Conoce tus Expresiones Regulares

El concepto de una expresión regular (regexps) es una anotación que describe un patrón de una cadena de caracteres, se usa comúnmente en varios lenguajes y programas y aunque su implementación puede ser diferente los principios para construir una son comunes entre ellas.

Este articulo describe algunas herramientas y técnicas para aprender y precisar la construcción de regexps para una amplia gama de aplicaciones Unix como:

- Resaltar resultados
- Enseñar solo los resultados y no las líneas
- Llamar a un Mago
- Estudiar documentos.

Resaltar Resultados en su contexto

Cuando construimos regexp ayuda a que seamos capaces de ver que cadenas el patrón encuentra en el contexto del conjunto de información en la que buscamos. Consideremos las 4 lineas de texto de ***Ejem1*** y la expresión regular t[a-z]

Ejem1

```
$ cat midsummer.txt
I know a bank where the wild thyme blows,
Where oxlips and the nodding violet grows,
Quite over-canopied with luscious woodbine,
With sweet musk-roses and with eglantine.
$ grep t[a-z] midsummer.txt
I know a bank where the wild thyme blows,
Where oxlips and the nodding violet grows,
Quite over-canopied with luscious woodbine,
With sweet musk-roses and with eglantine.
$
```

Debido a que encuentra por lo menos 1 resultado en cada lina, el comando **grep** da como salida todas las líneas de texto del fichero. Pero ¿cuáles son los caracteres que la expresión regular encontró en estas líneas de texto?.

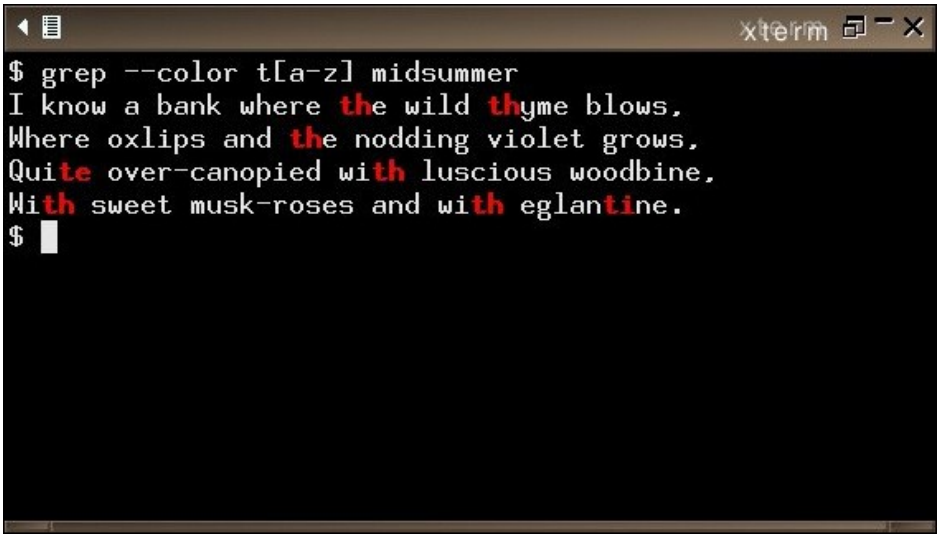
Con una regexp trivial como esta, es fácil que uno mismo la pueda encontrar “a ojo”. Pero con regexp más complicadas y en ficheros de datos más extensos, se puede tornar considerablemente más difícil encontrar que cadena o cadenas la expresión regular a encontrado. Es útil poder ver exactamente donde está la cadena que ha encontrado en

cada línea de texto la regexp. Una manera de ver tus regexp en contexto es marcándolas en la salida.

Puedes ahcerlo con varias aplicaciones como **grep**, **sed** y **Emacs.**

Resaltando con grep

En algunas de las nuevas versiones de grep (como GNU grep) resalta las regexp en color cuando usas el modificador *–color* como vemos en la siguiente imagen.



Si tu terminal soporta color, esta es una manera útil de ver exactamente que cadenas esta encontrando la regexp.

Resaltando con Sed

Tambien puedes resltar regexp en sed, el editor de flujo. El modificador a sed seria:

```
's/regexp/
```

Da como resultado una copia del fichero con todos los resultados marcado por corchetes “[ ]”. Como podemos ver en Ejem2

```
$ sed 's/t[a-z]/[&]/g' midsummer
I know a bank where [th]e wild [th]yme blows,
Where oxlips and [th]e nodding violet grows,
Qui[te] over-canopied wi[th] luscious woodbine,
Wi[th] sweet musk-roses and wi[th] eglan[ti]ne.
$
```

Puedes marcar las regexp de otras maneras. Si tu fichero de entrada es Groff, puedes agregar negrillas a la regexp y mandárselo a groff para que lo procese

```
$ sed 's/t[a-z]/\\fB&\\fP/g' infile.roff | groff -
```

Tambien puedes escribir un corto programa sed para que de los resultados en color. Si tu Shell soporta secuencias de escape, puedes resaltar todas las regexps en el contexto del fichero. Pero como las secuencias de escape son engorrosas para escribir, sin duda vas a querer ejecutarlo desde un script como mostramos a continuación

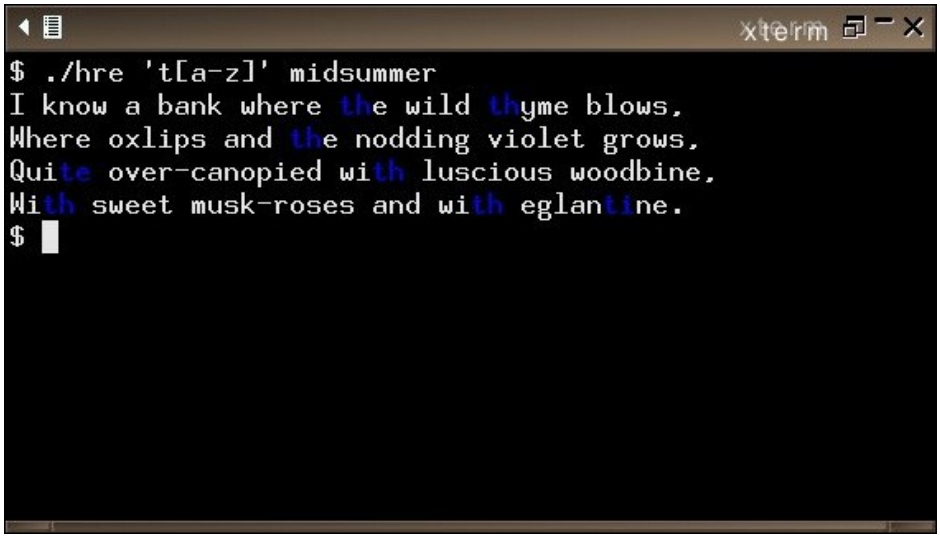
```
#!/bin/sh
# resalta el patron de regexp en el fichero
# uso: hre regexp fichero
sed 's/'$1'/^[[34m&^[[37m/g' < $2
```

El "^[" que aparece dos veces en el comando es un carácter de escape literal, por lo tanto tienes que introducir el comando con un editor que soporta la introducción de caracteres literales, como Emacs (donde escribes c-q ESC para introducirlo). El 34 y 37 son los códigos de escape de bash para especificar los colores azul y blanco respectivamente.

Para hacer el script ejecutable, escribimos:

```
$ chmod 744 hre
```

Luego ejecútalo como en el ejemplo 2 pero cambiando sed por el nombre de tu script (en este caso hre)



Este método tiene sus inconvenientes. El script solo funciona cuando el texto plano del terminal es blanco porque porque restaura el texto a ese color, si tu terminal uso un color diferente tienes que cambiar el código de escape en el script.

Resaltando con Emacs

En nevas versiones del editor Emacs GNU, las funciones `isearch-forward-regexp` y `isearch-backward-regexp` resaltan todos los resultados en el buffer. Si tienes instalado versiones recientes de Emacs en tu sistema, pruébalo ahora:

1. Ejecuta Emacs

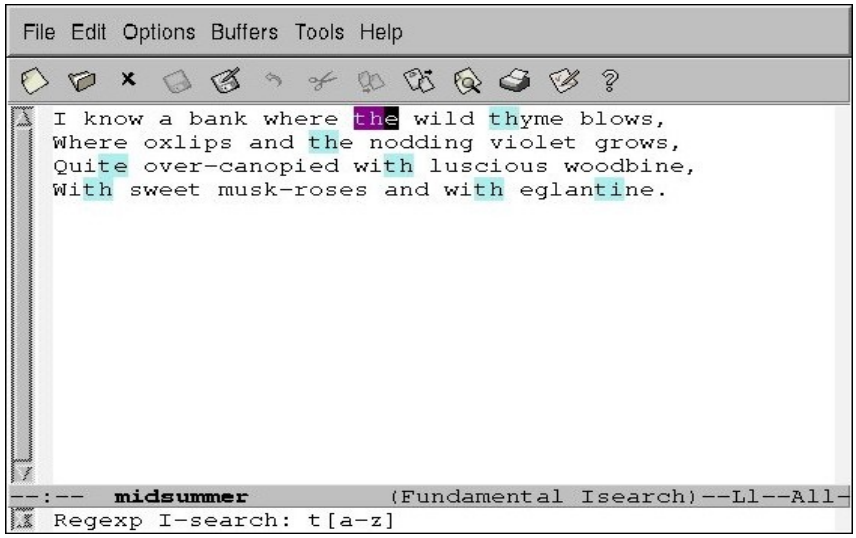
```
$ emacs midsummer
```

2. Escribe `M-x isearch-forward-regexp`.

   M-x es la notación de Emacs para la combinación de Meta-x, en la mayoría de los sistemas se realiza presionando la tecla “Alt” y X y soltándolas simultáneamente o presionando ESC soltándola y presionando la tecla X.

3. Escribe la regexp que usaras: t[a-z]

   Debido a que la búsqueda se realiza de manera incremental, Emacs empieza resaltando resultado mientras escribes cada carácter, en este caso cuando escribes “T”, todos los caracteres “T” en el bufer serán resaltadas. Notece que tan pronto escribes el carácter “[“ los resaltados desaparecen y el programa te avisa de que no tiene suficientes datos para realizar la búsqueda



4. Escribe C-x C-c Para salir de Emacs

   Escribes esta combinacion dejando presionado la tecla Crtl y presionando X, y luego dejando presionado la tecla Ctrl y presionando c

Las funciones `isearch-forward-regexp` y `isearch-backward-regexp` normalmente están atadas a los comandos M-S-s y M-S-r (para crearlas mantén presionado Alt y Ctrl, y luego presiona S o R)

Mostrar solo los resultados y no las líneas de texto

Hay otra manera de abarcar el problema del contexto del patrón y es de solo mostrar los resultados en si mas no la líneas enteras donde se encuentran. Hay maneras de hacer esto con **grep**, **sed** y **perl**

Muestra solo los resultados con grep

La opción "–only-matching" (también puede ser –o) cambia el comportamiento de grep de manera que de como salida solo los resultados en si. Como con la opción –color descripta anteriormente, esta función aparece en versiones recientes de grep, incluida GNU grep, que es "open source" y disponible para varios sistemas operativo.

Esta opción es para recolectar datos que coinciden con la regexp, es útil para recopilar direcciones IP, URLs, nombres, direcciones de correo electrónico, palabras y similares; pero también es una buena manera para aprender regexps. En Ejem 4 muestra cómo se recolectan todas las palabras de un fichero de texto, nos da cada palabra una por línea.

```
$ egrep -o '[A-Za-z]+' midsummer
I
know
a
bank
where
the
wild
thyme
blows
Where
oxlips
and
the
nodding
violet
grows
Quite
over
canopied
with
luscious
woodbine
With
sweet
musk
roses
and
with
eglantine
$
```

De hecho, esta es una manera simple de comprobar que tu regexp este correctamente construida cuando estés construyendo una particularmente complicada para cierto trabajo, amenudo puedes ver de inmediato si le hace falta algún arreglo

Digamos que solo quieres las palabras del fichero que contenga la cadena *th*, y has construido la regexp mostrada en Ejem5.

```
$ egrep -o 'th[a-z]*' midsummer
the
thyme
the
th
th
th
$
```

Puedes ver de inmediato que algunos de los resultados no son palabras. Intentémoslo de nuevo esta vez tomando en cuenta cualquier letras en la palabra que pueda venir antes de *th.*

```
$ egrep -o '[a-z]*th[a-z]*' midsummer
the
thyme
the
with
ith
with
$
```

Mejor, pero todavía no del todo correcto. Podemos ver en el caso de "*ith*" la regexp no busco letras en mayúsculas. Lo rectificamos usando la opción –i

```
$ egrep -o -i '[a-z]*th[a-z]*' midsummer
the
thyme
the
with
With
with
```

¡Listo!

El uso de –o y datos de prueba ayuda a la construcción de regexp, ya que has podido asumir que la regexp funciona

Mostrar Solo los resultados con Sed

Puedes hacer lo mismo con el comando sed, de la siguiente manera

```
s/.*\(regexp\).*/\1/p
```

Este comando muestra los resultados mas no la línea complet. Pero, solo muestra la ultima palabra que encontró en la línea

```
$ sed -n 's/.*\(th[a-z]\).*/\1/p'
midsummer
thy
the
$ grep -o th[a-z] midsummer
the
thy
the
$
```

M*ostrar solo los resultados con Perl*

Las regeps también son populares en el lenguaje Perl, pero las regeps en perl son diferentes de aquellas que usas en grep. La herramienta pcretest te deja probar regexps de Perl, la puedes usar para familiarizarte con con la librería de expresiones regulares compatible con Perl.

La regexps esta envuelto por un carácter "/" y puede venir seguido de un modificador que altera el comportamiento de la búsqueda

Modificador	Descripcion
8	Modifica el soporte para el conjunto de caracteres Unicode
g	Modifica la búsqueda para resultados globales
i	Ingora diferencia en Mayusuclas y Minisculas
M	Modifica la búsqueda en líneas multiples
X	Usa regexps exetendidas de Perl

Ejecuta pcretest de manera interactiva

```
$ pcretest
PCRE version 6.7 04-Jul-2006

  re> /[a-z]*th[a-z]*/ig
data> With sweet musk-roses and with eglantine.
 0: With
 0: with
data> Ctrl-c
$
```

Tambien puedes ejecutar pcrtest con un fichero de entrada. Ficheros de entrada contienen una regexp que deseas en la primera línea seguida por cualquier numero de líneas de datos para realizar la búsqueda. Tambien puedes tener varias regexps y sus respectivos datos separándolos entre ellos mediante una line vacia; pcrtest continua leyendo el fichero hasta que encuentra el final del fichero.

Si das el nombre de un segundo fichero, pcrtest escribe la salida en ese fichero, de otra manera solo muestra el resultado por la salida estándar.

```
$ cat midsummer.pre
/w[hi]|th/gi
I know a bank where the wild thyme blows,
Where oxlips and the nodding violet grows,
Quite over-canopied with luscious woodbine,
With sweet musk-roses and with eglantine.
$ pcretest midsummer.pre
PCRE version 6.7 04-Jul-2006

/w[hi]|th/gi
I know a bank where the wild thyme blows,
 0: wh
 0: th
 0: wi
 0: th
Where oxlips and the nodding violet grows,
 0: Wh
 0: th
Quite over-canopied with luscious woodbine,
 0: wi
 0: th
With sweet musk-roses and with eglantine.
 0: Wi
 0: th
 0: wi
 0: th
```

Llamando a un mago

El scipt txt2regex es un programa interactivo y multiplataforma echo para el Shell Bash. Cuando lo ejecutas hace una serie de preguntas sobre el patrón que quieres buscar y luego construye una regexps para varias aplicaciones como:

- awk
- ed
- egrep
- emacs
- expect
- find
- gawk
- grep
- javascript
- lex
- lisp
- mawk
- mysql
- ooo
- perl
- php
- postgres
- procmail
- python
- sed
- tcl
- vbscript

Ademas de ayudarte de manera interactiva en la construcción de regexps, txt2regx también trae un resumen de la sintaxis para varios lenguajes y aplicaciones como lista de regexps comunes y una tabla de metacarcteres.

Construye una regexp

Para construir una regexps para 1 o mas aplicaciones que soportan txt2regx debes dar los nombres de las aplicaciones en una lista delimitada pro comas como un argumento de –prog

Intenta construir la regexp trivial que hemos visto anteriormente, en la que encontraba el carácter T seguido por un carácter en minuxcula.

1) Ejecuta txt2regx y especifica regexps para grep, sed y Emacs:

```
$ txt2regex --prog grep,sed,emacs
```

2) Quieres encontrar el carácter T en cualquier parte de la lina no solamente al principio de la línea, por lo tanto escribe 2 para selección “cualquier parte de la línea”

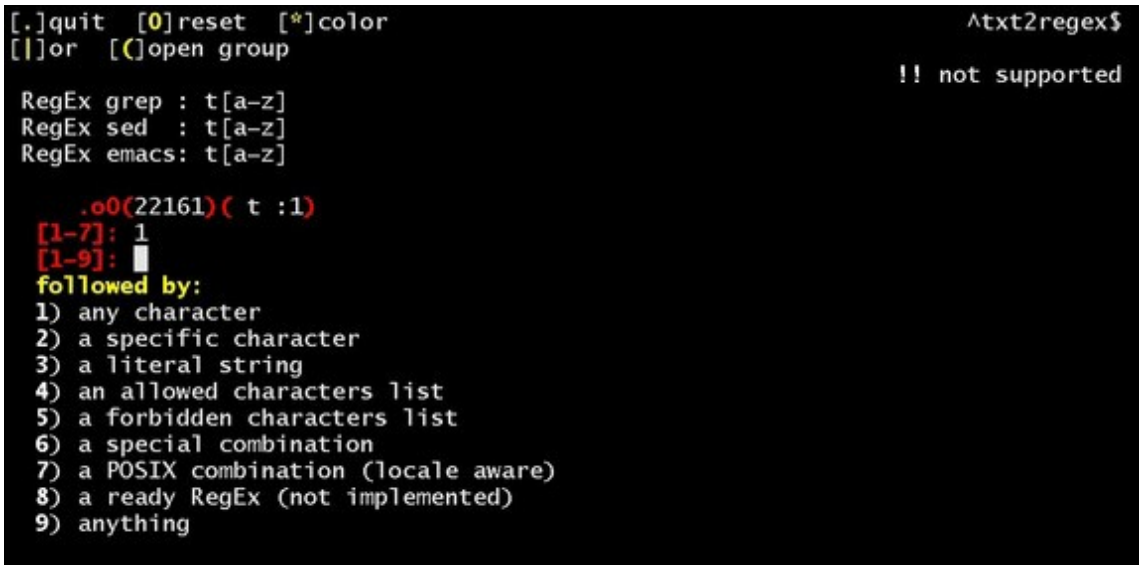
3) Escribe 2 otra vez para seleccionar un “carácter especifico” seguido por t cuando te pregunte que carácter deseas buscar

4) Escribe 1 para especificar exactamente una vez

5) Encuentra cualquier carácter en minúscula con la opción 6 para seleccionar “combinación especial” y luego la opción b para que solo sean minúsculas. Al finalizar escribe . para salir del menú de combinaciones

6) Encuentra las minúsculas escribien 1

Mientras vas por el proceso, txt2regx construye la regexp para cada una de las 3 aplicaciones que hemos seleccionado y las muestra en la parte superior de la pantalla. Ahora que hemos seleccionado exactamente lo que queremos, puedes ver las 3 regeps para las 3 aplicaciones.



Escribe .. para cerrar el programa. La lista de las regexp se mantendrán en tu terminal. Puedes usarlas tal y como se ven o editarlas y refinarlas, por ejemplo que tal si buscamos palabras que contenga un “`”, la regexp que acabamos de construir no lo mostrara de manera correcta como podemos ver a continuación

```
$ echo "Don't miss a word, just 'cause it's wrong." | egrep [A-Za-z]+
Don
t
miss
a
word
just
cause
it
s
wrong
$
```

Tendras que agregar el carácter dentro de los corchetes, de esta manera

```
$ echo "Don't miss a word, just 'cause it's wrong." | egrep "[A-Za-z']+"
Don't
miss
a
word
just
'cause
it's
wrong
$
```

Obten una regexp pre-echa

La opción –make la describe el autor como “un remedio para los dolores de cabeza”. Da una regexp para varios patrones comunes.

Argument	Description
`date`	This argument matches time in `hh:mm` format, from 00:00 to 99:99.
`hour`	This argument matches time in `hh:mm` format, from 00:00 to 99:99.
`hour3`	This argument matches time in `hh:mm` format, from 00:00 to 23:59.
`number`	This argument matches any positive or negative integer.

Por ejemplo, para obtener una regexp de hora military

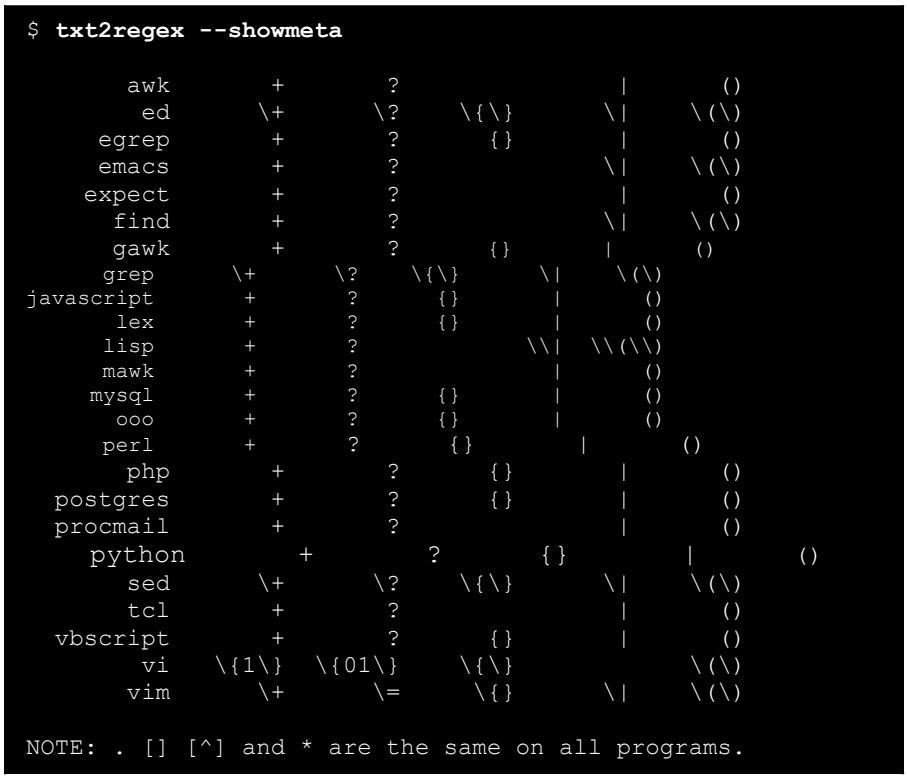
```
$ txt2regex --make hour3

 RegEx perl    : ([01][0-9]|2[0123]):[012345][0-9]
 RegEx php     : ([01][0-9]|2[0123]):[012345][0-9]
 RegEx postgres: ([01][0-9]|2[0123]):[012345][0-9]
 RegEx python  : ([01][0-9]|2[0123]):[012345][0-9]
 RegEx sed     : \([01][0-9]\|2[0123]\):[012345][0-9]
 RegEx vim     : \([01][0-9]\|2[0123]\):[012345][0-9]

$
```

Conoce tus metacaracteres

Con la opción –showmeta obtenemos una tabla con todos los metacaracteres



Estudia la Documentacion

Ante cualquier duda la mejor fuente para aclararlo suele ser la documentación, por norma general en todos los sistemas existe el comando man o info que proporciona los manuales para las aplicaciones