

<b>Miembros del grupo de estudios .....</b>	<b>1</b>
<b>Materiales de estudio .....</b>	<b>1</b>
<b>Algunos criterios.....</b>	<b>2</b>
6.- Caracteres especiales y entrecomillado .....	3
Tabla 1.2 .....	3
El comando find .....	4
Desafío N° 1.....	6
Entrecomillado inverso .....	14

---

## Miembros del grupo de estudios

María, Fabricio, Diego --> **Argentina**

Jorge, Asterix --> **Chile**

Alejandro --> **Brasil**

---



## Materiales de estudio

*(Si tienen alguno más para recomendar, lo agregan)*

[Apuntes\\_bash\\_0.pdf / Apuntes\\_bash\\_1.pdf / bash.pdf Comandos.pdf](#)

[\(GRUPO BASH\) PRIMERA SEMANA "COMODINES"](#)

**\*\* (GRUPO BASH) SEGUNDA SEMANA "REDIRECCIONES Y PIPES" \*\*** **NUEVO!**

[Ejemplos de scripts - Ubuntu Forum](#)

[Un comando cada día - Ubuntu Forum](#)

[Bash scripting de supervivencia](#)

[Taller de programación shell](#)

[Shell Scripting](#)

[Google Docs](#)

[Redireccionamiento de Entrada-Salida](#)

**\*\*\* Redireccionamiento y tuberías \*\*\*** **NUEVO!**

---

## Algunos criterios

- Podemos reutilizar los iconos para señalar algunas cuestiones (copiar y pegar).

	Referencia al material de estudio
	Desafío
	Script
	Atención!
	Mi Reino por un caballo!!!
	Una duda...
	Premio "Rex Tux Scripting"
	Consejo
	Cuidado!!!
	Comando

- Delimitar las intervenciones con barras horizontales.
- No usar comentarios (en la publicación web no aparecen)
- Firmar con el nombre resaltado con color.

María, Fabricio, Diego, Jorge, Rodrigo, Alejandro, Asterix

- Fuente `Courier New` para los scripts y comandos. Aplicar sangrado de párrafo (ahora está en Formato-->Estilo de Párrafo).
- Si no visualizan correctamente las fuentes verdana y courier, instalen los paquetes:

```
sudo aptitude install msttcorefonts ttf-mscorefonts-installer
```

- Poner onda con el formato para facilitar la legibilidad del documento y el trabajo de publicación posterior.
- 



## 6.- Caracteres especiales y entrecomillado

---

Tabla 1.2

<b>Carácter</b>	<b>Descripción</b>
~	Directorio home
`	Sustitución de comando
#	Comentario
\$	Variable
&	Proceso en background
;	Separador de comandos
*	Comodín 0 a n caracteres
?	Comodín de un sólo carácter
/	Separador de directorios
(	Empezar un subshell
)	Terminar un subshell
\	Carácter de escape
<	Redirigir la entrada
>	Redirigir la salida

	Pipe
[	Empieza conjunto de caracteres comodín
]	Acaba conjunto de caracteres comodín
{	Empieza un bloque de comando
}	Acaba un bloque de comando
'	Entrecomillado fuerte
"	Entrecomillado débil
!	No lógico de código de terminación

Como siempre, lo más difícil es comenzar la semana...

A los **caracteres especiales** ya los hemos estado utilizando bastante, por lo tanto, creo que la idea ahora no sería trabajar con ellos específicamente, sino trabajar cómo se comportan en relación al **entrecomillado (fuerte y débil)** aplicado a los parámetros de los comandos (echo, cut, sort, find, ls, etc.)

Comencemos con la lectura, a ver qué se nos ocurre...



## Suerte!

Alejandro

---



## El comando find

A poco de comenzar la lectura aparece el comando "find". Basta ejecutar en consola para constatar que no es un comando sencillo:

```
$ man find
```

Les dejo una recopilación de aspectos "comprensibles" (para mí) que fui encontrando o que se me van ocurriendo, sobre este comando.

La sintaxis:

```
find [ruta...] [expresión]
```

Algunas opciones:

```
-name          busca según el nombre indicado en la expresión  
-perm         busca según los permisos del fichero  
-exec         permite ejecutar comandos sobre el resultado de  
la búsqueda
```

## Grupo de estudios: Programación Bash Script

Tercera semana: del 22 al 28 de octubre de 2009

---

```
-user          busca los ficheros que pertenecen al usuario
especificado
-group        lo mismo pero que pertenecen al grupo
especificado
-size         busca por tamaño
-type         busca ficheros de un determinado tipo: f
              (archivo regular), d (directorio), l (vínculo simbólico)
-print       imprime la búsqueda en la salida estándar
-empty       busca ficheros y directorios vacíos
-a           AND lógico
-o           OR lógico
```

Ej. 1 - busca todos los ficheros de backup generados automáticamente (son generados a partir del nombre y el caracter tilde al final) Al no ser ejecutado con privilegios administrativos (sudo), hay directorios que no serán listados y darán error, por eso se redirige la salida estándar de errores a /dev/null que es como los "agujeros negros" de Carl Sagan.

```
$ find / -name '*~' 2> /dev/null
```

Ej. 2 - busca en mi home todos los archivos que tienen permisos rwx (lectura, escritura, ejecución) para ugo (usuario, grupo y otros)

```
$ find /home/alejandro -perm 777
```

Ej. 3 - busca todos los archivos con extensión ".tmp" y los borra (las llaves indican que el comando es ejecutado sobre el resultado de la búsqueda; y la línea debe finalizarse con "\;")

```
$ find . -name '*.tmp' -exec rm {} \;
```

busca todos los ficheros terminados en ".conf" y muestra su contenido:

```
$ find /etc -name "*.conf" -exec cat {} \;
```

Ej. 4 - Selecciona archivos cuyo tamaño es exactamente el número especificado. Se puede agregar una de las letras [bckw] para indicar que el número representa bloques, bytes, kilobytes, o palabras de 2 bytes, respectivamente.

```
$ find /bin -size 20c -print 2> /dev/null
```

Selecciona archivos cuyo tamaño es menor que el número especificado (usando -).

```
$ find /bin -size -20c -print 2> /dev/null
```

Selecciona archivos cuyo tamaño es mayor que el número especificado (usando +).

```
$ find /bin -size +20c -print 2> /dev/null
```

Ej. 5 - Buscar una cosa o la otra (-o); una cosa con tal y cual criterio (-a):

```
$ find ~ -empty -a -name '*.txt' -exec rm {} \;
```

```
$ find ~ -user alejandro -o -user root
```



## Desafío N° 1

1. Qué archivos hay en el directorio /usr que pertenezcan a miembros del grupo del usuario?
2. Cuantos archivos se encontraron en la pregunta anterior? Utiliza redireccionamiento para mostrar este número en la pantalla.
3. Cuales archivos en tu cuenta son de tamaño igual a 300 bytes?
4. Cuales archivos en tu cuenta son de tamaño menor a 300 bytes?
5. Cuales archivos en tu cuenta son de tamaño mayor a 300 bytes?
6. Consulta el manual de find para determinar como mostrar los nombres de los archivos que tienes en tu cuenta que se modificaron en los últimos 30 días.
7. Consulta el manual de find para determinar como mostrar los nombres de los archivos que tienes en tu cuenta que se modificaron hace mas de 15 días.
8. Consulta el manual de find para determinar como mostrar los nombres de los archivos que tienes en tu cuenta que se modificaron hace de 5 días
9. Muestra el contenido de los archivos que tengas en tu cuenta cuyo tamaño sea mayor a 30 bytes.
10. Muestra el nombre de todos los archivos que tienes en tu cuenta cuyo nombre termine con el símbolo ~. Si no tienes ninguno, crea 4 y vuelve a hacer el ejercicio. (Nota: Los archivos de respaldo que crea vi tienen esta terminación.)
11. Muestra el contenido de los archivos que encontraste en el paso anterior mostrando una pantalla de información a la vez.
12. Crea un directorio que se llame para\_borrar y copia los archivos que se mostraron en el paso anterior.
13. Utiliza find para borrar los archivos del directorio que se creo en el paso anterior.
14. Explica para que sirven los parametros -amin y -used .
15. Da un ejemplo del uso de los parametros del ejercicio anterior.

Estos ejercicios fueron extraídos de [http://yaqui.mxl.uabc.mx/~taller\\_unix/sem03-1/pract-20.html](http://yaqui.mxl.uabc.mx/~taller_unix/sem03-1/pract-20.html)



En distintos lugares, inclusive en nuestro libro, aconsejan usar el entrecomillado simple (fuerte) para el nombre. Lo cual es bastante desconcertante, porque se supone que ese entrecomillado haría que los caracteres especiales sean interpretados como caracteres comunes y perderían sus características de comodines.

**¿Alguien consigue explicar por qué sucede eso?**

Alejandro

---

**Alejandro**, ¿cuándo pones en el terminal `$ man find`, el resultado te aparece en inglés o en castellano? Si es en castellano podrías indicar cómo lo haces porque a mí el `man` me aparece en inglés y como es inglés técnico se me vuelve incomprensible, claro que recurro a san Google y solucionado, pero sería mejor tenerlo en castellano. Lo del entrecomillado para el nombre no lo pude encontrar, sólo lo del ejemplo y que se entienda el por qué (pág. 19-20) ¿Podrías indicar alguna página?

Jorval

---



Tengo algunas inquietudes

```
$find . -name "*.tmp" 2>/dev/null
$find / -name "*.tmp" 2>/dev/null
$find . -name '*.tmp' 2>/dev/null
$find / -name '*.tmp' 2>/dev/null
```

- En todos los casos anteriores el resultado de mi búsqueda, es la misma. ¿es el "." y el "/" lo mismo?. ¿o es solo coincidencia?
- ¿Se puede usar el comando `find` con mas de un criterio? ¿y cual sería su sintáxis si es que es así?. (Por ejemplo si quiero buscar archivos `*.txt` y que además tengan permisos `rx`)
- Al ejecutar el comando, este busca en el directorio definido, pero además en sus subdirectorios. ¿Se puede relizar la búsqueda en un solo directorio?.

Bueno esas por el momento serían mis dudas, investigaré para ver si las puedo resolver y si les encuentro solución las comentaré.

Asterix

---

**Asterix:** De lo que he leído hasta ahora el punto "." significa que deseas buscar en tu directorio actual y sus subdirectorios. Si pones "/" la búsqueda será desde el directorio raíz, todos los archivos y por ahí leí que hay que tener cuidado porque puede demorarse muuuuuucho, no en este caso que le dices cuales archivos buscar, pero en otros sí. Respecto a más de un criterio, entiendo comandos, sí, por ejemplo puede buscar y borrar lo que encontró. Respecto a c) ni idea, pero san Google quizás lo sepa. En todo caso [AQUÍ](#) puedes encontrar algo más. Otra cosa, los nombres de archivo deben ir entre comillas sencillas, me imagino que no dobles como pusiste en algunos de tus ejemplos. Saludos.

Jorval

---

**Jorge**, respecto a tu pregunta por el "man" de "find", el resultado me aparece en inglés. A veces, lo que me aparece en castellano es "--help", pero trae mucha menos información que `man`. Así que, como yo también tengo problemas con el inglés, voy mirando en "help", en "man" y en Google, y de todo eso voy tratando de sacar algo en limpio.

Sobre el entrecomillado fuerte en `find`, mi desconcierto es por lo siguiente:

Si yo quiero imprimir en pantalla

## Grupo de estudios: Programación Bash Script

Tercera semana: del 22 al 28 de octubre de 2009

---

```
2 * 3 = 6
```

tengo que hacer

```
$ echo '2 * 3 = 6'
```

para que no interprete el asterisco como un comodín. Ahora, en el comando

```
$ find '*.tmp'
```

el entrecomillado no anula el asterisco ¿no sé si me explico?

.....

**Asterix**, se puede usar find con más de un criterio, fijate en el ejemplo 5 que dejé más arriba. Para eso se usan las opciones -a (AND lógico) y -o (OR lógico)

```
$ find -name '*.txt' -a -perm 777
```

El ejemplo busca ficheros con el nombre terminado en ".txt" y (and) con permisos de lectura, escritura y ejecución para el propietario, el grupo y otros.

Y existen dos opciones que son **-maxdepth** y **-mindepth** que van acompañadas de un número entero. Sirven para regular la profundidad de la búsqueda en el árbol de directorios. Habría que investigarlos para ver cómo funcionan.

```
$ find -maxdepth 1 -name '*.txt'
```

Supuestamente buscaría sólo en el directorio actual.

Alejandro

---

### Mis respuestas al desafío N° 1

```
1.- $ find /usr -group jorval -print
2.- No apareció ninguno
3.- $ find . -size 300c -print
4.- $ find . -size -300c -print
5.- $ find . -size +300c -print
6.- $ find . -mtime -29 -print
7.- $ find . -mtime -14 -print
8.- $ find . -mtime -4 -print
9.- $ find . -size +30 -print
10.- $ find . -name '*~' -print
11.- $ find . -name '*~' -print
```

## Grupo de estudios: Programación Bash Script

Tercera semana: del 22 al 28 de octubre de 2009

---

```
12.- $ mkdir para_borrar
     $ cp "archivo-que-me-salio-once" para_borrar
13.- $ find . -name 'para_borrar' -exec rm {} \;
14.- -amin n Buscar el último archivo al cual se accedió n
minutos atrás
     -used n Buscar el último archivo al cual se accedió n
días después que su estatus fue cambiado
15.- $ find . -amin 10 -print
     $ find . -used 3 -print
```

A todos les puse -print para poder verlos en pantalla.

**Jorval**

---

**Jorge**, encontré estos paquetes en synaptic para colocar el man en castellano:

```
sudo aptitude install manpages-es manpages-es-extra
```



Que lo disfrutes!

**Alejandro**

---

**Alejandro**, INSTALADOS Y YA DISFRUTÁNDOLO. Gracias, millones de gracias. Ahora sí que podré avanzar en mis estudios. Ya me parecía que en algunos sitios había visto páginas man en castellano, pero. En realidad en ubuntu y linux en general no hay nada o casi nada que no haya sido hecho por tantos entusiastas que trabajan permanentemente en mejorar el sistema y hacerlo extensivo a más personas.

**Jorval**

(yo también te agradezco! excelente!! **Fabricio** )

---

Ahí van mis pseudo-soluciones (elimino las líneas que son iguales a las de **Jorge**):

```
2.- ~$ find /usr -group alejandro >findUsr.txt ; wc -l
findUsr.txt ; cat findUsr.txt      (no respeta la consigna, pero
sirve)

6.- ~$ find ~ -mtime -30

7.- ~$ find ~ -mtime +15

8.- ~$ find ~ -mtime 5
```

```
11.- ~$ find ~ -name '*~' -exec cat {} \; | less

12.- ~$ mkdir para_borrar

    ~$ find ~ -name '*~' -exec cp {} para_borrar \;    (no sé
explicar por qué, pero no copia todos los que listó antes)

13.- ~$ find ~ -name '*~' -exec rm {} \;

15.- Creo que lo que pide aquí es un ejemplo, no de la sintaxis,
sino de para qué lo usaríamos, y no se me ocurre para qué...
```

### Comentarios sobre las soluciones de **Jorge**:

- Líneas 6, 7 y 8: la consigna dice "en los últimos 30 días" (-30); "más de 15 días" (+15); "hace 5 días" (5). El + y el - representan > y <.
- Línea 11: yo entendí que tiene que visualizar el "contenido" de los ficheros, para eso es necesario agregar el cat, y además paginar.
- Línea 12: el problema es que el find de la línea 11 puede arrojar una lista grande de ficheros, que deberían moverse al nuevo directorio.
- Línea 13: si lo que quieres es borrar el directorio "para\_borrar", debes usar "rm -R" (recursivo).

Alejandro

---

### Aclaraciones:

- Líneas 6, 7 y 8 : en el MAN en inglés lei y creí entender que -mtime y otras expresiones no consideraban las fracciones de día por lo tanto, cuando uno colocaba n +1 quería decir que era el día dos y no el uno. Bueno lo busqué en el MAN en castellano y esa parte no aparece traducida.
- Línea 11 Al leer el Man en castellano me enteré que no es necesario colocar -print ( se lo puse a todos) porque por defecto el comando find envía a la pantalla.
- Línea 12 Me salio un solo archivo en la línea 11. Es uno de "combinado" que hicimos la primera semana. Si fueran muchos, aún el comando cp con comodines creo que sería apropiado.
- Línea 13 , pero tú tampoco lo usaste, ¿por qué recursivo? si sabes que esa carpeta no tiene subdirectorios adentro, sólo archivos.
- Otra cosa, también en el MAN en inglés leí un asunto que al colocar el tamaño en -size había que colocarlo, por ejemplo 'c', lo probé con y sin entrecomillado y funciona igual. En el MAN en castellano ni lo menciona.

Jorval

---



Creo que sería interesante que los demás dejaran sus opiniones sobre las soluciones dadas hasta aquí, más que entablar una conversación entre dos personas. Si encuentran errores, si hay una mejor solución, si no dio el resultado esperado, etc. Esa retroalimentación siempre enriquece la comprensión de todo el grupo, es importante.

Alejandro

He aquí mis soluciones:

```
1) $ find /usr -group asterix
2) $ find /usr -group asterix | wc -l
3) $ find -size 300c -print 2>dev/null
4) $ find -size -300c -print 2>/dev/null (para contar se agrega
| wc -l)
5) $ find -size +300c -print 2>/dev/null
6) $ find ~ * -mtime -30 2>/dev/null | more
7) $ find ~ * -mtime +15 2>/dev/null | more
8) $ find ~ * -mtime 5 2>/dev/null | more
9) $ find / -size +30c -print 2>/dev/null
10) $ find ~ -name "*~" 2>/dev/null
11) $ find ~ -name "*~" 2>/dev/null | more
12) $ mkdir para_borrar; find ~ -name "*~" 2>/dev/null -exec cp
{} para_borrar \;
13) $ find ~ -name "*~" 2>/dev/null -exec rm {} para_borrar \;
14)
    a) El parámetro -amin se usa para buscar ficheros a los
    cuales se accedieron por última vez hace n minutos.
    b) El parámetro -used se usa para buscar ficheros n días
    después, desde que se cambió por última vez su estado.
15)
    a) $ find ~ "*" -amin 25 2>/dev/null
    b) $ find ~ "*" -used 25 2>/dev/null
```

**Asterix**

---

Corrijo el ejercicio 11, ya que de primera no entendía la sintaxis de **Alejandro**, pero leyendo el enunciado más detenidamente, me di cuenta del error, así es que gracias.

Quedaría del siguiente modo 11) `$ find ~ -name "*~" 2>/dev/null -exec cat {} \;`  
`| less`

Los ejercicios de **Jorval** (6,7 y 8) contienen algunos errores en la sintaxis, los cuales fueron detallados por **Alejandro**.

En el ejercicio 12 no tuve el problema que describe **Alejandro**, es decir me copió todos los listados.



**Dudas:**

- a) Veo que Jorval después del find usa -print, pero no sé si será tan necesario ya que con o sin el -print, a mi al menos me dan los mismos resultados.
- b) Al estar en mi directorio personal, al parecer no es necesario usar el carácter ~ o . después de find. Obtengo los mismos resultados si no los coloco.
- c) Algo similar me ocurre con el comodín \*. Si voy a buscar cualquier archivo que cumpla un cierto criterio, ¿es necesario colocarlo?
- d) En el man de find aparece used n con la siguiente descripción: "Se accedió por última vez al fichero n días después de que se cambió por última vez su estado". ¿Qué se entiende por estado.

De igual modo agradezco a todos, por la ayuda brindada a mis dudas.

**Asterix**

---

**Asterix:** Respecto a tus dudas

- a.- Tienes toda la razón, si lees más arriba puse unas Aclaraciones al respecto
- b.- También tienes razón, es lo mismo
- c.- No sabría qué decirte. Un tipo de entrecomillado que no es mencionado en el capítulo del libro que estamos siguiendo es el "entrecomillado inverso". Permite construir comandos tomando como argumentos la salida de otro comando entrecomillado entre comillas inversas --> ` ` (yo las tengo al lado de la tecla P)
- d.- Entiendo que el estado de un archivo son los permisos que tienen el usuario, el grupo y los otros sobre ese archivo. Cuando los cambias, cambia su estado.

**Jorval**

---

- Ejercicio 2: A mí me aparecieron 35 ficheros en /usr que pertenecen a mi grupo. Yo lo resolví de una forma bastante rebuscada. Pasa que cuando probé una solución similar a la que dio **Asterix**, no me funcionó, entonces intenté por otro lado. Pero ahora probé nuevamente y anda perfecto.
- Estoy de acuerdo en que colocar **-print** no ayuda mucho, porque find envía los datos a su salida estándar (pantalla)
- Si bien, para los scripts puede ser útil, para probar comandos, redirigir la salida de errores a **/dev/null** evita que veamos, justamente, los errores. Y de los errores se aprende...
- Es cierto que el caracter tilde "~" y "." no son necesarios cuando uno está ubicado en su home de usuario. Únicamente se justifica si la búsqueda se realiza desde otro lugar (que a veces sucede).
- Tampoco es necesario el "\*" cuando la búsqueda es para todos los archivos. Pero lo que yo revisaría es la sintaxis, porque:
  1. ¿qué está expandiendo el comodín "\*"?

- ¿el nombre? Entonces debería llevar la opción **-name** ¿no? Tipo: `find ~ -name '*' size -300c`
- ¿la ruta de acceso o path? En ese caso, no sé bien cómo se comportará el espacio en blanco que queda entre los dos comodines "`~ *`". ¿No sé si se entiende? Ahí hay algo extraño.

Una forma de poner a prueba lo que está sucediendo:

```
~$ find ~ -size -300c | wc -l
938
~$ find . -size -300c | wc -l
938
~$ find -size -300c | wc -l
938
~$ find ~ * -size -300c | wc -l
1033
~$ find * -size -300c | wc -l
95
~$ find -name '*' -size -300c | wc -l
938
```

Ahí, se ve claramente que en la cuarta y la quinta búsqueda pasa algo extraño. Sería realmente interesante averiguar "qué es lo que pasa" en esas líneas.



### ¿Alguien tiene idea?

Y por último, **Asterix**, veo un problema en tu línea 13.

```
13) $ find ~ -name "*~" 2>/dev/null -exec rm {} para_borrar \;
```



No está buscando en el directorio "`para_borrar`", sino en tu home. Y luego los borra. Si fueran ficheros importantes, **eso hubiese probocado un desastre en tu sistema**. Como son temporales, no pasó nada. Por otro lado, para borrar, no es necesario indicar un destino (`rm {} para_borrar`), eso sólo cuando copiamos o movemos.

Interesante lo que va surgiendo. Por eso insisto, **es bueno el intercambio**. Tenemos que buscarlos "los pelos y señales", como dice el dicho. No por ser criticones, sino para ayudarnos a aprender. Si Asterix no planteaba sus dudas, ni se me hubiese ocurrido pensar en eso. Cuando vi sus ejercicios por primera vez no noté nada que me hiciera pensar, los "\*" me parecieron lógicos.



Alejandro

---

**Alejandro**, tienes razón en el ejercicio 13.

Mirando las distintas opciones de `find` que pusistes, la uno, dos, tres y seis las probé y al

igual que tú me dieron los mismos resultados. No se me había ocurrido lo planteado por tí en la N° 4 y 5. Se me ocurre realizar un cat a las tres opciones (podría ser la uno, cinco y seis), redirigir la salida a un archivo, ordenar las líneas y ver cuales archivos (por ende en que directorio están), están repetidos, y analizar el resultado.

Asterix

---

**Asterix**, ¿Pudiste hacer esa prueba que dijiste? ¿Qué resultado dio?

### Entrecomillado inverso



Un tipo de entrecomillado que no es mencionado en el capítulo del libro que estamos siguiendo es el "entrecomillado inverso".

Permite construir comandos tomando como argumentos la salida de otro comando entrecomillado entre comillas inversas --> `` (yo las tengo al lado de la tecla P)

Ej:

```
~$ cat `find -name '*.sh'`
```

Muestra el contenido de todos los scripts ".sh" que encuentre en mi home. Viene a ser lo mismo que (pero con una sintaxis más transparente):

```
~$ find -name '*.sh' -exec cat {} \;
```

Interesante ¿no? **Lo bueno es que se puede aplicar a cualquier comando**, y no solamente a "find".

Alejandro

---

Qué raro... miren el resultado que me da este comando:

```
fb91@fb91-desktop:~$ find ~ -size -300c | wc -l
find: «/home/fb91/Desktop»: Permiso denegado
10513
```

Fabricio

---

**Fabricio**: lo que sucede es que al comando wc le pusiste -1 (uno) y debe ser -l (ele)

Jorval

---

Descubrí una cosa muy loca:

**938 + 95 = 1033**

No, no es que estoy aprendiendo a sumar. La cantidad de ficheros que encuentra "find ~ \*" es igual a la suma de "find ~" y "find \*"

¿Será que busca por los dos patrones?

"~" (todo lo de home) y "\*" **¿todo lo qué?**

Alejandro

---

Les comento mis observaciones:

Tuve que modificar mis criterios ya que con los ejemplos descritos arriba, me daba una enormidad de ficheros: resumo lo que hice.

```
a) ~$ find ~ -size 16384c | wc -l
    39
b) ~$ find . -size 16384c | wc -l
    39
c) ~$ find -size 16384c | wc -l
    39
d) ~$ find ~ * -size 16384c | wc -l
    72
e) ~$ find * -size 16384c | wc -l
    33
f) ~$ find -name '*' -size 16384c | wc -l
    39
```

Todos estos archivos los dirigí a prueba\_a, prueba\_d, y prueba\_e; luego los uní en uno solo, para finalmente ordenarlos por líneas, luego hice las siguientes observaciones:

- 1) Ejercicios a) b) c) y f) dieron exactamente el mismo resultado con los mismos archivos encontrados.
- 2) El ejercicio d) es como si me listara dos veces, excepto que "en el primer listado" me aparecen archivos en directorios ocultos (6 archivos), y en el "segundo listado" solo los archivos de directorios no ocultos (33 archivos). Por lo que 33 ("primer listado")+ 6 (archivos ocultos e el "primer listado")+33 en el "segundo listado"= 72
- 3) En el ejercicio e) solo me lista los archivos no ocultos (33), si le sumamos los 6 ocultos no dá el N° **39**.

Con lo anterior queda clarificado un poco lo raro del resultado de Alejandro.

Saquen ustedes sus propias conclusiones, si lo desean les puedo enviar las copias de lo que hice.

Saludos

Asterix

---

Jorge, puse una l (ele)

## Fabricio

---

Gracias **Asterix!** Ahora me empieza a quedar claro.

- el comodín tilde "~" es expandido por bash (en mi caso) como /home/alejandros/ y busca todo lo que hay ahí que cumpla los requisitos del patrón. Eso incluye archivos y directorios, tanto ocultos, como no. Dando por ejemplo:

```
/home/alejandros/Documentos/Textos/papeles.txt
/home/alejandros/Documentos/planillas/mensual.odt
/home/alejandros/.themes/wallpaper.png
/home/alejandros/.oculto.sh
```

- en cambio, el comodín asterisco "\*" es expandido como el nombre de un fichero o directorio, pero sin tomar los ocultos. Una vez encontrado, muestra la ruta de esta manera:

```
Documentos/Textos/papeles.txt
Documentos/planillas/mensual.odt
```

- entonces, al sumar los dos patrones de búsqueda "~ \*", va a listar todo, y aparecen líneas repetidas (pero no iguales):

```
/home/alejandros/Documentos/Textos/papeles.txt
/home/alejandros/Documentos/planillas/mensual.odt
/home/alejandros/.themes/wallpaper.png
/home/alejandros/.oculto.sh
Documentos/Textos/papeles.txt
Documentos/planillas/mensual.odt
```

Para find no es lo mismo /home/alejandros/Documentos... que Documentos... directamente.

**Conclusión interesante:** se pueden dar distintas rutas a la búsqueda en find. Por ejemplo, sería posible buscar archivos en varias carpetas, sin necesidad de buscar en todo el sistema:

```
~$ cp `find ~/.icons /usr/share/icons /usr/local/share/icons -
name '*.svg'` MisIconos
```

De esa forma copio todos los iconos que andan dando vueltas por ahí en una carpeta. Cosa que no podría hacer buscando todos los .svg del directorio raíz /, porque saldrían imágenes que no son iconos.

**Lo que no termino de entender** es ¿por qué el comodín asterisco "\*" no lista archivos ocultos?

Alejandro

---

**Fabricio** ¿Cómo están los permisos del directorio Desktop?

```
~$ ls -l ~/Desktop
```

¿Tiene permiso de lectura "r"? ¿fb91 aparece en la columna de propietario y grupo?

Alejandro

---

El comando `ls -l ~/Escritorio` me dice lo siguiente:

```
drwx----- 2 root root 4096 2009-07-20 17:19 Desktop
```

seguramente tengo que darle permisos con el `chmod` (aunque los archivos que dejo "tirados" en el escritorio me los guarda en Escritorio y no en Desktop)... igualmente la duda que me quedó no es que no tengo permisos para buscar en Desktop, sino el número tan grande que me dá el resultado de la búsqueda con respecto al de ustedes!

A mi me dá **10513** y a ustedes al rededor de 900

Fabricio

---

Ah, **Fabricio**, pero no te preocupes por eso. Cuando yo hice la prueba estaba en una versión de Ubuntu Openbox recién instalada. Pero, cuando la hice en Jaunty me tiró unos 30 mil, más o menos.

Alejandro

---



Tengo **una gran duda**: ¿Qué pretende ser la ruta `acceso ~*`? En realidad no comprendo que es lo que se quiere encontrar. Pues `ruta_acceso` es la ruta de acceso del directorio en que se desea buscar.

Jorval

---

**Jorge**, no sé si entiendo bien la pregunta.

La búsqueda `find ~ *` (con espacio en el medio de los dos comodines) es lo que estuvimos analizando hasta aquí. Se trata en realidad de dos búsquedas simultáneas: `find ~` y `find *`. Son dos rutas, dos path.

La otra búsqueda era `find -name '*~'`, que buscaba los archivos de backup generados automáticamente, tipo `sources.list~`. En este caso no se trata del path, sino del nombre del fichero o directorio.

¿Responde eso tu pregunta? ¿o se trata de otra cosa?

## **Grupo de estudios: Programación Bash Script**

*Tercera semana: del 22 al 28 de octubre de 2009*

---

---

Alejandro

---

---

**Alejandro**, Sí esa era mi duda. Para mí "find" es para buscar en un árbol de directorios, si uno le indica un directorio el buscará automáticamente en todos los subdirectorios del directorio. En todo caso, demos por finalizado este aspecto.

Jorval

---

---