

Linux Shell Scripting

CONTENIDO:

- **Introducción**
- **Nuestro primer script**
- **Variables**
- **Comandos del sistema**
- **Estructuras condicionales**
- **Ciclos**
- **Ingresando datos**
- **Operadores**
- **Recomendaciones**
- **Agradecimientos**

INTRODUCCIÓN

- El shell es un ambiente de programación capaz de automatizar casi cualquier cosa en su sistema Linux.
- El shell de uso más común en Linux es bash, pero existen otros (ksh, sh, entre otros)

INTRODUCCIÓN

- El shell provee:
 - Una interfase interactiva textual al sistema operativo.
 - Un ambiente operacional.
 - Facilidades para iniciar y administrar comandos y programas.
 - Un lenguaje de programación.

INTRODUCCIÓN

- Los scripts son archivos que contienen comandos a ser ejecutados por una shell.
- Puede ser otro comando que pueda teclear a partir del prompt.
 - comando que invoque una utilidad
 - Un programa compilado
 - Otro script :)

INTRODUCCIÓN

- Los scripts soportan varias características de programación como pueden ser: ciclos, variables, arreglos, funciones, etc.
- Redirecciones de salida y entrada, PIPES, Expansiones, control de trabajo (jobs)
- Comandos contruidos de forma personalizada :)

NUESTRO PRIMER SCRIPT

```
--- holaScript.sh ---
```

```
#!/bin/bash
```

```
clear
```

```
echo Hola script
```

```
--- holaScript.sh ---
```

VARIABLES

- De ambiente: Variables globales. Son pasadas a todos los procesos iniciados por el shell, incluyendo otros shells. Esto significa que los procesos hijo *heredan* el ambiente. Por convención se expresan en mayúsculas.
- De shell: Son variables locales. Son específicas al shell corriente y no son heredadas por procesos hijo. En bash, las variables de shell pasan a ser de ambiente cuando son exportadas.

VARIABLES

- Muchas de las variables son necesarias para la ejecución de programas.
- Para que esas variables estén disponibles deben ser exportadas para convertirlas en variables de ambiente.

```
$ export MIVAR
```

- Para listar las variables de ambiente:

```
$ env
```

VARIABLES

- `$0` : Nombre del programa ó script.
- `$#` : Cantidad de argumentos.
- `$1..$n` : Argumentos según su posición en la línea de comandos.
- `$@` : Lista de argumentos.
- `$?` : Resultado de la ejecución del ultimo comando.
- `$$` : PID del shell actual.

VARIABLES

```
--- variables3.sh ---
```

```
echo "Nombre del script: $0"
```

```
echo "Número de argumentos: $#"
```

```
echo "Lista de argumentos: $@"
```

```
echo "PID del proceso actual: $$"
```

```
echo "PID del proceso hijo: $!"
```

```
--- variables3.sh ---
```

COMANDOS DEL SISTEMA

```
--- comando.sh ---
```

```
#!/bin/bash
```

```
HOLA="Hola, hoy es el día $(date +%j)  
del año."
```

```
echo $HOLA
```

```
--- comando.sh ---
```

Apostrofes y comillas

Cuando se asignan cadenas de caracteres que contiene espacios o caracteres especiales, la cadena debe estar encerrada entre apostrofes o comillas

El uso de comillas dentro de una cadena de caracteres permitira que cualquier variables dentro de las comillassea interpretado

COMANDOS DEL SISTEMA

```
--- comando.sh ---
```

```
#!/bin/bash
```

```
HOLA="date +%j"
```

```
echo $HOLA
```

```
--- comando.sh ---
```

COMANDOS DEL SISTEMA

```
--- comando.sh ---
```

```
#!/bin/bash
```

```
HOLA=`date +%j`
```

```
echo $HOLA
```

```
--- comando.sh ---
```

DEL SISTEMA

COMANDOS

```
--- variables.sh ---
```

```
#!/bin/bash
```

```
var="test de cadenas"
```

```
var2="Valor de la variable es $var"
```

```
echo $var2
```

```
--- variables.sh ---
```


COMANDOS DEL SISTEMA

```
--- variables.sh ---
```

```
#!/bin/bash
```

```
var='test de cadenas'
```

```
var2='Valor de la variable es $var'
```

```
echo $var2
```

```
--- variables.sh ---
```

OPERADORES (CADENAS DE TEXTO)

- [`s1 = s2`]: `s1` coincide con `s2`
- [`s1 != s2`]: `s1` no coincide con `s2`
- [`s1 < s2`]: `s1` es alfabéticamente anterior a `s2`, con el locale actual
- [`s1 > s2`]: `s1` es alfabéticamente posterior a `s2`, con el locale actual
- [`-n s1`]: `s1` no es nulo (contiene uno o más caracteres)
- [`-z s1`] : `s1` es nulo

ESTRUCTURA CONDICIONALES

```
--- si2.sh ---
```

```
#!/bin/bash
```

```
if [ $(whoami) = root ]; then
```

```
    echo "Hola ROOT"
```

```
else
```

```
    echo "No eres root"
```

```
fi
```

```
--- si2.sh ---
```

ESTRUCTURA CONDICIONALES

```
--- si3.sh ---
```

```
#!/bin/bash
```

```
if [ $USER = root ]; then
```

```
echo "El usuario es root"
```

```
elif [ $(whoami) = usted ]; then
```

```
    echo "El usuario es usted"
```

```
else
```

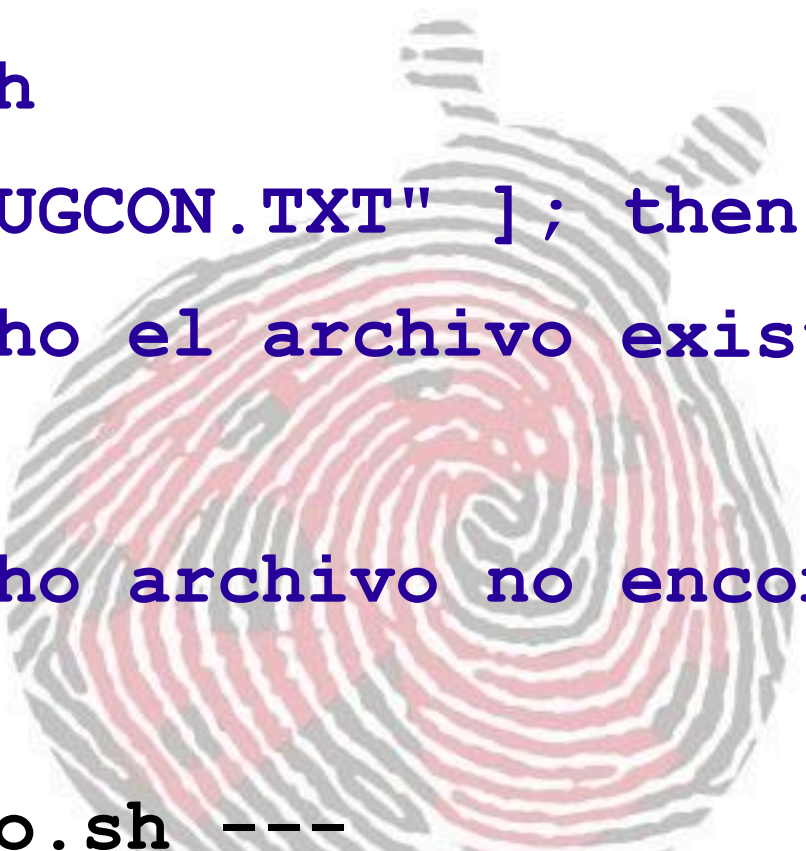
```
    echo "quien eres"
```

```
fi
```

```
--- si3.sh ---
```

ESTRUCTURA CONDICIONALES

```
--- archivo.sh ---  
#!/bin/bash  
if [ -f "BUGCON.TXT" ]; then  
    echo el archivo existe  
else  
    echo archivo no encontrado  
fi  
--- archivo.sh ---
```



INGRESANDO DATOS

```
--- leer.sh ---
```

```
#!/bin/bash
```

```
echo Por favor, introduzca su nombre:
```

```
read NOMBRE
```

```
echo "¡Hola $NOMBRE!"
```

```
--- leer.sh ---
```

OPERADORES (ARITMETICOS)

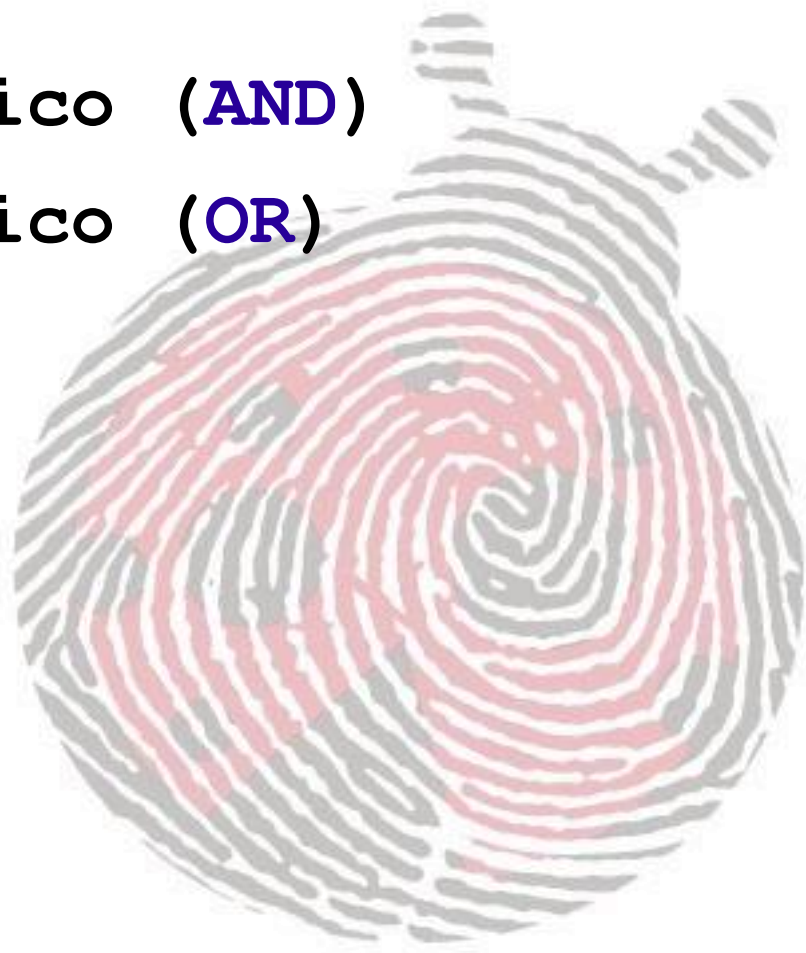
- `+` (adición)
- `-` (sustracción)
- `*` (producto)
- `/` (división)
- `%` (módulo)
- `**` (exponenciación)

OPERADORES (RELACIONES ARITMETICAS)

- [a -lt b] equivale a ((a < b))
- [a -gt b] equivale a ((a > b))
- [a -le b] equivale a ((a <= b))
- [a -ge b] equivale a ((a >= b))
- [a -eq b] equivale a ((a == b))
- [a -ne b] equivale a ((a != b))

OPERADORES (LOGICOS)

- `&&` Y lógico (AND)
- `||` O lógico (OR)



OPERADORES (SOBRE ARCHIVOS)

-d /ruta/archivo: Verdadero si *archivo* existe y es un directorio

-e /ruta/archivo: Verdadero si *archivo* existe

-f /ruta/archivo: Verdadero si *archivo* existe y es un archivo común

-l /ruta/archivo: Verdadero si *archivo* existe y es un enlace suave

-r /ruta/archivo: Verdadero si *archivo* existe y puede leerse

OPERADORES (SOBRE ARCHIVOS)

-s /ruta/archivo: Verdadero si *archivo* existe y tiene tamaño mayor que 0

-w /ruta/archivo: Verdadero si *archivo* existe y es escribible

-x /ruta/archivo: Verdadero si *archivo* existe y es ejecutable

arch1 -ot arch2: Verdadero si *arch1* es más viejo que *arch2*

CICLOS

- El `for` es distinto a los de otros lenguajes de programación. Básicamente, le permite iterar sobre una serie de 'palabras' contenidas dentro de una cadena.

CICLOS

- El `for` es distinto a los de otros lenguajes de programación. Básicamente, le permite iterar sobre una serie de 'palabras' contenidas dentro de una cadena.
- El `while` ejecuta un Trozo de código si la expresión de control es verdadera, termina cuando es Falsa (o se encuentra una interrupción explícita dentro del código en ejecución)

CICLOS

- El `until` es casi idéntico al `while`, excepto en que el código se ejecuta mientras la expresión de control se evalúe como falsa.

CICLOS

```
--- contador.sh ---  
#!/bin/bash  
clear  
  
read -p "Introduce un numero: " numero  
for ( ( a=0; a<=$numero; a++ ) )  
do  
    echo "$a"  
done  
  
--- contador.sh ---
```

CICLOS

```
--- mientras.sh ---
```

```
#!/bin/bash
```

```
CONTADOR=0
```

```
while [ $CONTADOR -lt 10 ]; do
```

```
    echo El contador es $CONTADOR
```

```
    let CONTADOR=CONTADOR+1
```

```
done
```

```
--- mientras.sh ---
```


CICLOS

```
--- until.sh ---  
#!/bin/bash  
CONTADOR=20  
until [ $CONTADOR -lt 10 ]; do  
    echo CONTADOR $CONTADOR  
    let CONTADOR-=1  
done  
--- until.sh ---
```

CICLOS

```
--- caso.sh ---
```

```
#!/bin/bash
```

```
echo "Introduce un numero entre 1 y 5: "
```

```
read num
```

```
case $num in
```

```
1) echo "El valor de num es 1.>";;
```

```
2) echo "El valor de num es 2.>";;
```

```
3) echo "El valor de num es 3.>";;
```

```
4) echo "El valor de num es 4.>";;
```

```
5) echo "El valor de num es 5.>";;
```

```
0|6) echo "NUMERO FUERA DE RANGO.>";;
```

```
*) echo "valor no reconocido";;
```

```
esac
```

```
--- caso.sh ---
```

Ejercicio

- ▶ A) Verifique si el usuario es root, de ser afirmativo: Use un ciclo y la sentencia case- esac para mostrar el siguiente menú:
 - 1 Listar Procesos
 - 2 Listar Procesos en forma de árbol
 - 3 Guardar en un archivo el contenido del directorio actual
 - 4 salir

EJERCICIO

Script (ejercicio.sh) que haga:

- Una carpeta (BUGCON)
- Una serie de archivos (10 archivos) dentro de la carpeta BUGCON
- Comprima la carpeta de BUGCON
-
- PREMIO AL QUE LO REALICE :)