# Shell Script: Variables

Todo lenguaje de programación necesita lugar donde almacenar la información para luego procesarla según su uso. Este lugar se llama “Variable”, un lugar en la memoria principal que se reserva para que el programa pueda guardar o acceder de una forma muy rápida a la información que contiene.

Cada variable tendrá un identificador que estará compuesto por caracteres alfa numéricos y/o guion bajo, con la restricción de que no pueden empezar por un número. Como casi todo en Linux, los nombres de variables son Case Sensitive (se distingue entre mayúsculas y minúsculas), por lo que no será lo mismo VAR que Var

Si vamos a leer el contenido de una variable tenemos que añadir el símbolo del dólar al inicio del nombre, pero para “escribirla”, para asignarle contenido, se utiliza SIN el símbolo del dólar. Si quisiera guardar una frase para luego sacarla por pantalla, haría:

FRASE=”Soy una frase”

echo $FRASE

Es MUY importante que cuando le asignamos un valor a una variable NO dejemos ningún espacio entre el final del nombre de la variable y el signo igual (=) ni entre el signo igual y el contenido de la variable. Debe ir todo junto.

Cuando necesitamos acotar el nombre de la variable para evitar errores podemos utilizar la nomenclatura ${VAR} , así el nombre queda entre llaves y no se mezcla con ninguna otra parte del script.

## Variables predefinidas

El sistema dispone de numerosas variables con información valiosa que están a nuestra disposición para poder acceder a la información que contienen.

### Variables de entorno

Contienen información respecto al entorno de ejecución. Por ejemplo:

$USER contiene el nombre del usuario actual

$PWD directorio de trabajo actual

$HOSTNAME nombre del ordenador

$LANG idioma del sistema

$OSTYPE tipo de sistema operativo

Otras variables útiles

También disponemos de unas variables que contienen información referente a la propia ejecución del script. Son fundamentales las que hacen referencia al paso de parámetros (lo explicaremos más adelante).

$0 El nombre del script

$$ PID del proceso que se le ha asignado al script en ejecución

$? Resultado devuelto por el último proceso ejecutado.

$n la información de un parámetro en concreto, siendo n un número del parámetro

$\* todos los parámetros. Se presentan en una sola cadena de caracteres

$@ todos los parámetros. Es una lista con un elemento por cada parámetro recibido

$# la cantidad de parámetros con los que se ha invocado el script. Será una sola cifra

## Arrays

Se podría definir un array como una variable estructurada que permite almacenar secuencialmente un conjunto de datos del mismo tipo y relacionados. Para acceder a los elementos individuales de un array se necesita un índice.

Si nos imaginamos una variable como una caja vacía donde guardar información, un array sería un archivador con muchos cajones.

Definición de arrays

Para crear un array con valores definidos usamos:

miarray=(dato1 dato2 dato3)

Si queremos crearlo vacío:

miarray=()

Lectura de arrays

Para acceder a la información no basta con el nombre del array. Tenemos que indicar también la posición a la que queremos acceder (indice).

${miarray[indice]}

Ejemplo de uso:

animales=(perro gato loro)

echo “Yo tengo un ${animales[0]} y un ${animales[1]}”

echo “Pero también me gustaría tener un ${animales[2]}

Si utilizamos un array normal (no asociativo) los índices son numéricos y empiezan desde el 0 a n-1 (siendo n la cantidad de elementos que tenga el array). También podemos acceder empezando desde la última posición utilizando números negativos, desde la posición -1 (el último) hasta -n (el primero). Con este método el ejemplo anterior quedaría así:

echo . “Yo tengo un ${animales[-3]} y un ${animales[-2]}”

echo “Pero también me gustaría tener un ${animales[-1]}

### Añadir elementos

Si queremos añadir un elemento al final de un array:

miarray+=(elemento)

### Información del array

Existen unos método para manejar la información del propio array:

Mostrar todos los elementos: ${miarray[\*]} o ${miarray[@]}

Mostrar la cantidad de elementos: ${#miarray[@]}

Mostrar todos los indices: ${!miarray[@]}

Eliminar una posición: unset miarray[indice]

### Parámetros

Los comandos de Linux suelen recibir parámetros para indicarle el objeto sobre el que tienen que realizar la acción deseada, a un script también se le pueden pasar parámetros cuando se invoca. Después del nombre del ficheros del script cada palabra separada por espacios se considera un parámetro que se envía al script.

Ya hemos comentado las variable predefinidas que podemos consultar para saber más sobre los parámetros que se le han pasado al script ($\*,$@,$#), ¿pero cómo accedemos a la información de cada parámetro?.

Utilizaremos $1 para el primer parámetro, $2 para el segundo etc. hasta $9.

Ejemplo:

Si llamamos al script así:

./miscript.sh perro gato loro

Podemos acceder a los parámetros así:

echo “Yo tengo un $1 . y un $2”

echo “Pero también me gustaría tener un $3