EXPRESIONES REGULARES.

Las expresiones regulares son patrones que afectan a una cadena de caracteres. Son muy útiles para seleccionar con gran precisión elementos de un fichero. Se utilizan mucho en los scripts, y son la parte más difícil de entender. Veamos un ejemplo. Lo primero, creamos un fichero de texto llamado **expre** con el siguiente contenido:

cosas	**casa**	**libros**	**animal**	**verdugo**
LLUvia	**lanaa**	**madre**	**hermano**	**nido**
bicho	**limo**	**asno**	**vision**	**alma**
libro	**cosas**	**tipo**	**falso**	**oso**
bicha	**corbata**	**talon**	**barco**	**tigre**
ult				

Ahora vamos utilizar **grep** para buscar la cadena cosas dentro del fichero expre. Veamos, escribimos: **grep cosas expre**. Obtenemos:

libro	**cosas**	**tipo**	**falso**	**oso**
cosas	**casa**	**libros**	**animal**	**verdugo**

Como el fichero **expre** tiene cinco columnas, cuando **grep** selecciona la cadena cosas, nos muestra las filas completas en las que se encuentre. Ahora vamos a hacer uso de expresiones regulares para lograr selecciones más precisas. **“^cosas”** hace que **grep** busque la línea que comience por cosas. Esto es consecuencia del metacaracter: **^**. Las expresiones regulares son precisamente caracteres especiales (llamados metacaracteres) que influyen de una manera particular en la cadena de texto que referencian. Hagamos la prueba. Escribimos: **grep “^cosas” expre**. El resultado es:

cosas	**casa**	**libros**	**animal**	**verdugo**

Es importante no escribir ningún carácter previo a la primera columna de palabras al editar el fichero **expre**. No es correcto utilizar la barra espaciadora ni tabuladores. **grep** los tomará por caracteres y en este caso, como ninguna línea empieza por “cosas” no mostrará resultado alguno. Veamos otro ejemplo. Supongamos que queremos filtrar todos los subdirectorios del directorio: **/usr**. Para ello escribimos:

ls -la /usr | grep ^d

Obtenemos:

```
drwxr-xr-x    1 root root 368 Sep 15 00:56  .
drwxr-xr-x    2 root root 488 Sep 24 14:03  ..
drwxrwxrwx    8 root root 192 Aug 13 17:15  X11R6
drwxrwxrwx    2 root root 696 Sep 15 20:13  bin
drwxr-xr-x    2 root root 384 Sep 15 01:27  games
drwxrwxrwx    5 root root 120 Aug 13 17:08  i586-suse-linux
drwxrwxrwx    9 root root 864 Sep 8 00:02   include
drwxrwxrwx    1 root root 648 Sep 15 20:13  lib
drwxrwxrwx    1 root root 304 Sep 7 00:57   local
drwxr-xr-x    4 root root  96 Sep 15 00:56  man
drwxrwxrwx    2 root root  84 Sep 8 00:03   sbin
drwxr-xr-x    1 root root 64 Sep 15 20:15   share
drwxrwxrwx    5 root root 60 Aug 26 00:09   src
```

El comando **grep** ha seleccionado únicamente aquellos ficheros que comienzan por la letra **d**. La primera columna, comenzando por la izquierda son los permisos. El primer carácter indica el tipo de fichero. d significa directorio. Bien, esta forma de proceder va a ser nuestra estrategia para administrar el sistema. filtraremos los ficheros de configuración del sistema en busca de la información que necesitemos para después hacer algo con ella. Continuemos. Si escribimos:

grep “^...$” expre

Obtenemos:

ult

En este caso, hemos seleccionado solamente aquella línea que empiece por una palabra de tres caracteres. Si la expresión regular fuese: **^....$**, entonces seleccionaríamos palabras de cuatro caracteres. Basta, con escribir tantos puntos como caracteres tenga la palabra que queramos seleccionar. Otro ejemplo:

grep “^bich[oa]” expre

Obtenemos:

bicho limo asno vision alma
bicha corbata talon barco tigre

Grep ha seleccionado las líneas que contienen bicho o bicha. Veamos otras posibilidades:

grep “[A-Z] [A-Z]*” expre

Obtenemos:

LLUvia lanaa madre hermano nido

Lo que **grep** ha hecho ahora es seleccionar la única línea que empieza por mayúsculas. Ahora vamos a buscar las líneas que terminan por el carácter: e. Escribimos:

grep 'e$' expre

Obtenemos:

bicha corbata talon barco tigre

Por último, probemos a buscar aquella línea donde se repite el carácter: a. Escribimos:

grep “a\{2,\}” expre

Obtenemos:

LLUvia lanaa madre hermano nido

Como no se trata de probar todas y cada una de las expresiones, he preparado el siguiente listado. Es buena idea practicar utilizando cada una de las expresiones y ver el resultado.

PATRÓN	REPRESENTA
Bicho	**La cadena bicho**
^bicho	**La cadena bicho al comienzo de una línea**
bicho$	**La cadena bicho al final de una línea**
^bicho$	**La cadena bicho formando una única línea**
bich[ao]	**Las cadenas bicha y bicho**
bi[^aeiou]o	**La tercera letra no es una vocal minúscula**
bi.o	**La tercera letra es cualquier carácter**
^....$	**Cualquier línea que contenga cuatro caracteres cualesquiera**
^\.	**Cualquier línea que comience por un punto**
^[^.]	**Cualquier línea que no comience por un punto**
bicho$	**bicho, bichos, bichoss, bichosss, etc**
“*bicho”*	**bicho con o sin comillas dobles**
[a-z] [a-z]*	**Una o más letras minúsculas**
[^0-9a-z]	**Cualquier carácter que no sea ni número ni letra mayúscula.**
[A-Za-z]	**Cualquier letra, sea mayúscula o minúscula.**
[Ax5]	**Cualquier carácter que sea A, x o 5**
bicho | bicha	**Una de las palabras bicho o bicha**
(s | arb)usto	**Las palabras susto o arbusto**
\<bi	**Cualquier palabra que comience por bi**
bi\>	**Cualquier palabra que termine en bi**
a\{2,\}	**Dos o más aes en una misma fila.**

LOS FILTROS.

Los comandos que sirven para filtrar información se denominan filtros. Uno de los más conocidos es **grep**. Sin embargo, existen otros.

CUT

Vamos a ver en primer lugar el filtro **cut**. Permite cortar columnas o campos de un fichero de texto y enviarlos a la salida estándar. La opción **-c** es para cortar columnas y **-f** para cortar campos. La opción **-d** se utiliza para indicar que carácter hace de separación entre campos, es decir el delimitador. Si no se indica nada, el delimitador es el tabulador. Para que **cut** sepa que campos o columnas tiene que cortar hay que indicarlo con una lista. Esta lista tiene tres formatos posibles:

1-2	**Campos o columnas de 1 a 2.**
1-	**Campo o columna 1 toda la línea.**
1, 2	**Campos o columnas 1 y 2.**

Veamos unos ejemplos. Para ello editamos un fichero de texto llamado amigos con el siguiente contenido:

ANT:3680112
SEX:6789450
COM:3454327
VIM:4532278
DAO:5468903

Escribimos en la consola:

cut -c 1-3 amigos

Obtenemos:

ANT
SEX
COM
VIM
DAO

Es decir, hemos cortado las tres primeras letras de cada línea del fichero personas. Ahora vamos a intentar algo un poco más difícil. Vamos a cortar campos individuales. El carácter delimitador del fichero amigos, es **:**, Si escribimos:

cut -d ':' -f 1 amigos

Obtenemos:

ANT
SEX
COM
VIM
DAO

O sea el primer campo. Si escribimos:

cut -d ':' -f 2 amigos

Obtenemos el segundo campo:

3680112
6789450
3454327
4532278
5468903

Ahora vamos a ver un ejemplo más útil. Vamos a filtrar el fichero passwd, que se encuentra en /etc y vamos a seleccionar los usuarios que utilizan el interprete de ordenes bash. Lo primero para ver el contenido del fichero escribimos: **cat /etc/passwd**. Obtenemos:

root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/bin/bash
daemon:x:2:2:Daemon:/sbin:/bin/bash
lp:x:4:7:Printing daemon:/var/spool/lpd:/bin/bash
mail:x:8:12:Mailer daemon:/var/spool/clientmqueue:/bin/false
news:x:9:13:News system:/etc/news:/bin/bash
uucp:x:10:14:Unix-to-Unix CoPy system:/etc/uucp:/bin/bash
games:x:12:100:Games account:/var/games:/bin/bash
man:x:13:62:Manual pages viewer:/var/cache/man:/bin/bash

at:x:25:25:Batch jobs daemon:/var/spool/atjobs:/bin/bash
wwwrun:x:30:8:WWW daemon apache:/var/lib/wwwrun:/bin/false
ftp:x:40:49:FTP account:/srv/ftp:/bin/bash
gdm:x:50:15:Gnome Display Manager daemon:/var/lib/gdm:/bin/bash
postfix:x:51:51:Postfix Daemon:/var/spool/postfix:/bin/false
sshd:x:71:65:SSH daemon:/var/lib/sshd:/bin/false
ntp:x:74:65534:NTP daemon:/var/lib/ntp:/bin/false
vdr:x:100:33:Video Disk Recorder:/var/spool/video:/bin/false
nobody:x:65534:65533:nobody:/var/lib/nobody:/bin/bash

Escribimos:

cat /etc/passwd | grep bash | cut -d ':' -f 1,7

Obtenemos:

root:/bin/bash
bin:/bin/bash
daemon:/bin/bash
lp:/bin/bash
news:/bin/bash
uucp:/bin/bash
games:/bin/bash
man:/bin/bash
at:/bin/bash
ftp:/bin/bash
gdm:/bin/bash
nobody:/bin/bash

Hemos utilizado, en primer lugar **cat** para pasar el fichero **passwd** por la salida estándar, depués mediante un pipeline (pipe o tubería) lo hemos filtrado con **grep** seleccionando solamente aquellas líneas que contienen la cadena bash. Finalmente, nos quedamos con los campos 1 y 7.

TR

Dejamos **cut**, y pasamos al comando **tr.** Es un filtro que se emplea como traductor, generalmente para convertir de minúsculas a mayúsculas y viceversa. Lo primero editamos un fichero de texto de nombre gnu que tenga el siguiente contenido:

Gnu/linux es un sistema operativo
QUE PRESENTA UN GRAN FUTURO

Escribimos en la consola:

tr [A-Z] [a-z] < gnu

Obtenemos:

gnu/linux es un sistema operativo
que presenta un gran futuro

Hemos convertido todos los caracteres entre A y Z en sus homólogos entre a y z. Lógicamente, si queremos hacer lo contrario escribimos:

tr [a-z] [A-Z] < gnu

Obtenemos:

GNU/LINUX ES UN SISTEMA OPERATIVO
QUE PRESENTA UN GRAN FUTURO

Ahora vamos a ver como sustituir caracteres. Escribimos:

tr [A-Z] g < gnu

Obtenemos:

gnu/linux es un sistema operativo
ggg gggggggg gg gggg gggggg

Todas las mayúsculas han sido sustituidas por el carácter **g**. Naturalmente, podemos modificar el rango de caracteres implicados si en lugar de utilizar: **[A-Z]** utilizamos: **[A-M]**, es decir que solo se van a convertir en el carácter **g**, las mayúsculas entre la **A** y la **M**.
Probemos:

tr [A-M] g < gnu

Obtenemos:

gnu/linux es un sistema operativo
QUg PRgSgNTg UN gRgN gUTURO

También podemos utilizar las técnicas de sustitución para eliminar determinados caracteres. Se utiliza la opción **-d**. Veamos:

tr -d [A-Z] < gnu

Obtenemos:

nu/linux es un sistema operativo

Como era de esperar, todas las consonantes han sido eliminadas. Conviene señalar, que **tr** no afecta para nada al fichero de entrada, que como recordaremos se llama gnu. Lo toma, lo procesa, y envía el resultado al monitor. El original no es modificado en ningún caso.

TEE

Por último, vamos a ver el filtro **tee**. Tiene como misión bifurcar la salida de una orden enviando los resultados simultáneamente a la salida estándar (monitor) y a un fichero. Su sintaxis es:

tee [-a] archivo(s)

La opción **-a** significa append o sea añadir. Con esta opción la salida a un fichero no sobreescribe la información sino que la añade al final del fichero de texto. Este filtro es útil en algunos casos.