

Shell Scripts

Operaciones aritméticas (+ - / *)

```
a=64;b=16
```

```
resultado=$((a+b+10)); echo $resultado
```

```
((a++)); echo $a
```

Evaluación de expresiones

test ← equivalente a []

man test (según indica el manual verifica tipos de ficheros y compara valores)

En el manual puedes encontrar todos los posibles operadores (cadenas, números, ficheros)

Es importante respetar los espacios entre operandos y operadores.

En la asignación de valores a variables no debe haber espacios antes/después del signo =

Comparar cadenas:

```
a="AAA";b="BBB";c="AAA"
```

```
test $a = $b && echo iguales || echo distintos
```

```
[ $a = $b ] && echo iguales || echo distintos
```

```
if [ $a = $b ]; then echo "son iguales"; else echo "son distintos"; fi
```

```
if [ $a = $b ]
then
    echo "iguales"
else
    echo distintos
fi
```

Comparar números:

```
n1=64; n2=87; n3=64
```

```
[ $n1 -eq $n3 ] && echo iguales || echo distintos
```

```
[ $n1 -lt $n2 ] && echo "$n1 es menor que $n2"
```

Verificar ficheros:

```
if [ -d rutaDirectorio ]; then echo "el directorio existe"; fi
```

```
[ -O archivo ] && echo el archivo es de tu propiedad
```

```
if [ $1 -gt 0 ]; then
    echo "$1 es positivo"
elif [ $1 -lt 0 ]; then
    echo "$1 es negativo"
elif [ $1 -eq 0 ]; then
    echo "es cero"
else
    echo "$1 no es un número"
fi 2>/dev/null
```

Shell Scripts

Parámetros posicionales

Cuando se ejecuta un *script* se pueden pasar parámetros (argumentos) que se podrán procesar refiriéndose a ellos por el orden en que se expresaron.

`./scriptParametros uno "dos tres" cuatro`

\$0 → nombre del script (*scriptParametros.sh*)

\$1 → *uno*

\$2 → *dos tres*

\$3 → *cuatro*

Se pueden desplazar todos los parámetros con el comando **shift** ($\$2 \rightarrow \1 , $\$3 \rightarrow \2 ,...)

\$* → todos los parámetros como un solo valor

```
for p in "$*"; do echo $p; done  
uno dos tres cuatro
```

\$@ → todos los parámetros recogidos de forma individual

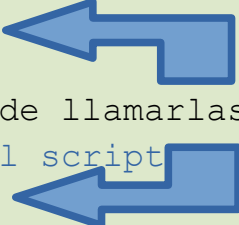
```
for p in "$@"; do echo $p; done  
uno  
dos tres  
cuatro
```

\$# → total de parámetros

\$? → valor devuelto tras la ejecución del script (**return** valor, **exit** valor o **código de error**)

Funciones

```
#!/bin/bash  
function funcion1() {  
    echo "función1"  
}  
function2() {  
    echo -n "La suma es... "  
    resultado=$(( $1+$2 ))  
    return $resultado  
}  
#Las funciones deben estar definidas antes de llamarlas  
#A partir de aquí se inicia la ejecución del script  
funcion1  
funcion2 54 10 #paso como parámetros los valores 54 y 10  
echo $?, $resultado #muestro el valor de retorno de la función
```



Shell Scripts

Alternativas con **case**:

```
#!/bin/bash
tipo=$1 # $1: argumento del script
case $tipo in
"A")
    echo "es de tipo A"
    ;;
"B")
    echo "es de tipo B"
    ;;
"C")
    echo "es de tipo C"
    ;;
*)
    echo "tipo desconocido"
esac
```

Los ;; provocan la salida del *case* una vez verificado el caso.

Bucles

```
#!/bin/bash
n=1
while [ $n -le 10 ]
do
    echo "vuelta $n"
    ((n++))
done
```

```
#!/bin/bash
n=1
until [ $n -gt 10 ]
do
    echo "vuelta $n"
    ((n++))
done

#!/bin/bash
for (( n=1; n<=10; n++ ))
do
    echo "vuelta $n"
done
```

Shell Scripts

```
#!/bin/bash
for n in {01..10}
do
    echo "vuelta $n"
done
```

```
for n in {A..Z}; do echo -n $n; done; echo
```

```
for f in *; do echo $f; done
```

```
for e in uno dos "tres cuatro" cinco; do echo $e; done
```

```
lista=(uno "dos tres" cuatro)
for e in ${lista[*]}; do echo $e; done
for e in "${lista[@]}"; do echo $e; done #para evitar problemas con los delimitadores IFS
```

```
#!/bin/bash
opciones=(uno dos tres cuatro salir)
echo "seleccione su opción:"
select op in "${opciones[@]}"
do
    echo "has elegido $op"
    [ $op = "salir" ] && break
done
```

expresiones regulares

Validar una dirección de correo electrónico:

```
unset a
regex="^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,4}$"
until [[ $a =~ $regex ]]; do read a; done
```

Validar un parámetro numérico:

```
if [[ ! ($1 =~ ^[[:digit:]]+$) ]]; then
    echo "no es un número"
fi
```

Shell Scripts

Archivos

```
#!/bin/bash
# crea un archivo agenda.csv
while true
do
    read -p "Nombre: " nombre
    if [ -z "$nombre" ]; then
        exit
    fi
    read -p "Apellidos: " apellidos
    read -p "Teléfono: " telefono
    echo $nombre,$apellidos,$telefono
done >> agenda.csv #redirigimos la salida del echo al archivo agenda.csv
#como es una redirección doble (>>), se añadirán las líneas al archivo
#si la redirección fuera simple (>) se crearía el archivo de nuevo
```

```
#!/bin/bash
# lee el archivo agenda.csv
IFS="," #establezco como separador de campos la ,
while read nombre apellidos telefono
do
    echo "Nombre: " $nombre
    echo "Apellidos: " $apellidos
    echo "Teléfono: " $telefono
done < agenda.csv #redirecciono la entrada del archivo hacia el read
echo "*****"

#otra forma, leyendo cada línea como un array:
while read -a registro
do
    echo "Nombre: " ${registro[0]}
    echo "Apellidos: " ${registro[1]}
    echo "Teléfono: " ${registro[2]}
    echo -----
done < agenda.csv
```