

# SHELL: SCRIPTS

---



---

IMPLANTACIÓN DE SISTEMAS  
JUAN CARLOS NAVIDAD GARCÍA

1. Realizar un script llamado '01-hola-mundo.sh' que muestre por pantalla: "Hola mundo!".

```

[+] juan@juan-vmware: ~
GNU nano 6.4                                01-hola-mundo.sh *
echo "Hola mundo!"

```

```

juan@juan-vmware:~$ sudo chmod +x 01-hola-mundo.sh
juan@juan-vmware:~$ ./01-hola-mundo.sh
Hola mundo!

```

2. Realizar un script llamado ('02-hola-parametros.sh') que muestre:

**NOTA: Ejecuta el script con 3 parámetros posicionales**

- 2.1. Nombre del script.
- 2.2. Todos los parámetros introducidos.
- 2.3. El valor del segundo parámetro introducido.
- 2.4. El número de parámetros introducidos.

```

[+] juan@juan-vmware: ~
GNU nano 6.4                                02-hola-mundo.sh
echo "Nombre del script:" $0;
echo "Todos los parametros menos el 0": $*;
echo "Parametro dos": $2;
echo "Numero de parametros": $#;

```

```

juan@juan-vmware:~$ ./02-hola-mundo.sh juan carlos navidad
Nombre del script: ./02-hola-mundo.sh
Todos los parametros menos el 0: juan carlos navidad
Parametro dos: carlos
Numero de parametros: 3

```

3. Realizar un script llamado '03-hola-al-menos-1-parametro.sh' que verifique que al menos hayamos introducido un parámetro.

```

[+]
juan@juan-vmware: ~
GNU nano 6.4
02-hola-mundo.sh
echo "Se han introducido: " $# "parametros";

```

```

juan@juan-vmware:~$ sudo nano 02-hola-mundo.sh
juan@juan-vmware:~$ sudo chmod +x 02-hola-mundo.sh
juan@juan-vmware:~$ ./02-hola-mundo.sh
Se han introducido: 0 parametros
juan@juan-vmware:~$ ./02-hola-mundo.sh hola
Se han introducido: 1 parametros

```

4. Ídem y que además separe cada argumento por ", " ('04-hola-parametros-separados.sh').

```

[+]
juan@juan-vmware: ~
GNU nano 6.4
04-hola-mundo.sh *
echo "$* | tr ' ' ',';

```

```

juan@juan-vmware:~$ ./04-hola-mundo.sh juan carlos navidad
juan,carlos,navidad

```

5. Realiza un script que verifique que los parametros pasados sean usuarios conectados al sistema ('05-hola-usuario.sh').

```

jua@jua-vmware: ~
GNU nano 6.4                                05-hola-mundo.sh *
for usuario in "$@"; do
    if who | grep -qw "^$usuario "; then
        echo "$usuario está conectado al sistema."
    else
        echo "$usuario no está conectado al sistema."
    fi
done
```

```

jua@jua-vmware:~$ ./05-hola-mundo.sh jua migue antonio
jua está conectado al sistema.
migue no está conectado al sistema.
antonio no está conectado al sistema.
jua@jua-vmware:~$
```

6. Realizar un script llamado 'usuarioconectado' que retorna un SI si el primer parámetro coincide con algún usuario conectado o NO en caso contrario.

```

jua@jua-vmware: ~
GNU nano 6.4                                usuarioconectado.sh
if who | grep -qw "^$1 "; then
    echo "SI"
else
    echo "NO"
fi
```

```

jua@jua-vmware:~$ ./usuarioconectado.sh jua migue antonio
SI
jua@jua-vmware:~$ ./usuarioconectado.sh migue jua antonio
NO
```

7. Modificar el fichero `'.bashrc'` para modificar el PATH y añadir la carpeta de estos ejercicios. Para ello añade la siguiente línea: `export PATH=$PATH:~/ruta_carpeta_ejercicios`

```

jua@jua-vmware: ~/scripts
GNU nano 6.4                               ../.bashrc *
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc pack

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need t
# this, if it's already enabled in /etc/bash.bashrc and /et
# sources /etc/bash.bashrc).
if ! shopt -oq posix; then
    if [ -f /usr/share/bash-completion/bash_completion ]; the
        . /usr/share/bash-completion/bash_completion
    elif [ -f /etc/bash_completion ]; then
        . /etc/bash_completion
    fi
fi

export PATH=$PATH:~/scripts"

```

8. Modificar el script `'05-hola-usuario.sh'` para que llame a `'usuarioconectado'` (`'08-hola-usuario.sh'`).

```

jua@jua-vmware: ~/scripts
GNU nano 6.4                               08-hola-usuario.sh
for usuario in "$@"; do
    if ./usuarioconectado.sh "$usuario" | grep -q "SI"; then
        echo "$usuario está conectado al sistema."
    else
        echo "$usuario no está conectado al sistema."
    fi
done

jua@jua-vmware:~/scripts$ ./08-hola-usuario.sh jua
jua está conectado al sistema.

```

9. Realizar un script llamado 'usuariosistema' que retorna un SI si el primer parámetro coincide con algún usuario del sistema o NO en caso contrario.

```

[+] juan@juan-vmware: ~/scripts
GNU nano 6.4 usuariosistema.sh
if who | grep -qw "^$1 "; then
    echo "SI"
else
    echo "NO"
fi

juan@juan-vmware:~/scripts$ ./usuariosistema.sh juan miguel antonio
SI
juan@juan-vmware:~/scripts$ ./usuariosistema.sh luis miguel antonio
NO

```

10. Modificar el script '08-hola-usuario.sh' para que llame a 'usuariosistema' ('10-hola-usuario.sh').

```

[+] juan@juan-vmware: ~/scripts
GNU nano 6.4 10-hola-usuario.sh *
for usuario in "$@"; do
    if ./usuariosistema.sh "$usuario" | grep -q "SI"; then
        echo "$usuario está conectado al sistema."
    else
        echo "$usuario no está conectado al sistema."
    fi
done

juan@juan-vmware:~/scripts$ ./10-hola-usuario.sh juan
juan está conectado al sistema.
juan@juan-vmware:~/scripts$ ./10-hola-usuario.sh luis
luis no está conectado al sistema.

```

11. Realizar un script llamado 'suma' que realice la suma de 2 parámetros introducidos (tendrá que poder sumar números decimales, como  $2.2 + 3$ ).

```

[+] juan@juan-vmware: ~/scripts
----- GNU nano 6.4 ----- suma.sh *
resultado=$(echo "$1 + $2" | bc)
echo "La suma de $1 + $2 es igual a $resultado"

juan@juan-vmware:~/scripts$ ./suma.sh 1 2.3
La suma de 1 + 2.3 es igual a 3.3
```

12. Realizar un script llamado 'resta' que realice la resta de 2 parámetros introducidos (tendrá que poder restar números decimales, como  $2.2 - 3$ ).

```

[+] juan@juan-vmware: ~/scripts
----- GNU nano 6.4 ----- resta.sh *
resultado=$(echo "$1 - $2" | bc)
echo "La resta de $1 - $2 es igual a $resultado"

juan@juan-vmware:~/scripts$ ./resta.sh 5 2.3
La resta de 5 - 2.3 es igual a 2.7
```

13. Realizar un script llamado 'multiplica' que multiplique los 2 parámetros introducidos (tendrá que poder multiplicar números decimales, como  $2.2 * 3$ ).

```

[+] juan@juan-vmware: ~/scripts
----- GNU nano 6.4 ----- multiplica.sh *
resultado=$(echo "$1 * $2" | bc)
echo "La multiplicacion de $1 * $2 es igual a $resultado"

juan@juan-vmware:~/scripts$ ./multiplica.sh 5 2.3
La multiplicacion de 5 * 2.3 es igual a 11.5
```

**14. Realizar un script llamado 'division' que realice la división de 2 parámetros introducidos (tendrá que poder sumar números decimales, como 2.2 / 3).**

```

juan@juan-vmware: ~/scripts
GNU nano 6.4                                division.sh *
resultado=$(echo "$1 / $2" | bc)
echo "La division de $1 / $2 es igual a $resultado"

juan@juan-vmware:~/scripts$ ./division.sh 10 2.5
La division de 10 / 2.5 es igual a 4

```

**15. Realizar un script llamado 'calc01.sh' que realice operaciones básicas entre 2 números llamando a cada uno de los scripts anteriormente creados (suma, resta, multiplicación y división).**

```

juan@juan-vmware: ~/scripts
GNU nano 6.4                                calc01.sh
./suma.sh $1 $2
./resta.sh $1 $2
./multiplica.sh $1 $2
./division.sh $1 $2

juan@juan-vmware:~/scripts$ ./calc01.sh 8 3
La suma de 8 + 3 es igual a 11
La resta de 8 - 3 es igual a 5
La multiplicacion de 8 * 3 es igual a 24
La division de 8 / 3 es igual a 2

```



## 16. Ídem pero sin llamar a los scripts ('calc02.sh').

```
juan@juan-vmware: ~/scripts
GNU nano 6.4 calc02.sh
suma=$(echo "$1+$2" | bc)
echo "La suma de $1 + $2 es igual a $suma"
resta=$(echo "$1-$2" | bc)
echo "La resta de $1 - $2 es igual a $resta"
multiplica=$(echo "$1*$2" | bc)
echo "La multiplicacion de $1 x $2 es igual a $multiplica"
division=$(echo "$1/$2" | bc)
echo "La division de $1 / $2 es igual a $division"
```

```
juan@juan-vmware:~/scripts$ ./calc02.sh 8 3
La suma de 8 + 3 es igual a 11
La resta de 8 - 3 es igual a 5
La multiplicacion de 8 x 3 es igual a 24
La division de 8 / 3 es igual a 2
```

## 17. Realizar un script llamado 'citas-menu.sh' que muestre un menú con las siguientes opciones:

1. Añadir cita nueva.
2. Buscar por nombre del paciente.
3. Buscar citas por hora inicial.
4. Buscar citas por hora final.
5. Listar las citas ordenadas por nombre del paciente.
6. Listar las citas ordenadas por hora inicial.
7. Salir del programa.

Después, modifica el script citas.menu.sh al que llamaras citas.sh para que defina las funciones para realizar las opciones del menú.

NOTA: necesitaras un fichero citas.txt con el formato:  
nombrePaciente:horaInicio:horaFin

```

juan@juan-vmware: ~/scripts
GNU nano 6.4 citas-menu2.sh
opcion1() {
echo " "
read -p "Escriba el nombre del paciente: " nombre
read -p "Escriba la hora de inicio de la cita: " hora_inicio
read -p "Escriba la hora de finalizacon de la cita : " hora_fin
echo "$nombre $hora_inicio $hora_fin" >> citas.txt
clear
}
opcion2() {
clear
echo " "
read -p "Escriba el nombre del paciente que quiere buscar: " nombre
echo " "
cat citas.txt | grep "$nombre"
}

```

```

opcion6() {
clear
echo " "
sort -k 2 citas.txt
}

```

```

while true; do
    mostrar_menu
    read -p "Elija una opción: " opcion

    case $opcion in
        1)
            opcion1
            ;;
        2)
            opcion2
            ;;
        3)
            opcion3
            ;;
        4)
            opcion4
            ;;
        5)
            opcion5
            ;;
        6)
            opcion6
            ;;
        7)
            echo "Hasta luego..."
            break
            ;;
        *)
            echo "Opción no válida. Introduce una opción del 1 al 7."
            ;;
    esac
done

```

```

esac
done

```

```

opcion3() {
clear
echo " "
read -p "Escriba la hora de inicio de la cita: " hora_i
echo " "
cat citas.txt | grep "$hora_i"
}
opcion4() {
clear
echo " "
read -p "Escriba la hora de finalizacion de la cita: " hora_f
echo " "
cat citas.txt | grep "$hora_f"
}
opcion5() {
clear
echo " "
sort -k 1 citas.txt
}

```

```

mostrar_menu() {
echo " "
echo "1. Añadir cita nueva."
echo " "
echo "2. Buscar por nombre del paciente."
echo " "
echo "3. Buscar citas por hora inicial."
echo " "
echo "4. Buscar citas por hora final."
echo " "
echo "5. Listar las citas ordenadas por nombre del paciente."
echo " "
echo "6. Listar las citas ordenadas por hora inicial."
echo " "
echo "7. Salir del programa."
echo " "
}

```

**18. Realizar un script llamado 'ordena' que liste el contenido del directorio actual ordenado por tamaño del archivo de menor a mayor. El listado sólo mostrará el nombre de los archivos y el número de línea correspondiente. En el caso de que se introduzca algún parámetro se mostrará el siguiente mensaje de error: "No se permiten parámetros." y retornará un código de retorno igual a 1.**

```

  GNU nano 6.4                                ordena.sh
  if [ $# -ne 0 ]; then
    echo "No se permiten parámetros."
    exit 1
  fi
  ls -S | cat -n | cut -f 2
```

```

juan@juan-vmware:~/scripts$ ./ordena.sh hola
No se permiten parámetros.
```

```

juan@juan-vmware:~/scripts$ ./ordena.sh
citas-menu2.sh
citas-menu.sh
calc02.sh
08-hola-usuario.sh
10-hola-usuario.sh
05-hola-mundo.sh
02-hola-mundo.sh
ordena.sh
citas.txt
multiplica.sh
division.sh
resta.sh
suma.sh
calc01.sh
usuarioconectado.sh
usuariosistema.sh
04-hola-mundo.sh
01-hola-mundo.sh
```

**19. Realizar un script llamado 'jaula' que cree, sólo si no existe, el directorio jaula en la \$HOME del usuario y mueva los ficheros pasados por parámetro a dicho directorio.**

**19.1. En el caso de que no se le pase ningún parámetro se mostrará el siguiente mensaje de error: "Hay que introducir al menos un parámetro." y retornará un código de retorno igual a 1.**

**19.2. En el caso de que algún fichero introducido por parámetro no exista se mostrará el siguiente mensaje de error: "El fichero '\$FICHERO' no existe." y retornará un código de retorno igual a 2.**

**19.3. Si el fichero .jaula existe en la \$HOME del usuario pero no es un directorio mostrará el siguiente mensaje de error: "El fichero '\$HOME/.jaula' no es un directorio." y retornará un código de retorno igual a 3.**

```

jaula.sh *
GNU nano 6.4
if [ $# -eq 0 ]; then
    echo "Debe proporcionar al menos un parámetro"
    exit 1
fi
if [ -e "$HOME/.jaula" ]; then
    if [ ! -d "$HOME/.jaula" ]; then
        echo "El fichero '$HOME/.jaula' no es un directorio."
        exit 3
    fi
else
    echo "Creando el directorio .jaula en $HOME..."
    mkdir "$HOME/.jaula"
fi
for archivo in "$@"; do
    if [ -e "$archivo" ]; then
        mv "$archivo" "$HOME/.jaula"
        echo "Ficheros movidos exitosamente."
    else
        echo "El fichero '$archivo' no existe."
        exit 2
    fi
done
```

```

juan@juan-vmware:~/scripts$ ./jaula.sh
Debe proporcionar al menos un parámetro
```


```

juan@juan-vmware:~/scripts$ ./jaula.sh hola
Creando el directorio .jaula en /home/juan...
El fichero 'hola' no existe.
```

```

juan@juan-vmware:~/scripts$ ./jaula.sh $HOME/frutas.txt
Ficheros movidos exitosamente.
```

20. Crear un script llamado 'array.sh' que declare un array, lo rellene con datos y luego itere sobre el mismo para mostrar los datos.



```

jua@jua-vmware: ~/scripts
GNU nano 6.4 array.sh
array=("$1" "$2" "$3" "$4")

# Iterar sobre el array y mostrar los datos
echo "Los datos del array son:"
for dato in "${array[@]}"
do
    echo "$dato"
done

jua@jua-vmware:~/scripts$ ./array.sh jua carlos navidad garcia
Los datos del array son:
jua
carlos
navidad
garcia
```