

Práctica 8

Programación Shell

Scripts

- Un script es un archivo que contiene un conjunto de instrucciones que serán interpretadas por la Shell.
- La primera línea indicará la Shell que ejecutará nuestro script

`#!/bin/bash`

- Para los comentarios se usa el símbolo #.

Ejemplo 1. Crear y ejecutar el siguiente script para listar los archivos del directorio donde se ejecute.

```
herramientas@rogelio-VirtualBox:~$ vi ejemplo1.sh
```

Se crea el archivo

```
#!/bin/bash  
echo "Hola Mundo."  
#Termina mi programa
```

Se teclea su contenido. Aplica lo que aprendiste en la práctica de vimtutor

Comentario

```
herramientas@rogelio-VirtualBox:~$ chmod u+x ejemplo1.sh
```

Se le da permiso de ejecución

```
herramientas@rogelio-VirtualBox:~$ ./ejemplo1.sh  
Hola Mundo.
```

Se ejecuta el script

Variables y parámetros de entrada

- Una variable no necesita ser declarada y pueden ser de cualquier tipo.

Ejemplo 2. Teclea, analiza y ejecuta el siguiente script con variables.

```
#!/bin/bash
var1=20
var2=10
echo "El valor de var1 es $var1"
echo "El valor de var2 es $var2"
echo "La suma de var1 + var2 es $(( $var1 + $var2 ))"
```

Ejemplo 3. Realiza el ejercicio 2 nuevamente, pero ahora pide los valores por teclado.

```
#!/bin/bash
echo 'Escribe el valor de var1:'
read var1
echo 'Escribe el valor de var2:'
read var2
echo "El valor de var1 es $var1"
echo "El valor de var2 es $var2"
echo "La suma de var1 + var2 es $(( $var1 + $var2 ))"
```

Parámetros de entrada

A un script también se le pueden pasar parámetros de entrada, para usarlos dentro del código se usa el símbolo \$ y el número del parámetro. El parámetro 0 es el nombre de nuestro script.

Ejemplo 4. Realiza la suma de dos números en un script que reciba los dos números que se van a sumar.

```
herramientas@rogelio-VirtualBox:~$ vi suma.sh
```

```
#!/bin/bash  
echo "La suma es $(( $1 + $2 ))"
```

```
herramientas@rogelio-VirtualBox:~$ chmod u+x suma.sh  
herramientas@rogelio-VirtualBox:~$ ./suma.sh 5 10  
La suma es 15
```

Estructuras de control

If/elseif/else

La sintaxis de esta estructura de control es la siguiente:

```
if [ condición ]  
then  
    "comandos"  
elseif [ condición ]  
then  
    "comandos"  
else [ condición ]  
    "comandos"  
fi
```

Comparación de cadenas alfanuméricas

Operador	Verdad (TRUE) si:

<code>cadena1 = cadena2</code>	cadena1 es igual a cadena2
<code>cadena1 != cadena2</code>	cadena1 no es igual a cadena2
<code>cadena1 < cadena2</code>	cadena1 es menor que cadena2
<code>cadena1 > cadena 2</code>	cadena1 es mayor que cadena 2
<code>-n cadena1</code>	cadena1 no es igual al valor nulo (longitud mayor que 0)
<code>-z cadena1</code>	cadena1 tiene un valor nulo (longitud 0)

Comparación de valores numéricos

Operador	Verdad (TRUE) si:

<code>x < y</code>	x menor que y
<code>x <= y</code>	x menor o igual que y
<code>x == y</code>	x igual que y
<code>x >= y</code>	x mayor o igual que y
<code>x > y</code>	x mayor que y
<code>x != y</code>	x no igual que y

Ejemplo 5. Crear un script que reciba como parámetros a dos números. Compara el primer parámetro con respecto al segundo. Escribe los tres posibles casos con un if.

```
#!/bin/bash
if [ $1 -lt $2 ]
then
    echo "El número $1 es menor que $2"
elif [ $1 -eq $2 ]
then
    echo "El número $1 es igual que $2"
else
    echo "El número $1 es mayor que $2"
fi
```

```
rmh@rmh-VirtualBox:~/Escritorio/p8$ ./compara.sh 20 100
El número 20 es menor que 100
rmh@rmh-VirtualBox:~/Escritorio/p8$ ./compara.sh 50 50
El número 50 es igual que 50
rmh@rmh-VirtualBox:~/Escritorio/p8$ ./compara.sh 10 5
El número 10 es mayor que 5
```

Estructuras de control

for. el bucle **for** permite iterar sobre una serie de elementos de una cadena, lista, rango, etc.. La sintaxis de esta estructura de control es la siguiente:

```
for variable [in lista]
do
    Instrucciones / comandos que pueden utilizar $variable
done
```

Ejemplo 6. Como ejemplo didáctico genera la tabla de multiplicar del 5 usando un ciclo for.

Opción 1

```
#!/bin/bash
for i in 1 2 3 4 5 6 7 8 9 10
do
    echo "5 x $i = $((5*$i))"
done
```

```
rmh@rmh-VirtualBox:~/Escritorio/p8$ ./ciclo.sh
```

```
5 x 1 = 5
5 x 2 = 10
5 x 3 = 15
5 x 4 = 20
5 x 5 = 25
5 x 6 = 30
5 x 7 = 35
5 x 8 = 40
5 x 9 = 45
5 x 10 = 50
```

Opción 2

```
#!/bin/bash
for i in {1..10}
do
    echo "5 x $i = $((5*$i))"
done
```

Ejemplo 7. Poner en minúsculas y mayúsculas la lista de directorios y archivos que contiene nuestra carpeta de usuario

```
#!/bin/bash
for doc in $(ls)
do
    mayus=$(echo $doc | tr a-z A-Z)
    echo "$doc - $mayus"
done
```

Con \$(instrucción) se ejecuta una instrucción y el resultado se puede guardar en una variable.

```
rmh@rmh-VirtualBox:~/Escritorio/p8$ ./mayus.sh
ciclo2.sh - CICLO2.SH
ciclo.sh - CICLO.SH
compara.sh - COMPARA.SH
ejemplo1.sh - EJEMPL01.SH
ejemplo2.sh - EJEMPL02.SH
ejemplo3.sh - EJEMPL03.SH
mayus.sh - MAYUS.SH
suma.sh - SUMA.SH
```

Estructuras de control

while.

El ciclo **while** también permite iterar, pero verifica una condición antes de cada iteración. La sintaxis de esta estructura de control es la siguiente:

```
while [condición]
do
    #comandos a ejecutar
done
```

Ejemplo 8. Usar un ciclo **while** para imprimir en pantalla los números del 1 hasta un límite que le mandes como parámetro a tu script.

```
#!/bin/bash
numero=1
while [ $numero -le $1 ]
do
    echo "$numero"
    numero=$((numero+1))
done
```

```
rmh@rmh-VirtualBox:~/Escritorio/p8$ ./ciclow.sh 10
```

```
1
2
3
4
5
6
7
8
9
10
```

```
#!/bin/bash
#Coloca aquí tu nombre          #Coloca aquí tu matrícula
opcion="inicio"
while [ $opcion != "fin" ]; do
    echo -e '-----'
    echo "Selecciona una opción"
    echo "1. Sumar"
    echo "2. Restar"
    echo "3. Multiplicar"
    echo "4. Dividir"
    echo "fin. Salir"
    read opcion
    echo "${clear}"
    if [ $opcion != "fin" ]; then
        echo "Teclea el primer número"; read primero
        echo "Teclear el segundo número"; read segundo
        if [ $opcion -eq 1 ]; then
            echo "La suma de $primero + $segundo = $((primero+segundo))"
        elif [ $opcion -eq 2 ]; then
            echo "La resta de $primero - $segundo = $((primero-segundo))"
        elif [ $opcion -eq 3 ]; then
            echo "La multiplicación de $primero * $segundo = $((primero*segundo))"
        elif [ $opcion -eq 4 ]; then
            echo "La división de $primero / $segundo = $((primero/segundo))"
        fi
    fi
done
```

Proyecto: Creación de una calculadora sencilla en shell