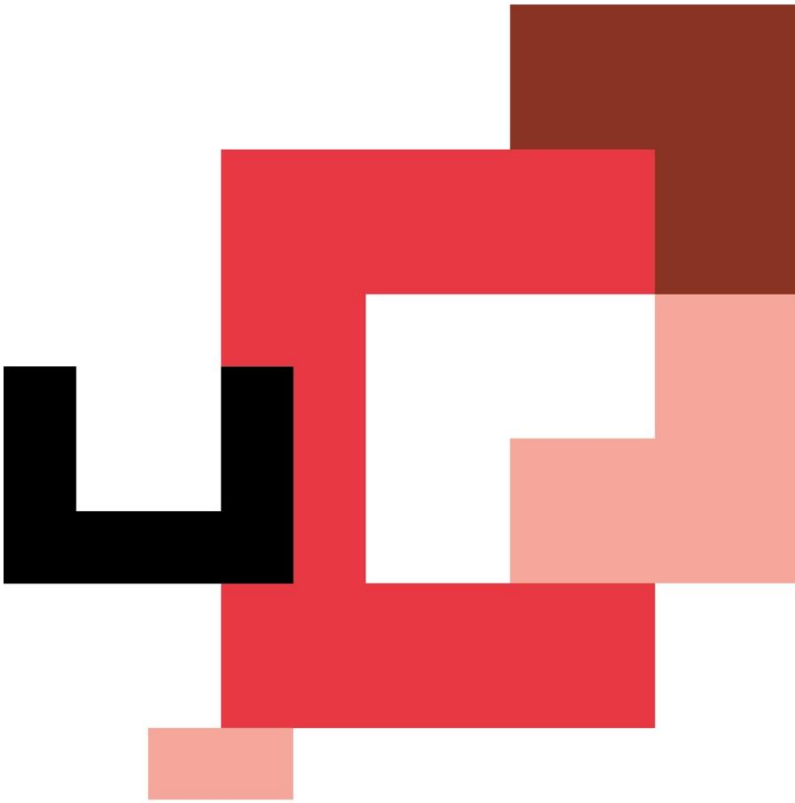
THE BRIDGE

BOOTCAMP Ciberseguridad en formato online



1. Cómo ejecutar scripts Bash

Para escribir en Bash puedes hacerlo directamente desde la consola/terminal de GNU/Linux, sin embargo, su mayor potencia radica en poder escribir 'scripts' y ejecutar piezas de código para automatizar tareas. Aquí muestro paso a paso un ejemplo.

1.1. Escribe un 'script' Bash

Abre tu editor de textos favoritos y escribe el siguiente código.

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

Guárdalo como **holamundo.sh** (la extensión no importa pero ésta es renococida por muchos editores de texto) .

1.2. Dale permisos

Para lanzar el 'script' Bash, dále permisos de ejecución. Para ello escribe en tu terminal/consola de comandos:

chmod u+x holamundo.sh

Recuerda ejecutar este comando y los siguientes en la misma carpeta/directorio donde se encuentra holamundo.sh

1.3. Ejecútalo

Lanza el 'script' ejecutando en tu terminal/consola:

./holamundo.sh

Si no ha habido ningún problema se ejecutará el 'script' de Bash mostrando la frase "Hola mundo".

2. Características Bash

Aquí se mostrarán las características de dicho intérprete de comandos o *shell* de GNU. Recuerda que los **comandos** son las acciones a ejecutar desde la consola/terminal Linux. En la siguiente sección hay una recopilación de ellos.

2.1. Comentarios

Los comentarios en Bash se hacen línea a línea con el símbolo **#**.

```
#!/bin/bash

#
# Hola Mundo comentado
#

echo "Hola mundo"
```

2.2. Variables

Las variables no tienen tipo, es decir, una variable puede contener una cadena, un número, etc. sin necesidad de definirlo.

La sintaxis es:

nombre_variable=valor_variable

Es obligatorio no dejar espacios antes o despues del simbolo '=' ya que sino Bash interpretaría la variable como un comando Linux.

Para acceder a una variable simplemente escribe como prefijo **$** en el nombre de la variable.

echo $varname

Un ejemplo sencillo:

```
#!/bin/bash

# Asignación y salida de variables

mivariable="Me llamo Nacho"
echo $mivariable
```

2.2.1. Paso de variables

Cuando ejecutas desde tu terminal/consola tienes la posibilidad de pasarle más argumentos. Por ejemplo:

./miScript.sh hola 4

Para recoger estos valores escribe **$** y a continuación el número de posición del argumento pasado. El primer argumento tiene valor **$1**, que sería 'hola', y el segundo argumento sería $2, en el ejemplo sería el número 4. La variable **$0** es el propio nombre del archivo.

```
#!/bin/bash

#
# Paso de variables
#
```

```
echo "Tu primer argumento es" $1
echo "Tu segundo argumento es" $2
```
También hay que destacar que **$?** guarda el valor de salida del último comando ejecutado. **$*** almacena todos los argumentos y **$#** es el número de argumentos pasados.

2.3. Comparaciones y/o expresiones

Los condicionales y bucles se rigen mediante la evaluación de una expresión. Por eso lo primero es saber cómo programar en Bash las evaluaciones de una expresión.

La evaluación de una expresión da como resultado **verdadero** o **falso**. Si la comparación o evaluación de la expresión es verdadera se ejecutará el bucle o la condicional, si es falsa la evaluación no se ejecutará.

En Bash, la sintaxis es la siguiente:

test *expresión*
ó

[*expresión*]
Un ejemplo de expresión sería **[3 -eq 5]** que comprueba si el valor 3 es igual a 5. Como es incorrecto, el valor que devuelve es falso. Los símbolos **[** y **]** tienen que estar obligatoriamente separados por un espacio.

2.3.1. Comparaciones numéricas

numero1 -eq numero2: Devuelve verdadero si 'numero1' es igual a 'numero2'.
numero1 -ge numero2: Devuelve verdadero si 'numero1' es igual o mayor a 'numero2'.
numero1 -gt numero2: Devuelve verdadero si 'numero1' es mayor a 'numero2'.
numero1 -le numero2: Devuelve verdadero si 'numero1' es igual o menor a 'numero2'.
numero1 -lt numero2: Devuelve verdadero si 'numero1' es menor a 'numero2'.
numero1 -ne numero2: Devuelve verdadero si 'numero1' no es igual a 'numero2'.

2.3.2. Comparaciones de cadenas

cadena1 = cadena2: Devuelve verdadero si 'cadena1' es idéntica a 'cadena2'.
cadena1 != cadena2: Devuelve verdadero si 'cadena1' no es idéntica a 'cadena2'.
cadena1: Devuelve verdadero si 'cadena1' es nulo (no significa que su longitud sea cero).

-n cadena1: Devuelve verdadero si la longitud de caracteres de 'cadena1' es mayor que cero.
-z cadena1: Devuelve verdadero si la longitud de caracteres de 'cadena1' es cero.

2.3.3. Comparaciones de ficheros

-d nombrefichero: Devuelve verdadero si el fichero es un directorio.
-f nombrefichero: Devuelve verdadero si el fichero es un archivo.
-r nombrefichero: Devuelve verdadero si el fichero puede ser leído.
-w nombrefichero: Devuelve verdadero si el fichero puede ser escrito.
-x nombrefichero: Devuelve verdadero si el fichero es ejecutable.

2.3.4. Comparaciones de expresiones

!*expresión*: Devuelve verdadero si la expresión no se cumple.
expresión1* -a *expresión2: Devuelve verdadero si la expresión1 y la expresión2 se cumplen (también vale **&&**).
expresión1* -o *expresión2: Devuelve verdadero si la expresión1 o la expresión2 se cumplen (también vale **||**).

2.4. Condicionales

En programación, una sentencia condicional es una instrucción que se pueden ejecutar o no en función del valor de una expresión. En Bash, las condicionales más populares son los siguientes:

2.4.1. If – Then

```
if [ expresión ]
then
comandos
fi
```

2.4.2. If – Then – Else

```
if [ expresión ]
then
comandos
else
comandos
fi
```

2.4.3. If – Then – Else if – Else

```
if [ expresión1 ]
then
comandos
elif [ expresión2 ]
then
comandos
else
comandos
fi
```

2.4.4. Case

```
case cadena in
```

```
cadena1)
comandos
;;
cadena2)
comandos
;;
*)
comandos
;;
esac
```

Se comprueba ***cadena***. Si concuerda con ***cadena1*** se ejecutará los comandos correspondientes hasta llegar a **;;**. Lo mismo ocurre con ***cadena2***. Si ***cadena*** no coincide con ***cadena1*** o ***cadena2*** entonces se ejecutará *****. Se puede añadir tantas cadenas de verificación como uno desee.

2.5. Bucles

Un bucle repite los comandos que uno ha escrito tantas veces hasta que la expresión se verifique.

2.5.1. For

Existen muchas maneras de realizar un bucle *for* en Bash. Yo sólo uso ésta:

```
for (( inicializador; condición; incremento ))
do
        comandos
done
```

Su sintaxis es casi idéntica a C. Aquí un ejemplo:

```
#!/bin/bash
for (( c=1; c<=5; c++ ))
do
        echo "Bienvenido $c veces..."
done
```

2.5.2. While

```
while [ expresión ]
do
comandos
done
```

2.5.3. Until

```
until [ expresión ]
do
comandos
done
```

3. Comandos Linux

Lista de comandos más importantes según la Wikipedia.

3.1. Ayuda

man: muestra manual del comando que le indiquemos.
--help: da una ayuda de los comandos.

3.2. Archivos y directorios

ls: lista los archivos y directorios.
sort: ordena alfabéticamente una lista de archivos.
cd: cambio de directorio.
pwd: muestra la ruta al directorio actual.
tree: muestra la estructura de directorios y archivos en forma gráfica.
mkdir: crea un directorio.
rmdir: borro directorios.
rm -r: borra directorios no vacíos.
cp: copia archivos.
rm: borra archivos.
mv: mueve o renombra archivos y directorios.
cat: ve el contenido de uno o varios archivos.
more: ve el contenido de los archivos.
less: ve el contenido de los archivos.
split: dividir archivos.
find: busca archivos.
locate: localiza archivos según una lista generada.
updatedb: actualiza la lista de los archivos existentes.
whereis: muestra la ubicación de un archivo.
file: muestra el tipo de archivo.
whatis: muestra descripción del archivo.
wc: cuenta líneas palabras o caracteres en un archivo.
grep: busca un texto en archivos.
head: muestra el inicio de un archivo.
tail: muestra el final de un archivo.
tailf: muestra el final de un archivo y lo que se añada en el instante (logs).
tr: reemplaza caracteres en un fichero de texto.
sed: cambia una cadena de caracteres por otra.
join: cruza la información de dos archivos y muestra las partes que se repiten.
paste: toma la primera línea de cada archivo y las combina para formar una línea de salida.
uniq: elimina líneas repetidas adyacentes del archivo entrada cuando copia al archivo salida.
cut: sirve para seleccionar columnas de una tabla o campos de cada línea de archivo.
ln: crea enlaces a archivos o carpetas.
diff: muestra las diferencias entre dos archivos.
fuser: muestra que usuario tiene en uso o bloqueado un archivo o recurso.
tar: empaqueto archivos.
gzip: comprime archivos gz.
gunzip: descomprime archivos gz.
compress: comprime archivos Z.
uncompress: descomprime archivos Z.
chmod: cambio permisos a archivos y directorios.
chown: cambio de propietario.
chgrp: cambio de grupo.
vi: abre el editor de texto vi.
pico: edita un fichero de texto.

3.3. Usuarios

adduser: agregó nuevo usuario.
useradd: agregó nuevo usuario.
userdel: borra un usuario.
passwd: permite cambiar la contraseña.
su: cambio de usuario.
whoami: muestra el nombre de usuario.
logname: muestra el nombre de usuario.
id: muestra datos de identificación del usuario.
finger: da información de usuario.
chfn: cambia la información del finger.
who: muestra los usuarios del sistema.
w: muestra un detalle de los usuarios.
last: información de los últimos usuarios que han usado el sistema.
mail: programa de correo.
pine: lector de correo en modo texto.
write: manda un mensaje a la pantalla de un usuario.
mesg: activo o desactivo recibir mensajes.
wall: mensaje a todos los usuarios.
talk: establecer una charla con otro usuario.
banner: saca letrero en la pantalla.
set: da información sobre el entorno del usuario.
addgroup: agregó nuevo grupo.
groupadd: agregó nuevo grupo.
chown: cambia el propietario de un fichero.

3.4. Procesos

top: muestra los procesos que se están ejecutando y permite matarlos.
ps: muestra la lista de procesos del usuario.
ps aux: muestra la lista de procesos de la máquina.
kill: mata proceso por ID.
killall: mata proceso por nombre.
time: mide el tiempo que tarda un proceso en ejecutarse.
fg: trae a primer plano un proceso parado o en segundo plano.
bg: pone un proceso en segundo plano.
&: colocado al final de la línea de comando ejecuta en segundo plano.
nice: ajusta la prioridad de un proceso de -20 a 19.

3.5. Discos

mount: monta un disco.
umount: desmonta un disco.
df: muestra el espacio libre de los discos.
du: muestra el espacio usado por el disco o un directorio.
mkfs: formateo un disco.
fsck: estado del disco.
fdisk: gestión de particiones.

3.6. Red

netstat: muestra estado de la red.
ifconfig: muestra la configuración del dispositivo de red.
iwconfig: muestra la configuración del dispositivo de red inalámbrico.
nmap: escanea la red y muestra los puertos que se encuentran disponibles.
ping: indica si hay respuesta por parte del servidor.
nslookup: me da la IP de nuestro servidor DNS.

telnet: me conecto a un equipo remotamente.
netconf: configuro la red.
ntop: muestra los procesos de la red.
route -n: muestra la tabla de rutas.

3.7. Sistema

rlogin: se conecta a otra máquina de forma remota (remote login).
rsh: se conecta a otra máquina de forma remota (remote shell).
ftp: se conecta a otra máquina por el protocolo ftp.
reboot: reinicia la máquina.
halt: apaga el sistema.
shutdown: apaga el sistema.
init0: apaga la máquina.
init6: reinicia la máquina.
uptime: muestra el tiempo transcurrido de encendida la máquina.
exit: cierro sesión actual.
logout: salgo del sistema.
nohup: proporciona inmunidad frente a rupturas de comunicación..
dmesg: muestra mensajes del arranque del ordenador.
history: muestra todos los comandos digitados por el usuario.
uname: da información del sistema operativo.
tee: copia la entrada estándar a la salida estándar y a un archivo.
host: muestra la dirección IP del servidor en una red local.
hostname: muestra el nombre del servidor.
umask: muestra y permite cambiar la máscara de usuario.
chroot: cambia la raíz para que root ejecute algo en forma particular.
chsh: cambia el login shell.
free: estado de la memoria.
date: muestra fecha y hora actual.
cal: muestra calendario.
clear: borro la pantalla.
at: ejecuta un comando más tarde.
env: ver variables de entorno.
export: permite el uso de variables por programas en todos los caminos del usuario.
modprobe: cargo modulo.
startx: arranca el servidor X.
xev: muestra los eventos de las teclas y el ratón.
lspci: muestra los periféricos conectados al puente pci.
lsmod: muestra los modulos cargados en el sistema.
echo: escribe un mensaje en la salida estándar.
alias: crear un alias. Un comando largo abreviado en pocas letras.
unalias: borrar un alias.
bc: calculadora.
mc: ejecuta Midnight Commander.
xkill: mata una ventana gráfica.
rpm: instala los paquetes rpm RedHat.
dpkg: instala los paquetes deb Debian.
kernelcfg: manejo los modulos cargados en el kernel.
insmod: inserta modulos en el kernel.
rmmod: elimina modulos del kernel.
updatedb: actualiza la base de datos interna de archivos.
sh: cambia al bash shell.
setxkbmap: por si no funcionan las teclas con AltGr en modo X.

4. Ejemplos

Introducir dos números diferentes e indicar cuál es el mayor

```
#!/bin/bash
echo "Introducir dos números:"
read A
read B
if [ $A -gt $B ]
then
        echo $A "es el mayor"
else
        echo $B "es el mayor"
fi
```

Pasar dos números como parámetros e indicar el menor

```
#!/bin/bash
echo $#
if [ $# -ne 2 ]
then
        echo "Falta algún parámetro"
elif [ $1 -eq $2 ]
        then
                echo "Son iguales"
        elif [ $1 -lt $2 ]
                then
                        echo $1 "es menor"
                else
                        echo $2 "es menor"
fi
```

Ver los procesos que está ejecutando un usuario concreto

```
#!/bin/bash
RES=s
while [ $RES = s ]
do
        echo "Introducir nombre de usuario:"
        read USU
        ps aux|grep $USU
        echo "¿Desea continuar?"
        read RES
done
```

Mostrar los usuarios que pasamos como parámetros y saber si están conectados

```
#!/bin/bash
for i in $*
do
        if who|grep -s $i>/dev/null
        then
                echo $i si está conectado
        else
                echo $i no está conectado
        fi
done
```

THE BRIDGE