Filtrado de entradas

El comando `grep` es un filtro de texto que busca líneas en una entrada y devolverá aquellas que coincidan con un patrón determinado.

```
grep [OPCIONES] PATRÓN [ARCHIVO]
```

Siga leyendo

Utilice el siguiente comando para cambiar al directorio `Documents` :

```
sysadmin@localhost:~$ cd ~/Documents
```

Si el siguiente ejemplo le devuelve un mensaje de error, repita el ejemplo de la *Sección 11*:

```
sysadmin@localhost:~/Documents$ cp /etc/passwd .
```

Por ejemplo, el archivo `passwd` que copiamos anteriormente al directorio `Documents` contiene los detalles de cuentas especiales del sistema y cuentas de usuarios en el sistema. Este archivo puede ser muy grande, sin embargo, el comando `grep` se puede utilizar para filtrar y obtener información sobre un usuario específico, como por ejemplo el usuario `sysadmin` . Utilice `sysadmin` como argumento de patrón y `passwd` como argumento de archivo:

```
sysadmin@localhost:~/Documents$ grep sysadmin passwd
sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin:/bin/bash
```

El comando anterior devolvió la línea del `passwd` que contiene el patrón `sysadmin` .

Nota

Esta línea es la entrada `/etc/passwd` que pertenece al usuario `sysadmin` y proporciona información que está más allá del alcance de este curso. Para obtener más información sobre este archivo, consulte NDG Linux Essentials.

El ejemplo anterior utiliza un término de búsqueda simple como patrón; sin embargo `grep` es capaz de interpretar patrones de búsqueda mucho más complejos.

Expresiones regulares

Las expresiones regulares tienen dos formas comunes: la forma básica y la forma extendida. La mayoría de los comandos que utilizan expresiones regulares pueden interpretar expresiones regulares básicas. Sin embargo, las expresiones regulares extendidas no están disponibles para todos los comandos y normalmente requieren una opción de comando para funcionar correctamente.

En la siguiente tabla se resumen los caracteres básicos de las expresiones regulares:

Caracteres Básicos Regex	Significado
`.`	Cualquier carácter individual
`[ ]`	Cualquier carácter especificado
`[^]`	Cualquier carácter que no es el carácter especificado
`*`	Cero o más del carácter anterior
`^`	Si es el primer carácter del patrón, el patrón deberá estar al principio de la línea para coincidir, si no es así se tratará como un `^` literal.
`$`	Si es el último carácter del patrón, el patrón deberá estar al final de la línea para coincidir, si no es así se tratará como un `$` literal.

En la siguiente tabla se resumen las expresiones regulares extendidas, que se deben utilizar con el comando `egrep` o la opción `-E` con el comando `grep`:

Caracteres Básicos Regex	Significado
+	Uno o más del patrón anterior
?	El patrón es opcional
{ }	Especificar mínimo, máximo, o coincidencias exactas en el patrón anterior
|	Alternancia - el “o” lógico
( )	Se usa para crear grupos

Patrones básicos

Las expresiones regulares son patrones que sólo ciertos comandos pueden interpretar. Las expresiones regulares se pueden expandir para que coincidan con ciertas secuencias de caracteres en el texto. Los ejemplos que se muestran en esta página harán uso de expresiones regulares para demostrar su potencia cuando se utilizan con el comando `grep`. Además, estos ejemplos proporcionan una demostración muy visual de cómo funcionan las expresiones regulares, el texto que resulta de estas expresiones se mostrará en color rojo en el resultado (*output*).

Siga leyendo

Utilice el siguiente comando `cd` para cambiar al directorio `Documents`.

```
sysadmin@localhost:~$ cd ~/Documents
```

La más simple de todas las expresiones regulares solo usa caracteres de significado literal, como el ejemplo de la página anterior:

```
sysadmin@localhost:~/Documents$ grep sysadmin passwd
sysadmin:x:1001:1001:System Administrator,,,,:/home/sysadmin:/bin/bash
```

Caracteres de anclaje

Los caracteres de anclaje son una de las formas con que se pueden utilizar expresiones regulares para limitar los resultados de una búsqueda. Por ejemplo, el patrón `root` aparece muchas veces en el archivo `/etc/passwd`:

```
sysadmin@localhost:~/Documents$ grep 'root' passwd
root:x:0:0:root:/root:/bin/bash
operator:x:1000:37::/root:
```

> Para evitar que el shell los malinterprete como caracteres especiales del shell, estos patrones deben escribirse protegidos por comillas sólidas, lo que simplemente significa colocarlos entre comillas.

El primer carácter de anclaje `^` se utiliza para indicar que el patrón debe aparecer al *principio* de la línea. Por ejemplo, para encontrar todas las líneas en `/etc/passwd` que *comienzan* con `root` use el patrón `^root`. Tenga en cuenta que `^` debe ser el *primer* carácter del patrón para ser efectivo.

```
sysadmin@localhost:~/Documents$ grep '^root' /etc/passwd
root:x:0:0:root:/root:/bin/bash
```

Para el siguiente ejemplo, examine primero el archivo `alpha-first.txt` . El comando `cat` se puede utilizar para mostrar el contenido del archivo:

```
sysadmin@localhost:~/Documents$ cat alpha-first.txt
A is for Animal
B is for Bear
C is for Cat
D is for Dog
E is for Elephant
F is for Flower
```

El segundo carácter de anclaje `$` se puede utilizar para indicar que el patrón debe aparecer al *final* de la línea, reduciendo eficazmente los resultados de la búsqueda. Para encontrar las líneas que terminan con una `r` en el archivo `alpha-first.txt` , utilice el patrón `r$` :

```
sysadmin@localhost:~/Documents$ grep 'r$' alpha-first.txt
B is for Bear
F is for Flower
```

Una vez más, la posición de este carácter es importante; el `$` debe ser el *último* carácter en el patrón para ser eficaz como anclaje.

Encontrar caracteres coincidentes usando .

Los siguientes ejemplos usan el archivo `red.txt` :

```
sysadmin@localhost:~/Documents$ cat red.txt
Red
Reef
Rot
Reeed
Rd
Rod
Roof
Reed
Root
reel
read
```

Una de las expresiones más útiles es el carácter `.` (punto). Representa cualquier carácter excepto el carácter de nueva línea. El patrón `r..f` encontrará cualquier línea que contenga la letra `r` seguida de exactamente dos caracteres (que pueden ser cualquier carácter excepto el de nueva línea) y luego la letra `f` :

```
sysadmin@localhost:~/Documents$ grep 'r..f' red.txt
reef
roof
```

El mismo concepto se puede repetir usando otras combinaciones. El comando en el ejemplo siguiente encontrará palabras de cuatro letras que comienzan con `r` y acaban con `d`:

```
sysadmin@localhost:~/Documents$ grep 'r..d' red.txt
reed
read
```

Este carácter se puede utilizar tantas veces como se desee. Para encontrar todas las palabras de al menos cuatro caracteres se puede utilizar el siguiente patrón:

```
sysadmin@localhost:~/Documents$ grep '....' red.txt
reef
reeed
roof
reed
root
reel
read
```

La línea no tiene que coincidir exactamente, simplemente debe contener el patrón, como se ve a continuación cuando se busca `r..t` en el archivo `/etc/passwd`:

```
sysadmin@localhost:~/Documents$ grep 'r..t' /etc/passwd
root:x:0:0:root:/root:/bin/bash
operator:x:1000:37::/root:
```

Encontrar un carácter único usando [ ]

Los corchetes `[ ]` se utilizan para indicar caracteres *únicos* o rangos de caracteres posibles en una lista.

Por ejemplo, dado el archivo `profile.txt` :

```
sysadmin@localhost:~/Documents$ cat profile.txt
Hello my name is Joe.
I am 37 years old.
3121991
My favorite food is avocados.
I have 2 dogs.
123456789101112
```

Para encontrar todas las líneas en el archivo `profile.txt` que contienen un número, utilice el patrón `[0123456789]` o `[0-9]` :

```
sysadmin@localhost:~/Documents$ grep '[0-9]' profile.txt
I am 37 years old.
3121991
I have 2 dogs.
123456789101112
```

Por otro lado, para encontrar todas las líneas que contienen caracteres no numéricos, inserte un `^` como primer carácter dentro de los corchetes. Este carácter *niega* los caracteres que lo siguen:

```
sysadmin@localhost:~/Documents$ grep '[^0-9]' profile.txt
Hello my name is Joe.
I am 37 years old.
My favorite food is avocados.
I have 2 dogs.
```

Nota

No confunda el resultado de `[^ 0-9]` con *las líneas que no contienen números*. En realidad, esta expresión se refiere a *líneas que contienen no-números* (caracteres no numéricos). Examine el archivo original para ver la diferencia. Las líneas tercera y sexta sólo contienen números, *no contienen caracteres no numéricos*, y por lo tanto no han sido incluidas en este último resultado.

Cuando otros caracteres de expresión regular se colocan dentro de corchetes, se tratan como caracteres literales. Por ejemplo, el carácter `.` normalmente indica cualquier carácter. Pero si se coloca dentro de corchetes simplemente se referirá al carácter . (punto). En el siguiente ejemplo, sólo las líneas que contienen el carácter `.` se mostrarán en el resultado.

```
sysadmin@localhost:~/Documents$ grep '[.]' profile.txt
Hello my name is Joe.
I am 37 years old.
My favorite food is avocados.
I have 2 dogs.
```

Indicar un carácter o patrón repetido utilizando el `*`

El carácter de expresión regular * se utiliza para indicar la ausencia o la presencia una o más veces del carácter o patrón que lo precede. Por ejemplo, e* indicaría la ausencia (cero) o la presencia, una o más veces, de la letra e:

```
sysadmin@localhost:~/Documents$ cat red.txt
red
reef
rot
reeed
rd
rod
roof
reed
root
reel
read
sysadmin@localhost:~/Documents$ grep 're*d' red.txt
red
reeed
rd
reed
```

También es posible indicar la ausencia o presencia una o más veces de una lista de caracteres utilizando los corchetes. El patrón `[oe]*` utilizado en el siguiente ejemplo se refiere a líneas con ausencia o presencia una o más veces del carácter `o` o del carácter `e`:

```
sysadmin@localhost:~/Documents$ grep 'r[oe]*d' red.txt
red
reeed
rd
rod
reed
```

Cuando se usa con un solo carácter, `*` no resulta muy útil. Note que cualquiera de los siguientes patrones coincide con cada una de las líneas (string) del archivo: `.*` `e*` `b*` `z*`.

```
sysadmin@localhost:~/Documents$ grep 'z*' red.txt
red
reef
rot
reeed
rd
rod
roof
reed
root
reel
read
```

```
sysadmin@localhost:~/Documents$ grep 'e*' red.txt
red
reef
rot
reeed
rd
rod
roof
reed
root
reel
read
```

Esto se debe a que `*` puede indicar *cero* ocurrencias de un patrón. Para que el `*` sea útil, es necesario crear un patrón que incluya algo más que el carácter que precede al `*`. Por ejemplo, los resultados anteriores se pueden refinar agregando otra `e` para hacer que el patrón `ee*` coincida efectivamente con cada línea que contenga al menos una `e`.

```
sysadmin@localhost:~/Documents$ grep 'ee*' red.txt
red
reef
reeed
reed
reel
read
```

Entrada estándar

Si no se proporciona un nombre de archivo, el comando `grep` actuará sobre la entrada estándar, que normalmente proviene del teclado y de la entrada proporcionada por el usuario que está ejecutando el comando. Esto permite que el uso de `grep` se convierta en una experiencia interactiva donde el usuario escribe y `grep` filtra la entrada a medida que avanza. Pruébelo. Simplemente presione **Ctrl+D** cuando esté listo para volver al intérprete de comandos (*prompt*).

```
sysadmin@localhost:~$ grep 'red'
The girl in the red dress had red hair and a matching red bow.
The girl in the red dress had red hair and a matching red bow.
The horse in the red saddle was bred for racing.
The horse in the red saddle was bred for racing.
sysadmin@localhost:~$
```

Siga leyendo

Utilice el siguiente comando `cd` para volver al directorio principal:

```
sysadmin@localhost:~/Documents$ cd ~
```