Manipulación avanzada de archivos y directorios

El comando find

El comando **find** sirve para buscar recursivamente archivos o directorios. Es una herramienta muy potente, algunas de cuyas opciones se mencionan a continuación.

Opciones

`-name patrón` La el nombre del archivo (sin la ruta) concuerda con el patrón del shell “patrón”.

`-iname patrón` Como -name, pero la concordancia no hace caso de mayúsculas ni minúsculas.

Ejemplos: **`find /home/yo/Documentos -name 'cancion.mp3'`** Busca el elemento llamado "cancion.mp3" en la carpeta /home/yo/Documentos o en alguna de sus subcarpetas.

`find /home/yo/fotos -iname 'paseo*.jpg'` Busca elementos que coincidan con el patrón 'paseo*.jpg' (el * representa cero o más caracteres), sin distinguir mayúsculas de minúsculas.

Los argumentos numéricos pueden especificarse como: **`+n`** para "mayor que n", **`-n`** para "menor que n", y **`n`** para "exactamente n".

`-amin n` Se ha accedido al fichero por ultima vez hace “**n**” minutos.

`-atime n` Se ha accedido al archivo por ultima vez hace (**nx24**) horas.

`-cmin n` El archivo se modificó (en su contenido, propietario, grupo, permisos, etc) por última vez hace “**n”** minutos.

`-ctime n` El archivo se modificó (en su contenido, propietario, grupo, permisos, etc) por última vez hace (**nx24**) horas.

`-mmin n` El contenido del archivo fue modificado por última vez hace “n” minutos.

`-mtime n` El contenido del archivo fue modificado por última vez hace (**nx24**) horas.

Ejemplo: Buscar todos los archivos a los que se haya accedido en el transcurso de la última hora: **`find . -amin -60`**

`-user usuario` El propietario del archivo es el usuario "usuario" (se permite un UID numérico).

`-group grupo` El archivo pertenece al grupo "grupo" (se permite un GID numérico).

`-path patrón` El nombre del archivo (incluyendo la ruta hacia él) concuerda con el patrón del shell "patrón". Los metacaracteres no tratan de forma especial a '/' o a '.'; así, por ejemplo, **`find . -path './sr*sc'`** mostrará una entrada para un directorio llamado './src/misc' (si es que existe).

`-follow` Sigue los enlaces simbólicos (ver más adelante).

`-lname` busca los enlaces simbólicos que “apunten” al archivo (ver más adelante).

`-perm modo` Los bits de permiso del archivo son exactamente "modo" (octal o simbólico).

`-perm -modo` Todos los bits de permiso "modo" estan activos para el fichero, no importa lo que pase con los demás bits.

`-perm /modo` Cualquiera de los bits de permiso de "modo" esta activo para el fichero.

Ejemplos: Encontrar todos los archivos de la carpeta /home/alguien/ (y sus subcarpetas) que sean ejecutables para su dueño (no importa cuáles sean los otros permisos)

`find /home/alguien -perm -u+x`

Encontrar todos los archivos de la carpeta /home/alguien/ (y sus subcarpetas) que tengan permiso de lectura y escritura para el dueño, de lectura solamente para el grupo y ningún permiso para los otros: **`find /home/alguien -perm 640`**

Encontrar todos los archivos de la carpeta /home/alguien/ (y sus subcarpetas) que tengan algún permiso (cualquiera de los tres) para los otros usuarios: **`find /home/alguien -perm /o+rwx`**

-maxdepth N (donde N es un número). Desciende recursivamente hasta “N” niveles por debajo de la ruta especificada (con **-maxdepth 1** busca en la carpeta actual, sin ingresar en ninguna subcarpeta).

-mount no ingresa a directorios que sean puntos de montaje de otros sistemas de archivos.

-regex patrón El nombre del archivo concuerda con la expresión regular "patrón". Esta concordancia es en la ruta entera. Por ejemplo, para un archivo llamado './fubar3', se puede emplear la expresión regular **'.*bar.'** o **'.*b.*3'**, pero no **'b.*r3'**.

-size n El archivo emplea "n" unidades de espacio. Las unidades son bloques de 512 bytes si no se dice otra cosa, bloques de bytes si se pone la letra 'c' después del número "n", o kilobytes con la 'k'.

-type X El archivo es de tipo "X". Se debe reemplazar la "X" por una d si se quiere buscar directorios, por una f para archivos, por una l para enlaces simbólicos, etc.

Ejemplo: Buscar en la carpeta actual (y sus subcarpetas) todos los archivos (no directorios) cuyo nombre empiece con "M" (mayúscula o minúscula), que ocupen más de 1KB: find . -type f -size +1k -iname 'M*'

Acciones

-exec orden \; Ejecuta orden. Todos los argumentos que siguen a find se toman como parte de la "orden" hasta que se encuentre **\;** La cadena **{}** se reemplaza por el nombre del archivo que se está procesando. Ambas construcciones pueden necesitar 'escaparse' (con una '\') o entrecomillarse, para protegerlas de la expansión que efectuaría el shell. La orden se ejecuta en el directorio de comienzo.

-ok orden \; Como **-exec** pero pregunta primero al usuario.

Ejemplo: Buscar y *borrar* el elemento llamado "cancion.mp3" en la carpeta /home/yo/Documentos o en alguna de sus subcarpetas:

```
find /home/yo/Documentos -name 'cancion.mp3' -exec rm {} \;
```

Si deseamos que pregunte antes de borrar, debemos cambiar -exec por -ok:

```
find /home/yo/Documentos -name 'cancion.mp3' -ok rm {} \;
```

-print Imprime el nombre completo del fichero en la salida estándar, seguido por un salto de linea. Es la acción por defecto.

-fprint archivo Escribe el nombre completo de cada archivo encontrado en "archivo". Si "archivo" no existe cuando se ejecuta find, se crea; si existe, se reemplaza.

-printf "formato" Imprime "formato" en la salida estándar, interpretando secuencias de escape '\' y directivas '%'. *No* añade un salto de linea al final de la cadena. Las secuencias de escape y directivas son:

\a La campana de alarma.

\b Espacio atrás.

\f Salto de página.

\n Salto de línea.

\t Tabulador horizontal.

\v Tabulador vertical.

**** Una barra invertida literal: '\'

%% Un signo de porcentaje literal: '%'.

%a El tiempo de último acceso al archivo.

%c El tiempo del último cambio del estado (contenido, permisos, propietario, etc) del fichero.

%f El nombre del fichero sin los directorios (solo el último elemento de la ruta).

%F Tipo del sistema de archivos donde esta el archivo.

%g El nombre de grupo al que pertenece el fichero, o el GID numérico si el grupo no tiene nombre.

%G El GID numérico del grupo al que pertenece el fichero.

%h Los directorios de la ruta del fichero (todos los componentes del camino menos el último elemento).

%k El tamaño del fichero en bloques de un kB (redondeado).

%l El objeto de un enlace simbólico (la cadena vacía si el fichero no es un enlace simbólico).

%m Los bits de permiso del fichero (en octal).

%p El nombre del fichero.

%s El tamaño del fichero en bytes.

%t El tiempo de última modificación del fichero.

%u El nombre del usuario propietario del fichero, o el UID si el usuario no tiene nombre.

%U El UID numérico del propietario del fichero.

Ejemplo: Buscar en la carpeta actual (y sus subcarpetas) todos los elementos cuyo nombre empiece con "M" (mayúscula o minúscula). Imprimir el nombre de archivo, tiempo del último acceso y propietario:

```
find /home/yo/Documentos -iname 'm*' -printf "Archivo: %f - Accedido: %a - Propietario: %u\n"
```

Operadores

Listados en orden de precedencia decreciente:

(expr) Fuerza la precedencia.

-not expr o bien **! expr** Negación de expr.

expr1 expr2 o bien **expr1 -a expr2** o bien **expr1 -and expr2** Operador lógico Y; expr2 no se evalúa si expr1 es falsa.

expr1 -o expr2 o bien **expr1 -or expr2** Operador lógico O; expr2 no se evalúa si expr1 es verdadera.

Ejemplos: Mostrar todos los elementos de la carpeta actual (y sus subcarpetas) que no pertenezcan al usuario "yo": **find . ! -user yo**

Mostrar todos los elementos que pertenezcan al usuario "yo", o bien que tengan todos los permisos para los otros usuarios: **find . -user yo -o -perm -o+rwx**

Redirección de salida para mensajes de error.

Como ya sabemos, los signos **>** y **>>** indican que la salida de un comando se escribirá en un archivo en lugar de mostrarse por la salida estándar. Esta redirección **no** afecta a los mensajes de error, que se siguen mostrando por pantalla.

Para redirigir los mensajes de error, puede seguirse el mismo procedimiento, utilizando los símbolos **2>** y **2>>**.

Puede ocurrir que queramos deshacernos de los mensajes de error, sin mostrarlos por pantalla ni escribirlos en ningún archivo. En ese caso, lo redirigimos a **/dev/null** (equivale a redirigirlo "a ninguna parte").

Ejemplo: supongamos que no existe el elemento **x.txt**, y hacemos **ls x.txt > aaa**

Veremos por pantalla el mensaje de error, y **aaa** estará vacío, pues la redirección ">" no afecta a los mensajes de error. Pero si hacemos **ls x.txt 2> bbb** el archivo bbb contendrá el mensaje de error, que no se mostrará por pantalla. Si no necesitamos guardar en ningún lado el mensaje de error, podemos escribir el comando **ls x.txt > /dev/null**

Este procedimiento es válido para todos los comandos. **En el caso particular del comando find, suele ser útil deshacerse de los mensajes de error que nos señalan que no tenemos permiso para acceder a determinado directorio.** Ejemplo: **find / -iname 'algo.txt' 2> /dev/null**

Enlaces "duros" y enlaces simbólicos

Suele ser útil tener acceso a un archivo desde distintas ubicaciones en un sistema de archivos. Para no tener que hacer muchas copias del archivo, podemos usar enlaces (links). Los enlaces ocupan muy poco espacio, y permiten una gestión mucho más eficiente de la información.

Hay dos tipos de enlaces:

- **Enlaces simbólicos (symbolic links)**: Es simplemente un puntero hacia otro archivo. Cuando Linux abre un enlace simbólico, lee el puntero y luego busca el archivo al que apunta, que es el que realmente contiene los datos. Los enlaces simbólicos pueden apuntar a otros sistemas de archivos, locales o remotos, y pueden apuntar a directorios. Al ejecutar el comando ls -l, aparecen con una letra "l" al principio (así como las carpetas aparecen con "d", y los archivos con "-"). Los permisos de un enlace simbólico son los mismos que tiene el archivo apuntado. Si un enlace simbólico apunta a un archivo que no existe, se dice que está "roto".
- **Enlaces "duros" (hard links)**: Un enlace duro no es, en sentido estricto, un enlace. Es más bien otra entrada en el directorio para el mismo archivo. Las dos entradas (la original y la del enlace) tienen nombres diferentes, pero apuntan al mismo *inode*, es decir, a los mismos datos, con su contenido, propietario, grupo, permiso, etc. Si a un archivo X se le hace un enlace duro Y, los datos se borrarán únicamente cuando se hayan borrado tanto X como Y. Mientras uno de los dos permanezca, los datos no se perderán. Todos los enlaces duros de un archivo deben estar en su mismo sistema de archivos. No se puede crear un enlace duro a un directorio.

El comando ln

Los enlaces se crean con el comando ln. Su sintaxis es:

`ln [opciones] archivo enlace` Crea un enlace de nombre "enlace" que apunta a "archivo". Si se usa la opción -s el enlace es simbólico. Si no, es un enlace duro.

Ejemplo: Crear un enlace duro y un enlace simbólico para el archivo /home/fulano/texto.txt

Enlace simbólico: `ln -s /home/fulano/texto.txt simbolico`

Enlace duro: `ln /home/fulano/texto.txt duro.txt`

```
ls -l
lrwxrwxrwx 1 fulano fulano 27 ago 1 20:09 simbolico -> /home/fulano/texto.txt
-rw-r--r-- 2 fulano fulano  5 ago 1 20:09 duro.txt
```

Importante: Al crear enlaces simbólicos, conviene usar la ruta absoluta (comenzando con /).

El comando **cp** no copia los enlaces simbólicos, sino el archivo al que el enlace apunta. Si queremos que copie el enlace en lugar del archivo completo, hay que usar la opción -d.

Ejemplo (los comandos van precedidos por el signo #, las salidas no):

```
# ls -l dir1
total 13
lrwxrwxrwx 1 root root  19      Jan 4 02:43     file1 -> /file1
-rw-r--r-- 1 root root  10240   Dec 12 17:12    file2
# cp -r dir1 dir2
# ls -l dir2
total 3117
-rw-r--r-- 1 root root  3164160 Jan 4 02:43 file1 #No copió el enlace, sino
                                                   el archivo apuntado.
-rw-r--r-- 1 root root  10240   Jan 4 02:43 file2
# cp -rd dir1 dir3
# ls -l dir3
total 13
lrwxrwxrwx 1 root root  19      Jan 4 02:43 file1 -> /file1 #Copió el enlace
-rw-r--r-- 1 root root  10240   Jan 4 02:43 file2
```

Como ya vimos, el comando **find** no "sigue" los enlaces simbólicos, a menos que usemos la opción **-follow**

Si queremos saber cuántos enlaces simbólicos tiene un archivo determinado, podemos hacerlo con el comando find:

```
find / -lname /home/fulano/archivo
```

La salida es:

```
/home/fulano/unArchivo
/home/mengano/archivo1
/root/miEnlace
```

Cabe recordar que los enlaces simbólicos pueden estar en otros sistemas de archivos que no estén accesibles en este momento (por ejemplo, un pendrive que no está enchufado en la PC). Es obvio que, en estos casos, el comando find no los encontrará.

Al usar find con la opción -lname, también es conveniente utilizar la ruta absoluta.

Para buscar los enlaces duros de un archivo, hay que buscar primero el sistema de archivos en el que está: **df archivo** La salida es:

```
S.ficheros  blocks de 1K    Usados      Disponibles Uso%    Montado en
/dev/sda3   380622108       236181908   125082608   66%     /home
```

Sabemos que el sistema de archivos al que pertenece archivo, está montado en /home

Ahora hay que saber el inodo del archivo: **ls -i archivo** La salida es:

```
1442989 archivo
```

Vemos que el número de inodo es 1442989

Entonces ejecutamos el comando find con la opcion -inum:

find /home -inum 1442989 La salida es la lista de enlaces duros que tiene el archivo:

```
/home/fulano/Documentos/otroArchivo
/home/fulano/unArchivo
/home/mengano/archivo
```