Programación en la shell (II)

Xaro Benavent Garcia

xaro.benavent@uv.es

Herramientas de Programación

Órdenes sobre archivos de texto: grep|egrep

Sintaxis: `egrep` [opciones] patrón archivo

Buscar en archivos líneas que cumplan un patrón (expresión regular)

```
grep "#define" *.cpp

grep "*\\" *.cpp
```

Expresiones regulares

- Expresión regular: patrón que define conjuntos de cadenas de caracteres
- Se forman mediante operadores que combinan expresiones más pequeñas.
- Construcción:
  - La mayoría de los caracteres son expresiones que se representan a sí mismos. (p.ej., letras y dígitos).
  - Existen algunos símbolos que son metacaracteres (tienen un significado especial):
    - si van precedidos del carácter " \ " ("se escapan") se traducen por sí mismos.

+	?	.	*	$	(	)	[	]	{	}	|	&	\

Formación de expresiones regulares

- La construcción [ ] (clase de caracteres) permite definir una lista de caracteres que tú quieres admitir en tu expresión.
- Existen clases de caracteres identificadas de forma predeterminada:

[:alnum:] [:alpha:] [:cntrl:] [:digit:] [:graph:]

[:print:] [:punct:] [:space:] [:upper:] [:xdigit:]

```
$ egrep "ca[bv]allo"
$ egrep "[0123456789]"
$ egrep "[0-9]" (Rango)
```

Expresiones regulares (2)

METACARACTERES

.	Cualquier carácter.
^	Principio de cadena.
$	Final de línea.
\<	Inicio de una palabra.
\>	Final de una palabra.

```
Todas las vocales que empiezan por aeiou
$ egrep "^[aeiou]"
Palabra dos a principio de línea
$ egrep "^\<dos\>"
```

Expresiones regulares (3)

OPERADORES DE REPETICIÓN

El elemento que precede es

?	opcional y aparece, como mucho, una vez.
*	aparece cero o más veces.
+	aparce una o más veces.
{n}	aparece n veces.
{n,}	aparece n o más veces.
{n,m}	aparece, al menos n veces per no más de m veces.

```
NIF
$ egrep "[0-9]{8}[A-Z]{1}" datos.txt
Números de teléfono en formato: 96-354-35-00
$ egrep "[0-9]{2}.[0-9]{3}.[0-9]{2}.[0-9]{2}" tel.txt
```

Expresiones regulares (4)

- Dos expresiones regulares se pueden concatenar
- Dos expresiones regulares se pueden unir mediante el operador | (infijo):
- Usar una clase de manera negativa: $[\hat{}\ \ ]$

Expresión regular	Selecciona
" $[\hat{}0-9]$"	no contengan ningún número
" $\hat{}[0-9]$"	contenga números a principio de línea
hola|adi[o|ó]s	hola adios adiós
chic[o|a]	chicho chicha

Expresiones regulares: ejemplos (1)

Expresión	**Selecciona**
a.b	axb aab abb aSb a#b ...
a..b	axxb aaab abbb a4$b
[abc]	a b c
[aA]	a A
[aA][bB]	ab aB Ab AB
[A-Za-z]	A B C ... Z a b .. z
$[0-9]*$	cadena_vacía 0 1 9 00 99 9999
$[0-9][0-9]*$	0 1 9 00 99 9999
$\hat{}.*\$$	cualquier línea completa

Expresiones regulares: ejemplos (2)

<table>
<tr><th>Expresión</th><th>Selecciona</th></tr>
<tr><td>abc+d</td><td>abcd abcccccd abccd</td></tr>
<tr><td>abc*</td><td>abd abcd abccccd</td></tr>
<tr><td>123[a|b]*</td><td>123 123ab 123a 123b 123aab</td></tr>
<tr><td>123a{2,4}</td><td>123aa 123aaa 123aaaa</td></tr>
<tr><td>123a{2,}</td><td>123aa 123aaa 123aaaaaaaaa</td></tr>
<tr><td>ab[c|C|d]</td><td>abc abC abd</td></tr>
<tr><td>\ <hola|adios|ciao\ ></td><td>hola adios ciao</td></tr>
<tr><td>^hola|adios</td><td>hola al prinicipio y adios donde sea</td></tr>
<tr><td>^(hola|adios)</td><td>hola y adios al principio</td></tr>
</table>

Órdenes sobre archivos de texto: sort

Sintaxis: `sort` [opciones] [archivo]

Ordenar las líneas de un fichero (no modifica el fichero) sobre `stdout`.
La ordenación se realiza por defecto por orden alfabético.

Opciones

-r	Invertir el orden de las comparaciones
-n	Ordenar numéricamente
-k'campo'	Ordenar por el campo indicado

```
$ ls | sort
$ ls -l | sort -k7 (campo hora)
```

Más Órdenes (1): `uniq`

Sintaxis: `uniq` [opciones] [entrada [salida]]

Eliminar líneas duplicadas de un fichero ordenado.

Opciones

-c	Anteponer a cada línea del resultado la cuenta del número de veces que aparece la línea en la entrada

Más órdenes (2): `cut`

Sintaxis: `cut` [opciones] [archivo]

Elimina (corta) secciones de las líneas de un archivo.

Opciones	
-d 'delim'	El carácter que sigue a -d es el delimitador de campos
-f 'lista'	El carácter que sigue a -f es una lista de campos separados por -d

```
$ ls|sort
$ sort|uniq -c
$ ps|cut -d ' ' -f2
```

Más Órdenes (3): `wc`, `head` y `tail`

Sintaxis: `wc` [opciones] [archivo]

Cuenta el número de bytes, líneas o palabras de un archivo.

Sintaxis: `head` [opciones] [archivo]

Muestra la primera parte (10 líneas) de un archivo.

Sintaxis: `tail` [opciones] [archivo]

Muestra la última parte (10 líneas) de un archivo.

```
$ wc mi_programa
$ head mi_programa
$ head -n 5 mi_programa
$ tail -n 2 mi_programa
```

SED: Stream Editor

Sintaxis: `sed` ′lista de comandos′ [archivo]

Utilizado para hacer transformaciones básicas sobre textos.

	Opciones
p	imprimir línea
s/old/new/f	Sustituir old por new. Si f=g, reemplazar todas las ocurrencias; f=p, imprimir

```
#!/bin/sh
Sustituir el identificador "max" por "MAX"
$ sed -e 's/max/MAX/' mi_programa.cpp
```

Ejemplos (I)

- Escribe un programa de shell que nos muestre por pantalla información sobre los usuarios conectados en este momento al sistema. Además nos debe mostrar la lista de usuarios ordenada por hora de conexión.

```
$cat usuarios
#!/bin/sh
who|sort -k5
```

- Escribe un programa de shell que nos muestre por pantalla el nombre de los usuarios conectados en este momento al sistema

```
$cat usuarios2
#!/bin/sh
who|cut -d ' ' -f1|uniq
```

Ejemplos (II)

- Escribe un programa de shell que nos diga cuántos usuario están conectados en este momento al sistema

```
$cat usuarios3
#!/bin/sh
who|cut -d ' ' -f1|uniq|wc -l
```

- Realiza un programa que diga qué usuarios con uid 8xxx y gid 8000(alumnos) están conectados a la máquina en este momento.

```
$cat filtro_usuarios
#!/bin/sh
ps -eo user,uid,gid|sort -k2 |uniq | egrep
'8[0-9]{3}[ ]*8000'
|cut -f1 -d ' '
```

Ejemplos (III)

- Escribe un programa de shell que nos diga el uso de disco de cada usuario del sistema ordenado por dicho uso

```
$cat uso_disco
#!/bin/sh
cd home
for dir in *
do
du -sk $dir
done| sort -k2 -rn
```

Ejemplos (IV)

- Escribe un programa de shell que nos diga cuántos includes hay por directorio en usr include

```
$cat cuantos_includes
#!/bin/sh
find /usr/include -type f -name "*.h"|
sed -e "s/\/usr\/include\/ / /"|grep "/" |
sed -e "s/\/.*\/ / /"|uniq -c |sort -rn
```