Mientras que /usr/bin más bien contiene aplicaciones para los usuarios.
/sbin, /usr/sbin	Contienen programas para el administrador del sistema.
/usr	Contiene diversos archivos que dan soporte a las aplicaciones de usuario: **/usr/share/X11** — Archivo de soporte del sistema gráfico X Window. **/usr/share/dict** —Diccionarios para el corrector ortográfico. Por ejemplo, para los programas **look** y **aspell**. **/usr/share/doc** — varios archivos de documentación en diversos formatos. **/usr/share/man** — las páginas de **man**. **/usr/src** — archivos de código fuente. Por ejemplo, si se instala el código fuente del núcleo del sistema, estará aquí.
/usr/local	**/usr/local** y sus subdirectorios se usan para la instalación de programas que no pertenecen a la distribución de Linux que estamos usando de manera oficial. Por ejemplo, si encontramos un nuevo programa y queremos instalarlo, por lo general lo copiaremos a `/usr/local/bin`.
/var	Contiene archivos que cambian durante el funcionamiento del sistema. Esto incluye:

	/var/log — contiene los archivos de registro de las distintas actividades desarrolladas por los servicios del sistema y los errores encontrados. **/var/spool** — contiene archivos que han sido encolados para algún servicio, por ejemplo: emails, trabajos de impresión, etc. Cuando llega un correo al sistema (suponiendo que tenemos instalado un servidor de correo electrónico en el sistema) los mensajes se almacenan en **/var/spool/mail**.
/lib	Contiene las librerías compartidas, similares a las DLL de Microsoft® Windows®.
/home	Contiene los directorios personales de los usuarios. Generalmente, el directorio personal de cada usuario es el único lugar donde estos pueden escribir.
/root	Es el directorio personal del administrador del sistema.
/tmp	Es un directorio donde crear archivos temporales.
/dev	Contiene los dispositivos que están disponibles para el sistema. En los sistemas estilo UNIX los dispositivos son tratados como archivos. Así que estos son archivos especiales en los que al leer o escribir se está accediendo al dispositivo correspondiente. Por ejemplo `/dev/sda`, que es el primer disco duro del sistema.
/proc	Directorio especial que ofrece acceso a algunos aspectos del núcleo, por lo que en realidad no ocupa espacio de almacenamiento. **Hay un grupo de directorios cuyo nombre es un número, cada uno de los cuales corresponde a un proceso del sistema**. Además, hay cierto número de archivos que permiten acceder a algunos aspectos de la configuración del sistema. Por ejemplo /proc/cpuinfo, que indica detalles de la CPU.
/media, /mnt	Ambos directorios son usados para los puntos de montaje. Los distintos dispositivos de almacenamiento se pueden hacer accesibles al sistema de archivos en distintos lugares. **Al proceso de hacer accesible uno de estos dispositivos se lo denomina montaje**. **Cuando el sistema arranca, lee la lista de instrucciones de /etc/fstab que describe qué dispositivo debe ser montado en cada punto de montaje del árbol de directorios**. Según como hayamos instalado el sistema, es posible que /boot, /home o, incluso, /usr y /var sean puntos de montaje. **Los dispositivos adicionales no extraíbles se suelen montar automáticamente** —durante el arranque, si así se indica en /etc/fstab— **o manualmente en subdirectorios de /mnt.**

Dado que los dispositivos extraíbles no pueden ser montados permanentemente, y generalmente no están disponibles durante el arranque, **el sistema los montan automáticamente en subdirectorios de /media cuando son insertados o conectados al equipo**.

Cuando se requiere montar manualmente algún dispositivo extraíble, usar subdirectorios de /mnt es lo más conveniente. Así que no resulta raro encontrar directorios como /mnt/usbbackup o /mnt/dvd.

Para ver los puntos de montaje en uso, se puede usar el comando mount.

Mirando a nuestro alrededor

Ahora que sabemos movemos de directorio de trabajo en directorio de trabajo, vamos a ver algunas herramientas que pueden resultarnos útiles en nuestro paseo por el laberinto del árbol de directorios:

- ls — listar archivos y directorios.
- find — buscar archivos y directorios en el árbol de directorios.
- less — ver archivos de texto.
- file — identificar el tipo de contenido de un archivo.

ls

Como ya hemos visto, el comando ls permite mostrar el contenido de un directorio, por lo que seguramente sea el comando de Linux que se usa con más frecuencia. Veamos algunos ejemplos:

Ejemplos del comando ls

Comando	Resultado
ls	Lista los archivos del directorio de trabajo
ls /bin	Lista los archivos del directorio /bin —o cualquier otro directorio que se especifique—,
ls -l	Lista los archivos del directorio de trabajo usando el formato largo.
ls -l /bin /etc	Lista los archivos de los directorios /bin y /etc usando el formato largo.
ls -la ..	Lista todos los archivos —incluso los ocultos, cuyo nombre comienza por punto "."— en el directorio padre del directorio de trabajo usando el formato largo.

Como comentamos en la introducción, ls admite múltiples opciones y rutas a archivos y

---

directorios como argumentos.

Detalles del formato largo

Si usamos la opción "-l" con ls obtendremos una lista de archivos con información muy valiosa acerca de los archivos listados. A continuación mostramos un ejemplo:

```
-rw-------   1 jmtorres jmtorres    576 Apr 17  1998 weather.txt
drwxr-xr-x   6 jmtorres jmtorres   1024 Oct  9  1999 web_page
-rw-rw-r--   1 jmtorres jmtorres 276480 Feb 11 20:41 web_site.tar
-rw-------   1 jmtorres jmtorres   5743 Dec 16  1998 xmas_file.txt
----------     -------- -------- ------ ------------ -------------
    |               |        |        |         |            |
    |               |        |        |         |          Nombre
    |               |        |        |         |
    |               |        |        |         +--- Instante modif.
    |               |        |        |
    |               |        |        +------------ Tamaño en bytes
    |               |        |
    |               |        +--------------------- Grupo
    |               |
    |               +------------------------------ Propietario
    |
    +---------------------------------------------- Permisos
```

- **Nombre** — Nombre del archivo o directorio
- **Instante de modificación** — Instante en el que el archivo fue modificado por última vez. Si dicho momento ocurrió hace más de seis meses, se muestra la fecha con el año. En cualquier otro caso, se muestra la hora del día junto a la fecha.
- **Tamaño** — El tamaño del archivo en bytes.
- **Grupo** — El grupo del sistema al que se le asignan permisos de acceso al archivo.
- **Propietario** — El usuario propietario del archivo.
- **Permisos** — Una representación de los permisos de acceso al archivo:
    - El primer carácter es el tipo de archivo. Un “-“ indica que se trata de una archivo regular. Mientras que una “d“ indica un directorio.
    - El segundo conjunto de 3 caracteres muestra los permisos de lectura, escritura y ejecución del propietario.
    - El tercer conjunto de tres caracteres muestra los derechos del grupo del archivo.
    - El último conjunto de tres caracteres muestra los permisos para cualquier otro usuario del sistema.

find

El comando find es muy útil para buscar archivos y directorios que cumplan con unas condiciones específicas. Por ejemplo:

- Que el nombre contenga cierta cadena de caracteres o encaje con un patrón determinado.
- Que sea un enlace a un archivo cuyo nombre contiene cierta cadena o encaja con un patrón determinado.
- Que haya sido usado por última vez en cierto período de tiempo.
- Que tenga un tamaño comprendido dentro de cierto intervalo.
- Que sea de cierto tipo: regular, directorio, enlace simbólico, etc.
- Que pertenezca a cierto usuario o grupo.
- Que tenga ciertos permisos de acceso.

Una vez ubicados los archivos puedes realizar diversas acciones sobre ellos:

- Verlo o editarlo.
- Guardar sus nombres en otro archivo.
- Eliminarlos, renombrarlos, moverlos, etc.
- Cambiar sus permisos de acceso.
- etc.

Simplificando, el formato de la línea de comandos de find es el siguiente:

```
find [directorio] [condiciones] [acciones]
```

de tal forma que el comando buscará recursivamente desde `directorio` archivos y directorios que cumplan las condiciones establecidas. Para cada archivo encontrado, se ejecutarán las acciones indicadas. Si no se indica un `directorio`, la búsqueda tiene lugar desde el directorio actual de trabajo. Si no se indican acciones, la acción por defecto es `-print`, que imprime la ruta del archivo o directorio encontrado por la salida estándar.

Lo más sencillo para entender cómo se usa es ver algunos ejemplos concretos:

- `find /var -name "*.log"`
  busca en el directorio `/var` los archivos terminados en ".log" e imprime sus nombres en la salida estándar.
- `find /tmp -size +200k -print`
  busca en el directorio `/tmp` archivos mayores de 200k e imprime sus nombres por la salida estándar (como `-print` es la acción por defecto, realmente no hace falta indicarla). En los argumentos numéricos —como es el caso de la opción "-size"— "+N" significa mayor que N, "-N" significa menor que N, "N" significa exactamente igual a N.
- `find -user yo`
  buscar en el directorio actual todos los archivos y directorios del usuario “yo” e imprime sus nombres por la salida estándar. También se pueden buscar los de un grupo concreto mediante la opción “-group”.
- `find /var/spool/mail -atime +30`
  busca en el directorio `/var/spool/mail` archivos no accedidos hace más de 30

---

días. La opción "-atime" se refiere al tiempo transcurrido desde la última lectura, "-mtime" desde la última modificación de estado o permisos y "-ctime" desde la última modificación de contenido.

- `find /var/tmp -empty -exec rm {} \;`
  busca en el directorio `/var/tmp` archivos vacíos y los borra.
- `find -type d`
  buscar directorios en el directorio actual. También se pueden buscar archivos regulares (`-type f`), enlaces simbólicos (`-type l`) y tuberías (`-type p`), etc.
- `find /home -nouser -ls`
  busca archivos en `/home` cuyo propietario no existe entre las cuentas del sistema. Esta situación se da cuando la cuenta de usuario ha sido borrada, pero han permanecido los archivos creados por ese usuario. La acción `-ls` indicada, sirve para mostrar información sobre cada archivo encontrado en un formato similar al del comando **ls**.

less

Si queremos ver el contenido de un archivo de texto, el comando adecuado es less. Se trata de un comando muy útil dado que muchos de los archivos usados para el control y la configuración de Linux son simples archivos de texto.

El programa less puede ser invocado simplemente de la siguiente manera:

```
yo@mihost:~$ less ruta_del_archivo
```

Por ejemplo:

```
yo@mihost:~$ less /etc/passwd
```

Una vez iniciado, less mostrará la primera página del archivo de texto.

Comandos de less

Dentro de less tenemos la posibilidad de usar alguno de los siguientes comandos:

Comando	Acción
Re. Pág. o b	Retroceder una página.
Av. Pág. o espacio	Avanzar una página.
G	Ir al final del archivo.
1G	Ir al principio del archivo.
/caracteres	Buscar la siguiente ocurrencia de los *caracteres* especificados.
n	Repetir la búsqueda hecha previamente.
h	Mostrar la lista completa de comandos y opciones soportadas

---

	por el programa.
q	Salir

Como hemos visto, otro comando que permite mostrar el contenido de un archivo es cat, aunque carece de las funcionalidades del comando less.

file

Según vamos moviéndonos por el sistema, puede ser muy útil determinar qué clase de datos contiene un archivo antes de intentar verlos, ya sea con less, cat o con cualquier otro programa. Para eso utilizaremos el programa file de la siguiente manera:

```
yo@mihost:~$ file ruta_del_archivo
```

Por ejemplo:

```
yo@mihost:~$ file /etc/hosts
/etc/hosts: ASCII text
```

El comando file puede reconocer muchos tipos de archivos, entre ellos los siguientes:

Tipo de archivo	Descripción	¿Visible como texto?
ASCII text	Archivo de texto en ASCII.	sí
Bourne-Again shell script text	Un script de BASH.	sí
ELF 32-bit LSB core file	Un archivo de volcado de memoria. Se crea cuando un programa explota.	no
ELF 32-bit LSB executable	Un programa binario ejecutable.	no
ELF 32-bit LSB shared object	Una librería compartida.	no
GNU tar archive	Un archivo en formato TAR. Se trata de una forma muy común de almacenar grupos de archivos.	no, usa **tar tvf** para ver la lista de su contenido.
gzip compressed data	Un archivo comprimido con “gzip“.	no
HTML document text	Una página web.	sí
JPEG image data	Una imagen comprimida en JPEG.	no

PostScript document text	Un documento en formato PostScript.	sí
RPM	Un paquete en formato Red Hat Package Manager usado por ciertas distribuciones de Linux	no, usa **rpm -q** para examinar su contenido
Debian binary package	Un paquete en formato Debian usado por ciertas distribuciones de Linux.	no, usa **dpkg -c** para examinar su contenido.
Zip archive data	Un archivo comprimido con “zip“.	no

Manipulando archivos

En esta lección introduciremos los siguientes comandos:

- cp — copiar archivos y directorios.
- ln — enlazar archivos y directorios.
- mv — mover y renombrar archivos y directorios.
- rm — eliminar archivos y directorios.
- mkdir— crear directorios.

que son los usados con más frecuencia en sistemas Linux para manipular archivos y directorios.

Sin duda, muchas de las tareas que hacen estos programas se pueden hacer de forma más cómoda con un gestor de archivos gráficos. Pero entonces, **¿por qué aprender a usar estos viejos comandos?** La respuesta es **por potencia y flexibilidad**. Aunque algunas manipulaciones de archivo son más sencillas usando el entorno gráfico, es muy probable que tareas complejas se hagan de forma más fácil con programas de línea de comandos.

Por ejemplo, supongamos que queremos copiar todos los archivos HTML de un directorio a otro, pero solo aquellos que no existen en el directorio de destino o que son más recientes que los de dicho directorio. ¿Cómo lo haríamos con un gestor de archivos gráficos? No parece sencillo, pero con la línea de comandos:

```
yo@mihost:~$ cp -u *.html destination
```

Comodines (wildcards)

Ya comentamos en su momento que la shell no sabe nada de los argumentos que pasa a los comandos. No sabe qué tipo de argumentos esperan los comandos ni cuáles son válidos. Sin embargo, existe un caso especial en el que la shell presupone que el argumento que intentamos pasar es una ruta o nombre de archivo.

Cuando en un argumento aparece alguno de los comodines o *wildcards*, la shell usa ese argumento como plantilla con la que buscar en el sistema de archivos. Si encuentra archivos o directorios que encajan con la plantilla, sustituye el argumento por las rutas

---

correspondientes. En bash a este proceso se le denomina *expansión de nombre de ruta*.

Estos son algunos comodines de uso frecuente:

Comodín	Significado
*	Encaja con 0 o más ocurrencias de cualquier carácter.
?	Encaja con una ocurrencia de cualquier carácter
[caracteres]	Encaja con una ocurrencia de cualquier carácter de entre los indicados entre corchetes. Por ejemplo, [abc], [0a-_] o [1234567890]
[!caracteres]	Encaja con una ocurrencia de cualquier carácter que no sea uno de los indicados entre corchetes. Por ejemplo, [!abc]
[[:clase:]]	Encaja con una ocurrencia de cualquier carácter según indica la clase POSIX especificada:
	[[:alnum:]] Caracteres alfanuméricos
	[[:alpha:]] Caracteres alfabéticos
	[[:digit:]] Dígitos del 0 al 9
	[[:upper:]] Caracteres alfabéticos en mayúsculas
	[[:lower:]] Caracteres alfabéticos en minúsculas
	[[:space:]] Caracteres de espacio, como el espacio o el tabulador
	[[:punct:]] Símbolos de puntuación
	[[:xdigit:]] Dígitos hexadecimales
[inicio-fin]	Encaja con una ocurrencia de cualquier carácter en el rango indicado. Por ejemplo, [a-z], [A-Z] o [0-9]
[!inicio-fin]	Encaja con una ocurrencia de cualquier carácter que no esté incluido en el rango indicado. Por ejemplo, [!a-z] o [!0-9]

Estos son algunos ejemplos de patrones de selección de archivos usando comodines:

Pattern	Matches
*	Todos los archivos
g*	Archivos cuyo nombre comienza por “g”
.*	Archivos cuyo nombre empieza por “.”

.	Cualquier archivo con extensión.
*.txt	Archivos que terminan en ".txt"
b*.txt	Archivos cuyo nombre comienza por “b” y termina en “.txt”
Data???	Cualquier archivo cuyo nombre comienza por "Data” seguido por exactamente 3 caracteres
[abc]*	Cualquier archivo cuyo nombre comienza por "a”, "b” o "c” seguido por cualquier otro carácter
[aeiou]*	Archivos que empiecen por vocal
[A-Z]*	Archivos que empiecen por letra mayúscula
[0-9]*	Archivos que empiecen por un dígito
[Dd]ocument*	Archivos que empiecen por "Documen" o "document"
*[!~]	Archivos que NO terminen en ~ (muchas veces estos archivos son copias de seguridad)
*.[Tt][Xx][Tt]	Archivos ".txt" en cualquier combinación de mayús/minús
archivo[!0-9]*	Archivos que empiecen por "archivo" pero NO seguido de un dígito
foto[0-9][0-9].jpg	Archivos como foto01.jpg, foto23.jpg, etc.
backup_????_??.tar	Archivos como backup_2024_01.tar (año y mes)
[[:upper:]]*	Cualquier archivo cuyo nombre comienza por una letra en mayúsculas.
[[:space:]]	Archivos con espacios en el nombre
[[:alpha:]]*[[:digit:]]	Archivos que empiecen por una letra y terminen en un dígito
[A-Z]*[a-z]	Archivos que empiecen en mayúscula y terminen en minúscula
BACKUP.[[:digit:]][[:digit:]]	Cualquier archivo cuyo nombre comienza por "BACKUP.” seguido por exactamente dos dígitos numéricos.
*[![:lower:]]	Cualquier archivo cuyo nombre no termine en minúsculas.
.[!.]*	Archivos ocultos (empiezan por ".") pero no "." ni ".."

Se pueden usar comodines en cualquier comando, pero solo tiene sentido usarlos cuando se aceptan múltiples rutas o nombres de archivos o directorios.

---

cp

El programa cp copia archivos y directorios. En su forma más simple copia un único archivo:

```
yo@mihost:~$ cp archivo1 archivo2
```

pero también puede copiar múltiples archivo a un directorio diferente:

```
yo@mihost:~$ cp archivo1 archivo2 archivo3 directorio_destino
```

Ejemplos del comando cp

Comando	Resultado
cp archivo1 archivo2	Copia el contenido de "archivo1" en "archivo2". Si "archivo2" no existe, es creado. Si existe, será sobreescrito en silencio con el contenido de "archivo1".
cp -i archivo1 archivo2	Como el ejemplo previo, pero la opción "-i" (interactivo) le indica que si "archivo2" existe, debe preguntar para pedir confirmación antes de sobreescribir su contenido.
cp archivo1 directorio1	Copia el contenido de "archivo1" (se pueden especificar varios archivos) en el directorio "directorio1".
cp -R directorio1 directorio2	Copia "directorio1" y su contenido. Si "directorio2" no existe, es creado. En caso contrario, "directorio1" es creado dentro de "directorio2".

Enlaces simbólicos

Durante el tour que hicimos por el sistema de archivos es posible que hayamos visto unos archivos un tanto particulares, especialmente en los directorios "/boot" y "/lib". Por ejemplo:

```
yo@mihost:~$ ls -l /boot
lrwxrwxrwx     25 Jul  3 16:42 System.map -> System.map-2.0.36-3
-rw-r--r-- 105911 Oct 13  1998 System.map-2.0.36-0.7
-rw-r--r-- 105935 Dec 29  1998 System.map-2.0.36-3
-rw-r--r-- 181986 Dec 11  1999 initrd-2.0.36-0.7.img
-rw-r--r-- 182001 Dec 11  1999 initrd-2.0.36.img
lrwxrwxrwx     26 Jul  3 16:42 module-info -> module-info-2.0.36-3
-rw-r--r--  11773 Oct 13  1998 module-info-2.0.36-0.7
-rw-r--r--  11773 Dec 29  1998 module-info-2.0.36-3
lrwxrwxrwx     16 Dec 11  1999 vmlinuz -> vmlinuz-2.0.36-3
-rw-r--r-- 454325 Oct 13  1998 vmlinuz-2.0.36-0.7
-rw-r--r-- 454434 Dec 29  1998 vmlinuz-2.0.36-3
```

Obsérvese la notación usada después de los nombres de archivo de "System.map", "module-info" y "vmlinuz". Eso es así porque los tres archivos no son archivos regulares, sino *enlaces simbólicos*.

Los enlaces simbólicos son un tipo especial de archivo que apunta a otro archivo o directorio. Eso permite que un mismo archivo tenga múltiples nombres, incluso en varios directorios, al mismo tiempo. Cuando al sistema se le da un nombre de archivo que realmente es un enlace simbólico, él de forma transparente se encarga de hacernos acceder al archivo apuntado.

Los enlaces simbólicos se pueden crear tanto mediante el comando cp:

```
yo@mihost:~$ cp -s algun_archivo enlace_simbolico
```

como con el comando ln:

```
yo@mihost:~$ ln -s algun_archivo enlace_simbolico
```

mv

El comando mv mueve o renombra archivos y directorio. Para renombrar un archivo es tan sencillo como hacer lo siguiente:

```
yo@mihost:~$ mv archivo1 archivo2
```

Para mover archivos —o directorios— a un directorio diferente:

```
yo@mihost:~$ mv archivo1 archivo2 archivo3 directorio
```

Hay que tener en cuenta que mover archivos dentro de un mismo sistema de archivos es una acción rápida. Básicamente, es solo un cambio de nombre. Pero mover archivos a otro dispositivo o sistema de archivos implica copiar los archivos y luego borra los originales, por lo que suele llevar más tiempo.

Ejemplos del comando mv

Comando	Resultado
mv archivo1 archivo2	Si "archivo2" no existe, el "archivo1" es renombrado a "archivo2". Si "archivo2" existe, será sobreescrito en silencio con el contenido de "archivo1".
mv -i archivo1 archivo2	Como el ejemplo previo, pero la opción "-i" (interactivo) le indica que si "archivo2" existe, debe preguntar para pedir confirmación antes

---

	de sobreescribir su contenido.
mv archivo1 archivo2 archivo3 directorio1	Los archivos "archivo1", "archivo2" y "archivo3" son movidos a "directorio1". Si no existe, **mv** terminará con un error.
mv directorio1 directorio2	Si "directorio2" no existe, "directorio1" es renombrado. En caso contrario, "directorio1" es movido dentro de "directorio2".

rm

El programa rm elimina archivos.

```
yo@mihost:~$ rm archivo...
```

Y también puede ser usado para eliminar directorios:

```
yo@mihost:~$ rm -r directorio
```

Ejemplos del comando rm

Comando	Resultado
rm archivo1 archivo2	Borra "archivo1" y "archivo2".
rm -i archivo1 archivo2	Como el ejemplo previo, pero la opción "-i" (interactivo) le indica que debe preguntar para pedir confirmación antes de borrar.
rm -r directorio1 directorio2	Los directorios "directorio" y "directorio2" son borrados con todo su contenido.

Hay que tener cuidado con el comando rm, puesto que en la línea de comandos no hay papelera ni comandos para recuperar lo borrado. **Si se van a usar comodines, antes de usar rm se puede construir el comando usando ls para comprobar el efecto de los comodines antes del borrado**. Una vez se está seguro de lo que se va a borrar, se puede sustituir ls por rm.

mkdir

El comando mkdir se usa para crear directorios:

```
yo@mihost:~$ mkdir directorio1 directorio2 directorio3
```

Soporta la opción "-p" que sirve para pedirle que no solo cree el último directorio de una ruta sino también todos los directorios anteriores, si no existen:

```
yo@mihost:~$ mkdir -p /home/yo/Documentos/SSOO/práctica1
```

Usando comodines en los comandos

Dado que los comandos de los que hemos hablado aceptan múltiples archivos y directorios como argumento, es posible utilizar comodines para especificarlos de forma más eficaz.

Ejemplos de comandos usando comodines

Comando	Resultado
cp *.txt archivos_texto	Copia todos los archivos en el directorio actual cuyo nombre termina en ".txt" a un directorio llamado archivos_texto
mv mi_directorio ../*bak nuevo_directorio	Mueve el directorio "mi_directorio" y todos los archivos que terminan en ".bak" en el directorio padre del directorio actual de trabajo a un directorio llamado "nuevo_directorio". Dicho directorio debe existir.
rm *~	Borrar todos los archivos en el directorio actual de trabajo que terminan en "~". Algunas aplicaciones crean copias de seguridad de los archivos nombrándolas de esa manera. Con este comando se puede limpiar el directorio actual de esas copias.

Permisos

Los sistemas Linux no solo son multitarea sino también multiusuario. Eso significa que **más de un usuario puede tener abierta una sesión en el sistema al mismo tiempo**. Incluso aunque un ordenador solo tenga un monitor o un teclado, otros usuarios podrían conectarse al mismo tiempo a través de Internet —por medio de ssh o herramientas similares— y abrir sesión en el mismo ordenador.

Para que esto sea práctico, deben existir métodos para proteger a unos usuarios respecto a otros. Después de todo, el sistema no debe permitir aquellas acciones que pueden dañar al propio sistema ni las que puedan hacer que un usuario interfiera con otros.

En este tema estudiaremos los siguientes comandos:

- chmod — cambiar los permisos de acceso a archivos.
- su — convertirse en superusuario temporalmente.
- sudo — convertirse en superusuario temporalmente.
- chown — cambiar el propietario.
- chgrp — cambiar el grupo.

Permisos de archivo

En los sistemas Linux a cada archivo o directorio se le asignan permisos de acceso específicos para:

- El propietario del archivo.
- Los miembros del grupo propietario del archivo.
- Para todos los demás usuarios del sistema.

Estos permisos pueden ser:

- Leer
- Escribir
- O ejecutar —cargar y lanzar un archivo como un programa—.

Para ver los permisos de un archivo se puede usar el comando **ls**. Por ejemplo, si miramos el programa "bash" localizado en el directorio "/bin":

```
yo@mihost:~$ ls -l /bin/bash
-rwxr-xr-x 1 root root 975488 mar 30  2013 /bin/bash
```

Lo que nos indicaría que:

- El archivo "/bin/bash" es propiedad del usuario "root", que es el superusuario.
- El propietario tiene permisos para leer, escribir y ejecutar este archivo.
- El archivo es propiedad del grupo "root".
- Los miembros del grupo "root" pueden leer y ejecutar el archivo.
- El resto de los usuarios pueden leer y ejecutar el archivo.

En el siguiente diagrama se puede observar cómo interpretan los permisos en el listado:

```
- rwx rwx rwx
- --- --- ---
|  |   |   |
|  |   |   +------ Lectura (r), escritura (w) y ejecución (x)
|  |   |           para todos los usuarios.
|  |   |
|  |   +---------- Lectura (r), escritura (w) y ejecución (x)
|  |               para los miembros del grupo propietario.
|  |
|  +-------------- Lectura (r), escritura (w) y ejecución (x)
|                  para el propietario.
|
+----------------- Tipo de archivo:
                   - indica un archivo regular
```

```
d indica un directorio
```

chmod

El comando chmod se usa para cambiar los permisos de un archivo o directorio. Hay dos formas de especificar estos permisos, la notación octal y la simbólica. Aunque nosotros nos centraremos únicamente en la primera[1].

Es sencillo pensar en la configuración de permisos como una serie de bits —que es además como lo implementa el sistema operativo— donde un 1 significa que el permiso está activo y un 0 es que no:

```
rwx rwx rwx = 111 111 111
rw- rw- rw- = 110 110 110
rwx --- --- = 111 000 000
```

y donde:

```
rwx = 111 en binario = 7
rw- = 110 en binario = 6
r-x = 101 en binario = 5
r-- = 100 en binario = 4
```

Si se presenta cada uno de los 3 grupos de permisos —propietario, grupo y otros— con un único dígito —entre 0 y 7— obtenemos una forma muy sencilla de expresar cualquier configuración de permisos.

Por ejemplo, si queremos que un archivo `hola.txt` tenga permisos de escritura para el propietario, pero también queremos mantenerlo privado de cara a cualquier otro usuario:

```
yo@mihost:~$ chmod 600 hola.txt
```

A continuación, mostramos una tabla con los números de las configuraciones más comunes. Obviamente, aquellas que empiezan por 7 son usadas con los programas ejecutables —ya que activan el permiso de ejecución— mientras que el resto son para otro tipo de archivos.

Valor	Significado
777	**(rwxrwxrwx)** Sin restricciones. Cualquiera puede hacer cualquier cosa.
755	**(rwxr-xr-x)** El propietario del archivo puede leer, escribir y ejecutarlo. El resto de los usuarios solo pueden leerlo y ejecutarlo. Esta es la configuración más común para programas que puede usar cualquier usuario.

[1] Se puede acceder a un resumen de ambas notaciones en la dirección http://goo.gl/JeFh1F.

700	**(rwx------)** El propietario del archivo puede leer, escribir y ejecutarlo. El resto de los usuarios no pueden hacer nada. Esta configuración es útil para programas que solo el propietario debe usar.
666	**(rw-rw-rw-)** Todos los usuarios pueden leer y escribir.
644	**(rw-r--r--)** El propietario del archivo puede leerlo y escribirlo, mientras que el resto de los usuarios solo pueden leerlo. Se trata de una configuración común para archivos de datos que todo el mundo puede leer, pero solo el propietario la puede cambiar.
600	**(rw-------)** El propietario puede leer y escribir el archivo. Nadie más tiene permisos. Es una configuración común para archivos que el propietario quiere mantener en privado.

Permisos de directorio

El comando chmod también se puede usar para controlar los permisos de acceso a directorios. Nuevamente, usaremos la notación octal, aunque ahora el significado de los permisos "r", "w", "x" es ligeramente diferente:

- **r** — permite listar el contenido del directorio.
- **w** — permite crear, borrar y renombrar archivos dentro del directorio.
- **x** — permite entrar al directorio, por ejemplo, usando el comando **cd**

Estas son las configuraciones más comunes:

Valor	Significado
777	**(rwxrwxrwx)** Sin restricciones. Cualquiera puede listar los archivos, crearlos y borrarlos.
755	**(rwxr-xr-x)** El propietario del directorio tiene acceso completo. El resto de los usuarios solo pueden listar los archivos que contiene. Esta es la configuración más común para directorios que el propietario quiere compartir con otros usuarios.
700	**(rwx------)** El propietario del directorio tiene acceso completo, pero nadie más tiene permisos. Esta configuración es útil para directorios que el propietario quiere mantener en privado.

Superusuario por un instante

Es muy común tener que convertirse en superusuario para realizar ciertas tareas administrativas en el sistema. Sin embargo, siempre **es mejor no permanecer autenticados como superusuarios en el sistema durante largos periodos de tiempo**.

En la mayor parte de las distribuciones **se usa el comando su para obtener privilegios de superusuario de forma temporal**. Simplemente, tendremos que invocar el comando **su** y él nos pedirá la contraseña de superusuario:

```
yo@mihost:~$ su
Password:
root@mihost:/home/me#
```

Si tenemos éxito, a partir de ese momento, dispondremos de una nueva shell como superusuario —lo que por lo general viene indicado por el uso del caracter '**#**' en lugar de '$' en el *prompt*—. Para salir de dicha sesión, basta que ejecutemos el comando **exit**, volviendo así a la sesión previa.

En algunas distribuciones de Linux —fundamentalmente en Ubuntu y distribuciones derivadas— **en lugar de usar "su" se usa sudo**. Con el comando **sudo** uno o más usuarios pueden ejecutar comandos con privilegios de superusuario, sin tener que conocer la contraseña.

En este caso, para ejecutar un comando como superusuario, simplemente tenemos que proceder dicho comando con el comando **sudo**, que nos pedirá nuestra propia contraseña en lugar de la del superusuario:

```
yo@mihost:~$ sudo vi /etc/passwd
Password:
```

Cambiando el propietario de un archivo

Se puede cambiar el propietario de un archivo usando el comando chown. Por ejemplo, supongamos que queremos cambiar el propietario del archivo `hola.txt` a `usuario`:

```
yo@mihost:~$ sudo chown usuario hola.txt
```

Obsérvese que **para cambiar el propietario de un archivo hay que ser superusuario**. Esto se puede conseguir tanto usando el comando **su** como **sudo**.

Cambiando el grupo propietario de un archivo

El grupo propietario de un archivo o directorio puede cambiar mediante el comando chgrp de la siguiente manera:

```
yo@mihost:~$ chgrp nuevo_grupo hola.txt
```

En este caso **no hace falta ser superusuario, pero sí el propietario del archivo en cuestión**.

Tarjetas didácticas

Recuerda, para ayudarte a recordar los comandos de esta unidad y la anterior, su uso y las opciones más comunes, puedes apoyarte en estas tarjetas didácticas. La aplicación está disponible tanto en web como para móvil o tablet.
