

Esta serie de ejercicios te llevará desde la lógica más básica de comparación hasta la creación de un script profesional con estructuras de control complejas. El comando `test` es el "cerebro" detrás de las decisiones en cualquier script de Bash.

Lógica de comparación con el comando `test` y `[]`

Ejercicio: Comparación de cadenas con el formato `test`

- Objetivo: Entender cómo `test` evalúa verdades y mentiras (0 es verdadero en Bash).
- Paso a paso:
 1. Ejecuta: `test "linux" = "linux" ; echo $?`
 2. Ejecuta: `test "linux" = "windows" ; echo $?`
- Explicación: El comando `test` no devuelve texto, sino un código de salida. `echo $?` nos muestra ese código: 0 significa que la comparación es cierta, 1 que es falsa. 🧠

Ejercicio: Comprobación de archivos con el formato `[]`

- Objetivo: Usar la sintaxis moderna de corchetes para verificar la existencia de archivos.
- Paso a paso:
 1. Ejecuta: `[-f /etc/passwd] ; echo $?`
 2. Ejecuta: `[-d /etc] ; echo $?`
- Explicación: Los corchetes `[]` son un alias del comando `test`. La opción `-f` (file) comprueba si es un archivo regular y `-d` (directory) si es una carpeta. Importante: Siempre debe haber espacios después de `[` y antes de `]`. 📁

Ejercicio: Comparación numérica y estructura `if-then`

- Objetivo: Crear un pequeño script que tome una decisión basada en un número.
- Paso a paso:
 1. Crea un archivo `edad.sh` con:

```
edad=20
if [ $edad -ge 18 ]; then
    echo "Eres mayor de edad"
fi
```
 2. Ejecuta: `bash edad.sh`
- Explicación: `-ge` significa "Greater or Equal" (`\ge`). La estructura `if` evalúa el resultado de `test` encerrado en los corchetes. 🚫

Ejercicio: Estructura Completa `if-then-else`

- Objetivo: Gestionar dos caminos posibles en el flujo del programa.
- Paso a paso:

1. Crea `servidor.sh`:

```
servicio="ssh"
if systemctl is-active --quiet $servicio; then
    echo "El servicio $servicio está corriendo."
else
    echo "Alerta: $servicio está detenido."
fi
```

- Explicación: Aquí no usamos `test` explícitamente porque `systemctl` ya devuelve un código de salida (0 o 1). El `else` captura cualquier caso en el que el `if` falle. 🚨

Ejercicio: Iteración con el bucle `for`

- Objetivo: Repetir una acción sobre una lista de elementos definida.
- Paso a paso:

1. Ejecuta en la terminal:

```
Bash
for color in rojo verde azul; do
    echo "El color es: $color"
done
```

- Explicación: El bucle `for` asigna a la variable `color` cada uno de los valores de la lista de forma secuencial hasta terminarla. ↺

Ejercicio: Control de flujo con el bucle `while`

- Objetivo: Repetir una acción mientras se cumpla una condición numérica.
- Paso a paso:

1. Crea `cuenta.sh`:

```
contador=1
while [ $contador -le 5 ]; do
    echo "Número: $contador"
    contador=$((contador + 1))
done
```

- Explicación: El bucle `while` sigue ejecutándose mientras `test` (vía `-le`, Less or Equal) sea cierto. La línea `$((...))` se usa para realizar aritmética básica.

1	2
3	4

Ejercicio: Selección múltiple con `case`

- Objetivo: Evitar múltiples `if` anidados cuando hay muchas opciones.
- Paso a paso:

1. Crea `fruta.sh`:

```
fruta="manzana"
case $fruta in
    "manzana") echo "Es roja" ;;
    "platano") echo "Es amarillo" ;;
    *) echo "No conozco esa fruta" ;;
```

esac

- Explicación: **case** compara la variable con varios patrones. El símbolo *****) actúa como un "por defecto" si nada coincide. 🍏

El Script Maestro: Menú de Administración Profesional

Este ejercicio integra todo lo aprendido: bucle infinito, limpieza de pantalla, menú visual con Here Document y selección con case.

Ejercicio: Script de Menú Infinito

- Objetivo: Crear una interfaz de consola funcional para un administrador.
- Paso a paso:
 1. Crea un archivo llamado `menu.sh`.
 2. Copia y pega el siguiente código:

```
#!/bin/bash

# Bucle infinito
while true; do
    clear # Limpia la pantalla al inicio de cada ciclo

    # Menú creado con Here Document
    cat << EOF
=====
    MENU DE ADMINISTRACION LINUX
=====
1. Mostrar espacio en disco (df -h)
2. Ver memoria RAM (free -m)
3. Ver usuarios conectados (who)
4. Salir
=====
EOF

    read -p "Elige una opción [1-4]: " opcion

    case $opcion in
        1)
            df -h
            read -p "Pulsa Enter para continuar..."
            ;;
        2)
            free -m
            read -p "Pulsa Enter para continuar..."
            ;;
        3)
            who
            read -p "Pulsa Enter para continuar..."
            ;;
        4)
            echo "Saliendo del sistema..."
            exit 0
            ;;
        *)
            echo "Opción no válida."
```

```
        sleep 2
        ;;
    esac
done
```

3. Dale permisos: `chmod +x menu.sh`

4. Ejecútalo: `./menu.sh`

- Explicación: `* while true`: Crea un bucle que nunca termina a menos que usemos `exit`.
 - `clear`: Mantiene la interfaz limpia.
 - `cat << EOF`: Imprime el menú de forma elegante sin usar múltiples comandos `echo`.
 - `read -p`: Captura la entrada del usuario.
 - `case`: Ejecuta la acción elegida. Se añade un segundo `read` dentro de cada opción para que el resultado no desaparezca instantáneamente por el `clear` del siguiente ciclo. 🚀