Curso 1º SMR

Módulo: SOM

Sesión 2: Programación shell: comparación

¿Qué pretendemos conseguir?

Conocer los operadores de comparación.
Conocer los errores más comunes al utilizar los operadores de comparación.

Desarrollo de la sesión

Para las sesiones siguientes vamos a necesitar que nuestros script tomen decisiones. Y estas decisiones las tomarán comparando valores.

En programación shell se utilizan los corchetes (**[]**) para delimitar la expresión de comparación.

Los operadores de comparación si los valores son numéricos:

```
-eq  igual
-ne  no igual
-gt  mayor que
-lt  menor que
-ge  mayor o igual que
-le  menor o igual que
```

Ejemplo:

```
A=7
B=8
[ $A -eq $B ]
```

Si los valores son cadenas de caracteres:

```
=  igual
!= distinto
-z Cadena de longitud 0
-n Cadena de longitud mayor que 0
```

Aquí podemos observar que existen operadores unarios (sólo requieren un valor) como on -z y -n, y operadores binarios (requieren, igual que los operadores numéricos vistos anteriormente, de dos valores) = y !=

Si las variables almacenan nombre de archivos o directorios:

```
 a archivo existente
-d directorio
-w  archivo con permiso de escritura
```

-r archivo con permiso de lectura
-x archivo con permiso de ejecutable.

Ejemplo:

[-a '/etc/samba/smb.conf'] Será verdadero si existe el fichero que se encuentra entre comillas.

Para que una comparación esté correctamente creada debe guardarse un espacio entre el corchete de apertura y el primer carácter de la expresión, y un espacio entre el último carácter y el corchete de cierre.

Pueden aparecer errores si en las expresiones de comparación se utilizan variables y estas no tienen ningún valor. Pongamos por caso que la variable A está en blanco, la siguiente expresión dará un error:

```
[ $A -eq 1 ]
```

Ejercicio1: ¿Qué devuelven las siguientes expresiones de comparación:

```
[ 1 -eq 2 ]
[ $A -lt 7 ] → A=6
[ $B -gt 8 ] → B=6
[ -z 'ejercicio']
[ 'ejercicio' != $cadena ] → cadena='ejercicio'
[ -d '/home/madrid' ]?
```

Ejercicio2: ¿Qué está mál escrito en las siguientes expresiones:

```
[1 -eq $A]
[ A -lt 2 ]
[-z $cadena]
[ 7 = $Z ]?
```

Ejercicio3: Crea una expresión de comparación en la que se devuelva verdadero si el contenido de la variable fichero es el nombre de un fichero ejecutable en tu sistema.

Ejercicio4: Crea una expresión de comparación que devuelva verdadero si el valor contenido en la variable A es menor que 11.

Ejercicio5: Crea una expresión de comparación que devuelva verdadero si el valor contenido en la variable A es menor o igual que 11.

Las expresiones lógicas pueden combinarse para formar expresiones más complejas. Para ello utilizaremos los operadores **&&** (y lógico) o **||** (o logico).

Ten en cuenta las siguientes tablas de verdad:

Verdadero && Verdadero → Verdadero
Verdadero && Falso → Falso
Falso && Verdadero → Falso
Falso && Falso → Falso

Verdadero || Verdadero → Verdadero
Verdadero || Falso → Verdadero
Falso || Verdadero → Verdadero
Falso || Falso → Falso

Ejercicio6: Cuales de estas sentencias son verdaderas:
Para A = 5

```
[ -r '/home/madrid/.bashrc' ] && [ $A -eq 7 ]
[ -r '/home/madrid/.bashrc' ] || [ $A -eq 7 ]
[ $A -lt 4 ] || [ $A -gt 4 ]
[ $A -lt 4 ] && [ $A -gt 4 ]
```