

## INTRODUCCIÓN UNIX

**Fecha:** 30/06/2025  
**Práctica:** Bash Scripting - Flujos de Control en Estructuras Selectivas  
**Modalidad:** Individual/Grupal  
**Integrantes:** 1) Dilan Galvez  
2) Anyela carpio

### 1. Objetivos.

- Aplicar los conceptos de estructuras condicionales en Bash Scripting.
- Validar las estructuras condicionales en Bash Scripting.

### 2. Introducción.

#### Flujo de Control: Estructuras Selectivas

Un condicional es una estructura de control selectiva que ejecuta un bloque de código al evaluarse una o varias condiciones lógicas, esta puede ser verdadera (True) cuando la condición se cumple o falsa (False) cuando la condición no se cumple. Los scripts se vuelven dinámicos durante su ejecución, lo que los hace más flexibles y potentes.

Los diferentes operadores lógicos son similares a los que se encuentran en otros lenguajes de programación. Sin embargo, la sintaxis difiere para Bash Script.

Si se antepone el carácter !, se genera una negación de la condición.

Se ingresa diferentes comandos en una sola línea separando con el carácter ;

Condicionales cadena:

- ==
- !=

Condicionales numéricas:

- -eq: equal (==)
- -ne: not equal (!=)
- -lt: less than (<)
- -le: less or equal (<=)
- -gt: greater than (>)
- -ge: greater or equal (>=)
- -o: or (||)
- -a: and (&&)

Otras condiciones:

- -f: lo evaluado es un archivo y existe
- -x: lo evaluado es un ejecutable y existe
- -l: lo evaluado es un enlace directo
- -e: lo evaluado existe

- -d: lo evaluado es un directorio y existe
- -n: la variable no está vacía
- -z: la variable está vacía

fichero="/etc/passwd"

- -f "\$fichero" La salida es True
- -x "\$fichero" La salida es False
- -d "\$fichero" La salida es False
- -e "\$fichero" La salida es True
- -n "\$fichero" La salida es True
- -z "\$fichero" La salida es False

Los condicionales pueden limitar su funcionamiento, por lo que se puede controlar las variables de ingreso al script. Si se trabaja con variables y estas no se han ingresado se puede analizar como condición para su ejecución. Se analiza la variable #0 que identifica el número de parámetros.

```
#!/bin/bash  
# {Comment}
```

```
if [ $# -ne {Imput_Number} ]; then  
    echo "Invocar el programa $0 como se indica a continuación"  
    echo "${0}.sh {File} "  
    exit 1  
fi  
  
exit 0
```

### Formato del Condicional

El condicional `if` puede ser simple, complejo hasta anidado. Si la condición no se cumple, `||` ejecuta estos comandos.

Diferentes tipos de condicionales en bash.

```
[{Condition}]&&{{Command}}; { Command };;  
[{Condition}]||{{Command}}; { Command };;
```

```
if []  
then  
    ...  
fi
```

```
if []  
then  
    ...  
else  
    ...  
fi
```

```
if []
then

...
elif []
then

...
else
...
fi

case {Variable} in
    {Condition 1})
        {Command}
        ;;
    {Condition 2})
        {Command}
        ;;
    *)
        {Command}
        ;;
esac
```

Se verifica si la variable de entrada es un directorio o un fichero.

```
#!/bin/bash
# {Comment}

if [-f "$1"]; then
    echo "$1 es un fichero."
elif [-d "$1"]; then
    echo "$1 es un directorio"
else
    echo "Formato no compatible."
fi

exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ nano verifica_tipo.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ echo "Hola mundo" > archivo.txt

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ mkdir carpeta_prueba

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ chmod +x verifica_tipo.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./verifica_tipo.sh archivo.txt
archivo.txt es un fichero.

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./verifica_tipo.sh carpeta_prueba
carpeta_prueba es un directorio.
```

Se verifica si la variable de entrada es un fichero. Se cuenta el número de líneas.

```
#!/bin/bash
# {Comment}

if [-f "$1"]; then
    lineas=$(cat $1 | wc -l)
    echo "$1 es un fichero con $lineas líneas."
fi
exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ nano contar_lineas.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ echo -e "Primera línea\nSegunda línea\nTercera línea" > texto.txt

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ chmod +x contar_lineas.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ chmod +x contar_lineas.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./contar_lineas.sh texto.txt

texto.txt es un fichero con 3 líneas.
```

Se verifica si las variables de entrada son ficheros.

```
#!/bin/bash
# {Comment}

if [-f "$1" -a -f "$2"]; then
    echo "Los archivos $1 y $2 son ficheros."
else
    echo "Uno o ambos archivos no son ficheros."
fi
exit 0
```



```
(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ nano verifica_dos_ficheros.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ echo "Archivo uno" > archivo1.txt
echo "Archivo dos" > archivo2.txt

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ chmod +x verifica_dos_ficheros.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./verifica_dos_ficheros.sh archivo1.txt archivo2.txt

Los archivos archivo1.txt y archivo2.txt son ficheros.

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./verifica_dos_ficheros.sh carpeta_prueba texto.txt

Uno o ambos archivos no son ficheros.
```

Se verifica si hay dos variables.

```
#!/bin/bash
# {Comment}

[ $# -ne 2 ] &&{
    echo "$0 {Variable} {Variable}"
    exit 1
};
exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ nano verificar_dos_parametros.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ chmod +x verificar_dos_parametros.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ls
archivo1.txt  carpeta_prueba  texto.txt  verifica_dos_ficheros.sh
archivo2.txt  contar_lineas.sh  verifica_ambos_archivos.sh  verificar_dos_parametros.sh
archivo.txt   practica_bash    verifica_archivos.sh        verifica_tipo.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./verificar_dos_parametros.sh archivo1.txt archivo2.txt
Dos variables recibidas: archivo1.txt y archivo2.txt

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ cat verificar_dos_parametros.sh
#!/bin/bash
# Verifica si se ingresaron dos variables

[ $# -ne 2 ] && {
    echo "Uso correcto: $0 variable1 variable2"
    exit 1
}

echo "Dos variables recibidas: $1 y $2"
exit 0
```

```
#!/bin/bash
# {Comment}

[-f "$1" -a -f "$2"] &&{
    echo "Ambos archivos existen";
    exit 0;
};

exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ nano verifica_ambos_archivos.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ chmod +x verifica_ambos_archivos.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ls
archivo1.txt  carpeta_prueba  texto.txt  verifica_dos_ficheros.sh
archivo2.txt  contar_lineas.sh  verifica_ambos_archivos.sh  verificar_dos_parametros.sh
archivo.txt   practica_bash    verifica_archivos.sh        verifica_tipo.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ cat verifica_ambos_archivos.sh
#!/bin/bash
# Verifica si ambos archivos existen

[ -f "$1" -a -f "$2" ] && {
    echo "Ambos archivos existen"
    exit 0
}

echo "Uno o ambos archivos NO existen"
exit 1

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./verifica_ambos_archivos.sh archivo1.txt archivo2.txt
Ambos archivos existen

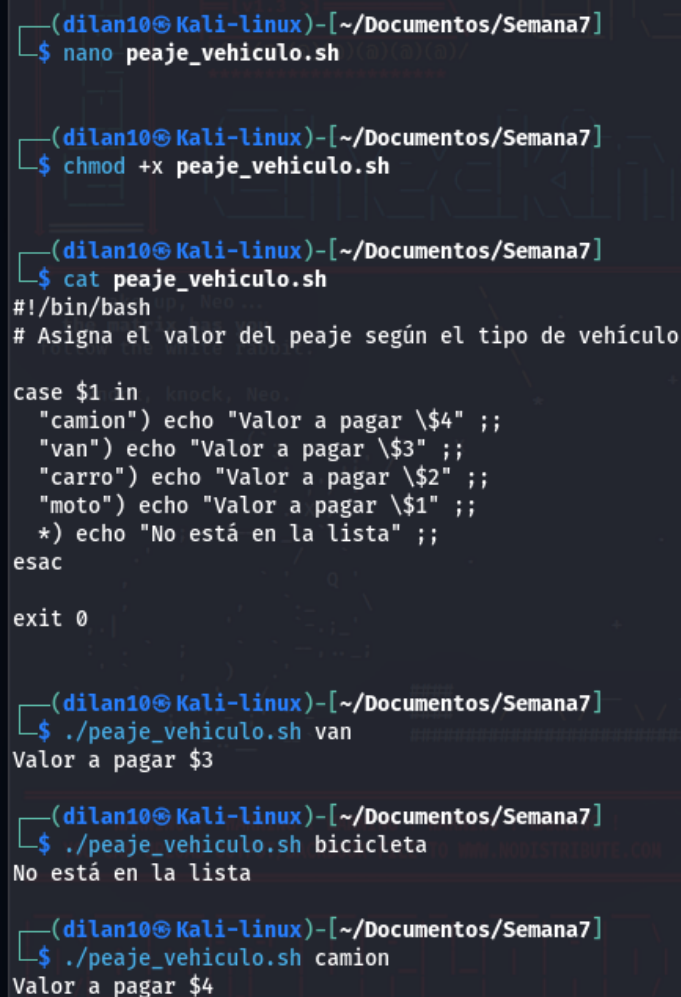
(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./verifica_ambos_archivos.sh archivo1.txt archivoX.txt
Uno o ambos archivos NO existen
```

Se asigna el valor del peaje según el tipo de vehículo que tiene.

```
#!/bin/bash
# {Comment}

case $1 in
    "camion") echo "Valor a pagar \"$4\"";;
    "van") echo "Valor a pagar \"$3\"";;
    "carro") echo "Valor a pagar \"$2\"";;
    "moto") echo "Valor a pagar \"$1\"";;
    *) echo "No está en la lista";;
Esac

exit 0
```



```
(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ nano peaje_vehiculo.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ chmod +x peaje_vehiculo.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ cat peaje_vehiculo.sh
#!/bin/bash
# Asigna el valor del peaje según el tipo de vehículo

case $1 in
    "camion") echo "Valor a pagar \"$4\"";;
    "van") echo "Valor a pagar \"$3\"";;
    "carro") echo "Valor a pagar \"$2\"";;
    "moto") echo "Valor a pagar \"$1\"";;
    *) echo "No está en la lista";;
esac

exit 0

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./peaje_vehiculo.sh van
Valor a pagar $3

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./peaje_vehiculo.sh bicicleta
No está en la lista

(dilan10@Kali-linux)-[~/Documentos/Semana7]
$ ./peaje_vehiculo.sh camion
Valor a pagar $4
```



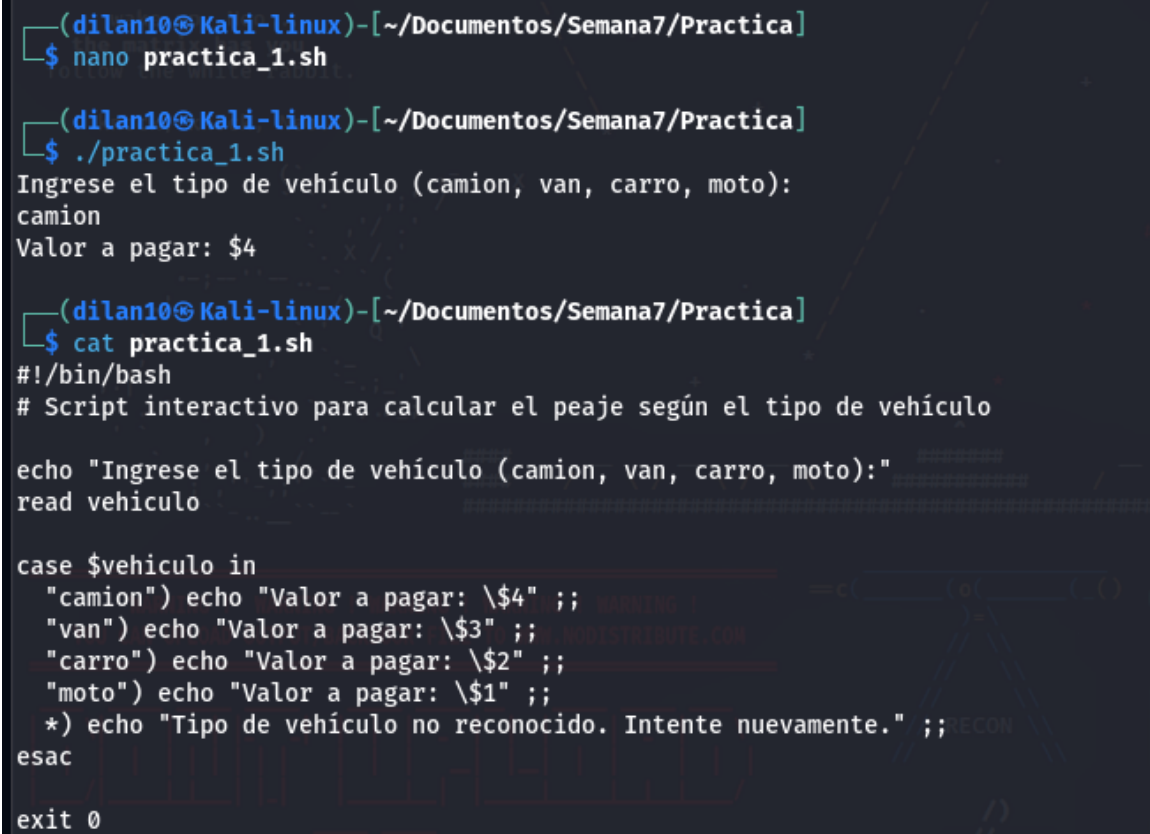
## Prácticas de Laboratorio

A continuación, se validan diferentes ejercicios.

Crear un script interactivo en Bash Script que implemente el pago de un peaje para diferentes tipos de automotores. Definir la lógica de control a través de un pseudocódigo o diagrama de bloques Presente en consola el resultado.

```
#!/bin/bash
# {Comment}
```

```
exit 0
```



```
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ nano practica_1.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ ./practica_1.sh
Ingrese el tipo de vehículo (camion, van, carro, moto):
camion
Valor a pagar: $4

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ cat practica_1.sh
#!/bin/bash
# Script interactivo para calcular el peaje según el tipo de vehículo

echo "Ingrese el tipo de vehículo (camion, van, carro, moto):"
read vehiculo

case $vehiculo in
    "camion") echo "Valor a pagar: \$4" ;;
    "van") echo "Valor a pagar: \$3" ;;
    "carro") echo "Valor a pagar: \$2" ;;
    "moto") echo "Valor a pagar: \$1" ;;
    *) echo "Tipo de vehículo no reconocido. Intente nuevamente." ;;
esac

exit 0
```

Crear un script interactivo en Bash Script que identifique los días de la semana. Definir la lógica de control a través de un pseudocódigo o diagrama de bloques. Presente en consola el resultado.

```
#!/bin/bash
# {Comment}
```

```
exit 0
```



```
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ nano practica_2.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ cat practica_2
cat: practica_2: No existe el fichero o el directorio

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ cat practica_2.sh
#!/bin/bash
# Script que identifica los días de la semana

echo "Ingrese un día de la semana:"
read dia

case $dia in
    "lunes") echo "Es el primer día laboral." ;;
    "martes") echo "Segundo día de la semana laboral." ;;
    "miércoles") echo "Mitad de semana." ;;
    "jueves") echo "Casi viernes." ;;
    "viernes") echo "Último día laboral." ;;
    "sábado") echo "Fin de semana, a descansar." ;;
    "domingo") echo "Domingo familiar." ;;
    *) echo "Eso no es un día válido." ;;
esac

exit 0

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ ./practica_2.sh
Ingrese un día de la semana:
lunes
Es el primer día laboral.
```

Crear un script interactivo en Bash Script que realice operaciones matemáticas básicas y que considere el cero como variable de entrada. Definir la lógica de control a través de un pseudocódigo o diagrama de bloques. Presente en consola el resultado.

```
#!/bin/bash
# {Comment}
```

```
exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ nano practica_3.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ ./practica_3.sh
Ingrese el primer número:
56
Ingrese el segundo número:
85
Seleccione la operación (+, -, *, /):
+
Resultado: 4760

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ cat practica_3.sh
#!/bin/bash
# Script de operaciones matemáticas básicas con validación de división por cero

echo "Ingrese el primer número:"
read num1

echo "Ingrese el segundo número:"
read num2

echo "Seleccione la operación (+, -, *, /):"
read op

case $op in
    "+") echo "Resultado: $((num1 + num2))" ;;
    "-") echo "Resultado: $((num1 - num2))" ;;
    "*") echo "Resultado: $((num1 * num2))" ;;
    "/")
        if [ "$num2" -eq 0 ]; then
            echo "Error: no se puede dividir para cero."
        else
            echo "Resultado: $((num1 / num2))"
        fi
        ;;
    *) echo "Operación no válida." ;;
esac

exit 0
```

Crear un script interactivo en Bash Script que identifique si un fichero tiene contenido, y que determine el número de líneas del fichero. Definir la lógica de control a través de un el pseudocódigo o diagrama de bloques. Presente en consola el resultado.

```
#!/bin/bash  
# {Comment}
```

```
exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]  
$ nano practica_4.sh  
  
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]  
$ ./practica_4.sh  
Ingrese el nombre del archivo:  
Stephano  
El archivo no existe.  
  
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]  
$ ./practica_4.sh  
Ingrese el nombre del archivo:  
preactica_10.sh  
El archivo no existe.  
  
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]  
$ cat practica_4.sh  
#!/bin/bash  
# Verifica si un fichero tiene contenido y cuenta las líneas  
  
echo "Ingrese el nombre del archivo:"  
read archivo  
  
if [ -f "$archivo" ]; then  
    lineas=$(wc -l < "$archivo")  
    if [ "$lineas" -gt 0 ]; then  
        echo "$archivo tiene $lineas líneas."  
    else  
        echo "$archivo está vacío."  
    fi  
else  
    echo "El archivo no existe."  
fi  
  
exit 0
```

Crear un script interactivo en Bash Script que identifique si el usuario por defecto tiene acceso al shell. Definir la lógica de control a través de un el pseudocódigo o diagrama de bloques. Presente en consola el resultado.

```
#!/bin/bash  
# {Comment}
```

```
exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ nano practica_5.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ cat practica_5.sh
#!/bin/bash
# Verifica si el usuario ingresado tiene acceso al shell

echo "Ingrese el nombre del usuario:"
read usuario

# Busca la línea del usuario en /etc/passwd y extrae el shell
shell=$(grep "^$usuario:" /etc/passwd | cut -d: -f7)

if [ -z "$shell" ]; then
    echo "El usuario no existe."
elif [[ "$shell" == */false || "$shell" == */nologin ]]; then
    echo "El usuario $usuario NO tiene acceso al shell."
else
    echo "El usuario $usuario SÍ tiene acceso al shell ($shell)."
fi

exit 0

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ ./practica_5.sh
Ingrese el nombre del usuario:
dilan10
El usuario dilan10 SÍ tiene acceso al shell (/usr/bin/zsh).

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ ./practica_5.sh
Ingrese el nombre del usuario:
yo
El usuario no existe.
```

Crear un script interactivo en Bash Script que implemente un semáforo de tránsito. Definir la lógica de control a través de un pseudocódigo o diagrama de bloques. Presente en consola el resultado.

```
#!/bin/bash
# {Comment}
```

```
exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ nano practica_6.sh

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ cat practica_6.sh
#!/bin/bash
# Simula un semáforo de tránsito

echo "Ingrese el color del semáforo (rojo, amarillo, verde):"
read color

case $color in
    "rojo") echo "ALTO. Deténgase." ;;
    "amarillo") echo "PRECAUCIÓN. Reduzca la velocidad." ;;
    "verde") echo "ADLNT. Puede continuar." ;;
    *) echo "Color no válido." ;;
esac

exit 0

(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]
$ ./practica_6.sh
Ingrese el color del semáforo (rojo, amarillo, verde):
verde
ADLNT. Puede continuar.
```

Crear un script interactivo en Bash Script que implemente un semáforo de calorías según rangos específicos. Definir la lógica de control a través de un pseudocódigo o diagrama de bloques. Presente en consola el resultado.

```
#!/bin/bash  
# {Comment}
```

```
exit 0
```

```
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]  
$ nano practica_7.sh  
  
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]  
$ ./practica_7.sh  
Ingrese la cantidad de calorías:  
.500  
Amrillo: Caloras moderadas  
  
(dilan10@Kali-linux)-[~/Documentos/Semana7/Practica]  
$ cat practica_7.sh  
#!/bin/bash  
# Semáforo de calorías según rangos específicos  
  
echo "Ingrese la cantidad de calorías:"  
read calorías  
  
if [ "$calorías" -lt 200 ]; then  
    echo "Verde: Bajo en calorías 🟢"  
elif [ "$calorías" -le 500 ]; then  
    echo "Amrillo: Caloras moderadas "  
else  
    echo "Rjo: Alto en calorías "  
fi  
  
exit 0
```

Crear un script interactivo en Bash Script que convierta las notas musicales entre el estándar americano y el tradicional. Definir la lógica de control a través de un pseudocódigo o diagrama de bloques. Presente en consola el resultado.

```
(kali㉿kali)-[~]
$ cat S7.sh
#!/bin/bash

echo "Conversor de Notas Musicales"
echo "1. Americano → Tradicional"
echo "2. Tradicional → Americano"
read -p "Elige una opción (1 o 2): " opcion

if [ "$opcion" = "1" ]; then
    read -p "Ingresa una nota americana (A-G): " nota
    case "$nota" in
        Clc) echo "Do" ;;
        Dld) echo "Re" ;;
        Ele) echo "Mi" ;;
        Flf) echo "Fa" ;;
        Glg) echo "Sol" ;;
        Ala) echo "La" ;;
        Blb) echo "Si" ;;
        *) echo "Nota no valida" ;;
    esac
elif [ "$opcion" = "2" ]; then
    read -p "Ingresa una nota tradicional (Do, Re ... ): " nota
    case "$nota" in
        Dolldo) echo "C" ;;
        Relre) echo "D" ;;
        Milmi) echo "E" ;;
        Falfa) echo "F" ;;
        Sollsol) echo "G" ;;
        Lalla) echo "A" ;;
        Silsi) echo "B" ;;
        *) echo "Nota invalida" ;;
    esac
else
    echo "Opcion invalida"
fi
```

```
(kali㉿kali)-[~]
$ sudo bash S7.sh
Conversor de Notas Musicales
1. Americano → Tradicional
2. Tradicional → Americano
Elige una opción (1 o 2): 1
Ingresa una nota americana (A-G): a
La
```

```
(kali㉿kali)-[~]
$
```

```
#!/bin/bash
# {Comment}
```

```
exit 0
```



Crear un script interactivo en Bash Script que convierta calificaciones entre el sistema numérico y el alfabético. Definir la lógica de control a través de un el pseudocódigo o diagrama de bloques. Presente en consola el resultado.

```

(kali㉿kali)-[~]
$ cat S7.sh
#!/bin/bash

echo "Convertor de Calificaciones"
echo "1. Numerico a Alfabetico"
echo "2. Alfabetico a Numerico"
read -p "Elige una opción (1 o 2): " opcion

if [ "$opcion" = "1" ]; then
    read -p "Ingresa una calificacion numerica (0-10): " nota
    if (( $(echo "$nota ≥ 9" | bc -l) )); then
        echo "Calificacion alfabetica: A"
    elif (( $(echo "$nota ≥ 8" | bc -l) )); then
        echo "Calificacion alfabetica: B"
    elif (( $(echo "$nota ≥ 7" | bc -l) )); then
        echo "Calificacion alfabetica: C"
    elif (( $(echo "$nota ≥ 6" | bc -l) )); then
        echo "Calificacion alfabetica: D"
    else
        echo "Calificacion alfabetica: F"
    fi
fi

elif [ "$opcion" = "2" ]; then
    read -p "Ingresa una calificacion alfabetica (A-F): " letra
    case "$letra" in
        A|a) echo "Equivale a 9 - 10" ;;
        B|b) echo "Equivale a 8 - 8.9" ;;
        C|c) echo "Equivale a 7 - 7.9" ;;
        D|d) echo "Equivale a 6 - 6.9" ;;
        F|f) echo "Equivale a menos de 6" ;;
        *) echo "Letra no valida" ;;
    esac
else
    echo "Opcion invalida"
fi

(kali㉿kali)-[~]
$ sudo bash S7.sh
Convertor de Calificaciones
1. Numerico a Alfabetico
2. Alfabetico a Numerico
Elige una opción (1 o 2): 1
Ingresa una calificacion numerica (0-10): 5
Calificacion alfabetica: F

(kali㉿kali)-[~]
$
  
```

#!/bin/bash

```
# {Comment}
```

```
exit 0
```

Crear un script interactivo en Bash Script que identifique si un estudiante está aprobado, en supletorio o reprobado. Definir la lógica de control a través de un el pseudocódigo o diagrama de bloques. Considerar el estándar de la UIDE. Presente en consola el resultado.

```
(kali@kali)-[~]
$ cat S7.sh
#!/bin/bash

read -p "Ingresa la calificacion del estudiante (0 - 10): " nota

if (( $(echo "$nota ≥ 7.00" | bc -l) )); then
    echo "El estudiante esta APROBADO"
elif (( $(echo "$nota ≥ 5.00" | bc -l) )); then
    echo "El estudiante esta en SUPLETORIO"
else
    echo "El estudiante esta REPROBADO"
fi

(kali@kali)-[~]
$ sudo bash S7.sh
Ingresa la calificacion del estudiante (0 - 10): 7
El estudiante esta APROBADO

(kali@kali)-[~]
$
```

```
#!/bin/bash
```

```
# {Comment}
```

```
exit 0
```

Crear un script interactivo en Bash Script que calcule la hipotenusa. Definir la lógica de control a través de un el pseudocódigo o diagrama de bloques. Presente en consola el resultado.



```
(kali㉿kali)-[~]
$ cat S7.sh
#!/bin/bash

echo "Calculo de la Hipotenusa (Pitagoras)"
read -p "Ingrese el cateto A: " a
read -p "Ingrese el cateto B: " b

hipotenusa=$(echo "scale=2; sqrt($a^2 + $b^2)" | bc -l)

echo "La hipotenusa es: $hipotenusa"

(kali㉿kali)-[~]
$ sudo bash S7.sh
Calculo de la Hipotenusa (Pitagoras)
Ingrese el cateto A: 3
Ingrese el cateto B: 4
La hipotenusa es: 5.00

(kali㉿kali)-[~]
$
```

```
#!/bin/bash
# {Comment}
```

```
exit 0
```

#### Conclusiones:

- Comprensión del flujo de control en Bash:  
Se logró entender que los condicionales (if, elif, else, case) permiten ejecutar comandos de forma controlada dependiendo de las condiciones. Esto aporta dinamismo e inteligencia al script.
- Desarrollo de scripts interactivos:  
Los ejercicios permitieron practicar scripts que interactúan con el usuario mediante entrada por parámetros o entrada directa (read), generando respuestas personalizadas según el flujo lógico.

#### Recomendaciones:

- Usar buena indentación y comentarios:  
Siempre comentar el propósito del script y usar indentación clara. Esto facilita la lectura y mantenimiento del código, especialmente en condicionales anidados.
- Validar entradas desde el inicio:  
Antes de procesar cualquier archivo o dato, validar que el usuario ha ingresado los parámetros correctos. Así se evitan errores inesperados o resultados incorrectos.
- Practicar más casos reales:  
Se recomienda seguir desarrollando scripts basados en situaciones reales de administración de sistemas, por ejemplo: automatizar backups, verificar servicios, monitorear logs, etc.

## Bibliografía

<https://www.kali.org/get-kali/#kali-installer-images>

GmbH, I. (17 de Enero de 2007). VirtualBox. *Obtenido de VirtualBox*:  
<https://www.virtualbox.org/wiki/Downloads>