Control de flujo

Introducción

Cuando hablamos de Control de Flujo, nos referimos puntualmente a las sentencias condicionales y los bucles. Estas son herramientas esenciales que nos permiten crear scripts dinámicos y poderosos. Las sentencias if nos dan la capacidad de tomar decisiones lógicas dentro de nuestros scripts, ejecutando diferentes bloques de código según se cumplan o no ciertas condiciones. Por otro lado, los bucles, como for, while y until, nos permiten automatizar tareas repetitivas, iterando sobre conjuntos de valores o ejecutando comandos mientras se cumplan o no ciertas condiciones. En esta guía, exploraremos en detalle estos conceptos fundamentales, desde la sintaxis básica de las sentencias if hasta el uso de operadores de comparación y la implementación de bucles de diferentes tipos.

Sentencias Condicionales if en Bash

Las sentencias condicionales if son fundamentales en la programación en Bash, ya que permiten controlar el flujo de ejecución de un script. Estas sentencias evalúan una condición y, basándose en el resultado (verdadero o falso), ejecutan diferentes comandos.

Sintaxis básica:

```
if [[ CONDICIÓN ]]; then
  COMANDO 1 si se cumple la condición
fi
```

- **if**: Palabra clave que indica el inicio de la sentencia condicional.
- **[[CONDICIÓN]]**: La condición a evaluar. Se utilizan dobles corchetes para realizar la evaluación.
- **then**: Palabra clave que indica el inicio del bloque de comandos a ejecutar si la condición es verdadera.
- **COMANDO 1 si se cumple la condición**: Uno o más comandos que se ejecutarán si la condición es verdadera.
- **fi**: Palabra clave que indica el final de la sentencia if.

Ejemplo:

```
if [[ $edad -ge 18 ]]; then
  echo "Eres mayor de edad"
fi
```

Clausula else:

Para especificar acciones a realizar cuando la condición no se cumple, se utiliza la cláusula else:

```
if [[ CONDICIÓN ]]; then
  COMANDO 1 si se cumple la condición
else
  COMANDO 2 si no se cumple la condición
fi
```

Ejemplo:

```
if [[ $hora -lt 12 ]]; then
  echo "Buenos días"
else
  echo "Buenas tardes"
fi
```

Clausula elif:
Cuando se necesitan evaluar múltiples condiciones, se puede utilizar la cláusula elif (abreviatura de "else if"):

```
if [[ CONDICIÓN 1 ]]; then
  COMANDO 1 si se cumple la condición 1
elif [[ CONDICIÓN 2 ]]; then
  COMANDO 2 si se cumple la condición 2
else
  COMANDO 3 si no se cumple ninguna condición anterior
fi
```

Ejemplo:

```
if [[ $nota -ge 9 ]]; then
  echo "Sobresaliente"
elif [[ $nota -ge 7 ]]; then
  echo "Notable"
else
  echo "Necesitas mejorar"
fi
```

Operadores de comparación:

Se utilizan para comparar valores en las condiciones.

- Números:
  - -eq: igual
  - -ne: no igual
  - -gt: mayor que
  - -lt: menor que
  - -ge: mayor o igual que
  - -le: menor o igual que

- Cadenas:
    - ==: igual
    - !=: no igual
    - <: menor que (orden lexicográfico)
    - >: mayor que (orden lexicográfico)
    - -n: no vacía
    - -z: vacía
- Archivos:
    - -e: existe
    - -f: es un archivo regular
    - -d: es un directorio
    - -r: tiene permiso de lectura
    - -w: tiene permiso de escritura
    - -x: tiene permiso de ejecución

Bucles (for, while, until):

Los bucles permiten ejecutar un bloque de código repetidamente.

for: itera (da vueltas) sobre una lista de valores.

```
for variable in lista_de_valores; do
   # Comandos a ejecutar en cada iteración
done
```

Ejemplo:

```
#!/bin/bash

for i in 1 2 3 4 5; do
   echo "Número: $i"
done
```

while: se ejecuta mientras una condición sea verdadera.

```
while [ condición ]; do
   # Comandos a ejecutar mientras la condición sea verdadera
```

done

Ejemplo:

```
#!/bin/bash
contador=1
while [ $contador -le 5 ]; do
   echo "Contador: $contador"
   contador=$((contador + 1))
done
```

until: se ejecuta mientras una condición sea falsa.

```
until [ condición ]; do
   # Comandos a ejecutar mientras la condición sea falsa
done
```

Ejemplo:

```
#!/bin/bash
contador=1
until [ $contador -gt 5 ]; do
   echo "Contador: $contador"
   contador=$((contador + 1))
done
```

Rangos numéricos:

Se pueden utilizar para generar una secuencia de números.

Ejemplo:

```
for i in {1..10}; do
   echo "Número: $i"
done
```

Listas de valores:

Se pueden especificar explícitamente los valores sobre los que iterar.

Ejemplo:

```
for i in "uno" "dos" "tres"; do
   echo "Valor: $i"
done
```

Comandos como entrada:

La salida de un comando se puede utilizar como entrada para un bucle.

Ejemplo:

```
for archivo in $(ls); do
   echo "Archivo: $archivo"
done
```

Recomendaciones:

- Utilizar nombres descriptivos para las variables de los bucles.
- Utilizar indentación para mejorar la legibilidad del código dentro de los bucles.
- Asegurarse de que la condición del bucle while o until eventualmente se vuelva falsa para evitar bucles infinitos.