

[geekland < https://geekland.eu/>](https://geekland.eu/) — Blog de Tecnología



Uso del comando grep en Linux y UNIX con ejemplos

 [Joan Carles < https://geekland.eu/author/admin/>](https://geekland.eu/author/admin/)

 [3 julio, 2022 < https://geekland.eu/uso-del-comando-grep-en-linux-y-unix-con-ejemplos/>](https://geekland.eu/uso-del-comando-grep-en-linux-y-unix-con-ejemplos/)

 [Deja un comentario < https://geekland.eu/uso-del-comando-grep-en-linux-y-unix-con-ejemplos/#respond>](https://geekland.eu/uso-del-comando-grep-en-linux-y-unix-con-ejemplos/#respond)

[Home < https://geekland.eu/>](https://geekland.eu/) » [Linux < https://geekland.eu/category/linux-2/>](https://geekland.eu/category/linux-2/) » Uso con ejemplos del comando grep

A continuación mediante ejemplos y explicaciones aprenderemos a usar el comando grep para poder filtrar texto y de esta forma poder programar scripts de una forma mucho más rápida y sencilla.

Nota: Este artículo es complementario de otros que escrito en este blog. Les

recomiendo que consulten los siguientes artículos para aprender a usar [awk < https://geekland.eu/uso-del-comando-awk-en-linux-y-unix-con-ejemplos/>](https://geekland.eu/uso-del-comando-awk-en-linux-y-unix-con-ejemplos/), [sed < https://geekland.eu/uso-del-comando-sed-en-linux-y-unix-con-ejemplos/>](https://geekland.eu/uso-del-comando-sed-en-linux-y-unix-con-ejemplos/), [cut < https://geekland.eu/uso-del-comando-cut-en-linux-y-unix-con-ejemplos/>](https://geekland.eu/uso-del-comando-cut-en-linux-y-unix-con-ejemplos/) y [tr < https://geekland.eu/uso-del-comando-tr-en-linux-y-unix-con-ejemplos/>](https://geekland.eu/uso-del-comando-tr-en-linux-y-unix-con-ejemplos/).

QUE USOS PODEMOS DAR AL COMANDO GREP

El comando grep nos permite buscar cadenas de texto y palabras dentro de un fichero de texto o de la entrada estándar de la terminal. Una vez encontrado el contenido que estamos buscando:

1. grep mostrará en pantalla la totalidad de la línea/s que contiene/n la cadena de texto o palabra que estamos buscando.
2. Con la opción pertinente únicamente mostrará la cadena de texto o palabra de cada una de las líneas que coincide con nuestro criterio de búsqueda.

Esto hará que podamos usar grep para conseguir los siguientes propósitos:

1. Hacer que únicamente se muestren las líneas descomentadas en un fichero de configuración.
2. En función del contenido de un fichero determinar la acción a realizar.
3. Extraer contenido de un fichero como por ejemplo direcciones IP.
4. Contar el número de líneas en las que aparece una palabra o cadena de texto.
5. Buscar de forma recursiva el nombre de los ficheros que contienen una determinada palabra o cadena de texto.
6. Etc.

SINTAXIS BÁSICA DEL COMANDO GREP

Para usar grep tendremos que usar un comando del siguiente tipo:

```
grep opcion/es 'cadena_de_texto' fichero_donde_buscar
```

Un ejemplo básico de uso del comando grep sería el siguiente:

```
> grep -w 'geekland' geekland.txt
```

Nota: Este comando se limita a mostrar la totalidad de líneas del fichero `geekland.txt` que contienen la palabra `geekland`

Opciones disponibles para el comando grep

Algunas de las opciones disponibles y que podemos usar para el comando grep son las siguientes:

Opción grep	Función
-F	Para indicar que queremos buscar una cadena de texto.
-w	Opción usada para encontrar una palabra.
-E	Sirve para indicar que queremos definir una expresión regular de búsqueda.
--color	Resalta en color las cadenas de texto que coinciden con nuestra búsqueda.
-H	Muestra el nombre del fichero en el caso que busquemos una palabra o cadena de texto en un solo fichero de texto.
-h	Oculto el nombre de los ficheros en el caso que busquemos una palabra o cadena de texto en varios ficheros de texto.
-n	Ver el número de línea/s en el que se encuentra la palabra o cadena de texto buscada.
-v	Tiene la utilidad de invertir. Por lo tanto esta opción permite mostrar las líneas que no coinciden con una determinada cadena de texto o palabra.
-i	Para no distinguir entre mayúsculas y minúsculas.
-o	Sirve para que la salida del comando <code>grep</code> únicamente muestre la cadena de texto o palabra de cada una de las líneas que coincide con nuestro criterio de búsqueda.
-x	Muestra las líneas de un fichero de texto o de la entrada estándar en que la palabra o cadena de texto que estamos buscando ocupa toda una línea completa.
-r	Leer de forma recursiva la totalidad de ficheros que están dentro de un directorio.
-R	Leer de forma recursiva la totalidad de ficheros que están dentro de un directorio. Esta opción también incluirá los enlaces simbólicos <code>< ht</code>

Opción grep	Función
	tps://geekland.eu/que-son-para-que-sirven-enlaces-duros-y-simbolicos/ .
-c	Nos permite obtener los números de líneas que contienen la cadena de texto o palabra que estamos buscando.
-A número	Muestra la línea que contiene la palabra o cadena de texto que estamos buscando. Además también muestra las x líneas anteriores a la línea que contiene la palabra o cadena de texto que estamos buscando.
-B número	Muestra la línea que contiene la palabra o cadena de texto que estamos buscando. Además también muestra las x líneas posteriores a la línea que contiene la palabra o cadena de texto que estamos buscando.
-C número	Muestra la línea que contiene la palabra o cadena de texto que estamos buscando. Además también muestra las x líneas anteriores y posteriores a la línea que contiene la palabra o cadena de texto que estamos buscando.
...	...

A continuación pasaremos a ver una serie de ejemplos para que puedan comprender a la perfección el uso del comando grep

EJEMPLOS DE BÚSQUEDA DE PALABRAS Y CADENAS DE TEXTO USANDO GREP

Algunos ejemplos que les deberían ayudar a dominar el uso del comando grep son los siguientes.

Mostrar todas las líneas que contengan una determinada cadena de caracteres con grep

Imaginemos el caso en que queremos imprimir la totalidad de líneas de nuestro

fichero `.bashrc` que contienen la cadena de caracteres `the`. Para realizar lo que acabo de comentar usaremos el siguiente comando:

```
> grep -F 'the' .bashrc
```

Nota: El parámetro `-F` es para indicar que `the` es una cadena de caracteres. En caso de no indicar `-F`, `the` se tomaría como una expresión regular.

Una vez ejecutado el comando obtendrán el siguiente resultado:

```
# Path to the bash it configuration
# Lock and Load a custom theme file.
# Leave empty to disable theming.
# location ~/.bash_it/themes/
# Some themes can show whether `sudo` has a current token or not.
...
```

Como pueden ver, la salida del comando muestra la totalidad de líneas que contienen la cadena de texto `the`. Por lo tanto se imprimen la totalidad de líneas que contienen palabras como `the`, `themes`, `theming`, etc.

Resaltar la sintaxis de las partes coincidentes con nuestra búsqueda mediante la opción `--color`

Si queremos que grep resalte las coincidencias de nuestra búsqueda tenemos que usar la opción `--color`. Por lo tanto si en el comando anterior añadimos la opción `--color`

```
> grep --color -F 'the' .bashrc
```

Obtendremos el siguiente resultado:

```
# Path to the bash it configuration
# Lock and Load a custom theme file.
# Leave empty to disable theming.
# location /.bash_it/themes/
# Some themes can show whether `sudo` has a current token or not.
...
```

Si observamos el resultado vemos las mismas frases que en el apartado anterior, pero ahora se resalta en color la cadena de texto `the` en cada una de las líneas. Esto es bastante útil porque nos ayuda a entender el porque se muestra cada una de las líneas.

Mostrar el nombre del fichero en que encontramos la coincidencia con la opción `-H` y `grep`

Si al inicio de la salida del comando quieren añadir el nombre del fichero en que se halla la línea mostrada en pantalla deberán usar la opción `-H` del siguiente modo:

```
> grep -H -F 'the' .bashrc
```

Y ahora el resultado obtenido será el siguiente:

```
.bashrc:# Path to the bash it configuration
.bashrc:# Lock and Load a custom theme file.
.bashrc:# Leave empty to disable theming.
.bashrc:# location /.bash_it/themes/
.bashrc:# Some themes can show whether `sudo` has a current token or not.
...
```

Por lo tanto obtenemos la totalidad de líneas del fichero `.bashrc` en que aparece la cadena de texto `the`. Además al inicio de cada una de las líneas podemos ver detallado el nombre del fichero en que aparece la línea.

Nota: Esta opción solo es necesaria en el caso que busquemos una palabra o cadena de texto en un solo fichero de texto. Si buscamos en varios ficheros la opción por defecto es mostrar los nombres de los ficheros.

Mostrar todas las líneas que no contengan una determinada cadena de caracteres usando la opción **-v** de grep

Si queremos mostrar en pantalla la totalidad de líneas del fichero `.bashrc` que no contienen la cadena de caracteres `the` tan solo tenemos que invertir la búsqueda realizada en el primer apartado de este artículo. Para invertir las búsqueda en grep deberemos usar la opción **-v**. Por lo tanto ejecutaremos el siguiente comando:

```
> grep -v -F 'the' .bashrc
```

Y el resultado será el siguiente:

```
#!/usr/bin/env bash

# If not running interactively, don't do anything
case $- in
    *i*) ;;
    *) return;;
esac

...

#bash-it enable plugin alias-completion
```

Mostrar todas las líneas que contengan una palabra determinada mediante la opción **-w**

Si queremos mostrar las líneas del fichero de configuración `.bashrc` que contengan la palabra `the` tendremos que usar la opción **-w** del siguiente modo:


```
> grep -w -F 'the' .bashrc
```

Y el resultado obtenido será el siguiente:

```
# Path to the bash it configuration
# (Advanced): Change this to the name of your remote repo if you
...
```

Ahora los resultados mostrados son mucho menores que en el primer ejemplo de este artículo. La razón es que solo se mostrarán las líneas que contienen la palabra específica `the`. Por lo tanto se excluirán las líneas que contienen palabras como `themes`, `theming`, etc.

Mostrar las líneas que contienen una palabra u otra palabra usando expresiones regulares gracias a la opción **-E**

Imaginemos que tenemos un fichero con el nombre `color.txt`. El contenido de este fichero es el siguiente:

```
color 1
colour 2
final
```

Si queremos mostrar únicamente las líneas que contienen la palabra `color` o `colour` lo podemos hacer mediante el uso de una de estas 2 expresiones regulares:

```
'colou?r'
'color|colour'
```

Nota: El símbolo `u?` significa que considere 2 casos. El primer caso que esté la letra `u` y el segundo caso que no esté la letra `u`.

Nota: El símbolo `|` significa o.

Si usamos grep con la primera de las expresiones regulares para mostrar la líneas del fichero `color.txt` que contienen la palabra `color` o `colour` lo haremos del siguiente modo obteniendo el siguiente resultado:

```
> grep -E -w 'colou?r' color.txt
color 1
colour 2
```

Si prefieren usar la segunda de las expresiones regulares obtendrán el mismo resultado.

```
> grep -E -w 'color|colour' color.txt
color 1
colour 2
```

Buscar todas las líneas que contengan una determinada palabra en la totalidad de ficheros que están dentro de un directorio con grep

En los comandos ejecutados hasta el momento estamos buscando si existe una palabra o una determinada cadena de texto en un solo fichero. Si en vez de analizar o mirar en un solo fichero queremos analizar o mirar en la totalidad de ficheros de un directorio deberemos proceder del siguiente modo.

En la ubicación `/home/joan/.config/i3` tengo la totalidad de ficheros de configuración de mi entorno de escritorio i3.

```
> ls
config i3blocks.conf scripts
```

Nota: `config` y `i3blocks.conf` son ficheros de texto. `scripts` se trata de un directorio.

Si quiero imprimir la totalidad de las líneas de los ficheros `config` y `i3blocks.conf` que contienen la palabra `i3` ejecutaremos el siguiente comando

dentro de la ubicación /home/joan/.config/i3:

```
> grep -w 'i3' ./*
```

Nota: También podría haber usado el comando `grep -w i3 /home/joan/.config/i3/*`.

Nota: Si ejecutáis el comando `grep -w i3 /home/joan/.config/i3` os dará error porque **grep no puede buscar en directorios**.

El resultado obtenido en mi caso es el siguiente:

```
./config:#          v1.0: 01/02/2021 - Primera configuración i3
./config:# Establemos que la tecla principal para trabajar con i3 sea la de Windows
./config:#bindsym $mod+Return exec i3-sensible-terminal
...
./i3blocks.conf:command=perl ~/.config/i3/scripts/mediaplayer
./i3blocks.conf:#command=/home/joan/.config/i3/scripts/temperature
./i3blocks.conf:command=~/.config/i3/scripts/bandwidth
...
grep: ./scripts: Es un directorio
```

En la salida veréis que aparecen la totalidad de las líneas del fichero `config` y `i3blocks.conf` que contienen la palabra `i3`. Además se indica el nombre del fichero en que está presente cada una de las líneas encontradas. Pero el comando no mirará en los ficheros que están dentro del directorio `scripts`. Si se quieren analizar la totalidad de ficheros del directorio `/home/joan/.config/i3` que contienen la palabra `i3` de forma recursiva y por lo tanto incluyendo el directorio `scripts`, lo tendrán que realizar del siguiente modo.

Buscar todas las líneas que contengan una determinada palabra en la totalidad de ficheros que están dentro de un directorio de forma recursiva con la opción `-r`

Para ver la totalidad de líneas de los ficheros que contienen la palabra `i3` y que están dentro del directorio y los subdirectorios de `/home/joan/.config/i3` tendremos que añadir la opción `-r` de búsqueda recursiva. Por lo tanto usaremos el siguiente comando para conseguir nuestro fin:

```
> grep -rw 'i3' /home/joan/.config/i3*
```

Nota: Como usamos la opción `-r` minúscula no se considerarán los enlaces simbólicos. En el caso que quisieran considerar los enlaces simbólicos deberían usar la opción `-R`.

y el resultado obtenido será el siguiente:

```
./config:#          v1.0: 01/02/2021 - Primera configuración i3
./config:# Establemos que la tecla principal para trabajar con i3 sea la de Windows
./config:#bindsym $mod+Return exec i3-sensible-terminal
...
./i3blocks.conf:command=perl ~/.config/i3/scripts/mediaplayer
./i3blocks.conf:#command=/home/joan/.config/i3/scripts/temperature
./i3blocks.conf:command=~/.config/i3/scripts/bandwidth
...
./scripts/calendar.sh:      i3-msg -q "exec yad --calendar \
```

En este caso podemos ver que los resultados de búsqueda son prácticamente iguales que en el apartado anterior. La única diferencia es que los resultados ahora mostrarán la totalidad de líneas de los ficheros de texto ubicados en `/home/.config/i3/scripts/` que contienen la palabra `i3`.

También observamos que al analizar varios ficheros de forma simultanea `grep` muestra el nombre del fichero en que se ha encontrado la línea que estamos buscando. Si queremos ocultar el nombre del fichero podemos usar la opción `-h` del siguiente modo:

```
> grep -h -rw 'i3' ./*
```

y el resultado que obtendremos será:

```
#          v1.0: 01/02/2021 - Primera configuración i3
# Establemos que la tecla principal para trabajar con i3 sea la de Windows
#bindsym $mod+Return exec i3-sensible-terminal
...
command=perl ~/.config/i3/scripts/mediaplayer
#command=/home/joan/.config/i3/scripts/temperature
command=~/.config/i3/scripts/bandwidth
...
i3-msg -q "exec yad --calendar \
```

Mostrar las líneas que contienen una determinada palabra o cadena de texto en los ficheros de un directorio de forma recursiva excluyendo ciertos directorios

Si queremos analizar la totalidad de líneas de los ficheros que están dentro del directorio y los subdirectorios de `/home/joan/.config/i3` excluyendo uno de los directorios, como por ejemplo el directorio `scripts`, deberemos usar la opción `--exclude-dir`. Por lo tanto para mostrar la totalidad de líneas de los ficheros contenidos en el directorio `/home/joan/.config/i3` que contienen la palabra `i3` excluyendo el directorio `scripts` procederemos del siguiente modo:

```
> grep -rw 'i3' /home/joan/.config/i3/* --exclude-dir=scripts
```

El resultado obtenido en mi caso es el siguiente:

```
./config:#          v1.0: 01/02/2021 - Primera configuración i3
./config:# Establemos que la tecla principal para trabajar con i3 sea la de Windows
./config:#bindsym $mod+Return exec i3-sensible-terminal
...
./i3blocks.conf:command=perl ~/.config/i3/scripts/mediaplayer
./i3blocks.conf:#command=/home/joan/.config/i3/scripts/temperature
./i3blocks.conf:command=~/.config/i3/scripts/bandwidth
...
grep: ./scripts: Es un directorio
```

En el caso que tuviéramos que excluir más de un directorio usaríamos la siguiente sintaxis:

```
> grep -rw 'i3' /home/joan/.config/i3/* --exclude-dir={directorio_1,directorio_2}
```

Obtener todas las líneas de un fichero que al menos contienen una de las letras definidas en una lista mediante grep

Para obtener la totalidad de líneas del fichero `.bashrc` que contienen las vocales `a`, o `i` lo haremos definiendo una búsqueda mediante expresiones regulares. El modo de realizarlo es el siguiente:

```
> grep -E '[ai]' ~/.bashrc
```

Nota: La opción `-E` es para indicar que el término de búsqueda se trata de una expresión regular. En este caso la expresión regular es `[ai]`.

y el resultado obtenido:

```
#!/usr/bin/env bash

# If not running interactively, don't do anything
case $- in
    *i*) ;;
    *) ;;
esac

...

# añadir rutas al PATH
export PATH="$HOME/.local/bin:$PATH"

#bash-it enable plugin alias-completion
```

Si quisiéramos imprimir la totalidad de líneas que contienen las vocales **a**, **o**, **e**, **o**, **i**, **u** entonces usaríamos el siguiente comando:

```
> grep -E '[aeiou]' ~/.bashrc
```

Nota: Para el último apartado también podríamos usar el comando `grep -E '[a-u]' ~/.bashrc`

Mostrar las líneas que contienen alguna letra que esté entre la **c** y la **f**

Si queremos mostrar la totalidad de líneas del fichero `.bashrc` que contienen una letra que esté entre la **c** y la **f** lo haremos del siguiente modo:

```
> grep -E '[c-f]' ~/.bashrc
```

Después de ejecutar el comando he obtenido el siguiente resultado:

```
#!/usr/bin/env bash

# If not running interactively, don't do anything
case $- in
    *) return;;
esac

...

source "$BASH_IT"/bash_it.sh

#alias ls='exa'

# añadir rutas al PATH
export PATH="$HOME/.local/bin:$PATH"

#bash-it enable plugin alias-completion
```

Ver la totalidad de líneas de un fichero de texto que empiezan por fo

Como hemos visto anteriormente, grep permite buscar líneas que contienen palabras o cadenas de texto determinadas mediante el uso de expresiones regulares. Por lo tanto si queremos buscar la totalidad de líneas de nuestro fichero de configuración `/home/joan/.config/i3/config` que empiezan por las letras fo usaremos el siguiente comando:

```
> grep -E '^fo' --color /home/joan/.config/i3/config
```

Nota: El símbolo ^ es una expresión regular cuyo significado es inicio.

Una vez ejecutado el comando el resultado obtenido es el siguiente:

```
font pango:System San Francisco Display 10
for_window [class="^.*"] border pixel 1
for_window [window_role="Msgcompose" class="(?!Thunderbird)] floating enable
```

Mostrar las líneas de un fichero que inician por fo añadiendo las 2 líneas posteriores al encuentro

Si aparte de mostrar las líneas que empiezan por las letras `fo` queremos añadir las 2 líneas posteriores a cada una de las líneas que empieza por las letras `fo` deberemos añadir la opción `-A 2` del siguiente modo:

```
> grep -E -A 2 '^fo' --color /home/joan/.config/i3/config
```

Ahora el resultado que obtendremos será el siguiente:

```
font pango:System San Francisco Display 10

# # Al tener como mínimo 2 ventanas abiertas presionamos la tecla Windows, Posicion
--
for_window [class="^.*"] border pixel 1

# Seleccionar las aplicaciones que arrancar al iniciar i3
--
for_window [window_role="Msgcompose" class="(?!Thunderbird)] floating enable
n
# Fijamos que la tecla imprimir pantalla sea encargada de realizar capturas de pant
```

Nota: Si observan la salida del comando verán que `--` hace de línea divisoria para cada una de las líneas que `grep` encuentra.

Hacer que grep no discierna entre mayúsculas y minúsculas con la opción `-i`

Si seguimos con el ejemplo del apartado anterior y queremos buscar todas las líneas del fichero de configuración `/home/joan/.config/i3/config` que empiezan por `Fo` ejecutaremos el siguiente comando:

```
> grep -E -A 2 '^Fo' --color /home/joan/.config/i3/config
```

La salida de ejecutar este comando será nula. En el fichero `/home/joan/.config/i3/config` hay líneas que empiezan por `fo`, pero no hay líneas que empiecen por

Fo. Para hacer que grep no diferencie mayúsculas de minúsculas usaremos la opción **-i** del siguiente modo:

```
> grep -E -A 2 -i '^Fo' --color /home/joan/.config/i3/config
```

Ahora la salida del comando mostrará la totalidad de líneas que empiezan por fo, Fo, f0 y F0.

```
font pango:System San Francisco Display 10
```

```
# # Al tener como mínimo 2 ventanas abiertas presionamos la tecla Windows, Posicion
```

```
--
```

```
for_window [class="^.*"] border pixel 1
```

```
# Seleccionar las aplicaciones que arrancar al iniciar i3
```

```
--
```

```
for_window [window_role="Msgcompose" class="(?!i)Thunderbird"] floating enable
```

```
# Fijamos que la tecla imprimir pantalla sea encargada de realizar capturas de pant
```

Encontrar las líneas de un fichero que inician por fo, Fo, f0 y F0 añadiendo las 2 líneas anteriores a la coincidencia mediante grep

Si en vez de mostrar las 2 líneas posteriores a una coincidencia queréis mostrar las 2 líneas anteriores deberéis reemplazar la opción **-A 2** por **-B 2** Por lo tanto si ejecutamos el comando del apartado anterior reemplazando **-A 2** por **-B 2**:

```
> grep -E -B 2 -i '^Fo' --color /home/joan/.config/i3/config
```

Obtendremos el siguiente resultado:

```
# Establecemos que la fuente del sistema sea la System San Francisco y que el tamaño
font pango:System San Francisco Display 10
--

# Quita la ventana de título y se define el grosor del borde que recubre las ventanas
for_window [class="^.*"] border pixel 1
--

# Hacer que la ventana de redacción de un email en thunderbird se abra en modo flotante
for_window [window_role="Msgcompose" class="(?!Thunderbird)] floating enable
```

Ver la totalidad de líneas de un fichero que inicien por **fo** y además ver las 2 líneas anteriores y posteriores

Si ahora queremos que se muestren:

- La totalidad de líneas del fichero de texto `/home/joan/.config/i3/config` que empiezan por **fo**
- Las **2** líneas anteriores a cada una de las líneas que empiezan por **fo**.
- Las **2** líneas posteriores a cada una de las líneas que empiezan por **fo**.

Tenemos que usar la opción **-C 2**. Por lo tanto para conseguir nuestro propósito en el fichero `/home/joan/.config/i3/config` ejecutaremos el siguiente comando:

```
> grep -E -C 2 '^fo' --color /home/joan/.config/i3/config
```

Y el resultado obtenido es el que muestro a continuación:

```
# Establecemos que la fuente del sistema sea la System San Francisco y que el tamaño
font pango:System San Francisco Display 10

# # Al tener como mínimo 2 ventanas abiertas presionamos la tecla Windows, Posicion
--

# Quita la ventana de título y se define el grosor del borde que recubre las ventan
for_window [class="^.*"] border pixel 1

# Seleccionar las aplicaciones que arrancar al iniciar i3
--

# Hacer que la ventana de redacción de un email en thunderbird se abra en modo flot
for_window [window_role="Msgcompose" class="(?!Thunderbird)] floating enable

# Fijamos que la tecla imprimir pantalla sea encargada de realizar capturas de pant
```

Mostrar los nombres de los ficheros que contienen una determinada palabra

Si nuestro propósito es obtener los nombres de los ficheros que contienen una palabra determinada lo podemos hacer del siguiente modo. Anteriormente vimos que ejecutando el siguiente comando:

```
> grep -H -w 'i3' ~/.config/i3/*
```

Obtenemos la totalidad de líneas de los ficheros ubicados en `~/.config/i3/` que contienen la palabra `i3` conjuntamente con el nombre del fichero en que están cada una de las líneas:

```
/home/joan/.config/i3/config:#          v1.0: 01/02/2021 - Primera configuración i3
/home/joan/.config/i3/config:# Establecemos que la tecla principal para trabajar co
...
/home/joan/.config/i3/i3blocks.conf:#command=/home/joan/.config/i3/scripts/rofi_cal
/home/joan/.config/i3/i3blocks.conf:# Add the following bindings to i3 config file:
grep: /home/joan/.config/i3/scripts: Es un directorio
```

Si además queremos limpiar y facilitar la lectura de los resultados usaremos el siguiente comando para conseguir nuestro objetivo final:

```
> grep -H -w i3 ~/.config/i3/* | cut -d ':' -f1 | uniq
```

Una vez ejecutado el comando el resultado es el siguiente:

```
/home/joan/.config/i3/config
/home/joan/.config/i3/i3blocks.conf
```

Por lo tanto tanto el fichero `config` como el fichero `i3blocks.conf` contienen líneas con la palabra `i3`.

Conocer los números de línea en que aparece una palabra determinada con la opción `-n`

En mi caso quiero conocer los números de línea del fichero `/home/joan/.config/i3/config` en que aparece la palabra `windows`. Para ello usaremos la opción `-n` del siguiente modo:

```
> grep -w -i -n 'windows' ~/.config/i3/config
```

Y el resultado obtenido es el siguiente:

```
7:# Establecemos que la tecla principal para trabajar con i3 sea la de Windows
13:# # Al tener como mínimo 2 ventanas abiertas presionamos la tecla Windows, Posic
16:# El atajo de teclado para abrir una terminal será windows + Enter. Aquí también
20:# La combinación de teclas para cerrar una ventana activa es windows+shift+q
28:# Combinación de teclas a usar para movernos entre ventanas. En mi caso uso la t
63:# change focus between tiling / floating windows
```

Por lo tanto ahora sabemos que la palabra `windows` aparece en las líneas 7, 13, 16, 20, 28 y 63.

Si únicamente quisiéramos obtener los números podríamos haber usado el siguiente comando:

```
grep -w -i -n 'windows' ~/.config/i3/config | cut -d ':' -f1
```

Conocer el número de líneas de un fichero que contienen la palabra `bindsym` mediante la opción `-c`

Toda línea del fichero de configuración del escritorio i3 que contiene la palabra `bindsym` está configurando un atajo de teclado. Por lo tanto si mi fichero de configuración es `~/.config/i3/config` y quiero saber el número de atajos de teclado que tengo configurados tan solo tendré que contar el número de líneas en que aparece la palabra `bindsym`. Para ello haremos uso de la opción `-c` del siguiente modo:

```
> grep -F -c 'bindsym' ~/.config/i3/config
```

La salida del comando es la siguiente:

```
114
```

Por lo tanto tengo configurados **114** atajos de teclado en mi fichero de configuración.

Mostrar las líneas descomentadas de un fichero de texto usando grep

Para mostrar las líneas descomentadas del fichero `.bashrc` tendremos que hacer uso de las expresiones regulares y de la opción `-v` para invertir los resultados de la salida. Concretamente deberemos usar el siguiente comando:

```
> grep -v -E '^(\#|$)' .bashrc
```

Nota: La expresión regular `^(\#|$)` significa todas las líneas que **inician** con un **símbolo de comentario** o con un **salto de línea**. Por lo tanto el comando ejecutado mostrará la totalidad de líneas del fichero `.bashrc` que no empiecen por el símbolo `#` y que no sean un salto de línea.

De esta forma obtendremos que el siguiente resultado:

```
case $- in
    *i*) ;;
    *) return;;
esac

export BASH_IT="/home/joan/.bash_it"
export BASH_IT_THEME='powerline-multiline'
export GIT_HOSTING='git@git.domain.com'
unset MAILCHECK
export IRC_CLIENT='irssi'
export TODO="t"
export SCM_CHECK=true
source "$BASH_IT"/bash_it.sh
export PATH="$HOME/.local/bin:$PATH"
```

Procediendo de esta forma podremos ver fácilmente la configuración que tenemos aplicada sin que nos molesten las opciones de configuración comentadas.

Mostrar únicamente las líneas descomentadas de un fichero de texto en el caso que existan comentarios anidados

El comando del apartado anterior no funcionará en el caso que el fichero de configuración tenga comentarios anidados mediante espacios o tabulaciones.

Por lo tanto si tenemos un fichero de texto con el nombre `comentario.txt` y contiene el siguiente contenido:

```
# Comentario 1
    #Comentario2
# Comentario 3
Esto es un ejemplo del comando grep en el blog de Geekland
```

El comando del apartado anterior no funcionará porque hay líneas que empiezan por espacios y/o tabulaciones. Por lo tanto para conseguir nuestro objetivo tendremos que redefinir la expresión regular del siguiente modo:

```
> grep -v -E '^ *# | ^\t*#' comentario.txt
```

Nota: La expresión regular `^ *#` define líneas que empiezan por 0 o varios espacios y además después de los 0 o más espacios tengan el símbolo de comentario `#`.

Nota: El símbolo `|` significa o.

Nota: La expresión regular `^\t*#` define líneas que empiezan por 0 o varias tabulaciones y además después de las 0 o más tabulaciones tengan el símbolo de comentario `#`

El resultado obtenido en aplicar el comando será el siguiente:

```
Esto es un ejemplo del comando grep en el blog de Geekland
```


Por lo tanto hemos conseguido eliminar la totalidad de comentarios anidados.

En el caso de eliminar la opción `-v` del comando que acabamos de ejecutar:

```
> grep -E '^ *# | ^\t*#' comentario.txt
```

Invertiríamos los resultados de salida y veríamos únicamente las líneas que contienen comentarios:

```
# Comentario 1
    #Comentario2
# Comentario 3
```

Mostrar la totalidad de líneas de un fichero de texto que finalizan con un dígito o carácter determinado

Para obtener la totalidad de líneas del fichero `/usr/share/dict/words` que terminan por una letra entre la `b` y la `c` lo haremos del siguiente modo:

```
> grep -E '[b-c]$' /usr/share/dict/words
```

Nota: El símbolo `$` es una expresión regular cuyo significado es final de la línea. Por lo tanto `[b-c]$` significa toda línea que termine por las letras `b` o `c`

Después de ejecutar el comando obtendremos el siguiente resultado:

Adriatic

Aelfric

Ahab

Alaric

...

web

womb

zinc

zodiac

Buscar las líneas en que la primera palabra empiece por un determinado carácter y cuya última palabra termine por un carácter determinado

Para encontrar la totalidad de líneas del fichero `/usr/share/dict/words` que empiecen por `a` y terminen por una `a` o una `b` lo haremos usando el siguiente comando:

```
> grep -E '^a.*[a-b]$' /usr/share/dict/words
```

La expresión regular `^a.*[a-b]$` se puede desglosar del siguiente modo:

- `^a`: Toda línea que empiece por `a`.
- `.*`: Seguida por cualquier carácter sin importar el número de repeticiones.
- `[a-b]$`: Que termine por letras comprendidas entre la `a` y la `b`.

El resultado de aplicar el comando ha sido el siguiente:

abracadabra

abscissa

absorb

addenda

adverb

aha

alb

...

atria

auditoria

automata

azalea

Extraer líneas de un fichero cuya primera palabra de cada línea cumpla un determinado criterio mediante grep

Para obtener las palabras del fichero `/usr/share/dict/words` que cumplan los siguientes criterios:

- Que empiecen por la letra `m`
- Que después de la `m` haya `1` carácter cualquiera.
- Que después del carácter cualquiera haya la letra `d`.
- Que después de la letra `d` existan `3` caracteres sin importar los que son.
- Y que finalmente después de los `3` caracteres cualquiera termine la palabra y la línea con la letra `e`.

Tendremos que usar la expresión regular `^m.d...e$`. Por lo tanto el comando que tendremos que ejecutar es el siguiente:

```
> grep -E '^m.d...e$' /usr/share/dict/words
```

Y el resultado obtenido es el que muestro a continuación:

```
mediate
```

```
midwife
```

Extraer direcciones IPv4 de un fichero de texto mediante la opción **-o** y definiendo una expresión regular

Imaginemos que el fichero `/etc/hosts` tiene el siguiente contenido:

```
# Host addresses
127.0.0.1    localhost
127.0.1.1    gk55
::1          localhost ip6-localhost ip6-loopback
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
```

Para extraer únicamente las direcciones IPv4 del fichero tenemos que tener en cuenta lo siguiente:

1. Las direcciones IPv4 están compuestas por números entre el 0 y el 9.
2. Las direcciones IPv4 están compuestas por cuatro números separados por un punto.
3. Cada uno de los 4 números que compone la dirección IPv4 puede tener 1, 2 o 3 cifras.

Si tenemos en cuenta estos 3 factores la forma más fiable para extraer las direcciones IPv4 es usando la siguiente expresión regular:

```
[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}
```

Respecto a la expresión regular hay que tener en cuenta los siguientes aspectos:

`[0-9]{1,3}`: Hace referencia a que pueden existir 1, 2 o 3 números que estén entre el 0 y el 9.

`\.`: Los 1, 2 o 3 números que estarán entre el 0 y el 9 irán seguido/s de un punto. El símbolo `\` es simplemente el símbolo escape para evitar que se tome `.` como una expresión regular.

Una vez tenemos clara la expresión regular a usar ejecutaremos el siguiente comando:

```
> grep -E --color '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' /etc/hosts
```

Y obtendremos el siguiente resultado:

```
127.0.0.1 localhost
```

```
127.0.1.1 gk55
```

Si os fijáis en el resaltado de color se muestra el contenido definido por nuestra expresión regular. En nuestro caso la expresión regular define 4 cifras entre el 0 y el 9, separadas por puntos. Cada una de las cifras entre el 0 y el 9 está separada puntos y podrá tener 1, 2 o 3 caracteres.

Si queremos que la salida del comando solo muestre las IP omitiendo las palabras `localhost` y `gk55` podemos usar la opción `-o`. De este modo `grep` solo mostrará la cadena de texto o palabra de cada una de las líneas que coincide con nuestro criterio de búsqueda. Por lo tanto si ejecutamos el siguiente comando:

```
> grep -E --color -o '[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}' /etc/hosts
```

Obtendremos el siguiente resultado:

127.0.0.1

127.0.1.1

Aplicar una acción A o B en función del contenido de un fichero de texto

Para aplicar una acción u otra en función de una determinada circunstancia tendremos que usar un condicional if. Imaginemos que tenemos un fichero `opciones.txt` y tiene el siguiente contenido:

blanco

negro

azul

Ahora queremos que si una de las líneas del fichero contiene la palabra `blanco` la terminal nos de como resultado 1. En el caso que el fichero de texto no disponga de la palabra `blanco` entonces queremos que nos devuelva 0. Para conseguir nuestro propósito podemos usar el siguiente comando:

```
> if [[ $(grep -x "blanco" /home/joan/opciones.txt) ]]; then echo "1"; else echo "0"
```

y el resultado obtenido será:

1

Si ahora borramos la palabra `blanco` del fichero `opciones.txt` y volvemos a ejecutar el comando:

```
> if [[ $(grep -x "blanco" /home/joan/opciones.txt) ]]; then echo "1"; else echo "0"
```

El resultado será el siguiente:

0

Nótese que en este ejemplo usamos `grep` con la opción `-x`. La opción `-x` hace

que grep solo considere las líneas cuyo contenido completo sea la palabra blanco. Para entender lo que estoy diciendo basta con este simple ejemplo.

Imaginemos que el contenido del fichero `opciones.txt` es el siguiente:

```
blanco
blanco es el mejor color
```

Si ejecuto el siguiente comando:

```
> grep -w "blanco" /home/joan/opciones.txt
```

Vemos que se mostrarán la totalidad de palabras que contienen la palabra blanco.

```
blanco
blanco es el mejor color
```

Pero si queremos obtener únicamente aquellas líneas cuyo único contenido sea la palabra blanco añadiremos la opción `-x` del siguiente modo:

```
> grep -w -x "blanco" /home/joan/opciones.txt
```

Y ahora finalmente el resultado obtenido será el siguiente:

```
blanco
```

Consultar la ayuda de una opción

grep tiene más opciones de las mostradas a lo largo de este artículo. Para profundizar más sobre el uso del comando grep les recomiendo que abran una terminal y ejecutan el siguiente comando para obtener ayuda:

```
> man grep
```

Del mismo modo también pueden consultar la siguiente

URL < <https://www.cyberciti.biz/faq/howto-use-grep-command-in-linux-unix/>



< <https://www.addtoany.com/share#url=linux-y-unix-con-ejemplos%2F&title=Uso%20del%20coman>



Dejar un comentario

Tu dirección de correo electrónico no será publicada. Los campos obligatorios están marcados con *

Comentario *

Nombre *

Correo electrónico *

Web

☐ Guarda mi nombre, correo electrónico y web en este navegador para la próxima vez que comente.

Publicar el comentario

Este sitio usa Akismet para reducir el spam.

[Aprende cómo se procesan los datos de tus comentarios < https://akismet.com/privacy/>](https://akismet.com/privacy/)



2024 geekland , **Funciona gracias a WordPress.** < <https://es.wordpress.org/>>



Todos los derechos reserva-