Facultad de Ingeniería
Universidad de Buenos Aires

75.08 Sistemas Operativos

Lic. Ing. Osvaldo Clúa
Lic. Adrián Muccio

Expresiones Regulares

Expresiones Regulares

- Se utilizan como solución al problema de realizar búsquedas de 2 caracteres en una cadena de 10 o un patrón definido en un archivo de millones de caracteres
- Las Expresiones Regulares (ER) constituyen un potente mecanismo para realizar manipulaciones y/o búsquedas de cadenas de texto

Expresiones Regulares

- Se define un espacio de búsqueda o *patrón* dentro del texto interpretando los caracteres en forma *literal* o *especial.*
- A los caracteres especiales se los llaman *metacaracteres* porque forman parte del mismo alfabeto del lenguaje pero se interpretan como descriptores de los caracteres *literales*

Expresiones Regulares

Ejemplos de *espacio de búsqueda o patrón*

- Dada la siguiente línea de texto:

  "-No puede ser pero es. El número de páginas de este libro es exactamente infinito. Ninguna es la primera; ninguna, la última."

  Extracto del cuento "El libro de arena" ed 1975 J. L. Borges

- Utilizando ER podríamos definir el siguiente *patrón*

  es. El número de páginas de este libro es exactamente infinito. Ninguna es

Expresiones Regulares

La ER para definir el patrón sería:

- 'es.*es'

Los caracteres espacio, e y ese son interpretados en forma literal

Los caracteres punto y asterisco actúan como Metacaracteres

es. El número de páginas de este libro es exactamente infinito. Ninguna es

Expresiones Regulares

- Una expresión regular es una forma de representar a los lenguajes regulares (finitos o infinitos) y se construye utilizando caracteres del alfabeto sobre el cual se define el lenguaje
- Es un conjunto de caracteres que especifican un patrón
- Toma su nombre de la notación utilizada por el matemático Stephen Cole Kleene en la década del 50

Aún antes de la creación de Unix, Ken Thompson implementó las RE para la búsqueda de patrones en un editor de línea de un sistema time-sharing

Expresiones Regulares

- Las Expresiones Regulares son interpretadas por un *Motor*
- Existen motores para el usuario final, por ejempo:
  - grep: ***g****lobal* ***r****egular* ***exp****resion,* comando de Unix
  - sed: ***s****tream* ***ed****itor,* comando de Unix
  - awk, ed, egrep, etc: comandos de Unix
  - Editores varios: vi, vim, etc

Expresiones Regulares

- Motores para el programador, por ejempo:
  - Perl
  - Python
  - PHP
  - Java/JavaScript
  - .Net Framework
- Motores para Base de Datos por ejempo:
  - Oracle, MySql, MariaDB, Teradata, Hadoop

Expresiones Regulares

IMPORTANTE

- Las ERs no forman parte del SHELL
- Si deseamos utilizar comandos que reciben ERs como parámetro, debemos protejernos de las distintas expansiones que realiza el SHELL

Expresiones Regulares

Metacaracteres de ER Simples

- . (punto): Cualquier caracter

de Anclaje

- ^ : Referencia al inicio de línea
- $: Referencia al fin de línea

Expresiones Regulares

de Repetición

- ? : Que el caracter anterior tenga como máximo una ocurrencia
- * : Expansión del caracter anterior en “n” ocurrencias subsiguientes del mismo. Incluye ocurrencia nula.
- \{x,y\} : Que el caracter anterior se repita entre *x* e *y* veces, siendo siendo opcional *y*

Expresiones Regulares

De Conjuntos

- [ ]: inicio y fin de conjunto
- - : rango dentro del conjunto
- ^ : negación del conjunto

Expresiones Regulares

grep –opciones RE lista_de_archivos

- El comportamiento por defecto es mostrar por std out todo el contenido de las líneas en las que se encuentre al menos una ocurrencia de la RE pasado por parámetro
- Es un típico comando *filtro*
- Ejemplo de opciones que modifican el comportamiento por defecto

  -l: solo muestra por std out el nombre de los archivos que en el que alguna de sus líneas contenga al menos una ocurrencia de la RE

Expresiones Regulares

grep –opciones RE lista_de_archivos

- -i: ignora diferencias entre mayúsculas y minúsculas
- -v: invierte el comportamiento por defecto
- -n: nro de línea + ‘:’ + línea
- -c: cuenta cuantas líneas contiene la RE

Expresiones Regulares

Ejemplos grep

muestra las líneas que comiencen con las palabras Hola u hola

```
> grep '^[Hh]ola' DATA
```

muestra las líneas no vacías

```
> grep -v '^$' DATA
```

muestra las líneas no vacías

```
> grep '.' DATA
```

Expresiones Regulares

muestra las líneas con un dígito < a 5 en su 3° caracter

> grep '^..[0-4]' DATA

muestra las líneas que no contengan un dígito entre 5 y 9 en su 3° caracter

> grep '^..[^5-9]' DATA

muestra las líneas que comienzan con una a y finalizan con por lo menos una b seguida de por lo menos una c seguida de una Z.

> grep '^a.*bcc*Z$' DATA

Ejemplo de salida:
aFAaJHads**bcZ**
a567cbccZ89jhgfdacasdf**bccccccccZ**

Expresiones Regulares

Ejercicio

Se tiene un archivo DATANET.conf con los códigos de transacción bancaria

Se desea saber si existe una inconsistencia entre los campos CREATE_TRX y PAYMENT_TYPE

la inconsistencia seria que existiera un valor Y en CREATE_TRX y el valor CHEQUE en campo PAYMENT_TYPE

Los campos están separados por ;

Expresiones Regulares

El formato de registro es

- BANK_ACCOUNT
- TRX_COD
- DESCRIPTION
- CREATE_TRX → Y
- RECEIPT_METHOD
- PAYMENT_TYPE → CHEQUE
- CODE_TYPE

Solo se puede usar el comando grep

Expresiones Regulares

sed: Stream Editor (editor en línea)

- sed es un verdadero editor de líneas, lo que hace es ejecutar comandos sobre cada línea de la entrada std o del archivo pasado como parámetro
- sed –opciones comando <parametros del comando> [archivo]

Expresiones Regulares

Algunos de los comandos son:

-a: agregar

-i : insertar

-d: borrar

-s: sustitur

Expresiones Regulares

Sustitución

- El comportamiento por defecto es recorrer todas las líneas y mostrarlas por la std out a menos que se encuentre una ocurrencia del patrón de búsqueda, en cuyo caso se muestra la línea cambiando el patrón por lo que se indique como patrón de sustitución.
- Solo se sustituye la primera ocurrencia del patrón de búsqueda

Expresiones Regulares

sed s/*busqueda*/*sustitucion/* lista_archivos

- El caracter que sigue al comando s queda a criterio del usuario, debe ser el mismo que separe el patrón de búsqueda del de la sustitución.

Expresiones Regulares

Tengo sed.

Tengo mucha, pero mucha sed.

cat archivo | sed 's/m.*a//'

El patrón se expande desde la primera m hasta la última a **mucha, pero mucha**

Tengo sed.

Tengo sed.

Expresiones Regulares

Se puede modificar el comportamiento por defecto

- Para sustituir todas las ocurrencias del patrón:

  sed s/find/replace/g

- Para sustituir en algunas líneas hay 2 opciones
  - Direccionamiento explícito por número de línea: sed 80s/a/B/
  - Direccionamiento por patrón:

    sed /^Hola/soAoBo

Expresiones Regulares

Sustitución con referencia

Ejemplo:

Se tiene un archivo de texto en el que aparecen fechas con el siguiente formato mm/dd/aaaa se desea cambiarle el formato a dd/mm/aaaa

1 2 3

-----mes---- -----día----- -----año----

sed ‘s-\([0-1] [0-9]\)/\([0-3][0-9]\)/\([0-9] \{4\}\)-\2/\1/\3-g’

Expresiones Regulares

Ejercicio

Se tiene un archivo con números enteros de 3 dígitos, se desea generar otro archivo con los capicúas de cada uno de los números

Ejemplo:

123 → 12321

231 → 23132

932 → 93239

Expresiones Regulares

Comando insert

- El comportamiento por defecto es imprimir por salida std nuevas líneas antes de una línea que contenga al menos una ocurrencia del patrón de búsqueda.
- sed '/patron/i\

  Nueva_linea_1

  Nueva_linea_N' lista_de_archivos

Expresiones Regulares

Archivo F.sql

```
CREATE OR REPLACE FUNCTION my_true RETURN NUMBER
IS
BEGIN
   RETURN TRUE;
END my_true ;
```

Expresiones Regulares

```
> sed '/my_true/i\
Nueva_linea'  F.sql

Nueva_linea
CREATE OR REPLACE FUNCTION my_true RETURN NUMBER
IS
BEGIN
   RETURN TRUE;
Nueva_linea
END my_true ;
```

Expresiones Regulares

Comando append

- El comportamiento por defecto es imprimir por salida std nuevas líneas después de una línea que contenga al menos una ocurrencia del patrón de búsqueda.
- sed '/patron/a\

  Nueva_linea_1

  Nueva_linea_N' lista_de_archivos

Expresiones Regulares

```
> sed '/my_true/a\
Nueva_linea'  F.sql

CREATE OR REPLACE FUNCTION my_true RETURN NUMBER
Nueva_linea
IS
BEGIN
   RETURN TRUE;
END my_true ;
Nueva_linea
```

Expresiones Regulares

Ejercicio PPV - Solo se permite el uso de grep, sed, wc, bc, let y echo

Una empresa operadora de televisión para su servicio de canales con cargo adicional, conocido como “Pay Per View”, desea que sus clientes puedan adquirir este servicio por medio de un mensaje SMS.

Nos pide desarrollar un script que sea invocado por una operadora telefónica y nuestro script registre la venta invocando a su vez a un API de su sistema CRM.

El script debe recibir como parámetros de entrada:

-Número de teléfono origen
-Código del cliente
-Canal

El Número de teléfono origen se envía con el siguiente formato fijo *(nn)(nnnnnn)nnnn*.
Donde ‘*n’:* significa dígito
Los caracteres 6 a 11 contienen el código de área
Los caracteres 13 a 16 contienen el número

Expresiones Regulares

El API a invocar es un programa que se llama RegistrarVentaPPV y recibe como parámetros el código de cliente y la señal PPV.
Por restricciones técnicas, en un mismo canal, no todos los clientes ven las mismas señales. La señal se determina en base al código de área del teléfono y el canal recibidos

Las relaciones se encuentran en el archivo Signals&Chanels.dat, que posee el siguiente formato:

Campo	Descripción
ID_RELACION	Identificación de Relación (*)
SEÑAL	Señal (*)
CODIGO_AREA	Nombre largo de la señal (*)
CANAL	Código del canal (*)
COMENTARIOS	Comentarios varios (*)

Todos los campos se encuentran separados por el caracter ;

El (*) indica que se desconocen tipo y formato del campo.

Existe a lo sumo un canal por cada dupla de SEÑAL y CODIGO_AREA

Numérico de 6 posiciones
Se rellena con 0s a la izquierda.

Expresiones Regulares

Ejercicio Control Sucursales - Solo se permite el uso de grep, sed, wc, bc, let y echo

Para la integración con el sistema de control de sucursales, la empresa nos pide desarrollar un script que reciba una novedad del inventario centralizado y la impacte en el stock de las sucursales.

El script se llamará RegistrarNovedadStock.sh, debe recibir como parámetro el centro logístico, la identificación del producto, el tipo de operación y la cantidad, debe deducir la sucursal relacionada y traducir el tipo de operación al código de novedad para finalmente invocar a un API del sistema de sucursales llamada UpdateStock.

El código de retorno del script debe ser el mismo código que devuelve el API.

En caso de no poder obtener la sucursal o el código de novedad, se debe invocar al API con el valor 'X' en el parámetro que no se pudo obtener.

Ejemplo de invocación del script:

RegistrarNovedadStock.sh <centro_logistico> <id_producto> <tipo_operación> <cantidad>

Expresiones Regulares

Ejemplo de invocación del API:

UpdateStock <sucursal> <código_novedad> <id_producto> <cantidad>

Para obtener la sucursal y el código de novedad se cuenta con el archivo MapeoSucursal.dat, que posee el siguiente formato:

Campo	Descripción
ID_MAPEO	Identificador de instancia de Mapeo (*)
CLASE_MAPEO	Clase de mapeo, para sucursal 'SUCURSAL', para novedad 'NOVEDAD'
VALOR_ORIGEN	Valor en sistema origen (*)
VALOR_DESTINO	Valor en sistema destino (*)
COMENTARIO	Comentarios varios (*)
ESTADO	Estado de la instancia, se debe verificar que la instancia a mapear se encuentre en estado 'ACTIVA'

Todos los campos se encuentran separados por el caracter ;

El (*) indica que se desconocen tipo y formato del campo.

Como máximo hay una sola instancia 'ACTIVA' para la dupla CLASE_MAPEO y VALOR_ORIGEN