Visualización de ficheros: uso de expresiones regulares

Como ya se ha visto, la potencia de las redirecciones y tuberias nos permiten personalizar muchisimo la busqueda de contenido en fichero y su posterior tratamiento. Un aspecto fundamental en el uso de estos filtros son las expresiones regulares, que nos permiten establecer carácteres comodin para realizar filtros en las búsquedas de texto.

Las herramientas que suelen utilizarse para el uso de expresiones regulares son grep y sus derivadas.

Comando grep

Muchos programas soportan el uso de estos elementos y el comando grep es el más común para realizar búsquedas en textos mediante patrones. Algunos caracteres tienen un significado especial en expresiones regulares como se muestra a continuación:

- **^** Inicio de línea.
- **$** Final de línea.
- **. (punto)** Cualquier carácter
- ***** Cualquier secuencia de cero o más caracteres.
- **[ ]** Cualquier carácter que esté presente en los corchetes.

Un uso común del comando **grep** es mostrar el contenido de archivos de configuración ignorando únicamente las líneas que corresponden a los comentarios, es decir, las líneas que se inician con el carácter almohadilla [#]. Para hacer esto, es necesario añadir el parámetro **-v**que invierte el patrón de busqueda, seleccionando las líneas que no coincidan con el patrón.

Por ejemplo, para hacerlo con el archivo */etc/default/grub* :

```
$ grep -v '^#' /etc/default/grub
```

En el siguiente ejemplo, el uso de los corchetes muestra las líneas de*/etc/default/grub* que contienen los términos hda o hdb.

```
$ grep 'hd[ab]' /etc/default/grub
```

Las opciones comunes del comando grep son:

-c: Cuenta las líneas en las cuales esta el patrón.

-i: Ignora la diferencia entre mayúsculas o minúsculas.

-f: Usa la expresión regular incluida en el archivo indicado por esta opción.

-n: Busca solamente en el número de línea indicada por esta opción.

-v: Realiza la acción inversa, es decir, muestra todas las líneas excepto la que corresponde al patrón.

Comandos egrep y fgrep

Los comandos **egrep** y **fgrep** complementan las funciones del comando**grep**.

El comando **egrep** es equivalente al comando **grep -E** que incorpora otras funcionalidades además de las expresiones regulares estándar. Por ejemplo, con **egrep** se puede utilizar el operador pipe "|" (tubería), que actúa como el operador O lógico:

```
$ egrep 'invención|invenciones'
```

Se devolverán todas las ocurrencias del término invención o invenciones.

El comando **fgrep** actúa de la misma forma que **grep -F** ya que deja de interpretar expresiones regulares. Es especialmente útil en los casos más sencillos donde lo que se desea es solamente encontrar la ocurrencia de algún término simple:

```
$ fgrep 'andalucia'
```

De este modo se ignora toda la operación de expresión regular lo que determinará que el proceso de búsqueda sea mucho más rápido. Incluso si se utilizan caracteres especiales, como dolar [$] o punto [.] se interpretarán literalmente y no por lo que representan en una expresión regular.

Expresiones regulares

Bueno en esta guía os quería explicar de forma mas clara y con ejemplos explicados que es esto de las expresiones regulares. Para comenzar lo primero que hay que aclarar es la diferencia entre expresiones regulares y patrones de ficheros, ya que se pueden confundir y a nivel funcional son diferentes. Las expresiones regulares son las utilizadas para buscar un texto dentro de un fichero, mientras que los patrones de ficheros son normalmente los argumentos pasados a los comandos como rm, cp, mv, etc. para hacer referencia a varios ficheros en el disco duro. Por tanto, podemos concluir en que tenemos un grupo de comandos que serán los que utilicen expresiones regulares y que están orientados al tratamiento de texto: grep, egrep, ed, sed, awk, vi, etc…

Existen dos tipos de expresiones regulares: las básicas y las extendidas. Como se puede deducir de su nombre las extendidas otorgan muchas mas opciones a la hora de tratar con los patrones a buscar. Antes de comenzar con la guía y ver ejemplos vamos a ver todas las expresiones regulares así como las reglas de uso de estas mismas:

Reglas

- Cuando hablamos de expresiones, estas pueden ser un carácter, un conjunto de caracteres o un metacáracter.
- Para utilizar las expresiones regulares es necesario ponerlas entre comillas simples ‘ ‘
- Para usar las expresiones regulares extendidas, es necesario otorgar al comando la funcionalidad. Por ejemplo, para el comando grep es necesario utilizar la opción -E o utilizar directamente el comando egrep, y para el comando sed es necesario añadirle la opción -r.
- Los caracteres son tratados de forma literal, es decir, que concuerdan consigo mismo. Por ejemplo x concuerda con x, abc concuerda con abc, etc…
- La excepción a la regla anterior esta en los metacaracteres:

```
. [ ] ^ $ * ( ) \
```

- Para utilizar estos metacaracteres como literales, es necesario escaparlos, esto se consigue antecediendole el metacaracter de barra invertida ****

A continuación seguiremos viendo algunas reglas más en conjunto de las diferentes expresiones regulares. Estas expresiones se pueden subdividir en varias categorías, que son las siguientes:

Expresiones regulares básicas de un solo carácter

Dentro de esta categoría tenemos las siguientes expresiones:

Expresión regular	Concuerda con….
.	Cualquier carácter
[ ]	Cualquiera de los caracteres que estén dentro de los corchetes
[^]	Cualquiera de los caracteres que NO estén dentro de los corchetes
^	El principio de línea
$	El final de la línea
*	Cero o mas ocurrencias de la expresión anterior (izquierda)
( )	Permite agrupar varias expresiones regulares
\	Escapa un metacarácter

Estas expresiones tienen las siguientes reglas especiales:

- Dentro de los corchetes [ ] los metacaracteres pierden su función especial y se tratan como literales.

- Hay varias excepciones a la regla anterior. Por ejemplo el carácter del “sombrerito” si se pone al inicio del corchete gana una funcionalidad diferente, como hemos visto en la tabla anterior.

Expresiones regulares básicas de repetición:

Podemos repetir una expresión regular tantas veces como queramos con la secuencia { }. Esta repetición se puede realizar de las siguientes formas:

Constructor	Proposito
{n}	Concuerda exactamente con n ocurrencias de la expresión anterior
{n,}	Concuerda con la menos n ocurrencias de la expresión anterior
{n, m}	Concuerda con entre n y m ocurrencias de la expresión anterior

Pongamos un ejemplo rápido de este tipo de expresiones. Por ejemplo para buscar números de tres cifras podemos utilizar [0-9]{3}. Esto significa que repetirá 3 veces la búsqueda de números del cero al nueve, podría igualarse a [0-9] [0-9] [0-9]

Expresiones regulares extendidas

Como hemos visto antes, para poder utilizar estas expresiones es necesario darle al comando que se ejecuta el soporte para poder usarlas. Antes de comenzar a verlas, es necesario ver reglas de conversión:

- El uso de las barras invertidas en los corchetes y los paréntesis no sirve en las expresiones regulares extendidas. Esto quiere decir que lo que hemos visto antes de ( ) y { } ahora se debe poner de este modo: ( ) y { }. Si queremos que sean caracteres literales es necesario escaparlos.

Expresiones regulares extendidas de un solo carácter

En este grupo se agregan las siguientes expresiones:

Expresión regular	Concuerda con…
+	Una o mas ocurrencias de la expresión anterior (izquierda)
?	Cero o una ocurrencia de la expresión anterior (izquierda)

Expresiones regulares extendidas de alternancia

Con el metacarácter tubería | podemos alternar entre dos expresiones regulares. Por ejemplo si queremos buscar una palabra que sea abc o bbc podemos indicarlo del siguiente modo **(a|b)bc**

Expresiones regulares extendidas de etiquetado

Un etiquetado consiste en referenciar una expresión regular en otro lugar del patrón de búsqueda. Para etiquetar una expresión regular debemos hacer uso de los paréntesis **( )** y luego para referenciar esa expresión debemos indicarlo con \n siendo n el número de la etiqueta en orden ascendente comenzando por uno (pueden existir varias etiquetas dentro del patrón).

Por ejemplo la expresión **(.)e\1e** concuerda con pepe y con nene.

Otros metacaracteres utilizados en las expresiones regulares extendidas

Existen otros metacaracteres para las expresiones regulares extendidas, entre los cuales solamente vamos a ver uno muy utilizado que hace referencia a la delimitación entre el principio y el fin de una palabra. Para referenciar el principio se debe usar **\<** y para referenciar el final **\>**

Casos prácticos

Bueno, una vez visto todo esto, vamos al caso práctico! Vamos a utilizar este texto para nuestros ejemplos:

```
Primeros pasos
Si desea empezar a usar Debian, puede obtener fácilmente una copia y seguir la
Guía de Instalació para instalarla.
Si est  actualizando a la última versión estable desde una versión anterior,
por favor, lea las Notas de Publicación antes de hacerlo.
Para obtener ayuda sobre el uso o instalación de Debian, consulte nuestras
páginas de documentación y soporte.
Los usuarios que hablen en idiomas que no sean el inglés deberían echar un
vistazo a nuestra sección internacional.
Los usuarios que usen sistemas distintos de Intel x86 deberían revisar la
sección de adaptaciones a otras arquitecturas.

RSS Últimas noticias.
[15 de feb de 2014] Updated Debian 6.0: 6.0.9 released
[8 de feb de 2014] Updated Debian 7: 7.4 released
[14 de dic de 2013] Updated Debian 7: 7.3 released
[20 de oct de 2013] Updated Debian 6.0: 6.0.8 released
[12 de oct de 2013] Updated Debian 7: 7.2 released
[28 de sep de 2013] Debian Edu / Skolelinux Wheezy - una solución completa
basada en Linux para la escuela

URLs
http://ftp.debian.org/debian/dists/wheezy/ChangeLog

http://ftp.debian.org/debian/dists/stable/

http://ftp.debian.org/debian/dists/proposed-updates

https://www.debian.org/releases/stable/

https://security.debian.org/
```

Lo primero que vamos a hacer para esta guía es configurar el comando grep con la opción de color. Esto nos va a ayudar mucho a saber realmente lo que estamos filtrando con el patrón que hemos puesto. Para activar esta opción se puede hacer de dos formas: o bien con la variable de entornoGREP_OPTIONS contenga el valor —color o bien creando un alias al comando alias grep=’grep —color=always’.

```
$ export GREP_OPTIONS = '--color'
```

Comenzamos con el ejemplo mas básico que es buscar una palabra concreta. En este caso vamos a buscar la palabra “Debian”:

```
$ grep 'Debian' texto
Si desea empezar a usar Debian, puede obtener fácilmente una copia y seguir la
Guía de Instalación para instalarla.
Para obtener ayuda sobre el uso o instalación de Debian, consulte nuestras
páginas de documentación y soporte.
[15 de feb de 2014] Updated Debian 6.0: 6.0.9 released
[8 de feb de 2014] Updated Debian 7: 7.4 released
[14 de dic de 2013] Updated Debian 7: 7.3 released
[20 de oct de 2013] Updated Debian 6.0: 6.0.8 released
[12 de oct de 2013] Updated Debian 7: 7.2 released
[28 de sep de 2013] Debian Edu / Skolelinux Wheezy - una solució  completa
basada en Linux para la escuela
```

Como podemos apreciar, no nos ha cogido las palabras “debian” que existen en el texto. Esto se debe a que, como ya sabemos, Linux discrimina entre mayúsculas y minúsculas. Para ignorar esta discriminación, debemos ponerle al comando grep la opción **-i**

Esto mismo podriamos abordarlo del siguiente modo, aunque no es el mas correcto. Usando el punto podemos sustituir cualquier carácter dentro del patrón de búsqueda, por tanto:

```
$ grep '.ebian' texto
Si desea empezar a usar Debian, puede obtener fácilmente una copia y seguir la
Guía de Instalación para instalarla.
Para obtener ayuda sobre el uso o instalación de Debian, consulte nuestras
páginas de documentación y soporte.
[15 de feb de 2014] Updated Debian 6.0: 6.0.9 released
[8 de feb de 2014] Updated Debian 7: 7.4 released
[14 de dic de 2013] Updated Debian 7: 7.3 released
[20 de oct de 2013] Updated Debian 6.0: 6.0.8 released
[12 de oct de 2013] Updated Debian 7: 7.2 released
[28 de sep de 2013] Debian Edu / Skolelinux Wheezy – una solución completa
basada en Linux para la escuela
http://ftp.debian.org/debian/dists/wheezy/ChangeLog
http://ftp.debian.org/debian/dists/stable/
http://ftp.debian.org/debian/dists/proposed-updates
https://www.debian.org/releases/stable/
https://security.debian.org/
```

De este modo buscaría cualquier palabra que contenga cualquier carácter seguido de “ebian”.

Ahora vamos a complicarlo mas. Vamos a realizar una búsqueda de todas las líneas que contengan la fecha de release pertenecientes a los dias 10 hasta el 19 de cualquier mes. Analizando el texto debemos notar que la expresión debe coincidir los números al inicio de línea. Viendo la estructura del fichero lo podemos acometer del siguiente modo:

```
$ grep '^\[1[0-9]' texto
[15 de feb de 2014] Updated Debian 6.0: 6.0.9 released
[14 de dic de 2013] Updated Debian 7: 7.3 released
[12 de oct de 2013] Updated Debian 7: 7.2 released
```

El acento circunflejo ^ lo utilizamos para indicar que la expresión se debe buscar al inicio de linea, despues escapamos el corchete para hacerlo literal, a continuación ponemos el uno y luego un rango que comprenda entre el 0 y el 9.

Lo siguiente que vamos a ver es el uso de una expresión regular extendida, que va a ser la interrogación. Por ejemplo, vamos a seleccionar las líneas que contengan tanto http como https. Para ello construimos el patrón del siguiente modo:

```
$ grep -E 'https?' texto
http://ftp.debian.org/debian/dists/wheezy/ChangeLog
http://ftp.debian.org/debian/dists/stable/
http://ftp.debian.org/debian/dists/proposed-updates
https://www.debian.org/releases/stable/
https://security.debian.org/
```

La interrogación permite que el carácter que esta a su izquierda sea opcional, por tanto nos va a buscar tanto cadenas http como https.

Para comprender como funciona el metacarácter + vamos a ver una sucesión de comandos para poder comprenderlo:

```
$ cat texto | cut -d " " -f 1-5 |grep -E '[0-1]+'
[15 de feb de 2014]
[8 de feb de 2014]
[14 de dic de 2013]
[20 de oct de 2013]
[12 de oct de 2013]
[28 de sep de 2013]
$ cat texto | cut -d " " -f 1-5 |grep -E '[0-2]+'
[15 de feb de 2014]
[8 de feb de 2014]
[14 de dic de 2013]
[20 de oct de 2013]
[12 de oct de 2013]
[28 de sep de 2013]
$ cat texto | cut -d " " -f 1-5 |grep -E '[0-3]+'
[15 de feb de 2014]
[8 de feb de 2014]
[14 de dic de 2013]
[20 de oct de 2013]
[12 de oct de 2013]
[28 de sep de 2013]
$ cat texto | cut -d " " -f 1-5 |grep -E '[0-4]+'
```

```
[15 de feb de 2014]
[8 de feb de 2014]
[14 de dic de 2013]
[20 de oct de 2013]
[12 de oct de 2013]
[28 de sep de 2013]
```

El siguiente ejemplo vamos a utilizar el concepto de repeticiones. Vamos a complicarlo un poco buscando palabras que solo contengan 4 letras. Hay que tener en cuenta que debemos delimitar el patrón con el inicio y el fin de una palabra, ya que si no buscaría dentro de las palabras secuencias de 4 letras, y eso no es lo que queremos. Por tanto la construcción se haría del siguiente modo:

```
$ grep -E '\<[a-zA-Z]{4}\>' texto
Si desea empezar a usar Debian, puede obtener fácilmente una copia y seguir la
Guía de Instalación para instalarla.
Si está actualizando a la última versión estable desde una versión anterior,
por favor, lea las Notas de Publicación antes de hacerlo.
Para obtener ayuda sobre el uso o instalación de Debian, consulte nuestras
páginas de documentación y soporte.
Los usuarios que hablen en idiomas que no sean el inglés deberían echar un
vistazo a nuestra sección internacional.
Los usuarios que usen sistemas distintos de Intel x86 deberían revisar la
sección de adaptaciones a otras arquitecturas.
[28 de sep de 2013] Debian Edu / Skolelinux Wheezy – una solución completa
basada en Linux para la escuela
URLs
http://ftp.debian.org/debian/dists/wheezy/ChangeLog
http://ftp.debian.org/debian/dists/stable/
http://ftp.debian.org/debian/dists/proposed-updates
```

Como podemos apreciar, con los delimitadores \< y > hacemos que solo encuentren un carácter comprendido entre la a y z tanto minúscula como mayúscula repetida la secuencia 4 veces.

Ahora algo mas complicado, vamos a seleccionar las líneas que tengan frases de dos palabras. Para ello lo primero a la hora de montar la expresión regular es saber que una palabra es una serie de letras consecutivas. Para afrontarlo como patrón de búsqueda podriamos indicar [a-zA-z]+

Lo siguiente que debemos entender que entre palabras hay un espacio, por tanto lo siguiente que deberiamos buscar sería cualquier cosa que no sea una letra, también hay que tener en cuenta que el final de la frase entendemos que se realiza con un punto, por tanto [^a-zA-Z]

Bien, ya tenemos una palabra y un espacio/punto. Ahora para saber una frase de dos palabras lo único que tendriamos que indicarle es que se repitiera 2 veces, pero claro, si lo ponemos de forma literal no va a comprender lo que queremos realizar. Por tanto es necesario agrupar las dos expresiones anteriores para realizar la repetición y delimitándolas para que entienda que es una frase, quedando una cosa así:

```
$ grep -E '^([a-zA-Z]+[^a-zA-Z]){2}.$' texto.txt
```

```
Últimas noticias.
```

Si analizamos la expresión regular, el inicio de la frase lo expresamos con ^ y el final con $. Lo siguiente es entre paréntesis la unión de una letra con un espacio o un punto, y ese conjunto se repite dos veces.

Espero que esta guía os haya servido. La he acortado en esta segunda edición de ejemplos, creo que en la otra me quedaron mas, pero bueno si necesitáis algún ejemplo concreto de algún metacarácter, solo tenéis que decírmelo.