# find

El comando find en Linux se utiliza **para buscar archivos y directorios** en un sistema de archivos basado en diversas condiciones que especifiques. Uso Básico

El formato básico del comando find es:

**find [ruta] [opciones] [acción]**

## Ruta

[**ruta**]: Directorio donde find empezará a buscar.

Si se utiliza el punto significa que se está buscando en el directorio actual.

Si se utiliza / significa en todos los directorios.

Ej:

**find . –name prueba.txt**

## Opciones

[**opciones**]: Criterios de búsqueda como el nombre del archivo, tipo, tamaño, etc.

### 1. -name

Busca archivos que coincidan exactamente con el nombre dado.

Ejemplo:

**find /home/usuario -name "nota.txt"**

Busca un archivo llamado "nota.txt" en el directorio /home/usuario.

### 2. -iname

Similar a -name, pero la búsqueda es insensible a mayúsculas y minúsculas.

Ejemplo:

**find /home/usuario -iname "nota.txt"**

Busca por "nota.txt", "Nota.txt", "NOTA.TXT", etc.

### 3. -type

Busca por tipo de archivo. Algunos valores comunes para tipo son f (archivo regular), d (directorio), l (enlace simbólico).

Ejemplo:

**find / -type d**

Busca todos los directorios en el sistema de archivos raíz.

### 4. -size

Busca archivos según su tamaño. Usa c (bytes), k (kilobytes), M (megabytes), G (gigabytes).

Ejemplo:

**find / -size +50M**

Busca archivos mayores de 50 megabytes.

### 5. -mtime, -atime, -ctime

Busca archivos por la última vez que fueron modificados (mtime), accedidos (atime) o cambiados (ctime). El tiempo se especifica en días.

Ejemplo:

**find /home/usuario -mtime -7**

Busca archivos modificados en los últimos 7 días.

### 6. -perm

Busca archivos por permisos.

Si se utiliza el signo menos puede terner más permisos además de los que se especifica.

Si no se utiliza el signo menos se hace una búsqueda estricta con esos permisos

Ejemplo:

find / -perm 644

Busca archivos con permisos 644.

***-perm [-]nnn*** : Buscaría todos los archivos cuyos permisos de acceso sean los indicados (nnn - octal). Si no se pone el signo menos delante buscará los archivos que tengan exactamente los permisos indicados por nnn y si se pone el signo menos delante nos dará los que coinciden en los permisos, aunque el resto sean distintos.

Los permisos se pueden indicar en formato simbólico. –perm /u=r

-perm /a=x todos ejecución.

-perm g=r equivale a perm 040

-perm –g=r que tenga el grupo lectura con independencia del resto de permisos.

find . -perm 644 🡪 busca archivos que tengan estos permisos exactamente y solo estos permisos y no más.

find . -perm -644 🡪Buscar archivos que tengan estos permisos, aunque puedan tener más permisos que estos

find . -perm /644 🡪 Busca los archivos que tengan alguno de solo de los permisos aunque solo tengan el de r de usuario.

Find . -perm g=r -type f 🡪 busca ficheros que solo tienen el permiso de r para el grupo resto 0040 resto nada.

U=r -type f 🡪 busca ficheros que solo tienen el permiso de r para el usuario 0400 resto nada.

A=r -type f 🡪 busca ficheros que solo tienen el permiso de r para el usuario, grupo y others 444 y que no tengan más permisos.

/u=r -type f 🡪 busca todos los ficheros que tengan permiso de r el usuario propietario, aunque tenga más permisos.

/a=r -type f 🡪 busca todos los ficheros que tengan permiso de r el usuario propietario ó el grupo ó other, aunque tenga más permisos.

**find / -perm u=rw,g=r,o=r**

Busca archivos donde el usuario tiene lectura y escritura, el grupo tiene lectura y otros tienen lectura.

### 7. -user, -group

Busca archivos que pertenecen a un usuario o grupo específico.

Ejemplo:

**find /home -user usuario**

Busca todos los archivos en /home que pertenecen al usuario "usuario".

### 8. -exec

Ejecuta un comando sobre los archivos encontrados.

Ejemplo:

**find /tmp -type f -exec rm {} \;**

Elimina todos los archivos en /tmp.

### 9. -or, -and, -not

Estos operadores lógicos permiten combinar condiciones.

Ejemplo:

**find /home -name "\*.tmp" -or -name "\*.bak"**

Busca archivos que terminan en ".tmp" o ".bak" en /home.

### 10. -maxdepth

Limita la búsqueda a una profundidad específica de directorios.

Ejemplo:

**find /home/usuario -maxdepth 1 -name "nota.txt"**

Busca archivos llamados "nota.txt" solo en el directorio /home/usuario sin entrar a subdirectorios.

### 11. -inum

Busca archivos por su número de inodo.

Ejemplo:

**find / -inum 12345**

Busca el archivo que tiene el número de inodo 12345.

### 12. -uid

Busca archivos por el UID (identificador de usuario) del propietario.

Ejemplo:

**find /home -uid 1000**

Busca archivos que pertenecen al usuario con UID 1000.

## Acción

[**acción**]: Qué hacer con los resultados encontrados, como imprimir los nombres de archivo, ejecutar otro comando, etc.

### 1. -print

Esta es la acción predeterminada de find, que imprime la ruta completa de los archivos encontrados.

Ejemplo:

find /home/usuario -name "\*.txt" -print

Imprime todas las rutas de archivos .txt en el directorio /home/usuario.

### 2. -exec

Permite ejecutar un comando en cada uno de los archivos encontrados. {} se reemplaza por el nombre del archivo actual y ; termina el comando.

rm “{}” “;” para borrar

cp para copiar

-i para preguntar

Ejemplo:

find /home/usuario -type f -name "\*.jpg" -exec chmod 644 {} \;

Cambia los permisos de todos los archivos .jpg a 644.

### 3. -execdir

Similar a -exec, pero el comando se ejecuta desde el directorio donde se encuentra el archivo.

Ejemplo:

find /home/usuario -type f -name "config.sh" -execdir pwd \;

Ejecuta el comando pwd en el directorio de cada archivo encontrado llamado config.sh.

### 4. -delete

Elimina los archivos encontrados. Es más seguro que usar -exec rm {} porque evita problemas con nombres de archivos extraños y no necesita invocar a otro proceso.

Ejemplo:

find /tmp -type f -name "\*.bak" -delete

Elimina todos los archivos con extensión .bak en el directorio /tmp.

### 5. -ls

Muestra detalles de los archivos encontrados, similar a la salida del comando ls -l.

Ejemplo:

find /home/usuario -type f -name "\*.pdf" -ls

Muestra detalles de todos los archivos .pdf encontrados en /home/usuario.

### 6. -ok

Similar a -exec, pero find pide confirmación en la terminal antes de ejecutar el comando para cada archivo.

Ejemplo:

find /home/usuario -type f -name "clean.sh" -ok chmod 755 {} \;

Pide confirmación antes de cambiar los permisos de cada archivo clean.sh a 755.

### 7. -prune

Evita que find descienda en los directorios especificados. Útil para excluir directorios de la búsqueda.

Ejemplo:

find /home/usuario -name ".git" -prune -o -type f -print

Busca todos los archivos en /home/usuario pero excluye los directorios .git y su contenido.

## Ejemplos Comunes (resumen)

Buscar un archivo por nombre:

find /ruta/a/buscar -name "nombre\_archivo.txt"

Esto buscará en el directorio /ruta/a/buscar y todos sus subdirectorios cualquier archivo que se llame nombre\_archivo.txt.

Buscar archivos por tipo:

find /ruta/a/buscar -type f

Aquí, -type f indica que solo quieres buscar archivos regulares (no directorios).

Buscar y ejecutar un comando en los archivos encontrados:

find /ruta/a/buscar -type f -name "\*.txt" -exec cat {} \;

Este comando buscará todos los archivos .txt y ejecutará el comando cat para mostrar su contenido. {} es un marcador que find reemplaza con el nombre del archivo encontrado.

Buscar archivos modificados en los últimos 7 días:

find /ruta/a/buscar -mtime -7

-mtime -7 busca archivos que han sido modificados en los últimos 7 días.

Buscar archivos por tamaño:

find /ruta/a/buscar -size +100M

-size +100M busca archivos que son mayores de 100 megabytes.

Consejos

Cuidado con los permisos: find buscará solo en directorios a los que el usuario tiene acceso. Si necesitas buscar en todos los directorios, puedes necesitar privilegios de superusuario (usando sudo).

Uso de comillas: Usa comillas para encerrar patrones que incluyan caracteres especiales o espacios, para que el shell no los interprete.

# grep

El comando grep en Linux se utiliza **para buscar texto específico dentro de archivos** o cualquier dato ingresado a través de un flujo. Te permite filtrar y mostrar las líneas que coinciden con una expresión regular o un patrón.

El formato básico del comando grep es:

**grep [opciones] 'patrón' [archivos]**

* 'patrón': Es la cadena de texto o expresión regular que quieres buscar.
* [archivos]: Uno o más archivos en los que buscar el patrón. Si no se especifica un archivo, grep espera entrada de datos a través de un flujo (como el resultado de otro comando).

## Opciones

### 1. -c

Cuenta el número total de líneas que contienen coincidencias del patrón especificado.

Ejemplo:

grep -c 'error' log.txt

Este comando cuenta cuántas líneas en el archivo log.txt contienen la palabra "error".

### 2. -i

Realiza búsquedas sin distinguir entre mayúsculas y minúsculas.

Ejemplo:

grep -i 'Server' config.ini

Busca la palabra "Server" en el archivo config.ini, ignorando si está en mayúsculas, minúsculas o una combinación de ambas.

### 3. -v

Muestra todas las líneas que no contienen el patrón especificado.

Ejemplo:

grep -v 'passed' results.log

Muestra todas las líneas del archivo results.log que no contienen la palabra "passed".

### 4. -n

Muestra el número de línea junto con las líneas que coinciden con el patrón.

Ejemplo:

grep -n 'todo' notes.txt

Busca la palabra "todo" en el archivo notes.txt y muestra las líneas coincidentes precedidas por su número de línea.

### 5. -l

Lista los nombres de los archivos que contienen al menos una coincidencia del patrón, sin mostrar las líneas coincidentes.

Ejemplo:

grep -l 'urgent' \*.txt

Lista todos los archivos .txt que contienen la palabra "urgent".

### 6. -r

Busca de forma recursiva el patrón especificado en todos los archivos del directorio dado y subdirectorios.

Ejemplo:

grep -r 'config' /etc/

Busca la palabra "config" en todos los archivos dentro del directorio /etc y sus subdirectorios.

### 7. -W

Busca exactamente la palabra especificada como un todo, no como parte de otra palabra.

Ejemplo:

grep -W 'log' system.log

Busca la palabra exacta "log" en system.log, asegurándose de que no sea parte de otras palabras como "blog" o "catalog".

## Expresiones regulares comunes en grep:

### . (punto)

Sustituye cualquier carácter en esa posición.Ejemplo:

grep 'e.r' document.txt

Busca cualquier palabra que tenga una 'e', seguida de cualquier carácter, y luego una 'r' (como "error", "earth", etc.).

### \* (asterisco)

Repite el carácter anterior cero o más veces.Ejemplo:

grep 'ho\*' report.txt

Busca 'h' seguido de cero o más 'o' (como "h", "ho", "hoo", etc.).

### [ ] corchetes

Coincide con cualquier carácter único dentro de los corchetes.Ejemplo:

grep 'b[aeiou]g' words.txt

Busca palabras que comienzan con 'b', seguido de cualquier vocal, y terminando en 'g' (como "bag", "beg", "big", etc.).

#### [ˆ ]

Excluye cualquier carácter dentro de los corchetes.Ejemplo:

grep 'b[^aeiou]g' words.txt

Busca palabras que empiezan con 'b', seguido de cualquier carácter que no sea una vocal, y terminando en 'g' (como "bng", "bfg").

### ˆ

Coincide con el inicio de una línea.Ejemplo:

grep '^start' script.sh

Busca líneas que comienzan con la palabra "start".

### $

grep 'end$' script.sh

Busca líneas que terminan con la palabra "end".

## Ejemplos Prácticos

### Buscar un texto en un archivo específico:

grep 'error' log.txt

Busca la palabra "error" en el archivo log.txt.

### Buscar de forma recursiva en todos los archivos bajo un directorio:

grep -r 'getConfig()' /var/www/html/

Busca el texto "getConfig()" recursivamente en todos los archivos bajo /var/www/html/.

### Buscar ignorando mayúsculas y minúsculas:

grep -i 'usuario' lista\_usuarios.txt

Busca la palabra "usuario" en lista\_usuarios.txt, sin importar si está en mayúsculas o minúsculas.

### Contar el número de coincidencias:

grep -c 'fatal' error\_log.txt

Cuenta cuántas veces aparece la palabra "fatal" en error\_log.txt.

### Mostrar las líneas que no coinciden con el patrón:

grep -v 'ok' status\_report.txt

Muestra todas las líneas en status\_report.txt que no contienen la palabra "ok".

### Mostrar el número de línea de las coincidencias:

grep -n 'warning' system\_log.txt

Muestra las coincidencias de "warning" en system\_log.txt junto con su número de línea.

### Listar archivos con coincidencias sin repetir el patrón:

grep -l 'test' \*.py

Lista los nombres de todos los archivos .py que contienen la palabra "test".

# sort

El comando sort en Linux es una herramienta esencial **para ordenar y fusionar ficheros de texto**.

La sintaxis básica del comando sort es:

**sort [opciones] [archivo1 [archivo2 ...]]**

## Funcionamiento sin Parámetros

Si sort se ejecuta sin parámetros, espera que se introduzcan datos manualmente a través del teclado. Cada línea de texto se separa pulsando Enter, y el proceso se finaliza con Ctrl+D. Luego, sort ordena los datos ingresados y los muestra por la salida estándar (el monitor).

## Ordenar Múltiples Archivos

Para ordenar múltiples archivos simultáneamente y mostrar el resultado en la salida estándar, puedes usar:

**sort fichero1 fichero2**

## Opciones Comunes de sort

### -c

sort -c paises

Explicación: Verifica si el archivo paises está ordenado. Si no está ordenado, sort terminará su ejecución y puede opcionalmente mostrar la primera línea que no cumple con el orden establecido.

### -u

sort -u master

Explicación: Ordena el archivo master, eliminando las líneas duplicadas. Esto es útil para obtener una lista de entradas únicas, por ejemplo, en una lista de nombres o elementos que pueden estar repetidos.

### -f

sort -f master

Explicación: Ordena el archivo master ignorando las diferencias entre mayúsculas y minúsculas. Es útil cuando los datos pueden tener inconsistencias en el uso de mayúsculas, asegurando que todos los datos se traten de manera uniforme.

### -n

sort -n precios

Explicación: Ordena el archivo precios tratando su contenido como números en lugar de texto. Esto asegura que los números se ordenen según su valor aritmético (por ejemplo, 10 vendrá después de 2), en lugar de por orden lexicográfico.

### -r

sort -r master

Explicación: Ordena el archivo master en orden inverso (de mayor a menor). Esta opción es especialmente útil cuando necesitas ver los valores más altos o más recientes al principio de la lista.

### -o

sort fichero1 -o ficheroOrdenado

Explicación: Ordena el contenido de fichero1 y guarda el resultado en ficheroOrdenado. Esta opción es útil para preservar los datos originales mientras se trabaja con una versión ordenada del archivo.

### -t

sort -t ':' -k 2 fichero

Explicación: Utiliza : como delimitador para separar campos en fichero y ordena basándose en el segundo campo. Esto es útil para archivos con múltiples campos separados por un delimitador específico, como CSV o archivos de configuración delimitados.

### -m

sort -m ordenado1 ordenado2

Explicación: Fusiona dos archivos previamente ordenados (ordenado1 y ordenado2) manteniendo el orden. Es eficiente cuando se combinan listas grandes que ya están en orden, evitando la necesidad de reordenar los datos combinados.

## Ejemplos Prácticos

### Ordenar y Guardar el Resultado en un Archivo:

sort fichero1 fichero2 > ficheroOrdenado

### Ordenar un Archivo y Sobrescribir el Original:

sort fichero -o fichero

### Ordenar por Idioma y luego por Continente:

sort -t: -k 4,4 -k 1,1 -k 2,2 paises

Este comando ordena el archivo paises por idioma (campo 4), continente (campo 1) y país (campo 2).

### Eliminar Líneas Duplicadas y Guardar el Resultado:

cat pass.txt | sort | uniq > pass-listos.txt

Crea un archivo con líneas únicas ordenadas, eliminando duplicados.

### Ordenación con Salida Específica

sort fichero1 fichero2 > fichero3

sort fichero2 > fichero2

sort fichero1 fichero2 -o fichero3

sort fichero2 -o fichero2

sort fichero1 -o fichero3 fichero2

Estos comandos muestran cómo ordenar uno o más ficheros y dirigir el resultado ya sea a otro archivo nuevo o reemplazar el contenido del archivo original. Utilizan tanto redirección con > como la opción -o para especificar el archivo de salida.

# cut

El comando cut en Linux es una herramienta para **extraer y manipular columnas o campos específicos de un archivo o entrada estándar**. Se usa principalmente para extraer partes de líneas de texto, lo que lo hace muy útil para procesar archivos de texto estructurado o flujos de datos.

La sintaxis básica del comando cut es:

**cut [opciones] [fichero]**

Si no se especifica un archivo, cut leerá la entrada estándar (lo que se digite en el teclado hasta que se introduzca Ctrl+D).

## Opciones Comunes

### -d

-d: Define el delimitador utilizado para separar campos en una línea. Si no se especifica, el delimitador por defecto es el tabulador.

**cut -d: -f1,4,6 fichero**

Explicación: Este comando extrae los campos 1, 4 y 6 de cada línea en fichero, usando : como el carácter delimitador.

### -c

-c: Selecciona columnas específicas o rangos de columnas para mostrar.

**cut -c1-15**

Explicación: Extrae los primeros 15 caracteres de cada línea de la entrada.

**cut -c1,15**

Explicación: Extrae el primer y el decimoquinto carácter de cada línea.

### -f

-f: Utilizado con -d, especifica los campos que se deben extraer.

**who | tr -s " " | cut -f2 -d " "**

Explicación: Este comando se usa en combinación con who y tr para limpiar el espacio en blanco y luego extraer el segundo campo, que generalmente indica el terminal del usuario.

## Ejemplos Prácticos

### Extraer Múltiples Campos Específicos Usando un Delimitador Personalizado

**cut -d: -f1-3 paises**

Explicación: Extrae los primeros tres campos de cada línea en el archivo paises, utilizando : como delimitador. Esto es útil para archivos de tipo CSV o similar donde los campos están claramente definidos por un delimitador específico.

### Filtrar y Extraer con grep

**cut -d: -f1-3 paises | grep EUROPA**

Explicación: Este comando primero extrae los tres primeros campos del archivo paises, y luego usa grep para filtrar las líneas que contienen la palabra "EUROPA". Es una forma eficaz de combinar herramientas para procesar y filtrar información basada en criterios específicos.

### Extraer Rangos de Caracteres Específicos

**cut -c1-6 paises**

Explicación: Extrae los primeros seis caracteres de cada línea en el archivo paises. Este comando es útil cuando se necesita un resumen o una vista previa de cada línea, especialmente en archivos con un formato de ancho fijo.

### tr

El comando tr en Linux es una herramienta útil **para transformar y manipular texto** de diversas maneras, incluyendo la compresión de caracteres repetidos, la eliminación de caracteres y la sustitución de caracteres por otros. Este comando opera en la entrada estándar y escribe en la salida estándar, lo que lo hace idóneo para uso en tuberías con otros comandos.

La sintaxis básica de tr se puede expresar de dos maneras principales:

## Compresión de caracteres repetidos y/o sustitución:

tr -s [cadena1] [cadena2]

Si cadena2 se omite, tr sustituirá todas las repeticiones contiguas de caracteres en cadena1 por una sola ocurrencia de ese carácter.

## Eliminación de caracteres:

tr -d [cadena]

Este formato elimina todos los caracteres en cadena de la entrada.

Opciones Comunes

-s: Comprime secuencias de caracteres repetidos (squash). Se utiliza para convertir múltiples repeticiones del mismo carácter en una sola.

-d: Elimina todos los caracteres especificados en cadena de la entrada.

Ejemplos Prácticos

### Compresión de Espacios en Blanco

who | tr -s " "

Explicación: Este comando convierte todas las secuencias de espacios en blanco en la salida del comando who en un único espacio, haciendo que la salida sea más legible y estandarizada.

### Sustitución de Espacios por Otro Carácter

who | tr -s " " "\*"

Explicación: Similar al ejemplo anterior, pero en lugar de comprimir los espacios en uno, los reemplaza por asteriscos. Esto puede ser útil para visualizar claramente la separación entre campos, especialmente en entornos donde el espacio puede ser menos visible.

### Eliminación de Espacios

who | tr -d " "

Explicación: Elimina todos los espacios de la salida del comando who. Esto podría ser útil si se necesitan procesar los datos sin espacios, por ejemplo, para generar identificadores únicos a partir de nombres de usuario y otros detalles.