There are no translations available.
Find es un comando muy utilizado por usuarios Linux, pero la mayoría de ellos utiliza una mínima parte de su potencia, descúbrela gracias a este artículo!

## 1.□□□ Introducción

*Find* es un comando que se utiliza para buscar archivos dentro de la jerarquía de directorios del sistema y opcionalmente, que cumplan una serie de reglas de búsqueda.

*Find* no solo busca archivos regulares sino cualquier clase de nodo que tenga una entrada en un directorio, como puede ser por ejemplo: un enlace simbólico, un subdirectorio, un fichero especial de bloques, etc□

Pero además de buscar archivos, *find* tiene la posibilidad de realizar operaciones sobre los ficheros que encuentra.

*Find* tiene una gran cantidad de parámetros y criterios de búsqueda lo que le hace muy flexible y potente, casi indispensable para realizar tareas administrativas.

Por último, *find* es un comando muy utilizado por usuarios Linux, pero la mayoría de ellos utiliza una mínima parte de su potencia, debido a que es difícil recordar todas sus opciones. Aquí, en este artículo veremos como aprovechar el gran potencial de este comando.

## 2.□□□ Findutils

Este paquete contiene programas para la búsqueda de ficheros, unos independientes para hacer búsquedas recursivas y otros, para crear, mantener y consultar una base de datos y realizar búsquedas por medio de ella.

Quizás estos últimos sean más rápidos, pero deberemos tener la base de datos actualizada antes de realizar la acción.

Entre estos programas contenidos en *findutils*, **se encuentra *find*** y otros como:

- **locate**: busca en una base de datos de nombres de ficheros y muestra los archivos que contienen la cadena indicada o cumplen un patrón dado.

*locate* guarda las rutas de los ficheros (además de permisos y características de pertenencia, para preservar la privacidad) en una base de datos llamada ***slocate***.

- **updatedb**: actualiza la base de datos que utiliza ***locale***. Explora en su totalidad el sistema de ficheros y actualiza e inserta en la base de datos todos los nombres de ficheros encontrados además de otros parámetros.

Una forma muy útil y práctica de automatizar la actualización de la base de datos ***slocate*** es programar la ejecución de un shell-script diario (por ejemplo cada noche) con la utilidad cron que ofrece Linux.

- **xargs**: comando para ejecutar determinadas acciones desde la entrada estándar. Normalmente se utiliza en conjunción con el resultado de una búsqueda (generalmente una lista de nombres de archivos) para aplicar una acción determinada posteriormente.

Seguramente todos estos programas estarán instalados en nuestro sistema Linux*.

Si no es así los tendremos que instalar.

- Para sistemas Debian:

Ejecutamos como root

miMaquina:~# apt-get install -y findutils

- Para sistemas RedHat:

Ejecutamos como root

[root@miMaquina~]# rpm □i
ftp://ftp.rediris.es/sites/fedora.redhat.com/core/version/arquitectura/os/Fedora/RPMS/findsutils-versio-arquitectura.rpm

ó

[root@miMaquina ~]# rpm -i findutils-version-arquitectura.rpm

* P ara comprobarlo

- En debian

usuario@miMaquina:~$ dpkg -l | grep -i findutils

- En sistemas RedHat

[usuario@miMaquina ~]$ rpm -qa | grep -i findutils

La potencia de comando *find* no reside en la rapidez que encuentra los ficheros deseados (el comando *locate* solo tiene que hacer una consulta a la base de datos y no realizar una búsqueda recursiva), sino en la gran cantidad de reglas por las que podemos filtrar la búsqueda, como por ejemplo: por su fecha de creación, tamaño del fichero, permisos, y un largo etcétera... además de posteriormente poder efectuar operaciones sobre el resultado.

## 3.□□□ Utilización de find

*Find* siempre devuelve un código de estado de tal forma que si es igual a 0 significa que la búsqueda ha acabado satisfactoriamente y puede continuar con la ejecución de alguna acción sobre el resultado (si es el caso) y, mayor que 0 si ha ocurrido algún error.

Aunque a priori la sintaxis de *find* se ve sencilla, en realidad se pueden utilizar una gran variedad de opciones como se puede constatar en la página de ayuda en línea correspondiente (*man find*).

*Find* siempre empieza la búsqueda en un directorio y luego desciende por sus subdirectorios buscando el patrón indicado (con algunas opciones podemos indicar □el nivel de profundidad□).

Si no se especifica ningún directorio de búsqueda, *find* recoge el valor de la variable PWD, es decir, asume el directorio actual.

| Para saber en que directorio estamos, ejecutamos: |
|---|

usuario@miMaquina :~$

*Find* utiliza las opciones *-name* y *-print* por defecto, es decir, si ejecutamos *find* a secas, listará el contenido del directorio donde estemos situados de forma recursiva.

usuario@miMaquina :~$ find

.

./sqlnet.log

./scripts

./scripts/backup.sql

./scripts/backup2.bat

./scripts/recuperacion.rman

./.gtkrc

./messages.1

./.bash_history

.

## 3.1. Sintaxis

***find [ruta] [expresión]***

- **ruta**: indica el punto de partida desde donde se deberá iniciar la búsqueda.
- **expresión**: indica un conjunto de opciones como la especificación del archivo a buscar, comparaciones, operaciones y acciones sobre el resultado.

**Expresión**

Como se ha comentado el parámetro expresión puede ser una combinación de opciones, comparaciones, operadores y acciones.

Para dotar más claridad a la expresión, es aconsejable poner primero las opciones y después las acciones sobre ellas, sino, nos puede aparecer una advertencia, como esta:

usuario@miMaquina :~$ find . -name '*.log' -maxdepth 2

find: atención: ha especificado la opción -maxdepth después de un argumento -name que no

es una opción, pero las opciones no son de posición (-maxdepth afecta tanto a

las evaluaciones especificadas antes de él como a las especificadas

después). Por favor especifique las opciones antes de otros argumentos.

Hay que reseñar que, la lista de opciones, comparaciones y acciones es muy extensa. Resaltaremos las más importantes. De nuevo, podemos recurrir a la ayuda en línea (*man find*) para ver todas.

## 3.2. Opciones

Lo primero que debe aparecer en la expresión son las opciones, ya que definen como realizar la búsqueda.

**-maxdepth {n}**. *Find* buscará de forma recursiva hasta un máximo de n niveles de subdirectorios por debajo del especificado.

Por ejemplo, con *-maxdepth 1* buscaremos en el directorio actual de forma no recursiva (no "bajaremos" a los subdirectorios). Lo vemos:

usuario@miMaquina :~$ find . -maxdepth 1 -name '*a*'

./.bashrc

./.bash_profile

./.bash_logout

./.Xauthority

./.gnome2_private

./Firefox_wallpaper.png

./cabeceraSamba.conf

.....

**-mindepth {n}**. Similar a maxdepth pero comenzará a buscar a partir de n niveles.

Por ejemplo, con *-mindepth 2* buscaremos sin tener en cuenta el directorio actual. Realizará la búsqueda a partir de los subdirectorios y de forma recursiva. Lo vemos:

usuario@miMaquina :~$ find . -mindepth 2 -name '*a*'

./.gconf/desktop/gnome/applications

./.gconf/desktop/gnome/applications/window_manager

./.gconf/desktop/gnome/accessibility

./.gconf/desktop/gnome/accessibility/keyboard

./.gconf/desktop/gnome/peripherals

./google-earth/resources/loading.png

./google-earth/resources/overviewframe.png

./google-earth/resources/palette-2.png

.....

**-mount**. No descender a directorios montados en otros sistemas de ficheros. Muy útil si tenemos mapeadas algunas unidades de red.

## 3.3.□□□□□□□□Comparaciones

Algunas comparaciones suelen llevar argumentos numéricos. Es importante que entendamos su significado:

Argumentos numéricos:

- +n: para un valor mayor que n
- n: exactamente ese valor n
- -n: para un valor menor que n

**Patrón**

Muchas veces realizaremos búsquedas que cumplan un patrón (expresión regular que incluye un conjunto de cadenas de caracteres), es decir, no buscar un determinado fichero sino un grupo seleccionado de ellos que tengan en común ciertos caracteres de su nombre o su extensión por ejemplo.

La sintaxis correcta del patrón es la siguiente:

- *: engloba cualquier o ningún carácter.
- ?: cualquier carácter sencillo, justamente uno.
- [cadena]: coincidencia exactamente en uno de los caracteres contenidos en la cadena.

También la cadena puede contener rangos como por ejemplo incluir todas las letras minúsculas y los números [a-z0-9].

- [^cadena]: no coincidencia con ninguno de los caracteres de la cadena. Es la negación de [cadena].
- : cualquier meta carácter debe ir precedido del símbolo backslash '' para que pierda su significado especial. Se consideran metacaractares *, ? y [].

Es aconsejable que el patrón vaya ente comillas simples.

Ejemplos:

· '?valen*' : podrían ser 1valentino y vvalen.

· '*.jpg': cualquier archivo con extensión jpg.

· '*1*.txt': cualquier archivo .txt que su nombre contuviera un 1.

· '*[1]*.txt: igual que el ejemplo anterior.

· '[0-9]?rt*.??q': coincidirían por ejemplo los archivos 03rt.44q y 1trtexample.exq

· 'ejercicio[0-9]*: coincidirían ejercicio1.txt, ejercicio1.c, etc...

· '[^0-9A-Z]example.php': coincidirían aexample.php. bexample.php. En definitiva cualquier fichero example.php cuyo primer carácter sea un letra minúscula.

· '*[a-z]*.odt: coincidirían *zprimero.odt, *ssss.odt; es decir, el primer carácter deberá ser un asterisco.

**Lista de comparaciones**

**-amin {n}**: búsqueda de ficheros que han sido leídos hace n minutos.

Por ejemplo:

usuario@miMaquina :~$ fi

./.bash_history

./.bash_profile

./.bashrc

Veríamos todos los ficheros que se han leído en los últimos 9 minutos partiendo desde el directorio actual.

| | |
|---|---|
| Nota: como vemos find aplica por omisión las acciones - | *name* |

**-atime {n}**: búsqueda de ficheros que han sido leídos por última vez hace nx24 horas.

Por ejemplo:

usuario@miMaquina :~$ find . -atime -1

./scripts

./.bash_history

./.kde

./.kde/Autostart

./.bash_profile

./.bashrc

./.bash_logout

Veríamos todos los ficheros que se han leído en las últimas 24 horas partiendo desde el directorio actual.

**-cmin {n}**: búsqueda de ficheros cuyos permisos se han cambiado hace n minutos.

Por ejemplo:

usuario@miMaquina :~$ find . -cmin -10

./.bash_history

Veríamos todos los ficheros cuyos permisos han sido cambiados en los últimos 9 minutos partiendo desde el directorio actual.

**-ctime {n}**: búsqueda de ficheros cuyos permisos se han cambiado por última vez hace nx24 horas.

Por ejemplo:

usuario@miMaquina :~$ fi

./.bash_history

Veríamos todos los ficheros cuyos permisos se han cambiado en las últimas 24 horas partiendo desde el directorio actual.

**-mmin {n}**: búsqueda de ficheros que han sido modificados (su contenido) hace n minutos.

Por ejemplo:

usuario@miMaquina :~$ fi

./sqlnet.log

./scripts

./scripts/backup.sql

./scripts/backup2.bat

Veríamos todos los ficheros que han sido modificados hace exactamente 4 minutos partiendo desde el directorio actual.

**-mtime n**; busqueda de ficheros que han sido modificados por última vez hace nx24h.

Por ejemplo:

usuario@miMaquina :~$ find . mtime +1

./.gtkrc

./messages.1

./.bash_profile

./.Xauthority

./.zshrc

./mbox

Veríamos todos los ficheros que han sido modificados a partir de 24h hacia adelante en el tiempo, partiendo desde el directorio actual.

**-empty**: búsqueda de ficheros o directorios vacíos. Muy útil para "hacer limpieza" del disco duro.

**-fstype {tipo}**: especificamos el tipo de sistema de archivos donde queremos realizar la busqueda (en /proc/filesystem podemos ver la lista de todos los sistemas de ficheros registrados en el sistema).

Por ejemplo:

usuario@miMaquina :~$ find . fstype ntfs

Buscaríamos todos los ficheros del sistema de archivos ntfs ignorando todos los demás.

**-gid {n}**: buscamos ficheros que pertenezcan al grupo con identificador igual a n.

Nota: con el comando id podemos saber cual es nuestro gid, uid así como a los grupos a los que tambie

**-group {nombre}**: igual que la expresión anterior, pero esta vez buscamos por el nombre del grupo.

**-lname {patrón}**: buscamos enlaces simbólicos que coincidan con el patrón.

Con **ilname** ignoramos las mayúsculas o minúsculas.

**-name {patrón}**: buscamos los ficheros que coincidan con el patrón.

Con **iname** ignoramos las mayúsculas o minúsculas.

**-links {n}**: buscamos ficheros con n enlaces.

**-nouser**: buscamos ficheros que no pertenezcan a ningún usuario. Es decir, el uid del fichero no coincide con el UID de ningún usuario.

**-nogroup**: igual que la expresión anterior pero con gid (identificador de grupo).

**-path {patrón}**: búsqueda del patrón que coincida con el path o ruta.

Por ejemplo:

usuario@miMaquina :~$ find . -path '*reg*'

./.gstreamer-0.10/registry.i486.xml

./.mozilla/pluginreg.dat

./.mozilla/firefox/zmeccrmm.default/compreg.dat

./.mozilla/firefox/pluginreg.dat

./.gdesklets/registry

./.gdesklets/registry/version

....

Buscaría y mostraría cualquier path que contuviera *reg*, como por ejemplo: */gnome2/Desktop/log/registro1/1.jpg*

Con **ipath** ignoramos las mayúsculas y minúsculas.

**-perm {modo}**: buscamos los ficheros que coincidan exactamente con los permisos en representación octal (de 0 a 7) o en representación simbólica (r para lectura, x para ejecución, w para escritura).

Ejemplo:

usuario@miMaquina :~$ fi

./prueba

./mbox

Para representación simbólica nos referiremos al propietario del fichero con **u**, al grupo con **g** y al resto con
**o**
.

Por ejemplo, el resultado de la instrucción anterior es idéntica a esta:

usuario@miMaquina :~$ fi

./prueba

./mbox

**-perm {-modo}**: buscaremos los ficheros que coincidan exactamente con los permisos o que al menos tengan estos permisos el propietario, el grupo y el resto en representación octal o en representación simbólica. Por ejemplo:

-666: lo cumplirían archivos que tuvieran de permisos rw-rw-rw-.

Un fichero con permisos rwx-rw-rw- si lo cumpliría.

Un fichero con permisos rwx-rw-r-x no lo cumpliría.

usuario@miMaquina :~$ fi

-rwxr-xr-x 1 alex alex 10090 2007-03-19 10:03 ./smb.conf.ORIGINAL

Es decir, este fichero tiene permisos 755 como vemos.

El mismo resultado utilizando permisos en representación simbólica:

usuario@miMaquina :~$ find . -perm -u=rwx,go=r

-rwxr-xr-x 1 alex alex 10090 2007-03-19 10:03 ./smb.conf.ORIGINAL

**-perm {+modo}**: buscaremos ficheros que coincidan exactamente con los permisos o que al menos tengan estos permisos el propietario ó el grupo ó el resto en representación octal o en representación simbólica. Por ejemplo:

+111: lo cumplirían archivos que tuvieran --x--x--x al menos. Si tienen más permisos, también cumplirían la condición.

Un fichero con permisos --x------ saldría en el resultado.

Un ficheros con permisos rw-r--r-- no saldría en el resultado.

usuario@miMaquina :~$ fi

-rw-r--r-- 1 root root 0 2007-07-16 13:56 httpd.conf

Vemos que el fichero tiene permisos 644.

Por ejemplo, sería lo mismo esta sentencia:

usuario@miMaquina :~$ fi

Con esta

usuario@miMaquina :~$ find . -perm +gou=w,u=rw

**-size {n} [ckMG]**: buscamos los ficheros que tengan como tamaño de bloque el valor n, es decir, el número bloques que utiliza el fichero en disco.

Un bloque por defecto son 512 bytes, pero desafortunadamente no tiene porqué serlo. Es recomendable buscar por unidades de almacenamiento que son las siguientes:

- c= bytes.
- k= kilobytes.
- M= Megabytes.
- G= GigaBytes.

Ejemplos:

usuario@miMaquina :~$ find . -size 40c

Buscará ficheros cuyos tamaños sean exactamente de 40 bytes.

usuario@miMaquina :~$ find . -size -24M

Buscará ficheros cuyos tamaños sean menos de 24 Megabytes.

usuario@miMaquina :~$ find . -size +46k

Buscará ficheros cuyos tamaños sean mayores de 46 kilobytes.

**-type {tipo}**: especificamos el tipo de fichero a buscar

- -d: directorio
- -f: fichero regular
- -l: enlace simbólicos
- -b: fichero especial de bloques*

· -c: fichero especial de caracteres*

*Los ficheros de dispositivos hacen referencia a dispositivos físicos o virtuales del sistema, de modo qu

| | |
|---|---|
| **De carácter** | : se accede a ellos a ra |

| | |
|---|---|
| **De bloque** | : se accede a ellos a ra |

-**uid {n}**: busca ficheros cuyo propietario tenga el uid especificado.

**-user {usuario}**: busca fichero cuyo propietario del mismo es el usuario.

## 3.4. Acciones

**-exec {orden}**: ejecutar una orden sobre el resultado de la búsqueda.

Utilizamos esta acción seguida de un espacio y el comando a ejecutar sobre el valor devuelto por *find* y después {} ;

La cadena **{}** se reemplaza por los fichero(s) encontrado(s).

Sintaxis:

**find ..... -exec accion_a_realizar {} ;**

Se verán ejemplos detallados en la sección Ejemplos.

**-ls**: la salida de *find* es un listado idéntico al formato utilizado por este comando.

**-print**: muestra el nombre completo de los ficheros encontrados por la salida estándar, seguido de un salto de línea.

*Find* lleva implícita esta acción por defecto.

Si la expresión no contiene ninguna acción (excepto **-prune**), se aplica por omisión la acción **-print**
siempre que la expresión devuelve verdadero (ó 0).

**-print0**: igual que *print0* pero sustituye el salto de línea por un carácter nulo.

**-fprint {fichero}**: escribe el resultado de la búsqueda en el fichero indicado, incluyendo un salto de línea por cada fichero encontrado.

**-fprint0 {fichero}**: igual que *fprint* pero sustituye salto de línea por carácter nulo.

**-fls {fichero}**: escribe en el fichero seleccionado el resultado de la busqueda con formato de salida idéntico al comando *ls*.

**-printf {formato}**: imprime el resultado en la salida estándar con el formato indicado.

El formato utiliza secuencias de escape y directivas. Es aconsejable utilizar la ayuda en línea (

*man find*
) para ver la gran cantidad de opciones.

Se ven algunos ejemplos en la sección Ejemplos.

**-fprintf {fichero} {formato}**: igual que *printf* pero escribe el resultado de la búsqueda en el fichero indicado.

### 3.5. Operadores

Las búsquedas pueden ser a veces complejas, con expresiones extensas. Podemos utilizar operadores para combinar expresiones.

- ( expr ): podemos utilizar paréntesis para agrupar las condiciones entre expresiones.
- ! expr: negación de la expresión.
- -not expr: idéntico operador que !.
- expr1 expr2: si omitimos el operador entre la expresion1 y expresion2, actúa como un AND lógico.
- expr1 -a expr1: lo mismo que la condición anterior (-a AND lógico).
- expr1 -o expr2: operador OR lógico entre las expresiones.

## 4. Acciones sobre el resultado

Como se ha mencionado antes, -**exec** se aplica para ejecutar una determinada acción sobre el resultado de la busqueda devuelto por *find*.

Pues bien, existe otra instrucción cuya finalidad es la misma pero con una sintaxis y funcionamiento diferentes, **xargs.**

Sintaxis xargs:

**find ..... | xarg [opciones] orden**

Opciones:

- -0 ó  null, se utiliza en el caso de que el nombre de los ficheros devueltos terminen con un carácter nulo, en lugar de con un espacio en blanco.

Esta opción deberá usarse cuando utilicemos la acción *print0* con *find*.

- -t ó  verbose: muestra por pantalla la acción a ejecutar antes de proceder con la orden.
- -p: pregunta al usuario si se debe ejecutar la acción.
- -i: recoge el valor devuelto por find y lo utilizamos mediante los corchetes "{}" (similar a la sintaxis de *exec*). Muy útil para comandos del tipo mv por ejemplo. Lo vemos:

usuario@miMaquina :~$ find . -name '1.log' | xargs -I -t mv {} 1.log.old

mv ./1.log 1.log.old

La cadena {} se reemplaza por el resultado devuelto por *find*, en este caso 1.log..

- --help: muestra un resumen de las opciones de *xargs*.

Para más opciones mirar la ayuda en línea (*man xargs*).

Entonces, ¿cual elegir?...

## 4.1. Diferencias entre xarg y exec

Fundamentalmente las diferencias son las siguientes:

- La sintaxis propia de cada uno.
- El tiempo de ejecución de la orden.

Respecto a la primera diferencia, pues es acostumbrarse a una sintaxis u otra, y respecto a la segunda, esta es definitiva porque es mucho más rápido el parámetro *xarg*.

Para comprobarlo bastaría con ejecutar:

```
usuario@miMaquina  :~$ time find -name *.jpg -print0 | xargs -0 ls

./google-earth/resources/google_earth_splash.jpg

./.opera/cache4/opr0003C.jpg

./.opera/cache4/opr0003M.jpg

.....

real 0m0.096s

user 0m0.020s

sys 0m0.060s
```

usuario@miMaquina :~$ time find -name *.jpg -exec ls {} ;

./google-earth/resources/google_earth_splash.jpg

./.opera/cache4/opr0003C.jpg

./.opera/cache4/opr0003M.jpg

.....

real 0m0.633s

user 0m0.012s

sys 0m0.428s

Resulta evidente la diferencia de tiempo.

## 5.□□□ Ejemplos comunes

Los criterios de búsqueda más comunes podrían ser:

- Buscar la ubicación exacta de un archivo por su nombre:

usuario@miMaquina :~$

- Buscar la ubicación(es) de un(os) archivo(s) que contiene(n) cierta cadena de caracteres o algún patrón:

usuario@miMaquina :~$

- Buscar enlaces simbólicos a ficheros:

usuario@miMaquina :~$

- Buscar archivos que fueron leídos por última vez en un cierto periodo de tiempo, por ejemplo en las últimas 24 horas:

usuario@miMaquina :~$

- Buscar ficheros que tienen un tamaño comprendido dentro de un intervalo, por ejemplo en MegaBytes:

usuario@miMaquina :~$

- Buscar archivos de un cierto tipo, como por ejemplo: directorios, enlaces simbólicos, etc :

usuario@miMaquina :~$

- Encontrar ficheros que pertenecen a un usuarios o grupo específico:

usuario@miMaquina :~$

usuario@miMaquina :~$

- Buscar ficheros que tienen determinados permisos de acceso:

usuario@miMaquina :~$

usuario@miMaquina :~$

Después de realizar estas búsquedas, podemos ejecutar estas acciones típicas como:

- Ver: -ls, -print, -print0, etc...
- Editar

usuario@miMaquina :~$

usuario@miMaquina :~$ find

| Nota: la última sentencia puede dar problemas ya que el acceso (o la entrada) al editor vi para modifica |
|---|

| Es recomendable en este caso, en ediciones de ficheros, utilizar la acción | *exec* |
|---|---|

- Guardar sus nombres en otro archivo: -fprint {fichero}, fprintf {fichero} {formato}, etc..

- Eliminar:

usuario@miMaquina :~$

usuario@miMaquina :~$ find

- Renombrar:

usuario@miMaquina :~$

usuario@miMaquina :~$

- Cambiar los permisos:

usuario@miMaquina :~$

usuario@miMaquina :~$

## 6.□□□ Ejemplos variados

La mejor manera de aprender a utilizar este comando es ver, comprender y practicar con todo tipo de ejemplos. Cuanto mas variados mejor.

Los vemos:

- Buscar en el directorio /var los archivos terminados en .log e imprimir sus nombres en la salida:

usuario@miMaquina :~$ find /var -name *.log -print

./log/dpkg.log

./log/auth.log

.....

- Busca archivos mayores de 200 kilobytes, ubicados en el directorio actual:

usuario@miMaquina :~$ find . -size +200k -print

/home/alex/google-earth/libauth.so

/home/alex/google-earth/libbase.so

/home/alex/google-earth/libbasicIngest.so

.....

Como se ha descrito antes, los argumentos numéricos son:

-
- +N es mayor que N.
- -N es menor que N.
- N es exactamente igual a N.

- Busca archivos no accedidos hace más de 30 días en todo el sistema:

usuario@miMaquina :~$

/usr/share/doc/iptables/changelog.gz

/usr/share/doc/iptables/changelog.Debian.gz

.....

Como recordatorio, la opción -atime se refiere al tiempo transcurrido desde la última lectura, -mtime el tiempo transcurrido desde última modificación de los permisos y, -ctime al tiempo transcurrido desde la última modificación del contenido.

- Busca archivos vacíos y borrarlos, ubicados en nuestro "home" por ejemplo:

usuario@miMaquina :~$ find /home/usuario -empty -exec rm {} ;

y con *xargs*:

usuario@miMaquina :~$ find /home/usuario -empty | xargs -t rm

rm /home/usuario/prueba

- Busca archivos sin propietario. Esta situación se puede dar por ejemplo cuando la cuenta de usuario ha sido borrada pero han permanecido los archivos creados por este usuario eliminado del sistema:

usuario@miMaquina :~$ find / -nouser -ls

193485 68 -rw-r--r-- 1 1003 1003 61952 abr 10 2001 /usr/lib/win32/acelpdec.ax

193486 40 -rw-r--r-- 1 1003 1003 38912 ene 7 2002 /usr /lib/win32/alf2cd.acm

193487 120 -rw-r--r-- 1 1003 1003 118784 mar 24 2004 /usr /lib/win32/aslcodec_dshow.dll

....

**Ignorar mensajes de error.**

Por ejemplo, buscamos los ficheros de nombre passwd en /etc y los imprimimos por pantalla:

usuario@miMaquina :~$ find /etc -name "passwd" -print

/etc/pam.d/passwd

/etc/passwd

find: /etc/ppp/peers: Permiso denegado

find: /etc/ssl/private: Permiso denegado

find: /etc/cups/ssl: Permiso denegado

find: /etc/chatscripts: Permiso denegado

Como vemos, a veces aparecen mensajes de error de acceso en el listado resultante. Esto se debe a que el usuario con el que ejecutamos el comando *find* no tiene permisos de acceso, lectura sobre ficheros y ejecución sobre directorios.

Para evitar estos mensajes, redireccionamos los errores (2>) a la "papelera de Linux" (/dev/null), en vez de verlos por la salida estándar (que es la pantalla, la opción por defecto). Lo vemos:

```
usuario@miMaquina  :~$ find /etc -name passwd -print 2>/dev/null

/etc/pam.d/passwd

/etc/passwd
```

Otra solución es ejecutar *find* como root, sólo aconsejable para tareas de administración.

## Más ejemplos

- Mostramos los ficheros cuyo nombres finalizan con los caracteres 'wd' y visualizamos su contenido:

```
usuario@miMaquina  :~$ find /etc -name "*wd" -exec cat {} ;
```

ó

usuario@miMaquina :~$ find /etc -name "*wd" | xargs -t cat

- Buscamos todos los ficheros pertenecientes al usuario root cuyo permiso especial SUID esta activo:

usuario@miMaquina :~$ find / -user root -perm -4000 -print

Permisos especiales:

- SUID -> 4000. Ficheros que al ejecutarlos un usuario diferente a root, adquieren los permisos de r

/usr/bin/passwd rwsrxrx 1 root root 28480 2006 09 17 10:05 /usr/bin/passwd

Aplicado al propietario.

Cuando termina la ejecución, se retorna a la situación previa.

- SGID -> 2000. Lo mismo que el SUID pero aplicado al grupo.
- Sticky bit (bit adhesivo) -> 1000. Es un bit que se aplica a los directorios. Si está activo, hace que un usuario sólo pueda borrar los ficheros que son de su propiedad en dicho directorio. Es común en el directorio /tmp.

- Buscar ficheros con permisos exactamente 777, cuyo propietario sea alex y restringimos la busqueda al sistemas de ficheros de Linux (omite el resto):

usuario@miMaquina :~$ find / -xdev -user alex -perm 777 -print

- Buscamos archivos ocultos en el sistema. Sólo buscamos en el sistema de ficheros propio de Linux (-xdev):

usuario@miMaquina :~$ find / -xdev -name ".*" -print

- Buscar los ficheros (sólo ficheros regulares por -type f) cuya extensión sea .jpg, que el propietario se alex y todos aquellos cuyos nombres no empiecen por cualquier letra, ya sea mayúsculas o minúsculas:

usuario@miMaquina :~$ find . -user alex -type f -name '*.jpg' ! -name '[a-zA-Z]*'

- Buscamos los ficheros del usuario root que han sido modificados en las últimas 24 horas ubicados a partir del directorio /etc:

usuario@miMaquina :~$ find /etc -user root -mtime -1

- Buscamos los ficheros del usuario root que han sido leídos en los últimos 2 minutos ubicados a partir del directorio /etc:

usuario@miMaquina :~$ find /etc -user root -amin -2

- Encontrar todos los ficheros que son escribibles por todos los usuarios:

usuario@miMaquina :~$ find / -perm -o=r

- Buscar todos los ficheros sin propietario y borrarlos. También los imprime por la salida estándar:

usuario@miMaquina :~$ find / -nouser -print0 | xarg -0 rm

- Buscamos todas las imágenes con extensión .gif, .jpg ó .png ubicadas en el escritorio y las movemos al directorio ~/imagenes:

usuario@miMaquina :~$ find ~/Desktop -name "*.jpg" -o -name "*.gif" -o -name "*.png" -print0 | xargs -0

*--target-directory: el valor de este parámetro es la ruta donde se encuentra la imagen encontrada.

Esta sentencia haría lo mismo:

usuario@miMaquina :~$ find ~/Desktop -name "*.jpg" -o -name "*.gif" -o -name "*.png" -print0 | xargs -0

- Restablecemos los permisos para los ficheros y directorios en nuestro directorio para aplicaciones Web:

usuario@miMaquina :~$ find /midirWeb/ -type d -print0 | xargs -0 chmod 755

usuario@miMaquina :~$ find /midirWeb -type f | xargs chmod 644

- Buscamos todos los ficheros regulares que han sido modificados en esta última semana y los empaquetamos con la utilidad tar:

usuario@miMaquina :~$ find / -type f -mtime -7 | xargs tar -rf fichero_semanal.tar

- Buscamos todos los ficheros a partir del directorio actual que no estén ocultos, sin 'bajar' por los subdirectorios (-maxdepth 1) y, los imprimimos en el siguiente formato:

usuario@miMaquina :~$ find . -maxdepth 1 -name '[!.]*' -type -f -printf 'Nombre: %10f Tamaño: %6s
'

Nombre: maria Tamaño: 31

Nombre: index.php Tamaño: 973

Nombre: @er Tamaño: 0

Nombre: httpd.conf Tamaño: 0

.....

El formato de la acción *-printf* en este caso es:

-
- Nombre:

- 10 espacios en blanco
- Nombre del fichero
- Tamaño:
- 6 espacios en blanco
- Tamaño del fichero en bytes
- Salto de línea

* Para ver todos los formatos de *-printf* mirar ayuda en línea.

- Buscar todos los ficheros cuyo tamaño sea mayor de 2 Megabytes y se hayan leído en el último mes:

usuario@miMaquina :~$ find / -size +2M -atime -30 -print

- Buscamos la version mas reciente del archivo httpd.conf. El fichero actual está ubicado en /etc/apache2/httpd.conf:

usuario@miMaquina :~$ find / -name httpd.conf -newer /etc/apache2/httpd.conf

## 7. Referencias

- **http://es.wikipedia.org**

- **Ayuda en línea de *find* (*man find*)**

- **Ayuda en línea de *xargs* (*man xargs*)**

- **Ayuda en línea de *exec* (*man exec*)**

- **http://www.gnu.org/**

- **http://bulma.net/**

- **http://www.linuxfocus.org/**

- **http://es.tldp.org/**

- **http://dmiessler.com/**

- **http://www.grymoire.com** /

- **http://www.grafikas.es/linux/**