Uso de expresiones regulares por comandos

1. El comando vi

Las ERb se usan en el editor **vi** para la búsqueda y sustitución de cadenas de caracteres.

Sintaxis

Búsqueda (modo comando):

```
/expresión-regular-básica
```

Sustitución (modo ex):

```
:[dirección[,dirección]]s/expresión-regular-básica/
expresión-de-remplazo/[flags]
```

En el capítulo El comando sed se muestran ejemplos de sustitución con el comando **sed**, que utiliza la misma sintaxis que el editor **vi**.

2. El comando grep

Esta sección ilustra el uso de expresiones regulares mediante el comando **grep** normalizado por POSIX. Usado con la opción **-E**, el comando entiende las expresiones regulares extendidas.

Sintaxis básica

```
grep [-iv...] expresión-regular-básica [ arch1 ... ]
grep [-iv...] -E expresión-regular-extendida [ arch1 ... ]
```

El comando **grep** usado con la opción **-E** remplaza el comando **egrep**. Para otras opciones, consultar el manual del comando.

Ejemplos

*A continuación se muestra el archivo **tel2.txt**:*

```
$ cat tel2.txt
Méndez Roca, Gisela|calle Ruiseñor|28023|Madrid|915351478
Ruiz del Castillo, Marcos|calle Balmes|08020|Barcelona|932282177

Hernández Darín, Alberto|plaza mayor|13190|Corral de Calatrava|
926448829

Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de Calatrava|
926443602

Martínez Parra, Marta|calle de la Santa Trinidad|38870|La Calera|
984122119
Expósito Heredia, Pedro|calle del castillo|38870|La Calera|
```

```
984122369

$
```

Búsqueda de la cadena "calatrava" en mayúsculas o minúsculas:

```
$ grep -i 'calatrava' tel2.txt
Hernández Darín, Alberto|plaza mayor|13190|Corral de
Calatrava|926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de
Calatrava|926443602
$
```

Búsqueda de las líneas que comiencen por la letra 'G':

```
$ grep '^G' tel2.txt
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de
Calatrava|926443602
$
```

Búsqueda de las líneas que terminen por 9:

```
$ grep '9$' tel2.txt
Hernández Darín, Alberto|plaza mayor|13190|Corral de
Calatrava|926448829
Martínez Parra, Marta|calle de la Santa Trinidad|38870|
La Calera|984122119
Expósito Heredia, Pedro|calle del castillo|38870|
La Calera|984122369
$
```

Búsqueda de las líneas que contengan dos ocurrencias sucesivas de la letra 'r':

```
$ grep 'rr' tel2.txt
Hernández Darín, Alberto|plaza mayor|13190|Corral de
Calatrava|926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de
Calatrava|926443602
Martínez Parra, Marta|calle de la Santa Trinidad|38870|
La Calera|984122119
```

Mismo ejemplo usando una ERb:

```
$ grep 'r\{2\}' tel2.txt
Hernández Darín, Alberto|plaza mayor|13190|Corral de Calatrava|
926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de Calatrava|
```

```
926443602
Martínez Parra, Marta|calle de la Santa Trinidad|38870|
La Calera|984122119
```

Mismo ejemplo usando una ERe:

```
$ grep -E 'r{2}' tel2.txt
Hernández Darín, Alberto|plaza mayor|13190|Corral de
Calatrava|926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de
Calatrava|926443602
Martínez Parra, Marta|calle de la Santa Trinidad|38870|
La Calera|984122119
$
```

Mostrar todas las líneas que no estén en blanco:

```
$ grep -v '^[ ␣ ⇔ ]*$' tel2.txt
Méndez Roca, Gisela|calle Ruiseñor|28023|Madrid|915351478
Ruiz del Castillo, Marcos|calle Balmes|08020|Barcelona|932282177
Hernández Darín, Alberto|plaza mayor|13190|Corral de Calatrava|
926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de Calatrava|
926443602
Martínez Parra, Marta|calle de la Santa Trinidad|38870|
La Calera|984122119
Expósito Heredia, Pedro|calle del castillo|38870|La Calera|
984122369
$
```

Una línea en blanco puede ser una línea vacía o una línea que contenga una serie de espacios o tabulaciones.

Mostrar las líneas que contengan las cadenas `calatrava` *o* `madrid` *(sin diferenciar mayúsculas y minúsculas):*

```
$ grep -iE 'Calatrava|madrid' tel2.txt
Méndez Roca, Gisela|calle Ruiseñor|28023|Madrid|915351478
Hernández Darín, Alberto|plaza mayor|13190|Corral de Calatrava|
926448829
Gómez Bádenas, Josefina|calle Sagasta|13190|Corral de Calatrava|
926443602
$
```

3. El comando expr

Sintaxis

```
expr cadena-de-caracteres: expresión-regular-básica
```

Este comando ofrece un operador "**:**" que permite comprobar la correspondencia entre una cadena de caracteres y una expresión regular.

Funcionamiento del operador "**:**":

- El número de caracteres de `cadena-de-caracteres` correspondiente a la ERb `expresión-regular-básica` se muestra por pantalla.
- Si `cadena-de-caracteres` se corresponde con `expresión-regular-básica`, el comando devuelve el código verdadero. Devuelve el código falso en caso contrario.
- La expresión regular se compara con relación al principio de la variable (**el "^" está implícito en la ERb**).
- Si una parte de la expresión regular se graba con \(\), el comando muestra en el terminal la parte de la cadena correspondiente.

Ejemplos

Verificar que el usuario haya introducido un número:

```
$ read num1
250
$ read num2
a8
```

*La variable **num1** contiene únicamente cifras:*

```
$ expr "$num1": '[0-9][0-9]*$'    # equivalente a '^[0-9][0-9]*$'
3
$ echo $?
0
```

*La variable **num2** contiene al menos un carácter que no es una cifra. La ERb no se satisface:*

```
$ expr "$num2": '[0-9][0-9]*$'    # equivalente a '^[0-9][0-9]*$'
0
$ echo $?
1
$
```

Contar el número de caracteres que hay en una variable:

```
$ linea="Méndez Roca, Gisela|calle Ruiseñor|28023|Madrid|915351478"
$ expr "$linea": '.*'
57
```

Mostrar la parte de la cadena correspondiente a la grabación (en este caso, el código postal):

```
$ expr "$linea": '.*|\([0-9]\{5\}\)'
28023
$
```

*Como muestra la figura 1, el carácter * es avaricioso: busca siempre la cadena más larga (en este caso, la "|" más a la derecha):*

```
$ expr "$linea": '\(.*\)|'
Méndez Roca, Gisela|calle Ruiseñor|28023|Madrid
```

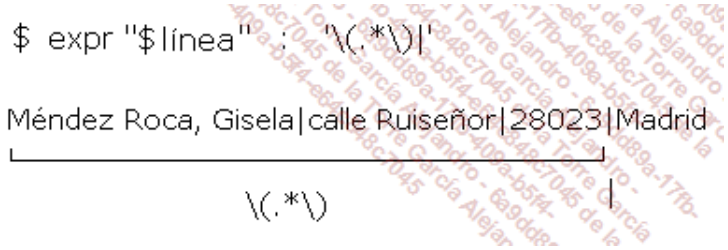

Figura 1: ¡El carácter "" es avaricioso!*

Solución que permite pararse en el primer "|" (ver figura 2):

```
$ expr "$linea": '\([^|]*\)|'
Méndez Roca, Gisela
$
```

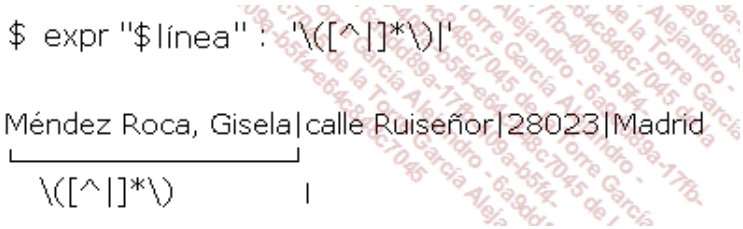

Figura 2: Recuperar la cadena más corta

*El script **testnum.sh** comprueba si el texto introducido por teclado es un número positivo o negativo. En caso afirmativo, realiza la suma de los números entrados.*

```
$ nl testnum.sh
     1  #! /usr/bin/sh

     2  suma=0

     3  while:
     4  do
     5    echo "Introduzca un número entero: \c"
     6    if read numero
```

```
 7    then
 8      if ( expr "$numero": '[0-9][0-9]*$' ||  \
 9           expr "$numero": '-[0-9][0-9]*$') > /dev/null

10      then
11        suma=`expr $suma + $numero`
12      else
13        echo "Entrada incorrecta"
14      fi
15    else
16      break
17    fi
18  done
19  echo "Resultado: $suma"
20  Exit 0
```

La barra invertida al final de la línea 8 permite enmascarar el carácter "salto de línea" para poder escribir el comando en dos líneas. La salida del comando **expr** se elimina, ya que en este caso se usa el código de retorno.

```
$ testnum
Introduzca un número entero: 1
Introduzca un número entero: 8000
Introduzca un número entero: -10
Introduzca un número entero: 2
Introduzca un número entero: aaa
Entrada incorrecta
Introduzca un número entero: ^d
Resultado: 7993
$
```

4. sed y awk

Las utilidades **sed** y **awk** se presentarán respectivamente en los capítulos El comando sed y El lenguaje de programación awk.