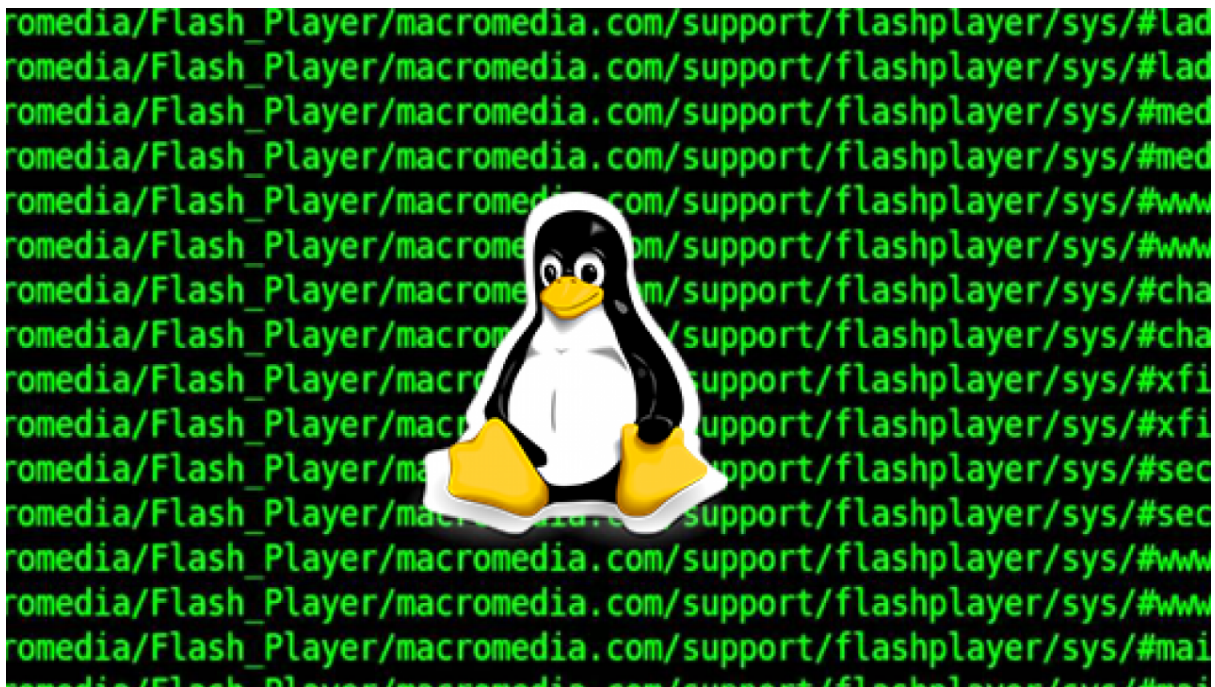
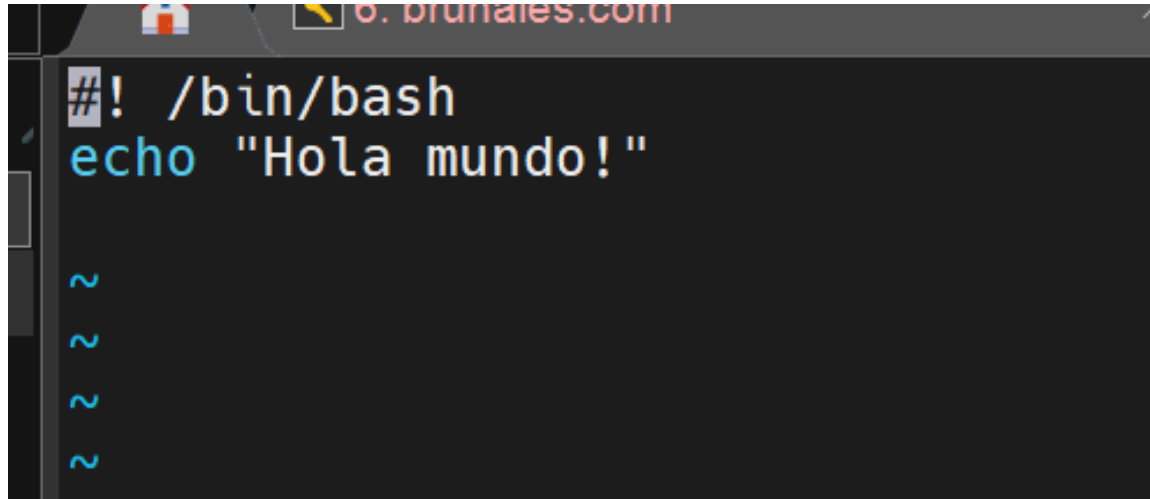


Prácticas shell script

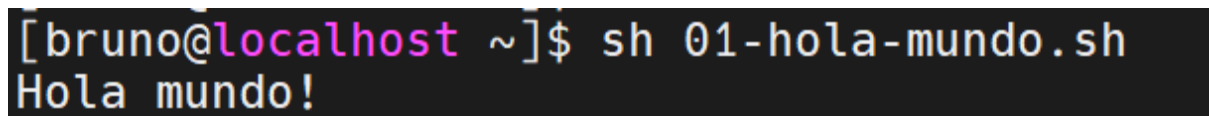


1 Realizar un script llamado '01holamundo.sh' que muestre por pantalla "Hola mundo!".

echo sirve para imprimir por pantalla. En este caso imprimimos **Hola mundo**.



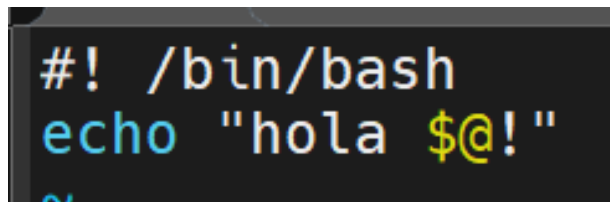
```
#!/bin/bash
echo "Hola mundo!"
~
~
~
~
```



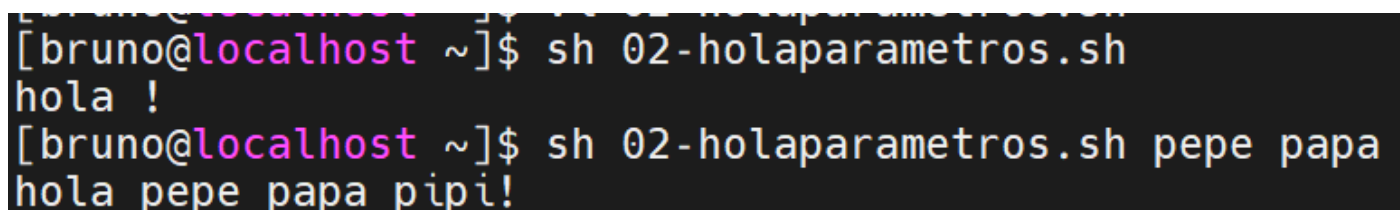
```
[bruno@localhost ~]$ sh 01-hola-mundo.sh
Hola mundo!
```

2 Ídem pero que en vez de "mundo" muestre los parámetros introducidos ('02holaparametros.sh').

"@" va a concatenar los parámetros que el usuario introduzca por teclado



```
#!/bin/bash
echo "hola $@"
~
```



```
[bruno@localhost ~]$ sh 02-holaparametros.sh
hola !
[bruno@localhost ~]$ sh 02-holaparametros.sh pepe papa
hola pepe papa pipi!
```

3 Ídem y que además verifique que al menos hayamos introducido un parámetro ('03holaalmenos1parametro.sh').

```
if [ $# -ne 0 ]; then
    echo "Hola $@"
    exit 0
fi
echo "Introduce al menos un parametro."
exit 1
```

“\$#” recoge la cantidad de parametros

“-ne” significa not equals

Aquí muestro un caso por si el usuario no introduce parámetros y otro si sí introduce nombre/s.

```
[bruno@localhost ProgramasShellScript]$ sh 03-hola-al-menos-1-parametro.sh Bruno Pepe
Hola Bruno Pepe!
[bruno@localhost ProgramasShellScript]$ sh 03-hola-al-menos-1-parametro.sh
Introduce al menos un parametro.
```

4 Ídem y que además separe cada argumento por ", " ('04holaparametrosseparados.sh').

En mi caso, he usado un bucle “for” para obligarme a aprender su sintaxis va desde 1 hasta n (la cantidad de parametros).

```
#!/bin/bash
primero=1
mensaje="Hola"
if [ $# -ne 0 ]; then
    for i in $(seq 1 $#); do
        if [ $primero -eq 1 ]; then
            mensaje="$mensaje $1"
            primero=0
        else
            mensaje="$mensaje, $1"
        fi
        shift
    done
else
    mensaje="Introduce al menos un parametro"
fi
echo $mensaje"!"
```

```
[bruno@localhost ProgramasShellScript]$ sh 04-hola-parametros-separados.sh Bruno P
Hola Bruno, Pepe, Papa, Papi!
```

5 Ídem y que además en caso de error muestra una ayuda ('05holaconayuda.sh').

```
#!/bin/bash
# función de ayuda
ayuda() {
cat << DESCRIPCION_AYUDA
SYNOPSIS
$0 NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]
DESCRIPCION
    Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.
CÓDIGOS DE RETORNO
1 Si el número de parámetros es menor que 1
DESCRIPCION_AYUDA
}
# si número de parámetros <= 0
if [ $# -le 0 ]; then
    echo "Hay que introducir al menos un parámetro."
    ayuda
    exit 1
fi
MENSAJE="Hola"
PRIMERO=1
# mientras haya parámetros
while [ -n "$1" ]; do
    if [ $PRIMERO -eq 1 ]; then
        MENSAJE="$MENSAJE $1"
        PRIMERO=0
    else
        MENSAJE="$MENSAJE, $1"
    fi
    # pasamos al siguiente parámetro
    shift
done
# mostramos la salida por pantalla
echo "$MENSAJE!"
exit 0
```

Se muestra la ayuda en caso de que no reciba argumentos.

```
bruno@localhost:~/ProgramasShellScript$ sh 05-hola-con-ayuda.sh pepe papa
Hola pepe, papa!
bruno@localhost:~/ProgramasShellScript$ sh 05-hola-con-ayuda.sh
Hay que introducir al menos un parámetro.
SYNOPSIS
05-hola-con-ayuda.sh NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]
DESCRIPCION
    Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.
CÓDIGOS DE RETORNO
1 Si el número de parámetros es menor que 1
```

6 Ídem y que además verifique que sean usuarios conectados al sistema ('06holausuario.sh').

Se usa el comando **who | grep \$1** ya que el comando **who** devuelve a los usuarios conectados y **grep** busca en el archivo la palabra que indiquemos (\$1). Si no encuentra ese argumento (el nombre del usuario), se cumplirá la condición del **if** e indicará que no está conectado

```
#!/bin/bash
# función de ayuda
function ayuda() {
cat << DESCRPCION_AYUDA
SYNOPSIS
$0 NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]
DESCRIPCION
Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.
CÓDIGOS DE RETORNO
1 Si el número de parámetros es menor que 1
2 Si el usuario no está conectado
DESCRPCION_AYUDA
}
# si número de parámetros <= 0
if [ $# -le 0 ] ; then
    echo "Hay que introducir al menos un parámetro."
    ayuda
    exit 1
fi
MENSAJE="Hola"
PRIMERO=1
# mientras haya parámetros
while [ -n "$1" ]; do
    ESTA_CONECTADO=`who | grep $1`
    if [ -z "$ESTA_CONECTADO" ]; then
        echo "El usuario $1 no está conectado"
        ayuda
        exit 2
    fi
    if [ $PRIMERO -eq 1 ]; then
        MENSAJE="$MENSAJE $1"
        PRIMERO=0
    else
        MENSAJE="$MENSAJE, $1"
    fi
    # pasamos al siguiente parámetro
    shift
done
# mostramos la salida por pantalla
echo "${MENSAJE}!"
```

```
[bruno@localhost ProgramasShellScript]$ sh 06-hola-usuario.sh Pepe Papa
El usuario Pepe no está conectado
SYNOPSIS
06-hola-usuario.sh NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]
DESCRIPCION
Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.
CÓDIGOS DE RETORNO
1 Si el número de parámetros es menor que 1
2 Si el usuario no está conectado
[bruno@localhost ProgramasShellScript]$ sh 06-hola-usuario.sh bruno
Hola bruno!
```

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

```
#!/bin/bash
function ayuda() {
cat << DESCRIPCION_AYUDA
SYNOPSIS
$0 NOMBRE_USUARIO
DESCRIPCIÓN
    Devuelve:
        SI si NOMBRE_USUARIO coincide con algún usuario conectado o
        NO si NOMBRE_USUARIO no coincide con ningún usuario conectado
CÓDIGOS DE RETORNO
1 Si el número de parámetros es distinto de 1
DESCRIPCION_AYUDA
}
# si número de parámetros distinto 1
if [ $# -ne 1 ]; then
    echo "El número de parámetros debe de igual a 1"
    ayuda
    exit 1
fi
ESTA_CONECTADO=`who | grep $1`
if [ -z "$ESTA_CONECTADO" ]; then
    echo "NO"
else
    echo "SI"
fi
```

En el if, se usa el comparador -z que significa zero. Justamente queremos saber eso sobre la variable porque si se cumple la condición, significará que no está conectado.

Aquí la prueba de su funcionamiento.

```
[bruno@localhost ProgramasShellScript]$ sh 07-usuarioconectado bruno
SI
[bruno@localhost ProgramasShellScript]$ sh 07-usuarioconectado bruna
NO
```

8 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"

vi para editar el archivo

```
t ~]$ vi .bashrc
```

Una vez en .bashrc, introducimos esta línea con la ruta de nuestros programitas.

```
# .bashrc
export PATH=$PATH":~/home/bruno/ProgramasShellScript"
# Source global definitions
```

Al tratar de usar nuestro nuevo comando, nos da permiso denegado. Por eso le asignamos a user el permiso de ejecución.

```
-rw-r--r-- 1 bruno bruno 651 Feb 27 15:38 06-nota-usuario.sh
-rw-r--r-- 1 bruno bruno 561 Feb 27 16:04 07-usuarioconectado
[bruno@localhost ProgramasShellScript]$ 07-usuarioconectado
-bash: /home/bruno/ProgramasShellScript/07-usuarioconectado: Permission denied
[bruno@localhost ProgramasShellScript]$ chmod u+x 07-usuarioconectado
```

Aquí ya nos deja ejecutar el programa desde cualquier ruta

```
[bruno@localhost ~]$ 07-usuarioconectado bruno
SI
[bruno@localhost ~]$ 07-usuarioconectado burna
NO
```

9 Modificar el script '06holausuario.sh' para que llame a 'usuarioconectado' ('07holausuario.sh')

Es igual que el **ejercicio 6**, la única modificación es que ya no nos hace falta tirar el comando. Ahora tenemos a disposición nuestro comando **07-usuarioconectado**, que devuelve si o no en función de si está conectado. Este valor se lo volcamos a **ESTA_CONECTADO** y lo usamos en la condición del if.

```
#!/bin/bash
# función de ayuda
function ayuda() {
cat << DESCRPCION_AYUDA
SYNOPSIS
$0 NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]
DESCRIPCION
Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.
CÓDIGOS DE RETORNO
1 Si el número de parámetros es menor que 1
2 Si el usuario no está conectado
DESCRPCION_AYUDA
}
# si número de parámetros <= 0
if [ $# -le 0 ] ; then
echo "Hay que introducir al menos un parámetro."
ayuda
exit 1
fi
MENSAJE="Hola"
PRIMERO=1
# mientras haya parámetros
while [ -n "$1" ]; do
ESTA_CONECTADO=`./07-usuarioconectado $1`

if [ "$ESTA_CONECTADO" == "NO" ]; then
echo "El usuario $1 no está conectado"
ayuda
exit 2
fi
if [ $PRIMERO -eq 1 ]; then
MENSAJE="$MENSAJE $1"
PRIMERO=0
else
MENSAJE="$MENSAJE, $1"
fi
# pasamos al siguiente parámetro
shift
done
# mostramos la salida por pantalla
echo "${MENSAJE}!"
~
```

Funciona perfectamente.

```
[bruno@localhost ProgramasShellScript]$ vi 06-hola-usuario.sh
[bruno@localhost ProgramasShellScript]$ 06-hola-usuario.sh pepe
El usuario pepe no está conectado
SYNOPSIS
/home/bruno/ProgramasShellScript/06-hola-usuario.sh NOMBRE_1 [NOMBRE_2] ... [NOMBRE_N]
DESCRIPCION
Muestra "Hola NOMBRE_1, NOMBRE_2, ... NOMBRE_N!" por pantalla.
CÓDIGOS DE RETORNO
1 Si el número de parámetros es menor que 1
2 Si el usuario no está conectado
[bruno@localhost ProgramasShellScript]$ 06-hola-usuario.sh bruno
Hola bruno!
```


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

He planteado este ejercicio de la siguiente forma. Vuelco a **user** el nombre de todos los usuarios del sistema a través del comando **cut -d: -f1 /etc/passwd** ya que este fichero es el que contiene todos los nombres. Saco solo los nombres gracias al argumento **-d** (delimitador) y al carácter ":", que es quien los delimita. Se utiliza **"-f1"** para extraer el primer campo de cada línea del archivo de entrada.

Luego meto en el array **users** los diferentes nombre de los usuarios

A continuación comparo el parametro con cada uno de los nombres. Devuelvo **"SI"** si se ha encontrado el nombre, por lo contrario muestro el mensaje **"NO"**.

```
#!/ bin/bash

user=$(cut -d: -f1 /etc/passwd)
users=( $user )
len=${#users[@]}
if [ -n "$1" ]; then
    for ((i=0; i<len; i++)) do
        if [ $1 == "${users[i]}" ]; then
            echo "SI"
            exit 0
        fi
    done

    echo "NO"
    exit 0
else
    echo "Se requiere un parametro"
    exit 1
fi
```

Un ejemplo para cada caso.

```
[bruno@localhost ProgramasShellScript]$ usuariosistema bruno
SI
[bruno@localhost ProgramasShellScript]$ usuariosistema pepe
NO
[bruno@localhost ProgramasShellScript]$ usuariosistema
Se requiere un parametro
```