

UT 04. Lenguajes de Scripts: actividades Bash (I)

1.- Escribir un script, **existe_usuario.sh**, que reciba como argumento el nombre de un usuario. El script mostrará por pantalla si el usuario existe o no en el sistema:

```
1  #!/bin/bash
2
3  if [ $# -eq 0 ]; then # $# contiene el número de parámetros pasados al script
4      echo "Debe introducir al menos 1 usuario como argumento."
5      echo "Ejemplo: $0 pepe"
6      exit 1
7  fi
8
9  getent passwd $1 >/dev/null
10
11 if [ $? -eq 0 ]; then
12     echo "El usuario $1 existe en el sistema."
13 else
14     echo "No existe el usuario $1 en el sistema."
15 fi
16 exit 0
```

```
1  #!/bin/bash
2
3  if [ $# -eq 0 ]; then # $# contiene el número de parámetros pasados al script
4      echo "Debe introducir al menos 1 usuario como argumento."
5      echo "Ejemplo: $0 pepe"
6      exit 1
7  fi
8
9  grep $1 /etc/passwd >/dev/null # buscamos al usuario con grep en /etc/passwd
10
11 if [ $? -eq 0 ]; then # si tiene éxito
12     echo "El usuario $1 existe en el sistema."
13 else # si no
14     echo "No existe el usuario $1 en el sistema."
15 fi
16 exit 0
```

```
edu@US-ST:~/scripts$ existe_usuario_v2.sh pepe
No existe el usuario pepe en el sistema.
edu@US-ST:~/scripts$ existe_usuario_v2.sh edu
El usuario edu existe en el sistema.
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

2.- Completa este script completando la opción 2.

```
#!/bin/bash
# Zona de funciones
function pausa {
    echo "Pulse una tecla para continuar...";
    read tecla;
}
while true
do
    clear;
    echo "1.- Carga del sistema";
    echo "2.- Sistema de inicialización";
    echo "3.- Nivel de ejecución";
    echo "4.- Fin";
    read -p "Su elección: " eleccion;

    case $eleccion in
        1) uptime;
           pausa;
           ;;
        2) echo "En construcción"
           pausa;
           ;;
        3) runlevel;
           pausa;
           ;;
        4) exit 0;
           ;;
        *) echo "Elección incorrecta";
           pausa;
           ;;
    esac
done
```

2) pstree | head -1 | cut -f1 -d" -";

3.- Crea un script que muestre la "tabla de multiplicar" (del 0 al 10) de un número solicitado de forma interactiva al usuario.

Propuestas

```
1  #!/bin/bash
2  #Tabla de multiplicar
3
4  read -p "Introduzca un número del 1 al 999: " NUMERO;
5
6  if [ $NUMERO -le 0 ] || [ $NUMERO -gt 999 ]; then
7      echo "Debe introducir un número entre 1 y 999."
8      exit 1
9  fi
10
11  CONTADOR=0
12  RESULTADO=0
13
14  while (($CONTADOR <= 10)); do
15      ((RESULTADO=$NUMERO*$CONTADOR));
16      echo "$NUMERO x $CONTADOR = $RESULTADO";
17      ((CONTADOR++))
18  done;
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

3.1.- Modifica el script de tal forma que se controle que el número introducido sea un número entero.

Observación: busca en Internet una solución. Lo más adecuado es que el control lo hagas en una función. Lo hago con una expresión regular.

```
tabla_multiplicar.sh x
1  #!/bin/bash
2  #Tabla de multiplicar
3
4  expresion='^[0-9]+$'
```

```
if [[ $NUMERO =~ $expresion ]]
```

```
40  else
41      echo "Debe introducir un número entero."
42      pausa;
43  fi
```

```
Introduzca un número del 1 al 999: 5.3
Debe introducir un número entero.
```

3.2.- Mejorar el script permitiendo al usuario ejecutar varias veces el programa (hasta que el usuario quiera).

```
existe_usuario_v2.sh x  sysinfo.sh x  tabla_multiplicar.sh x
1  #!/bin/bash
2  #Tabla de multiplicar
3
4  function pausa {
5      echo
6      echo "Pulse una tecla para continuar..."
7      echo "s para salir del programa."
8      read tecla
9      if [ $tecla = s ]; then
10         exit 0;
11     fi
12 }
13
14 while true
15 do
16     clear;
17     read -p "Introduzca un número del 1 al 999: " NUMERO;
18
19     if [ $NUMERO -le 0 ] || [ $NUMERO -gt 999 ]; then
20         echo "Debe introducir un número entre 1 y 999."
21         exit 1
22     fi
23
24     CONTADOR=0
25     RESULTADO=0
26
27     while (( $CONTADOR <= 10 )); do
28         ((RESULTADO=$NUMERO*$CONTADOR));
29         echo "$NUMERO x $CONTADOR = $RESULTADO";
30         ((CONTADOR++))
31     done;
32
33     pausa;
34
35 done;
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

4.- Crea un script que se llame **convertir.sh** que reciba uno o más argumentos (deben ser ficheros regulares) y les asigne el permiso 755. Controlar posibles errores. Nota: antes de resolver el problema vamos a ver la sentencia for...in.

```
1  #!/bin/bash
2  # Asignar permisos 755 a los ficheros pasados por parámetro
3
4  if [ $# -eq 0 ]; then # $# contiene el número de parámetros pasados al script
5      echo "Debe introducir al menos 1 argumento."
6      echo "Ejemplo: $0 fichero1, fichero2, fichero3 ..."
7      exit 1
8  fi
9
10 for FICHERO in $*; do # $* contiene un array con los parámetros pasados al script
11     if [ -f $FICHERO ]; then
12         # Asignar permisos 755
13         chmod 755 $FICHERO
14     else
15         echo "$FICHERO no es un fichero regular."
16     fi
17 done
18 exit 0
```

Observación: es un ejercicio para utilizar el **for...in**. Te adjunto algunos ejemplos de utilización de esta estructura de repetición.

for1.sh

```
GNU nano 2.8.6 File: for1.sh
#!/bin/bash
for i in `ls`
do
    echo $i
done
```

Ejecución:

```
edu@US-ST:~/scripts$ for1.sh
adivina.sh
alerta.sh
args.sh
buclewhile.sh
cambio.sh
es10.sh
esdir.sh
```

for2.sh

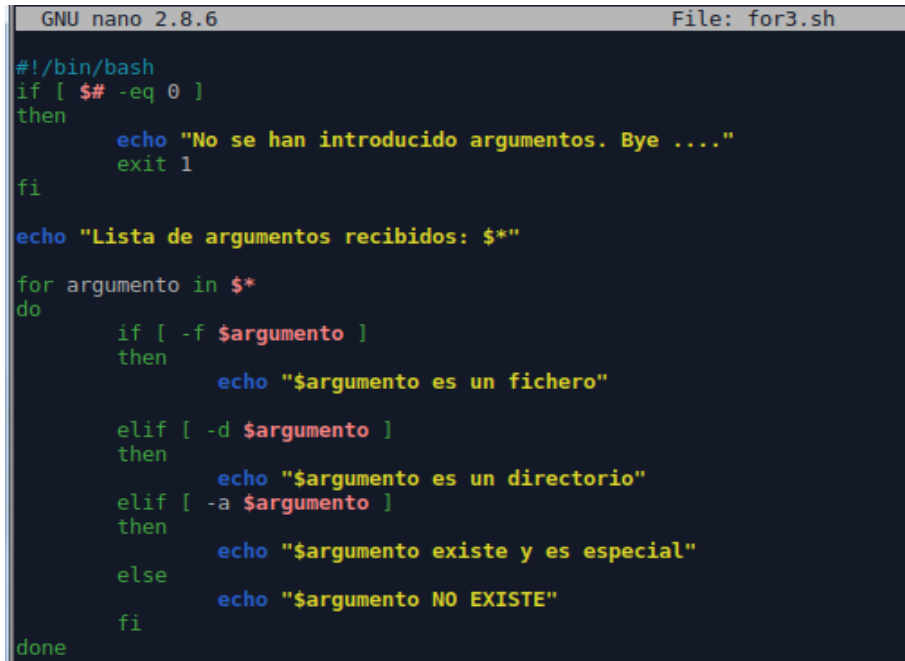
```
GNU nano 2.8.6 File: for2.sh
#!/bin/bash
for despegue in 4 3 2 1 "Fuego !!!"
do
    echo $despegue
done
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

Ejecución:

```
edu@US-ST:~/scripts$ for2.sh
4
3
2
1
Fuego !!!
```

for3.sh



```
GNU nano 2.8.6 File: for3.sh
#!/bin/bash
if [ $# -eq 0 ]
then
    echo "No se han introducido argumentos. Bye ...."
    exit 1
fi

echo "Lista de argumentos recibidos: $"

for argumento in $*
do
    if [ -f $argumento ]
    then
        echo "$argumento es un fichero"

    elif [ -d $argumento ]
    then
        echo "$argumento es un directorio"

    elif [ -a $argumento ]
    then
        echo "$argumento existe y es especial"

    else
        echo "$argumento NO EXISTE"
    fi
done
```

Observación: introducir algún fichero ubicado en /dev, por ejemplo /dev/zero

Ejecución:

```
edu@US-ST:~/scripts$ for3.sh /dev/zero
Lista de argumentos recibidos: /dev/zero
/dev/zero existe y es especial.
edu@US-ST:~/scripts$
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

for4.sh

¿Qué hace este script?

```
GNU nano 2.8.6 File: for4.sh

#!/bin/bash

for usuario in `cut -d":" -f1 /etc/passwd | grep "^st"`
do
    echo "Usuario....: $usuario"
done
```

Busca usuarios en el sistema que empiecen por las letras st.

```
edu@US-ST:~/scripts$ for4.sh
Usuario...: st_tv_01
Usuario...: st_tv_02
Usuario...: st_tfn_01
Usuario...: st_tfn_02
Usuario...: statd _
```

5.- Estudia el siguiente script, **compararFiles.sh**

```
#!/bin/bash
if diff $1 $2 > /dev/null
then
    echo "$1 y $2 son idénticos"
else
    echo "$1 y $2 son distintos"
fi
exit 0
```

Una vez que sepas lo que hace, pruébalo. A continuación, controla posibles errores que la versión actual no contempla.

```
compararFiles.sh x
1  #!/bin/bash
2
3  if [ $# -ne 2 ]
4  then
5      echo "Debe pasar 2 ficheros como argumentos."
6      exit 1
7  fi
8
9  if [ -f $1 ] && [ -f $2 ]
10 then
11     if diff $1 $2 >/dev/null
12     then
13         echo "$1 y $2 son idénticos."
14     else
15         echo "$1 y $2 son distintos."
16     fi
17 else
18     echo "Alguno de los parámetros no es un fichero."
19     exit 1
20 fi
```

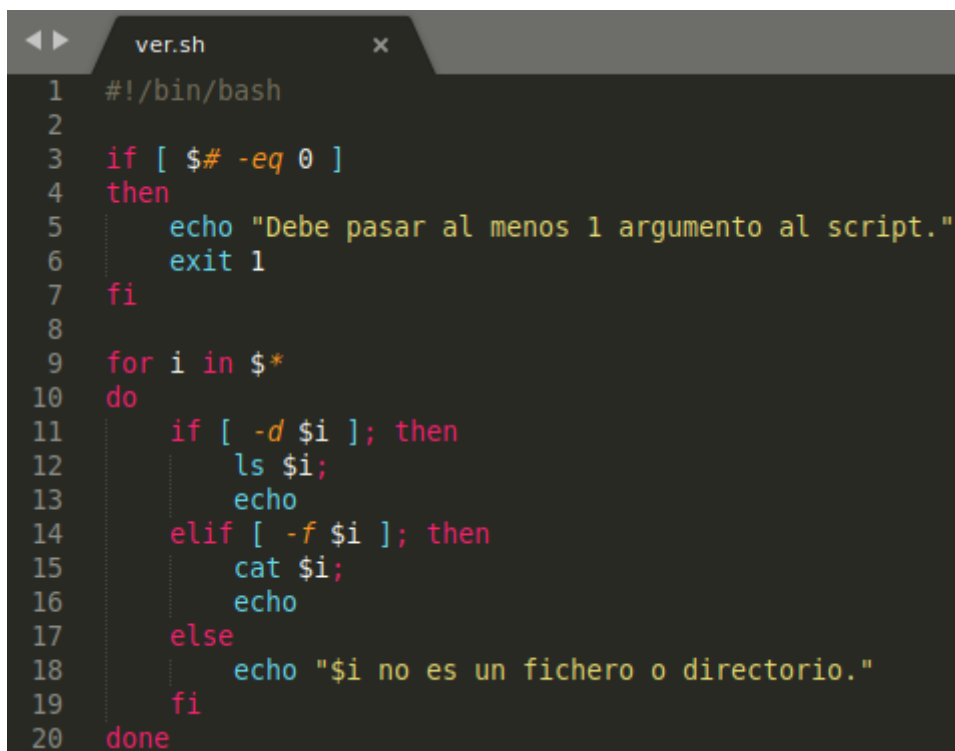
UT 04. Lenguajes de Scripts: actividades Bash (I)

Comprueba si 2 ficheros son iguales o distintos.

```
edu@US-ST:~/scripts$ compararFiles.sh perm.sh for1.sh
perm.sh y for1.sh son distintos.
edu@US-ST:~/scripts$ compararFiles.sh perm.sh perm.sh
perm.sh y perm.sh son idénticos.
edu@US-ST:~/scripts$ compararFiles.sh perm.sh 5
Alguno de los parámetros no es un fichero.
```

6.- Escribir un script, **ver.sh**, que, para cada argumento que reciba (puede recibir varios), realice una de las siguientes operaciones:

- si es un directorio ha de listar los ficheros que contiene.
- si es un fichero regular tiene que mostrar su contenido por pantalla.
- en otro caso, que indique que no es ni un fichero ni un directorio.



```
ver.sh
1  #!/bin/bash
2
3  if [ $# -eq 0 ]
4  then
5      echo "Debe pasar al menos 1 argumento al script."
6      exit 1
7  fi
8
9  for i in $*
10 do
11     if [ -d $i ]; then
12         ls $i;
13         echo
14     elif [ -f $i ]; then
15         cat $i;
16         echo
17     else
18         echo "$i no es un fichero o directorio."
19     fi
20 done
```

```
edu@US-ST:~/scripts$ ver.sh /home /home/edu
admin-01    alum-01    jefe_st    profe-02    st_tfn_01
admin-02    alum-02    lolamento psico-01    st_tfn_02
adm-nfs     crd_st     lost+found psico-02    st_tv_01
adm-samba   edu        matiasbuenas ricardoborriquero st_tv_02
aitortilla  enriquecido profe-01    samba       ubuntu

bin         Documentos Imágenes   PDF         Público    Vídeos
Descargas   Escritorio  Música     Plantillas  scripts
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

7.- La **DGT** pone en marcha nuevas medidas para el control de la velocidad. Uno de los cambios es que las sanciones dependerán del tipo de vehículo. Cada vez que *salta* el radar se produce una entrada en un fichero de un servidor Linux, al que están conectados los radares, que refleja el *tipo de vehículo, la cantidad en euros de la sanción y la matrícula*.

Vamos a considerar 5 tipos de vehículos:

A → Turismos

B → Camiones

C → Autobuses

D → Vehículos especiales

E → Motocicletas

El fichero en el que se van almacenando las multas se llama **sanciones.txt**.

Vemos la estructura (**tipo:cantidad:matrícula**) del mismo con un posible contenido:

```
A : 200 : AB123
A : 150 : HB123
B : 90 : XC42
D : 150 : ABCD
D : 200 : X123
C : 300 : CC12
```

Realizar en script, de nombre **multas.sh**, que cuando se ejecute muestre un **menú** en el que se solicite introducir un tipo de vehículo (A, B, C, D ó E). A continuación, debe mostrar en pantalla el total recaudado del mismo.

Por ejemplo, con los datos del fichero de muestra, si el usuario pulsa una A, la salida debería ser:

```
Total recaudado vehículos tipo A: 350 €
```

Si el usuario pulsa una opción incorrecta, la salida debería ser:

```
Elija opción...m
OPCIÓN INCORRECTA...
```

```
case $seleccion in
  A|B|C|D|E) for i in `grep ^$seleccion sanciones.txt | cut -f2 -d":"`
  do
    ((TOTAL=$TOTAL+$i))
  done
  echo "Total recaudado vehículos tipo $seleccion: $TOTAL€";
  pausa;
;;
```

```
/home/edu/scripts/multas.sh
```

```
Introduzca el tipo de vehículo: A, B, C, D o E.
Pulse s para salir del programa.
```

```
a
Total recaudado vehículos tipo A: 350€
```

```
Pulse una tecla para continuar...
```


UT 04. Lenguajes de Scripts: actividades Bash (I)

8.- Estudia el siguiente código de un script llamado f: hace un factorial del número introducido.

```
#!/bin/bash
clear
entrada=$1

if [ $entrada -ge 0 ]
then
    resultado=1
    while [ $entrada -gt 1 ]
    do
        resultado=$((resultado*entrada))
        entrada=$((entrada-1))
    done
    echo "El resultado es $resultado"
else
    echo "Debe introducir un número positivo"
fi
```

Este script debe recibir como argumento un número. Realiza una traza del programa si se invoca de las siguientes formas:

a) f 4

El resultado es 24.

b) f 0

El resultado es 1.

c) f -1

Debe introducir un número positivo.

9.- Tenemos un fichero llamado **empleados** cuyo contenido actual es el login y la compañía de teléfono que utiliza. El formato de este fichero es:

login:compañía

Realizar un script llamado **tfn.sh** que reciba un argumento que será una de las compañías de teléfono válidas: vodafone, yoigo, movistar y orange. La salida de este shell será:

a) Un mensaje de error si no se introduce uno y sólo un argumento:

"Debe introducir un sólo argumento, la compañía de tfn".

b) Un mensaje de error si no se introduce una marca válida:

"Debe introducir una marca válida (vodafone, yoigo, movistar y orange)".

c) En el caso de que todo sea correcto, un mensaje parecido a:

"El número de empleados que utilizan la compañía xxxxxxx es yy", donde

xxxxxxx es una de las marcas válidas e yy es un número entero.

Ejemplo del fichero empleados:

```
GNU nano 2.8.6 File: empleados
antonio:orange
juan:orange
pepe:vodafone
ernesto:movistar
luismi:yoigo
luisa:vodafone
carmen:orange
irene:orange
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

```
tfn.sh x empleados.txt x
1 #!/bin/bash
2
3 if [ $# -ne 1 ]
4 then
5     echo "Debe introducir un sólo argumento, la compañía de tlf."
6     exit 1
7 fi
8
9 if [ $1 = orange ] || [ $1 = vodafone ] || [ $1 = movistar ] || [ $1 = yoigo ]
10 then
11     resultado=`grep $1 empleados.txt | wc -l`
12     echo "$resultado empleados utilizan la compañía $1."
13 else
14     echo "Debe introducir una compañía válida. (movistar, orange, vodafone o yoigo)"
15 fi
```

```
edu@US-ST:~/scripts$ tfn.sh vodafone
2 empleados utilizan la compañía vodafone.
edu@US-ST:~/scripts$ tfn.sh orange
4 empleados utilizan la compañía orange.
```

Una vez realizado, ampliar este shell. Ahora, la salida debe ser así:
Total empleados: xx
Hay yy empleados que utilizan nnnnnnn, esto es, un nn% del total.

```
tfn.sh x empleados.txt x
1 #!/bin/bash
2
3 if [ $# -ne 1 ]
4 then
5     echo "Debe introducir un sólo argumento, la compañía de tlf."
6     exit 1
7 fi
8
9 if [ $1 = orange ] || [ $1 = vodafone ] || [ $1 = movistar ] || [ $1 = yoigo ]
10 then
11     total_empleados=`wc -l empleados.txt | cut -f1 -d" "`
12     resultado=`grep $1 empleados.txt | wc -l`
13     ((porcentaje=resultado*100/total_empleados))
14
15     echo "Total de empleados: $total_empleados"
16     echo "Hay $resultado empleados que utilizan $1, esto es, un $porcentaje% del total."
17
18 else
19     echo "Debe introducir una compañía válida. (movistar, orange, vodafone o yoigo)"
20 fi
```

```
edu@US-ST:~/scripts$ tfn.sh vodafone
Total de empleados: 7
Hay 2 empleados que utilizan vodafone, esto es, un 28% del total.
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

10.- Crea un script, **alerta.sh**, que cuando se ejecute busque todos los ficheros que contengan la palabra **hacker** dentro de **/home** y los guarde (**copie**) en el directorio **/cuarentena**.

```
1  #!/bin/bash
2
3  # Cuando se ejecute que busque todos los ficheros que
4  # contengan la palabra hacker dentro de /home y los guarde
5  # (copie) en el directorio /cuarentena
6
7  CONTADOR=0
8
9  for i in `find /home -type f -name "*hacker*" 2>/dev/null`; do
10     ((CONTADOR++))
11     cp $i /cuarentena
12 done
13
14 echo "Se han copiado $CONTADOR ficheros"
```

11.- Crea un script, **f_vulnerables.sh**, que busque, en **/bin** y **/home**, todos los ficheros que tengan permisos "7" en otros y los guarde, con toda su ruta, en el fichero **/root/archivos_peligrosos.txt**.

```
1  #!/bin/bash
2
3  rm /root/archivos_peligrosos.txt
4
5  for i in $(find /bin /home -type f -perm -o+w,o+w,o+x 2>/dev/null)
6  do
7      echo $i >> /root/archivos_peligrosos.txt
8  done
```

```
edu@US-ST:~/scripts$ sudo chown root:root f_vulnerables.sh
edu@US-ST:~/scripts$ sudo chmod 777 f_vulnerables.sh
```

Ejecutarlo como root.

```
edu@US-ST:~/scripts$ sudo su
root@US-ST:/home/edu/scripts# ./f_vulnerables.sh
```

```
root@US-ST:/home/edu/scripts# ./f_vulnerables.sh
root@US-ST:/home/edu/scripts# cat /root/archivos_peligrosos.txt
/home/edu/Descargas/uld/uninstall-scanner.sh
/home/edu/scripts/sanciones.txt
/home/edu/scripts/buclewhile.sh
/home/edu/scripts/f_vulnerables.sh
/home/edu/Descargas/uld/install-printer.sh
/home/edu/Descargas/uld/install-scanner.sh
/home/edu/Descargas/uld/noarch/printer-meta.pkg
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

12.- Crea un script, **d_to_h.sh**, que convierta un número decimal, introducido en **la línea de comandos**, a hexadecimal. Sólo se admiten números mayores o iguales a 0 y menores a 16.

```
f_vulnerables.sh x d_to_h.sh x
1  #!/bin/bash
2
3  if [ $# -eq 0 ] || [ $# -gt 1 ]
4  then
5      echo "Debe introducir como argumento un número decimal."
6      echo "Ejemplo $0 5."
7      exit 1
8  fi
9
10 if [ $1 -lt 0 ] || [ $1 -gt 16 ]
11 then
12     echo "El argumento debe ser un número entre 0 y 16."
13     exit 1
14 fi
15
16 printf '%x\n' $1
```

```
edu@US-ST:~/scripts$ d_to_h.sh 11
b
```

Arrays en bash

Ver concepto y funcionamiento en:

<https://gulvi.com/serie/curso-programacion-bash/capitulo/arrays-bash>

```
edu@US-ST:~/scripts$ numeros=(1 2 3 4 5)
edu@US-ST:~/scripts$ echo ${numeros[@]}
1 2 3 4 5
```

```
edu@US-ST:~/scripts$ cifrasLetras=( {A..Z} {a..z} {0..9} )
edu@US-ST:~/scripts$ echo ${cifrasLetras[*]}
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z a b c d e f g h i j k l m n
o p q r s t u v w x y z 0 1 2 3 4 5 6 7 8 9
```

```
edu@US-ST:~/scripts$ ficheros=(`ls`)
edu@US-ST:~/scripts$ echo ${ficheros[*]}
adivina.sh alerta.sh args.sh buclewhile.sh cambio.sh compararFiles.sh d_to_h.sh
empleados.txt es10.sh esdir.sh esfich.sh esfich_v2.sh esfich_v3.sh espacio.sh es
trellas.sh existe_usuario.sh existe_usuario_v2.sh for_1.sh for1.sh for2.sh for3.
sh for4.sh for5.sh f.sh funciones.sh func_suma2.sh func_suma.sh f_vulnerables.sh
holamundo.sh let.sh multas.sh nota_case.sh nota.sh num10.sh ocupacion.sh perm.s
h prueba.sh salida.sh saludo.sh sanciones.txt suma.sh sysinfo.sh tabla_multiplic
ar.sh tfn.sh ver.sh
edu@US-ST:~/scripts$ echo ${#ficheros[*]}
45
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

13.- Queremos poner a disposición de los empleados de una empresa un traductor de palabras español-inglés. Realizar un script, llamado **traductor.sh**, que cuando se ejecute pregunte al usuario si quiere español al inglés, inglés al español o salir. De español a inglés se debe mostrar, además de la traducción, la pronunciación. Si la palabra a traducir no existe, se mostrará el correspondiente mensaje.

Observación: tanto las palabras en español, la traducción y la pronunciación las tenemos almacenadas en arrays.

Código del programa:

```
#!/bin/bash

# Definición de los arrays
espaniol=(hola adiós coche azul)
ingles=(hello goodbye car blue)
pronunciacion=(jelou gudbai car blu)

# Función pausa para hacer el programa más interactivo
function pausa {
    echo
    echo "Pulse una tecla para continuar..."
    read tecla
}

# Menú
while true
do
    clear;

    echo "TRADUCTOR"
    echo
    echo "Elija una opción:"
    echo
    echo "1.- Español - Inglés."
    echo "2.- Inglés - Español."
    echo "3.- Salir."
    echo
    read eleccion;

    case $eleccion in
        #Español - Inglés
        1) read -p "Introduzca la palabra a traducir: " palabra

            contador=-1
            encontrada=1

            for i in `echo ${espaniol[*]}`
```

UT 04. Lenguajes de Scripts: actividades Bash (I)

```
do
  ((contador++))
  if [ $i == $palabra ]
  then
    encontrada=0
    echo
    echo "Español: $i"
    echo "Inglés: ${ingles[$contador]}"
    echo "Pronunciación:
${pronunciacion[$contador]}"
    echo
    break
  fi
done

if [ $encontrada -eq 1 ]
then
  echo "No se encontró la palabra."
fi

pausa;
;;

# Inglés - Español
2) read -p "Introduzca la palabra a traducir: " palabra

contador=-1
encontrada=1

for i in `echo ${ingles[*]}`
do
  ((contador++))
  if [ $i == $palabra ]
  then
    encontrada=0
    echo
    echo "Inglés: $i"
    echo "Español: ${espaniol[$contador]}"
    echo
    break
  fi
done

if [ $encontrada -eq 1 ]
then
  echo "No se encontró la palabra."
fi
```

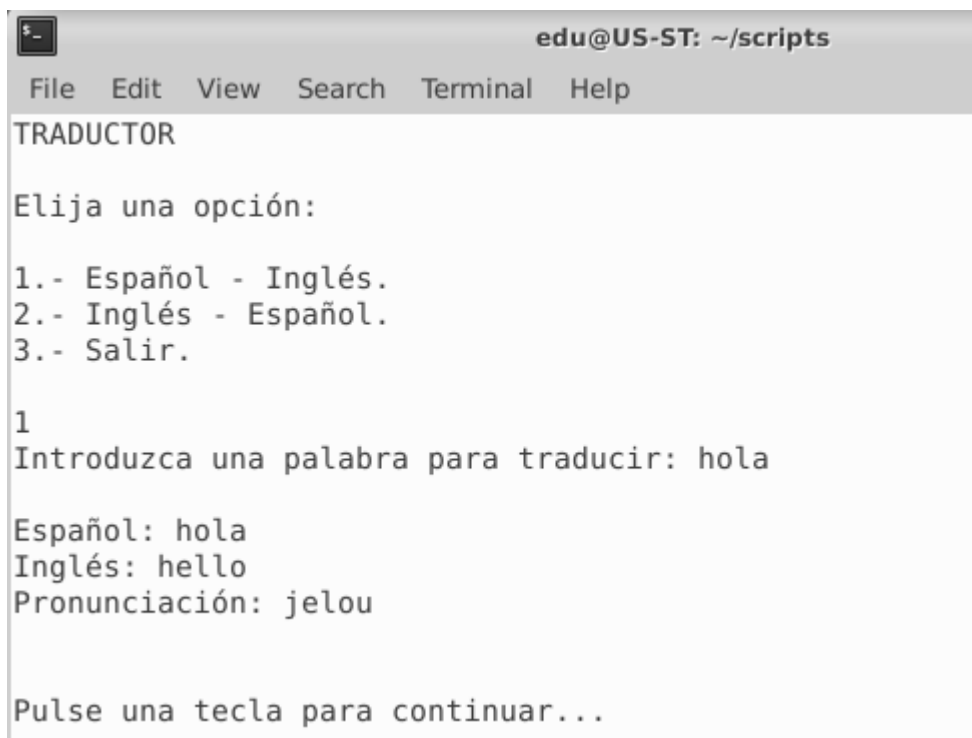
UT 04. Lenguajes de Scripts: actividades Bash (I)

```
        pausa;
        ;;
3) echo "Adiós."
   exit 0
   ;;

*) echo "Opción incorrecta."
   pausa;
   ;;

esac
done
```

Ejecución del programa:



```
edu@US-ST: ~/scripts
File Edit View Search Terminal Help
TRADUCTOR
Elija una opción:
1.- Español - Inglés.
2.- Inglés - Español.
3.- Salir.
1
Introduzca una palabra para traducir: hola
Español: hola
Inglés: hello
Pronunciación: jelou
Pulse una tecla para continuar...
```

Test sobre shell

- 1) ¿Cuál es la mejor descripción del shell?
a. Establece comunicación entre el usuario y el kernel
b. Es el programa command.com
c. Es un sistema anticuado, prácticamente en desuso.
d. Sólo para usuarios avanzados
- 2) ¿Cuál es el shell por defecto de Linux?
a. bash b. sh c. C-shell d. no hay ninguno por defecto
- 3) ¿Cuáles son ventajas del shell sobre el entorno gráfico?
a. ahorra tiempo
b. ahorra recursos

UT 04. Lenguajes de Scripts: actividades Bash (I)

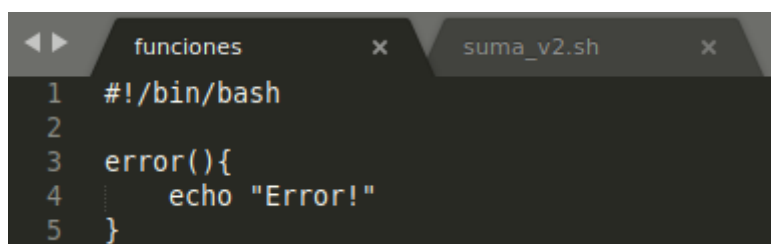
- c. es más sencillo
- d. no hay ninguna ventaja
- 4) ¿Qué es lo que entiende un shell?
 - a. comandos b. protocolos c. lenguaje C d. php
- 5) ¿Que se almacena en las variables del shell?
 - a. valores que describen el entorno o ambiente
 - b. valores de números hexadecimales
 - c. las IPs de los hosts
 - d. las tablas de inodo
- 6) ¿Cómo se visualiza una variable de nombre VAR?
 - a. echo \$VAR b. cat VAR c. bash VAR d. no se puede visualizar
- 7) ¿Cuál es la manera de obtener información sobre los comandos externos?
 - a. help comando b. ls comando c. bash comando d. man comando
- 5) ¿Qué comando permite cambiar de shell?
 - a. source
 - b. chsh
 - c. passwd
 - d. chown

Cuestiones

1.- ¿Qué permite el comando **source**? Sirve para ejecutar un script sobre el Shell sobre el que estamos, es decir, este no se ejecutará en un Shell hijo, por lo que después de su ejecución, los cambios que se hicieron a las variables de entorno se mantendrán.

2.- ¿Cómo se define una **función** en Bash? `function nombre_de_la_función()`
¿Cómo recibe y devuelve valores? Recibe valores llamando a la función pasándole los parámetros, por ejemplo, `nombre_de_la_función $1 "hola"`. Devuelve valores con la orden `return` o consultando la variable `$?`.

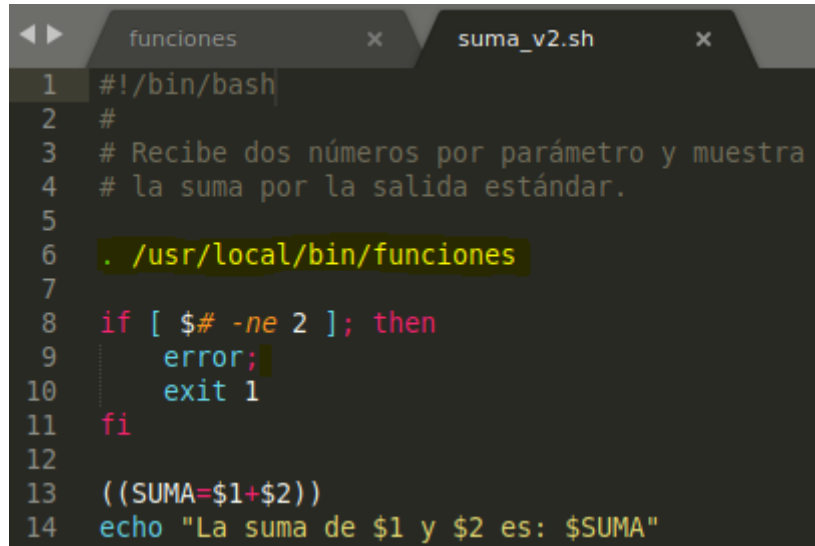
3.- Crea un fichero que sólo contenga funciones en `/usr/local/bin/funciones`, por ejemplo, con mensajes de error. Ahora crea un script que utilice alguna de esas funciones.



```
funciones suma_v2.sh
1  #!/bin/bash
2
3  error(){
4      echo "Error!"
5  }
```


UT 04. Lenguajes de Scripts: actividades Bash (I)

```
edu@US-ST:~$ ls -l /usr/local/bin
total 12
-rwxrwxrwx 1 root root 40 feb  5 07:47 funciones
-rwxr-xr-x 1 root root 403 nov 20 08:38 resamba.sh
-rwxr-xr-x 1 root root 75 nov 19 08:56 si.sh
```



```
1 #!/bin/bash
2 #
3 # Recibe dos números por parámetro y muestra
4 # la suma por la salida estándar.
5
6 . /usr/local/bin/funciones
7
8 if [ $# -ne 2 ]; then
9     error;
10    exit 1
11 fi
12
13 ((SUMA=$1+$2))
14 echo "La suma de $1 y $2 es: $SUMA"
```

```
edu@US-ST:~/scripts$ suma_v2.sh 2
Error!
```

Observación: consultar apuntes (IV).

4.- ¿Qué sucede cuando en un terminal bash tecleamos dash? Si existe, cambiamos al Shell dash.