

Mi github y los scripts : <https://github.com/adrianbii/Cyber>

1. Haz un script llamado **multiplo10.sh** que pida un número por teclado e indique si es un múltiplo de 10 o no.

```
➤ $ ./multiplos10.sh 10
es multiplo
[d4rkusx@tamosHACK]~[/scripts]
➤ $ cat multiplos10.sh
#!/bin/bash

#asignamos a una variable el resto de la division
x=$(( ${1%10} ))
#simple comparacion
if [ $x -eq 0 ];
then echo "es multiplo"
else echo "No es multiplo"
fi
```

2. Haz un script llamado **altura_mayor.sh**, que pida por teclado la altura en centímetros de tres personas y nos diga la mayor de las 3 en metros.

```
[d4rkusx@tamosHACK]~[/scripts]
➤ $ cat altura_mayor.sh
#!/bin/bash
read -p " altura en cm de la primera persona" f
read -p "altura de la segunda perosna en cm" se
read -p " altura de la tercera persoma" t

if [ $f -gt $se || $f -gt $t ];
then
$mayor = $f / 100
echo " la altura de la persona mayor es: $mayor"
elif [ $se -gt $f || $se -gt $t ];
then
$mayor = $se / 100
echo " la altura de la persona mayor es: $mayor"
else
$mayor = $t / 100
echo " la altura de la persona mayor es: $mayor"
fi
```

3. Haz un script llamado **cuenta10ficheros.sh**, que nos diga si en el directorio pasado como parametro hay más de 10 ficheros o no, es decir, los directorios no deben ser contados.

```
[d4rkusx@tamosHACK]~/scripts
$ cat cuenta10ficheros.sh
#!/bin/bash
#Cuenta los ficheros que hay en un directorio pasado como parámetro
cont=0
for i in `ls $1`
do
#Para poder referenciar al elemento correcto, debo dar la ruta completa si el script se lanza e
n otro directorio distinto
    if [ -f $1/$i ]
    then
        echo $1/$i
        ((cont++))
    fi
done
#comprobamos que hay mas de 10 ficheros
if [ $cont -gt 10 ]; then
echo "En este directorio hay mas de 10 ficheros"
else echo " en este directorio no hay as de 10 archivos"
fi
```

4. Haz un script llamado **decada_edad.sh**, que dada una edad que introducimos por pantalla, nos diga como resultado la década en la que nacimos (ejemplo 70, 80, 90 ...). Suponemos que la edad introducida está entre 15 y 60 años. El script debe coger el año actual de forma automática. La salida debe producirse en el siguiente formato: "Si naciste en 1983, naciste en la decada de 1980".

```
fi
[d4rkusx@tamosHACK]~/scripts
$ cat decadaedad.sh
#!/bin/bash

read -p "introduce tu edad " edad
año=$( date +%G )
decada=$(( "año"-"$edad" ))
case $decada in
1960-1969) echo " Si naciste en $decada , naciste en la decada de 1960";;
1970-1979) echo " Si naciste en $decada , naciste en la decada de 1970";;
1980-1989) echo " Si naciste en $decada , naciste en la decada de 1980";;
1990-1999) echo " Si naciste en $decada , naciste en la decada de 1990";;
2000-2010) echo " Si naciste en $decada , naciste en la decada de 2000";;
esac
```

5. Haz un script llamado **diadelmes.sh**, que nos diga cuántos días tiene el mes actual en el momento de su ejecución (se considera que Febrero tiene 28 días). La salida debe producirse en el siguiente formato: "Estamos en noviembre, un mes con 30 días".

```
esac
[d4rkusx@tamosHACK]--[~/scripts]
$ cat diames.sh
#!/bin/bash

mes=$( date +"%m" )
echo "$mes"
#dependiendo del mes que recoja el sistema, lo compara con cada caso y nos devuelve la opcion c
orrect...
case $mes in
1) echo "Enero tiene 31 dias ";;
2) echo "Febrero tiene 28 dias ";;
3) echo "Marzo tiene 31 dias ";;
4) echo "Abril tiene 30 dias ";;
5) echo "Mayo tiene 31 dias ";;
6) echo "Junio tiene 30 dias ";;
7) echo "Julio tiene 31 dias ";;
8) echo "Agosto tiene 31 dias ";;
9) echo "Septiembre tiene 30 dias ";;
10) echo "Octubre tiene 31 dias ";;
11) echo "Noviembre tiene 30 dias ";;
12) echo "Diciembre tiene 31 dias ";;
esac
```

6. Haz un script llamado **horoscopochino.sh**, que pida el año en que nacimos (4 cifras) y nos diga por pantalla qué animal nos corresponden según el horóscopo chino. Para calcularlo debemos dividir el año entre 12 y el resto nos indicará el animal según la siguiente tabla.

8	El mono	0	La rata	4	El dragón
9	El gallo	1	El buey	5	La serpiente
10	El perro	2	El tigre	6	El caballo
11	El cerdo	3	El conejo	7	La cabra

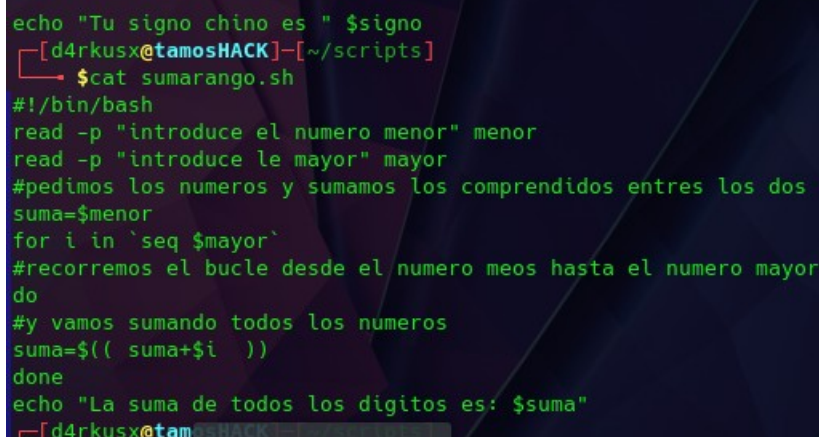
```
[d4rkusx@tamosHACK]--[~/scripts]
$ cat horoscopochino.sh
#!/bin/bash
#pedimos por parametro el año con 4 cifras
read -p "Inserta el año que naciste con 4 cifras: " ano
let zodiaco=ano%12
#comprobados a que caso pertenece y bum, nada de otro mundo
case $zodiaco in
0) signo=" el Mono" ;;
1) signo=" el Gallo" ;;
2) signo=" el Perro" ;;
3) signo=" el Cerdo" ;;
4) signo=" el Rata" ;;
5) signo=" el Buey" ;;
6) signo=" el Tigre" ;;
7) signo=" el Conejo" ;;
8) signo=" el Dragon" ;;
9) signo="el Serpiente" ;;
10) signo=" el Caballo" ;;
11) signo=" el Cabra" ;;
esac

echo "Tu signo chino es " $signo
```

7. Haz un script llamado **sumarango.sh**, que pida dos números por teclado y calcule la suma de los números que conforman ese rango. El resultado tiene que estar mostrado de dos formas, obtenido mediante el comando seq, y mediante la nomenclatura usada en lenguaje C.

Por ejemplo: Si introducimos el 5 y el 8 debemos mostrar el 26 por pantalla que es el resultado de sumar $5+6+7+8$.

OPCIONAL: El programa comprueba qué número es menor antes de calcular la suma del rango para poder invertir el orden en caso necesario, por si introdujeramos primero el mayor evitar entrar en un bucle erroneo.



```
echo "Tu signo chino es " $signo
[d4rkusx@tamoshack]~/scripts
$ cat sumarango.sh
#!/bin/bash
read -p "introduce el numero menor" menor
read -p "introduce le mayor" mayor
#pedimos los numeros y sumamos los comprendidos entres los dos
suma=$menor
for i in `seq $mayor`
#recorremos el bucle desde el numero meos hasta el numero mayor
do
#y vamos sumando todos los numeros
suma=$(( suma+$i ))
done
echo "La suma de todos los digitos es: $suma"
[d4rkusx@tamoshack]~/scripts
```

8. Implementar un script llamado **calculadora.sh** que muestre el siguiente menú:

- 1)Sumar
- 2)Restar
- 3)Dividir
- 4)Multiplicar
- 5)Salir

Después de mostrar el menú, se pedirá que se elija una opción. Si la opción elegida no está entre el 1 y el 4, se mostrará un mensaje de error. En caso de que la opción sea válida, se pedirán dos números por teclado y en función de la operación elegida, se devolverá el resultado por pantalla, manejando posibles errores al introducir caracteres.

- Cada operación será implementada haciendo uso de funciones
- Si la opción elegida no es válida, se volverá a mostrar el menú
- Si el script arroja algún error debe almacenarse en un fichero con log de errores creado para la ocasión (con contenido independiente cada vez que se ejecuta el script).
- El programa terminará, cuando pulse 5.

```
$ cat calculadora.sh
#!/bin/bash
sumar() { echo "Ingrese el primer número:"
    read a
    echo "Ingrese el segundo número:"
    read b
    result=`expr $a + $b`
    echo "El resultado es:$result"
}

resta() { echo "Ingrese el primer número:"
    read a
    echo "Ingrese el segundo número:"
    read b
    result=`expr $a - $b`
    echo "El resultado es:$result"
}

dividir() { echo "Ingrese el primer número:"
    read a
    echo "Ingrese el segundo número:"
    read b
    result=`expr $a / $b`
    echo "El resultado es:$result"
}

multiplicar() { echo "Ingrese el primer número:"
    read a
    echo "Ingrese el segundo número:"
    read b
    result=`expr $a \* $b`
    echo "El resultado es:$result"
}

falg="1"
while [ "$falg" -eq "1" ]
do
2>>/log_errores

echo "
echo "----CALCULADORA---"
echo "Elige una opcion:"
echo "1: suma"
echo "2: resta"
echo "3: dividir"
echo "4: multiplicar"
echo "5: ¡Sal de la calculadora!"
read check
```



```
    read b
    result=`expr $a \* $b`
    echo "El resultado es:$result"
}

falg="1"
while [ "$falg" -eq "1" ]
do
2>>/log_errores

echo "
echo "---CALCULADORA---"
echo "Elige una opcion:"
echo "1: suma"
echo "2: resta"
echo "3: dividir"
echo "4: multiplicar"
echo "5: ¡Sal de la calculadora!"
read check
echo "Valor: $check"
if [ "$check" -eq "1" ]
then
    sumar
elif [ "$check" -eq "2" ]
then
    resta
elif [ "$check" -eq "4" ]
then
    multiplicar
elif [ "$check" -eq "3" ]
then
    dividir
elif [ "$check" -ne "5" ]
then continue # con esto hacemos que cada vez ue se introduzca un numero distinto, el programa s
igue preguntandonos

else
    falg="0"
fi
done
```

9. Realizar un script que se llame **usuarios.sh** muestre la lista de información de un usuario del sistema pasado por parámetro (nombre de usuario, UID, GID y directorio de trabajo).

Cuando se muestre la información el script debe preguntar si quiere introducir otro usuario para mostrar su info o si se quiere salir del programa.

Se debe verificar que ese usuario está dado de alta en el sistema y por supuesto deberá mostrarse por pantalla el mensaje oportuno de darse tal circunstancia. Los errores arrojados por los comandos empleados serán enviados a un log destinado a ello, aunque el script avisará por pantalla si el usuario no existe.

```
done
[d4rkusx@tamosHACK]--[~/scripts]
→ $cat usuarios.sh
#!/bin/bash

if [ -z $1 ]; then $1=$USER
fi

if [ $USER = -list ]; then

echo
grep -w /bin/bash$ /etc/passwd | cut -d: -f1 | grep -v root | sort
echo
elif ! id $1 &>> /errores.txt; then
echo " el usuario $1 no existe en el sistema"
else
echo "Infomacion del usuario $1"
echo
echo "UID:  $(grep -w ^$1 /etc/passwd | cut -d: -f3)"
echo "GID:  $(grep -w ^$1 /etc/passwd | cut -d: -f4)"
echo "HOME: $(grep -w ^$1 /etc/passwd | cut -d: -f6)"

fi
[d4rkusx@tamosHACK]--[~/scripts]
→ $
```

10. Hacer un script llamado **copia_scripts.sh**, que haga un copiado de los scripts creados en la carpeta pasada como parámetro, deben ser empaquetados con el comando tar y el nombre del archivo debe tener el siguiente formato: "copia_scripts_ddmmaaaa.tar" siendo dd el día, mm el mes y aaaa el año en el que se produce la copia.

```
[d4rkusx@tamosHACK]--[~/scripts]
→ $cat copia_scripts.sh
#!/bin/bash
#damos formato a la hora que vamos añadir a los archivos backups
Date=$( date +%d-%m-%Y )
# al magia de este script, con el comando tar hacemos una copia empaquetada de la carpeta
tar -cvf copia-scripts-$Date.tar.gz /home/d4rkusx/scripts/*
[d4rkusx@tamosHACK]--[~/scripts]
```

11. OPCIONAL. Realiza un script que suba a tu cuenta de GitHub todos los ejercicios de la actividad pidiendo por parámetro el nombre de usuario correspondiente de la plataforma. Si encuentras problema con la autenticación en GitHub mediante token, hazlo de manera local en tu repositorio o introduce el token al conectar con github antes de lanzar el script.

IMPORTANTE: Todo el flujo de datos de entrada al script debe ser controlado, es decir, si se espera recibir por parámetro un número, una dirección o una cadena válida debe ser comprobado por el mismo.

IMPORTANTISIMO: Los scripts deben estar **COMENTADOS**.

FORMATO DE ENTREGA: T1A2_apellidos_nombre.pdf