UD 01: Actividades Scripts (I)

Cuestiones

1.- Explica la función de los siguientes comandos: **source** y **bash –x**. Ejemplo de aplicación de cada uno de ellos.

Source es un comando que permite ejecutar scripts sin que genere procesos hijos, con lo cual la información de las variables se pueden consultar variables en la consola, si lo ejecutamos con "./" no podremos consultar el valor de la variable ya que hace un proceso hijo. **Ejemplo**

Prueba.sh

```
#!/bin/bash
TFN=24
```

```
$./Prueba.sh
$echo $TFN
$
```

No nos devuelve contenido ya que la variable $TFN se ha declarado en un proceso hijo

```
$source Prueba.sh
$echo $TFN
$24
```

Nos muestra el contenido de la variable $TNF y que no ha realizado ningún proceso hijo

Bash -x sirve para depurar scripts y ver paso a paso lo que realiza y poder detectar posibles errores y comprender mejor su funcionamiento

```
$bash -x Prueba.sh
    $TFN=24   ← vemos como declara que $TFN vale 24
$   ← termina el script
```

Sin bash -x

```
$./Prueba.sh
$
```

2.- Contesta a las siguientes cuestiones

- ¿Qué significa #!/bin/dash y dónde lo podemos encontrar habitualmente?

Significa que el script lo debe ejecutar el intérprete de comandos el Dash. Habitualmente se encuentran al principio de un scripts de shell.

- ¿Qué sucederá tras ejecutar, dentro de un script, el comando echo -e "\tTexto \n\n"?

Con la opción -e activamos la interpretación de cirtis caracteres especiales \t para tabulador y \n para salto de línea con lo cual pondrá un tabulador antes de texto y después de texto dos saltos de linea

- ¿Qué sucederá tras ejecutar, dentro de un script, el comando echo -n "Nombre:"

Se utiliza para mostrar texto en la pantalla sin agregar una nueva línea al final sin un salto de línea entre la línea y el prompt.

- ¿Para qué se utilizan las órdenes export, set y unset?

- Esta orden se utiliza para exportar variables de entorno. (como hacer global una varaiable) Las variables de entorno son variables que están disponibles para todos los procesos hijo de la shell actual.
- Podemos usar set para ver variables del sistema y funciones.
- Podemos usar unset para eliminar variables declaradas

Scripts

1.- El comando ping –c 1 dir_IP tiene éxito si la dirección IP dir_IP es una dirección de red accesible. Crear un script, **red_ok.sh**, que compruebe si la dirección **IP que recibe por la línea de comandos es o no accesible**, mostrando en pantalla el mensaje correspondiente.

Hay que tener en cuenta que:

a) Este script sólo lo puede ejecutar root.

b) No se debe hacer nada si no recibe un solo argumento.

```
#!/bin/bash
#Comprobar numero de argumentos
if [ $# -eq 0 ] ; then
      echo "Debe intoducir una dirección IP"
      exit 5
fi
#Solo root o administardor puede ejectuarlo
if [ $UID -ne 0 ] ; then
      echo "Solo root"
      exit 6
fi
ACC=0
ERR=0
for dir_IP in $* ; do

      echo "Procesando $dir_IP"

      ping -c 2 $dir_IP &>/dev/null

      exito=$?
      echo "La ejecucion ha resultado en $exito"


      if [ "$exito" = "0" ] ; then
            echo "hay conectividad a $dir_IP"
            ((ACC++))
      else
            echo "No hay conectividad con $dir_IP"
            ((ERR++))
      fi

done

echo "Total direcciones recibidas: $#"
echo "Direcciones accesibles: $ACC"
echo "Direcciones inaccesibles o erróneas: $ERR"
```

```
#Segunda manera de resolverlo
#
#
#
#if ping -c 2 $dir_IP &>/dev/null ; then
#    echo "Hay conectividad a $dir_IP"
#else
#    echo "No hay conectividad con $dir_IP"
#fi
```

2.- Script **operar.sh** → recibe 2 argumentos → suma de esos argumentos → comprobar que los argumentos son números.

$operar.sh 2 3 → La suma de 2 + 3 es 5
$operar.sh 2 pepe → error en uno de los argumentos o en los dos
$operar.sh → Debe introducir 2 argumentos
Observación: asumimos que los números son enteros

```
#!/bin/bash
if [ $# -ne 2 ] ; then
     echo "Se requierer 2 argumentod"
     exit 4
fi
OP1=$1
OP2=$2

if [[ $OP2 == ?(-)+([0-9]) ]] && [[ $OP1 == ?(-)+([0-9]) ]]
; then
     echo "La suma de $OP1 + $OP2 es $((OP1 + OP2))"
else
     echo "Debes introducir numeros enteros"
fi
```

3.- Script, **cortesía.sh**, que muestre en pantalla el nombre completo, si es que existe, o el login, si no existen esos datos, del usuario que lo ejecuta, mostrando un saludo: Bueno días (antes de las 14h) y Buenas tardes (después de las 14h).

Salida en pantalla:

```
root@ls-st:/usr/local/bin# cortesia.sh
Buenos días
Gracias root por utilizar esta app
root@ls-st:/usr/local/bin# su adm_samba
adm_samba@ls-st:/usr/local/bin$ cortesia.sh
Buenos días
Gracias Antonio J. León por utilizar esta app
adm_samba@ls-st:/usr/local/bin$ exit
exit
root@ls-st:/usr/local/bin# su adm_nfs
adm_nfs@ls-st:/usr/local/bin$ cortesia.sh
Buenos días
Gracias Jose Sanchez por utilizar esta app
adm_nfs@ls-st:/usr/local/bin$ exit
exit
root@ls-st:/usr/local/bin# su adm_odoo
adm_odoo@ls-st:/usr/local/bin$ cortesia.sh
Buenos días
Gracias adm_odoo por utilizar esta app
```

```
#!/bin/bash
HORA=$(date +%H)
if [ $HORA -lt 14 ]; then
     echo Buenos dias
else
     echo Buenas tardes
fi
USUARIO=$(grep "^$USER:" /etc/passwd | cut -d":" -f5 | cut -d"," -f1)

if [ -z "$USUARIO"] ; then
     USUARIO=$USER
fi
echo "Gracias $USUARIO por utilizar esta app"
```


4.- El comando ping –c 1 dir_IP tiene éxito si la dirección IP dir_IP es una dirección de red accesible. Crear un script, red_ok.sh, que compruebe si las direcciones **IP que recibe por la línea de comandos son accesibles o no**, mostrando en pantalla el mensaje correspondiente.

Hay que tener en cuenta que:

a) Este script sólo lo puede ejecutar root.

b) No se debe hacer nada si no se recibe ningún argumento.

c) Debe mostrar al final una estadística sencilla en la que se muestre lo siguiente:

Total direcciones recibidas: xx

Direcciones accesibles: xx

```
Direcciones inaccesibles o erróneas: xx
#!/bin/bash
#Comprobar numero de argumentos
if [ $# -eq 0 ] ; then
      echo "Debe intoducir una dirección IP"
      exit 5
fi
#Solo root o administardor puede ejectuarlo
if [ $UID -ne 0 ] ; then
      echo "Solo root"
      exit 6
fi
for dir_IP in $* ; do

      echo "Procesando $dir_IP"

      ping -c 2 $dir_IP &>/dev/null

      exito=$?
      echo "La ejecucion ha resultado en $exito"


      if [ "$exito" = "0" ] ; then
            echo "hay conectividad a $dir_IP"
            ((ACC++))
      else
            echo "No hay conectividad con $dir_IP"
            ((ERR++))
      fi


done


echo "Total direcciones recibidas: $#"
echo "Direcciones accesibles: $ACC"
echo "Direcciones inaccesibles o erróneas: $ERR"
```

5.- Crea un script, **gestusuario.sh**, que solicite la entrada de un usuario (login) y diga si este existe o no. La comprobación debe hacerse en una **función:** *existe_usuario.* Esta función recibe el nombre del usuario y devuelve 0 si existe y 1 si no existe.

```
#!/bin/bash
existe_usuario(){
      grep "^$1:" /etc/passwd &> /dev/null
      return $?
}
read -p "introduce nombre: " LOGIN_US
existe_usuario $LOGIN_US
R=$?
```

```
if [ "$R" -eq "0" ] ;then
     echo "Elusaurio $LOGIN_US existe "
else
     echo "Elusaurio $LOGIN_US no existe "
fi
```

6.- Crea un script, ubicándolo en un directorio adecuado, que cuando se ejecute muestre en pantalla tanto los argumentos que recibe como el número de argumentos que recibe. En el caso de no recibir ningún argumento, hay que mostrar el mensaje "Usted no ha introducido ningún argumento".

```
#!/bin/bash
if [ $# -eq 0 ] ;then
     echo "usted no ha introducido ningún argumento"
     exit 4
fi
echo "Numero de argumentos: $#"
echo "Argumentos: "
for i in $*; do
     echo $i
done
```