



Introducción al Desarrollo de Software I - Lanzillotta

Capra - Chaves - Di Matteo - Sosa - Villegas - Palavecino

Agenda

- Condicionales e iterativas en Bash
 - Pipeline, redirección y listas
 - Expresiones Regulares
 - Script de búsqueda y reemplazo
-

Estructuras Condicionales e Iterativas

Estructuras

El Shell como intérprete puede resolver estructuras condicionales e iterativas, facilitando y ampliando la forma de interactuar con el usuario y permitiendo un gran abanico de posibilidad a la hora de crear Scripts.

Condicionales

La estructura de un condicional es:

```
if [[ CONDICIÓN ]] ; then
```

```
    echo "La condición se cumple, es True"
```

```
else
```

```
    echo "La condición no se cumple, es False"
```

```
fi
```

Condicionales - Cómo comparar

Números:

-eq: Si son Iguales

-ne: Si no son iguales

-gt: Si el primero es mayor al segundo

-lt: Si el primero es menor al segundo

-ge: Si el primero es mayor o igual al segundo

-le: Si el primero es menor o igual al segundo

Cadenas:

string1 == string2: Si son Iguales

string1 != string2: Si no son iguales

string1 =~ regex: Devuelve True si la regex coincide

string1 > string2: Si la primer cadena es mayor que la segunda por orden léxico. Lo contrario para <

-z string: Devuelve True, si la longitud es cero

-n string: Devuelve True, si la longitud NO es cero

Case

La estructura de un condicional tipo Case es:

case expression in

pattern1)

statements ;;

pattern2)

statements ;;

...

esac

Case - Ejemplo

Crear un script que reciba como parámetro el nombre de un archivo y muestre un menú con las siguientes opciones.

1- Ver el contenido del archivo

2- Editar el archivo con el editor nano

3- Ver los permisos del archivo

* - Salir

Estructuras cíclicas

- Una estructura cíclica nos permitirá repetir instrucciones dentro de un script. En Bash existen 3 estructuras for, while y until.
- Cada una de ellas tiene un uso específico.

Estructura For

El for nos permitirá iterar sobre una lista de valores.

donde:

i es la variable que tomará valor

[in LIST] es la lista de valores. Ejemplo

```
in {1. .5}
```

```
(( c=1; c<=5; c++ ))
```

```
in file1 file2 file3
```

```
in $(Linux-Or-Unix-Command-Here)
```

```
for i [in LIST ];
```

```
do
```

```
COMMANDS;
```

```
done
```

Ejemplo For

Crear un script que permita recorrer todos los archivos de un directorio y muestre el contenido de los que sean de extensión .txt

Estructura While

while [condition]

do

comandos

done

until [condition]

do

comandos

done

Ejemplo While y Until

1- Crear un script utilizando while que permita imprimir el mensaje

“Bienvenido por vez número: “ 5 veces.

2- Crear un script utilizando until que genere un número random entre 1 y 20, si los valores están entre 5 y 10 el bucle deberá terminar.

Pipelines |

Pipe o flujo de datos, nos permite utilizar la salida de un comando como entrada de otro.

Un ejemplo sería

comando 1 | comando 2 | comando 3

Lo que devuelva el comando 1 será utilizado por el comando 2 y la salida de este por el comando 3... así las veces que se desee.

Ejemplo Pipelines

1- Contar la cantidad de líneas de una archivo de texto.

```
cat ejemplo.txt | wc -l
```

2- Localizar una palabra en un archivo

```
cat ejemplo.txt | grep "Hola"
```


3- Ordenar las líneas de un archivo

```
cat ejemplo.txt | sort
```


Busqueda con grep

- Con grep se puede buscar una palabra o patrón y se imprimirá la línea o líneas que la contengan

grep **opcion/es** '**cadena_de_texto**' **fichero_donde_buscar**



Palabra o patrón a
buscar



Archivo en el cual
buscar

Ejemplos uso de grep

- Buscar una palabra

```
grep "Hola" ejemplo.txt
```

- Buscar una frase en un un archivo

```
grep "Hola mundo" ejemplo.txt
```

- Buscar múltiples palabras

```
grep búsqueda1 archivo | grep búsqueda2 archivo
```

- Buscar una palabra en un conjunto de archivos

```
grep -l palabra_a_buscar ./*
```

Búsqueda y reemplazo

Stream Editor (sed)

```
sed -i 's/old-text/new-text/g' input.txt
```

-i: Indica hacer un update en el archivo

s: Indica sustituir las palabras

g: Indica hacerlo en forma global

Expresiones regulares

- Una expresión regular es una cadena que sirve para expresar de forma compacta, un conjunto de cadenas.
- Son un potente mecanismo para resolver manipulación de textos y cadenas.
- Uno de los comandos que aceptan RE es grep.
- Por otra parte hay múltiples lenguajes que las interpretan

RE::Patrones básicos

Operación	Operador			Descripción	Ejemplo (ERE)
	BRE	ERE	PCRE		
Cuantificación	\?	?		Una vez o ninguna.	a?
	*			Las veces que sea (incluso ninguna)	a*
	\+	+		Al menos una vez	a+
	\{n,m\}	{n,m}		Entre <i>n</i> y <i>m</i> veces. Puede omitirse uno de los límites.	a{5,9}
Agrupación	Ver grupos.	(?regex)	(?:regex)	Para modificar el alcance de un operador	(?123)+
Alternativa	\			O lo uno o lo otro	(?Blas Luis)
Principio	^			El patrón comienza	^a
Fin	\$			El patrón acaba	a\$
Repr. universal	.			Cualquier carácter	.{2,3}
Escape	\			No interpretar un carácter especial	\.

BRE (*Basic Regular Expression*)

ERE (*Extended Regular Expression*) **La más usada**

PCRE (*Perl Compatible Regular Expression*)

RE::Patrones avanzados

Operación	Operador		Descripción	Ejemplo
	BRE	ERE/PCRE		
Alternativa	[...]		Uno de los caracteres incluidos. Puede indicarse un rango.	[A-Za-z]
	[^...]		Un caracter que no sea ninguno de los incluidos	[^A-Z]
Clases	Sin equivalencia	\w	Un carácter de palabra (letra, dígito o «_»).	\w+
	Sin equivalencia	\W	Un carácter que no sea de palabra.	^\W
	Sin equivalencia	\d	Un dígito, o sea, [0-9]	\d{4}
	Sin equivalencia	\D	Un carácter que no sea un dígito.	\D+
	Sin equivalencia	\s	Un carácter de espaciado, o sea, [\t\r\b\f]	\w+\s\w+
	Sin equivalencia	\S	Un carácter que no sea de espaciado.	\S+\s
	Sin equivalencia	[:nombre:]	Un carácter de la clase <i>nombre</i> .	[[:alpha:], ; .]+
	Sin equivalencia	[=x=]	Cualquier variante del caracter «x»	[[=a=]]
Límite de palabra	Sin equivalencia	\b[10]	Principio o fin de palabra.	\bdado\b
Grupos	\(regex\)	(regex)	Captura un grupo	(\w+)\s+\1
	\1, \2, ... \9		Refiere un grupo previamente capturado	

RE::Ejemplos

El uso de la opción -E nos permite colocar RE en grep

1- ¿Contiene el texto una o ninguna letra h?

grep -E 'h?' ejemplo.txt

2- ¿El texto contiene al menos dos letras o seguidas?

grep -E 'a{2,}' ejemplo.txt

3- ¿Comienza el texto por una letra mayúscula?

grep -E '^[A-Z]' ejemplo.txt

4- La frase no contiene ningún dígito:

grep -E '^\D+\$' ejemplo.txt

Operadores de redirección.


La salida estándar es la pantalla **stdout**, pero se puede redirigir la misma a otro lugar, por ejemplo un archivo. Para esto se utiliza el operador “>”

Realizar el siguiente ejemplo

```
bruno@bruno:~$ date
```

```
bruno@bruno:~$ date > fecha.txt
```

Se redirige la fecha
para ser guardada
en el archivo
fecha.txt



```
bruno@bruno:~$ cat fecha.txt
```


Operadores de redirección.

El operador `>`, lo que hace es sobrescribir el archivo, mientras que el operador `>>` sirve como agregar al final del archivo, como si fuese un `append`.

Ejemplo

```
[damon@localhost ~]$ date > ejemplo.txt
```

```
[damon@localhost ~]$ hostname >> ejemplo.txt
```

Operadores de redirección.

Así como existen los operadores de salida, también existe el de entrada

stdin, para esto se utiliza <

Ejemplo:

```
[damon@localhost ~]$ sort < milista.txt
```

Ejemplo 2: **(qué hace?)**

```
[damon@localhost ~]$ sort < mylist.txt > alphabetical-file.txt
```

Ordena el contenido de **milista.txt**, y lo muestra por pantalla

Operadores de redirección.

Por último existe un operador para **stderr**, que es `2>` y `2>>` (funciona como append)

Entonces se podría tener un archivo de log con errores como los ejemplos:

```
[damon@localhost ~]$ ccat 2> error.log
```

```
[damon@localhost ~]$ png 2>> error.log
```

Ejercitación

Ejercicio 1

Solicitar al usuario el ingreso de 10 valores enteros y calcular el promedio de los mismos.

Ejercicio 2

Crear un script llamado menu.sh que reciba un parámetro con el nombre de un archivo. El script deberá tener el siguiente menú que permita:

- 1- Ingresar una palabra y reemplazarla por **** en el archivo pasado por parámetro
- 2-abrir el archivo
- 3- realizar una copia del archivo llamada menu_copia.sh
- 4- Ingresar un email y validarlo mediante RE.
- 5- Salir