PARÁMETROS

- Un script puede recibir información mediante parámetros, que serán cadenas de texto que pondremos después del nombre del script
- Estarán separado por espacios
- Dentro del script podremos acceder a ellos mediante variables predefinidas del sistema:
 - √\$n => la información de un parámetro en concreto, siendo n un número del parámetro
 - $\sqrt{\$*}$ => todos los parámetros. Se presentan en una sola cadena de caracteres
 - $\sqrt{\$}$ => todos los parámetros. Es una lista con un elemento por cada parámetro recibido
 - $\sqrt{\$}$ # => el número de parámetros con los que se ha invocado el script. Será una cifra

VARIABLES

- > Sirven para guardar información y poder acceder a ella posteriormente
- Para referirnos a ellas utilizaremos un identificador que estará compuesto por caracteres alfa numéricos o guión bajo, pero no pueden empezar por un número
- > Se distingue entre mayúsculas y minúsculas. \$VAR no es igual que \$var

• LECTURA / ESCRITURA DE VARIABLES

Para guardar datos en una variable simplemente tendremos que asignarle un valor, utilizando su identificador SIN el símbolo del dólar:

```
variable="Hola" (es importante que todo esté junto, sin espacios)
```

Y para acceder a su contenido usaremos el símbolo del dólar seguido de su identificador:

```
echo $variable ollaves: echo ${variable}
```

MÁS VARIABLES PREDEFINIDAS

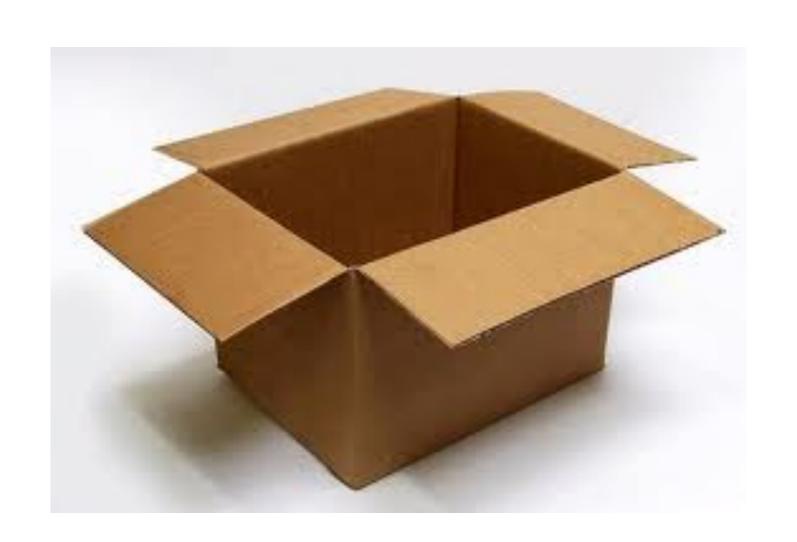
- Además de las variables para consultar los parámetros, disponemos de algunas similares pero que contiene otra tipo de información:
 - √ \$0 => El nombre del script
 - √ \$\$ => PID del proceso que se le ha asignado al script en ejecución
 - √ \$? => Resultado devuelto por el último proceso ejecutado.

VARIABLES DE ENTORNO

- > Son variables que están definidas en el sistema y que podemos usar si las necesitamos.
- > Son usadas por el propio sistema para su funcionamiento, por ejemplo \$PATH
- Podemos ver las disponibles con el comando

ARRAYS

- Es 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









DEFINICIÓN

```
miarray=(dato1 dato2 dato3)
```

LECTURA

```
${miarray[indice]} => echo ${miarray[0]} ${miarray[1]} ${miarray[-1]}
```

OTROS MÉTODOS

- ▶ Mostrar todos los elementos: \${miarray[*]} o \${miarray[@]}
- ▶ Mostrar la cantidad de elementos: \${#miarray[@]}
- Mostrar todos los indices: \${!miarray[@]}
- Eliminar una posición: unset miarray[indice]