# Operaciones aritméticas y lógicas en Shell Script

## Operadores lógicos || &&:

## Los operadores lógicos se suelen utilizar con valores booleanos (valores que devuelven true o false). En el caso de que los valores no sean booleanos, los operadores AND y OR tenderán a convertirlos a booleanos para poder trabajar con ellos.

## Operador OR

Cuando el operador || se usa con valores booleanos,**devuelve true si cualquiera de las dos expresiones es verdadera**. Si ninguna de las expresiones es verdadera, el operador devuelve false.

false || true  
//devuelve true

true || true  
//devuelve true

false || false  
//devuelve false

Cuando los valores no son booleanos, el operador OR convierte el valor de su izquierda a booleano para decidir qué hacer y, en caso necesario, también el de la derecha. Así, en el caso de que el valor de la izquierda sea verdadero, lo devuelve. En caso contrario, devuelve el de la derecha.

## Operador &&

Cuando se usa con valores booleanos, el operador AND **devuelve true si ambas expresiones son verdaderas** y devuelve false en caso contrario. El operador de la izquierda siempre es evaluado y el operador de la derecha solo es evaluado si el primer operador devuelve true.

true && true  
// devuelve true

true && false  
// devuelve false

Cuando no se usa con valores booleanos, el operador AND convierte los valores a booleanos y su funcionamiento es el siguiente: cuando el valor de la izquierda se convierte a falso, devuelve ese valor y, en caso contrario, devuelve el de la derecha.

## Operaciones que podemos hacer

### Aritméticas

En un Bash podemos realizar directamente sumas, restas, divisiones, multiplicaciones y resto de números enteros. Para ello tendemos que poner la operación dentro de $(( )).

Operadores: +, -, /, \*, % (% es el resto que resulta al realizar la división entre dos números)

Por ejemplo:

***$ echo $((25+6))***

***$ echo $((24%5)) daría como resultado 4***

Para operaciones con decimales tendríamos que ayudarnos de comandos externos como bc.

Ejemplo de uso de bc sin entrar en el prompt secundario del programa:

***$echo 6.5\*4.99 | bc***

Si quisiéramos integrarlo en un condicional:

**$if test $((25+6)) = 31; then echo “31”; fi**

ó

**$if [ $((25+6)) = 31 ]; then echo “31”; fi**

Ó en estructura de script:

***$n1=25***

***$n2=6***

***$if [ $(($n1+$n2)) = 31 ]; then echo “31”; fi***

### COMPARACIÓN DE NÚMEROS:

Si el objetivo es comparar dos valores podemos usar el comando test. La expresión sería:

**$ test “25” = “6”**

**$echo $?**

**1**

Si los dos valores fueran iguales, echo $? Devolverá 0.

Si quisiéramos probar con variables:

**$ n1=25**

**$ n2=6**

**$ test “$n1” = “$n2”**

**$ echo $?**

**1**

NOTA:

Si concatenamos los dos comandos en uno, de tal manera que quedaría:

**$ test “$n1” = “$n2”|| echo $?**

**1**

Se ejecutan los dos comandos siempre y cuando se cumpla que no hay igualdad. Si con los mísmos parámetros anteriores cambiamos a:

**$ test “$n1” = “$n2”&& echo $?**

No se ejecutará el segundo comando, dado que la primera igualdad no se cumple.

**NO ES ACONSEJABLE USAR || y && al comparar de manera generalizada, dado que funcionará o no según la salida del primer comando.**

### Lógicas

La finalidad de las operaciones lógicas es determinar si algo es verdadero o falso. Para poder hacer multitud de comprobaciones, tenemos la orden test. Se puede ejecutar con el comando test seguido de la comprobación que queremos hacer, o delimitar la comprobación entre corchetes.

Es decir, si quiero comprobar si $A es igual a $B puedo poner **test $A = $B**, pero también puedo poner **[ $A = $B ]**

#### Comparación de números

***Partimos de las siguientes variables:***

***$n1=25***

***$n2=6***

**SINTAXIS DESCRIPCIÓN**

**ENTERO1 -eq ENTERO2** ENTERO1 es igual que ENTERO2

***$if test $n1 -eq $n2; then echo "son iguales"; else echo "son distintos"; fi***

***$if [ $n1 -eq $n2 ]; then echo "son iguales"; else echo "son distintos"; fi***

**ENTERO1 -ge ENTERO2** ENTERO1 es mayor o igual que ENTERO2

**$if [ $n1 -ge $n2 ]; then echo "mayor"; else echo "menor"; fi**

**ENTERO1 -gt ENTERO2** ENTERO1 es mayor que ENTERO2

**$if [ $n1 -gt $n2 ]; then echo "mayor"; else echo "menor"; fi**

**ENTERO1 -le ENTERO2** ENTERO1 es menor que o igual que ENTERO2

**ENTERO1 -lt ENTERO2** ENTERO1 es menor que ENTERO2

**ENTERO1 -ne ENTERO2** ENTERO1 no es igual que ENTERO2

#### Comparación de texto (ojo depende de la implementación)

**SINTAXIS DESCRIPCIÓN**

**CADENA1 = CADENA2** CADENA1 es igual que CADENA2

**$if [ "hola" = "holas" ]; then echo "hola"; else echo "distintos";fi**

***$if [ "hola" == "hola" ]; then echo "hola"; else echo "distintos";fi***

**CADENA1 != CADENA2** CADENA1 no es igual que CADENA2

***$if [ hola != adios ] ; then echo distintos; else echo iguales; fi***

#### Comprobaciones sobre ficheros y directorios

Si echo $? Devuelve 0: el test es correcto

Si echo $? Devuelve 1:el test es incorrecto

SINTAXIS DESCRIPCIÓN

**-d RUTA Cierto si la RUTA existe y es un directorio**

Ejemplos:

$ test -d /etc/passwd

$ echo $?

$ [ -d /etc ] || echo "/etc no es un archivo estándar"

/etc no es un archivo estándar

**-e RUTA Cierto si la RUTA existe sea el elemento que sea**

$ test -e /etc/netplan

$ echo $?

$ [ -d /etc/netplan ] || echo "/etc/netplan no es un archivo estándar"

/etc no es un archivo estándar

-f RUTA Cierto si la RUTA existe y es un fichero normal:

$ test -f /etc/passwd

$ echo $?

ó

$ [ -f /etc ] || echo "/etc no es un archivo estándar"

/etc no es un archivo estándar

-r RUTA Cierto si la RUTA existe y se puede leer

-w RUTA Cierto si la RUTA existe y se puede escribir

-x RUTA Cierto si la RUTA existe y es ejecutable

-s RUTA Cierto si la RUTA existe y su tamaño es mayor que cero

Uso principal de las comparaciones y comprobaciones:

La utilidad que tienen las operaciones lógicas es la toma de decisiones.

Suponiendo que hacemos un script que le indicamos con el primer parámetro nuestra edad, haríamos:

Si $edad es mayor o igual que 18

muestro un mensaje de que eres mayor de edad

en caso contrario

muestro un mensaje diciendo que no lo eres

En el lenguaje de script:

***edad=14***

***if [ $edad -ge 18 ]; then***

***echo “Eres mayor de edad”***

***else***

***echo “Eres menos de edad”***

***fi***

***Así podemos construir estructuras condicionales.***