

## COMANDO grep:

El comando grep selecciona y muestra las líneas de los archivos que coincidan con la cadena o patrón dados.

## SINTAXIS:

La sintaxis es

**grep** [opciones] patrón [archivo]

## OPCIONES:

- A Muestra el número de líneas de texto que hay después de la línea coincidente.
- a No suprime las líneas de salida con datos binarios, las trata como texto.
- b Mostrar el offset de bytes del archivo de entrada antes de cada línea en la salida.
- c Muestra la cuenta de líneas coincidentes.  
Definir acción para acceder a los directorios
- d acción **read** lee todos los archivos del directorio.  
**skip** salta directorios.  
**recurse** lee reiterativamente todos los archivos y directorios
- e patrón Busca un patrón.
- h Muestra las líneas coincidentes pero no los nombres de archivo.
- i Ignora los cambios mayúsculos y minúsculos, las considera equivalentes.
- n Muestra la línea y el número de línea.
- q Mostrar en modo silencioso, no muestra nada.
- r Lee iterativamente todos los archivos en los directorios y subdirectorios encontrados.
- v Muestra todas las líneas que no coinciden.
- V Muestra la versión.
- w Coincidencia en palabras completas únicamente.

**También puedes usar patrones para la operación de búsqueda.**

- . Coincidencia de caracteres únicos.
- \* Carácter comodín.
- ^ Empieza por.
- \$ Termina en.

## EJEMPLO:

Vamos a asumir que tenemos un archivo file1.txt y tiene la siguiente información.

**hscripts has many valuable free scripts**  
**It is the parent site of www.forums.hscripts.com**  
**hscripts include free tutorials and free gif images**  
**Purchase scripts from us**  
**A webmaster/web master resource website**

1. Para mostrar todas las líneas que contengan hscripts:

```
grep 'hscripts' file1.txt
```

La salida será:

```
hscripts has many valuable free scripts  
It is the parent site of www.forums.hscripts.com  
hscripts include free tutorials and free gif images
```

2. Para mostrar la cuenta de líneas que contienen hscripts:

```
grep -c 'hscripts' file1.txt
```

La salida será:

```
3
```

3. Para mostrar las líneas que empiezan por hscripts:

```
grep '^hscripts' file1.txt
```

La salida será:

```
hscripts has many valuable free scripts  
hscripts include free tutorials and free gif images
```

4. Para buscar los archivos en el directorios HEC que contengan la cadena "include":

```
grep -c 'include' HEC/*
```

El comando anterior mostrará el nombre de archivo y la cuenta de líneas que contienen la cadena "include"

**Salida ejemplo:**

```
HEC/admin.php:3  
HEC/auth.php:1  
HEC/calendar.php:3  
HEC/checklogin.php:0  
HEC/colors.php:0  
HEC/msize.php:3
```

Grep es sin duda alguna no de los comandos más utilizados en el día a día por los administradores de sistemas. Muchos no van más allá de sus usos más básicos, que si bien cumplen a la perfección su función son una mínima parte de las posibilidades que nos ofrece. Algunos de estos ejemplos se pueden ejecutar tanto con grep como con egrep (**egrep= grep -E**).

Este es el texto que se va a utilizar para trabajar como ejemplo. Es un fichero de configuración de un Cluster MySQL de prueba. El nombre será test.

```
[ndb_mgmd]
hostname=192.168.0.10          # Hostname or IP address of management
node
datadir=/var/lib/mysql-cluster # Directory for management node log
files

# Options for data node "A":
[ndbd]
# (one [ndbd] section per data node)
hostname=192.168.0.30          # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data
files

# Options for data node "B":
[ndbd]
hostname=192.168.0.40          # Hostname or IP address
datadir=/usr/local/mysql/data   # Directory for this data node's data
files

# SQL node options:
[mysqld]
hostname=192.168.0.20          # Hostname or IP address
# (additional mysqld connections can
be
# specified for this node for various
# purposes such as running
ndb_restore)
```

- **Buscar dos o n strings distintas dentro de un mismo fichero**

La sintaxis es '(cadena1|cadena2|cadenaN)'. La salida será todas aquellas líneas que contienen cualquiera de esas strings. Esto nos permite hacer múltiples búsquedas en un único comando:

```
$ egrep '(192.168.0.30|192.168.0.40)' test
hostname=192.168.0.30          # Hostname or IP address
hostname=192.168.0.40          # Hostname or IP address
$ egrep '(192.168.0.30|192.168.0.40|192.168.0.20)' test
hostname=192.168.0.30          # Hostname or IP address
hostname=192.168.0.40          # Hostname or IP address
hostname=192.168.0.20          # Hostname or IP address
```

## Usar los corchetes para reducir el rango de búsqueda

Podemos hacer uso de los corchetes [] para definir, dentro de una misma string, que una sección contenga únicamente X caracteres, un rango de ellos, etc.

En este caso queremos sacar únicamente los resultados que contengan 192.168.0.30 y 192.168.0.40:

```
$ egrep 192.168.0.[30,40] test
hostname=192.168.0.30      # Hostname or IP address
hostname=192.168.0.40      # Hostname or IP address
```

Pero si quisiéramos definir los rangos que queremos mostrar para los dos últimos caracteres, en este caso numéricos (se podría hacer con alfanuméricos) podemos hacerlo separando el valor inicial y el final con “-“. En este ejemplo queremos que nos muestre aquello que cumpla la condición de que el primer número del último bloque tiene que ser 0,1 ó 2:

```
$ egrep 192.168.0.[0-2]0 test
hostname=192.168.0.10      # Hostname or IP address of management
node
hostname=192.168.0.20      # Hostname or IP address
```

Y en este que el primer número del último bloque sea 1,2 ó 3 y el último entre 0 y 9:

```
$ egrep 192.168.0.[0-3][0-9] test
hostname=192.168.0.10      # Hostname or IP address of management
node
hostname=192.168.0.30      # Hostname or IP address
hostname=192.168.0.20      # Hostname or IP address
```

Un ejemplo práctico con caracteres alfanuméricos sería usar por ejemplo **[a-c]test**, que buscaría atest, btest y ctest ó por ejemplo **[ar4d]test** que buscaría atest, rtest, 4test y dtest.

## Uso de clases predefinidas

Existen clases predefinidas que nos pueden llegar a ahorrar mucho trabajo. Son las siguientes:

```
[[:alnum:]], [[:alpha:]], [[:cntrl:]], [[:digit:]], [[:graph:]], [[:lower:]],
[[:print:]], [[:punct:]], [[:space:]], [[:upper:]], [[:xdigit:]]
```

Lo que hace cada una de ellas está claro por su nombre, no obstante vamos a ver un ejemplo. Podemos buscar toda línea que contenga un carácter en mayúsculas:

```
$ egrep [[:upper:]] test
hostname=192.168.0.10      # Hostname or IP address of management
node
datadir=/var/lib/mysql-cluster # Directory for management node log
files
...
...
```

- **Todo lo que comienza o termina por...**

Esto es sencillo, **las líneas comienzan por el carácter ^ y terminan con \$**. Por lo que:

Buscar todo lo que comience por “hostname”:

```
$ egrep ^hostname test
hostname=192.168.0.10      # Hostname or IP address of management
node
hostname=192.168.0.30      # Hostname or IP address
hostname=192.168.0.40      # Hostname or IP address
hostname=192.168.0.20      # Hostname or IP address
```

O todo lo que termine por “node”:

```
$ egrep node$ test
hostname=192.168.0.10      # Hostname or IP address of management
node
```

- **Más expresiones regulares**

- o **?** El carácter que precede es opcional y coincide al menos una vez.
- o **\*** El carácter que precede coincidirá 0 o más veces.
- o **+** El carácter que precede coincidirá 1 o más veces.
- o **{n}** El carácter que precede coincidirá exactamente n veces.
- o **{n,}** El carácter que precede coincidirá n o más veces.
- o **{,m}** El carácter que precede coincidirá como máximo m veces.
- o **{n,m}** El carácter que precede coincidirá entre n y m veces.

Podemos entonces buscar todas las líneas del fichero que contienen una IP. Usamos primero los corchetes para definir que puede haber números del 0 al 9, con las llaves decimos que habrá 1 o 3 números en cada bloque y así cuatro veces (nota: hay que escapar las llaves):

```
$ grep '[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}' test

hostname=192.168.0.10      # Hostname or IP address of management
node
hostname=192.168.0.30      # Hostname or IP address
hostname=192.168.0.40      # Hostname or IP address
hostname=192.168.0.20      # Hostname or IP address
```

- **Buscar palabras completas, contar resultados**

Para indicar a grep que queremos que la búsqueda se centre en palabras y no en caracteres sueltos especificamos el parámetro “-w”. Podemos ver la diferencia buscando “data” en el ejemplo, contamos los resultados satisfactorios con el parámetro “-c”:

```
$ grep -cw data test
5
$ grep -c data test
6
```

- **Excluir resultados, sensibilidad a mayúsculas**

Algo básico a la hora de usar grep es conocer los parámetros “-v” que excluye la cadena indicada en el resultado y “-i” que especifica que no se tendrá en cuenta si el resultado es mayúscula o minúscula.

Eliminamos las líneas que tienen comodines:

```
$ grep -v "#" test
[ndb_mgmd]

[ndbd]

[ndbd]

[mysqld]
```

- **Uso de pipes**

Todos usamos pipes (|) para acotar resultados. Hacemos un primer grep, con el resultado de ese hacemos otro y así sucesivamente:

```
$ grep -iw hostname test | grep 192.168.0.[0-9]0 | grep -v
192.168.0.40 | grep -vw node
hostname=192.168.0.30          # Hostname or IP address
hostname=192.168.0.20          # Hostname or IP address
```

- **Listar los archivos que contienen la cadena**

Con el parámetro -l podemos ejecutar grep contra varios archivos (y de forma recursiva en múltiples directorios con -R) y recibiremos el listado de archivos con coincidencias sin mostrar las líneas que contienen la cadena:

```
$ grep -lR SQL *
config.ini
header.xcf
iptables.sh
test
dir1/test.sql
dir2/prueba.txt
...
```

- **Número de línea en la que se encuentran los resultados**

Para saber el número de línea en la que se encuentra el resultado utilizaremos el parámetro -n:

```
$ grep -n hostname test
2:hostname=192.168.0.10          # Hostname or IP address of
management node
8:hostname=192.168.0.30          # Hostname or IP address
...
```

- **Cuántas veces se encuentran resultados**

Si “-n” nos mostraba el nº de línea en el que estaba la coincidencia, “-c” nos dice cuántas veces está la búsqueda en el fichero:

```
$ grep -c hostname test  
4
```

- **Ejemplos de uso**

```
$ cat archivo_demo1
```

```
“ESTA ES LA PRIMER LINEA EN MAYUSCULAS DE ESTE ARCHIVO
esta es la primer línea en minúsculas de este archivo
Esta Es La Primer Línea Con El Primer Carácter De Cada Palabra Con Mayúscula
```

```
Hay dos líneas vacías sobre esta
Y esta es la última línea”
```

```
$ cp /tmp/archivo_demo1 /tmp/archivo_demo2
```

- **Uso básico, buscar una cadena en un archivo**

```
$ grep “esta” archivo_demo1
esta es la primer línea en minúsculas de este archivo
Hay dos líneas vacías sobre esta
Y esta es la última línea
```

- **Buscar una cadena en varios archivos**

```
$ grep “esta” archivo_demo*
archivo_demo1:esta es la primer línea en minúsculas de este archivo
archivo_demo1:Hay dos líneas vacías sobre esta
archivo_demo1:Y esta es la última línea
archivo_demo2:esta es la primer línea en minúsculas de este archivo
archivo_demo2:Hay dos líneas vacías sobre esta
archivo_demo2:Y esta es la última línea
```

- **Buscar sin coincidencia entre mayúsculas y minúsculas (uso del argumento -i)**

```
$ grep -i “esta” archivo_demo1
ESTA ES LA PRIMER LINEA EN MAYUSCULAS DE ESTE ARCHIVO
esta es la primer línea en minúsculas de este archivo
Esta Es La Primer Línea Con El Primer Carácter De Cada Palabra Con Mayúscula
Hay dos líneas vacías sobre esta
Y esta es la última línea
```



- **Buscar expresiones regulares**

```
$ grep "esta.*línea" archivo_demo1
esta es la primer línea en minúsculas de este archivo
Y esta es la última línea
```

En este ejemplo buscamos aquellas líneas que contienen la palabra “esta” y luego de cualquier texto (\*) continúan con “línea”. Para más operadores de expresiones regulares se recomienda leer la documentación del grep.

- **Buscar palabras completas (excluir subcadenas) (uso del argumento -w)**

```
$ grep -iw "de" archivo_demo1
```

- **Buscar N líneas luego de la coincidencia**

```
$ grep -A2 "ESTA" archivo_demo1
ESTA ES LA PRIMER LINEA EN MAYUSCULAS DE ESTE ARCHIVO
esta es la primer línea en minúsculas de este archivo
Esta Es La Primer Línea Con El Primer Carácter De Cada Palabra Con Mayúscula
```

```
$grep -A1 "ESTA" archivo_demo1
ESTA ES LA PRIMER LINEA EN MAYUSCULAS DE ESTE ARCHIVO
esta es la primer línea en minúsculas de este archivo
```

- **Buscar N líneas antes de la coincidencia**

```
$ grep -B1 "ultima" archivo_demo1
Hay dos líneas vacías sobre esta
Y esta es la última línea
```

- **Buscar N líneas alrededor de la coincidencia**

```
$ grep -C1 "minúsculas" archivo_demo1
ESTA ES LA PRIMER LINEA EN MAYUSCULAS DE ESTE ARCHIVO
esta es la primer línea en minúsculas de este archivo
Esta Es La Primer Línea Con El Primer Carácter De Cada Palabra Con Mayúscula
```

- **Resaltar los resultados**

Es necesario setear las variables de entorno GREP\_OPTIONS y GREP\_COLOR

```
$ export GREP_OPTIONS='--color=auto' GREP_COLOR='100;8'
[/CODE]
```

- **Buscar en archivos recursivamente (uso del argumento -r)**

Cuando necesitas buscar una cadena en los archivos del directorio actual y sus subdirectorios. El siguiente ejemplo buscaría la palabra “auto” en todos los archivos del directorio actual y sus subdirectorios.

```
$ grep -r “auto” *
```

- **Buscar no coincidencias (uso del argumento -v)**

```
$ grep -iv “primer” archivo_demo1
```

Hay dos líneas vacías sobre esta  
Y esta es la ultima línea

- **Mostrar las líneas de varias no coincidencias (uso del argumento -v y los argumentos -e)**

```
$ cat archivo_demo3
```

```
a  
b  
c  
d
```

```
$ grep -v -e “a” -e “b” -e “c” archivo_demo3  
d
```

- **Contar el número de ocurrencias (uso del argumento -c)**

```
$ grep -c “esta” archivo_demo1  
3
```

```
$ grep -ic “esta” archivo_demo1  
5
```

- **Mostrar solo los nombres de los archivos los cuales contienen coincidencias (uso del argumento -l)**

```
$ grep -l “esta” archivo_demo*  
archivo_demo1  
archivo_demo2
```

- **Mostrar solo la cadena buscada (uso del argumento -o)**

Por defecto el comando grep muestra la línea completa de la cadena o patrón dado, pero si solo quieres mostrar dicha cadena o patrón puedes utilizar el argumento -o.

```
$ grep "es.*línea" archivo_demo1
esta es la primer línea en minúsculas de este archivo
Y esta es la ultima línea
$
$ grep -o "es.*línea" archivo_demo1
esta es la primer línea
esta es la ultima línea
```

- **Mostrar la posición en el archivo de la coincidencia (uso de argumento -b)**

```
$ grep -bo "es.*línea" archivo_demo1
54:esta es la primer línea
222:esta es la ultima línea
$
$ grep -bo "es.*línea" archivo_demo*
archivo_demo1:54:esta es la primer línea
archivo_demo1:222:esta es la ultima línea
archivo_demo2:54:esta es la primer línea
archivo_demo2:222:esta es la ultima línea
```

- **Mostrar la fila en el archivo de la coincidencia (uso de argumento -n)**

```
$ grep -no "es.*línea" archivo_demo*
archivo_demo1:2:esta es la primer línea
archivo_demo1:7:esta es la ultima línea
archivo_demo2:2:esta es la primer línea
archivo_demo2:7:esta es la ultima línea
```

A partir del fichero (frutas.txt), el cuál contiene las siguientes líneas:

```
La Manzana es una fruta pomácea comestible obtenida del manzano doméstico
La ManZana es una de las frutas más cultivadas del mundo
La manzana puede comerse fresca pelada o con piel
MANZANA Cameo
Mango
Mamey
Güayaba
Pera
```

Se elaboró los siguientes **ejemplos**:

**1. Sensible a mayúsculas y minúsculas**

```
$ grep manzana frutas.txt
```

```
La manzana puede comerse fresca pelada o con piel
```

**2. Insensible a mayúsculas y minúsculas**

```
$ grep -i manzana frutas.txt
```

```
La Manzana es una fruta pomácea comestible obtenida del manzano doméstico
```

```
La ManZana es una de las frutas más cultivadas del mundo
```

```
La manzana puede comerse fresca pelada o con piel
```

```
MANZANA Cameo
```

**3. Contar el número de coincidencias (Sensible a mayúsculas y minúsculas)**

```
$ grep -c manzana frutas.txt
```

```
1
```

**4. Contar el número de coincidencias (Insensible a mayúsculas y minúsculas)**

```
$ grep -ci manzana frutas.txt
```

```
4
```

**5. Prefijar número de línea**

```
$ grep -n manzana frutas.txt
```

```
3:La manzana puede comerse fresca pelada o con piel
```

**6. Sólo la parte que coincide (Sensible a mayúsculas y minúsculas)**

```
$ grep -o manzana frutas.txt
```

```
manzana
```

**7. Sólo la parte que coincide (Insensible a mayúsculas y minúsculas)**

```
$ grep -io manzana frutas.txt
```

```
Manzana
```

```
ManZana
```

```
manzana
```

```
MANZANA
```

**8. Encontrar las líneas que contienen manzana o Mamey**

```
$ grep -E 'manzana|Mamey' frutas.txt
```

```
La manzana puede comerse fresca pelada o con piel
```

```
Mamey
```

### 9. Invertir la búsqueda (Sensible a mayúsculas y minúsculas)

```
$ grep -v manzana frutas.txt
```

La Manzana es una fruta pomácea comestible obtenida del manzano doméstico

La ManZana es una de las frutas más cultivadas del mundo

MANZANA Cameo

Mango

Mamey

Güayaba

Pera

### 10. Invertir la búsqueda (Insensible a mayúsculas y minúsculas)

```
$ grep -iv manzana frutas.txt
```

Mango

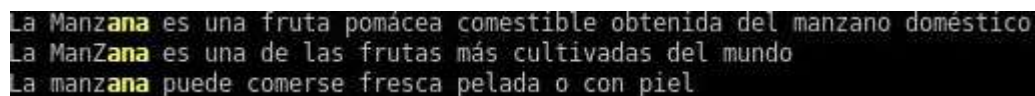
Mamey

Güayaba

Pera

### 11. Resaltar coincidencia

```
$ grep ana --color=always frutas.txt
```



The screenshot shows the output of the command 'grep ana --color=always frutas.txt'. It displays three lines from the file 'frutas.txt'. In each line, the word 'ana' (part of 'Manzana', 'ManZana', or 'manzana') is highlighted in yellow. The lines are: 'La Manzana es una fruta pomácea comestible obtenida del manzano doméstico', 'La ManZana es una de las frutas más cultivadas del mundo', and 'La manzana puede comerse fresca pelada o con piel'.

El color de resaltado puede cambiarse estableciendo la variable de entorno GREP\_COLOR, ejemplo export GREP\_COLOR='1;32'

### 12. Búsqueda recursiva

```
$ grep -lR estudiante proyectos/
```

El ejemplo anterior realiza una búsqueda recursiva dentro del DIR proyectos y encuentra/lista todos los todos los ficheros que contienen la palabra estudiante