Control del Flujo en scripts Bash

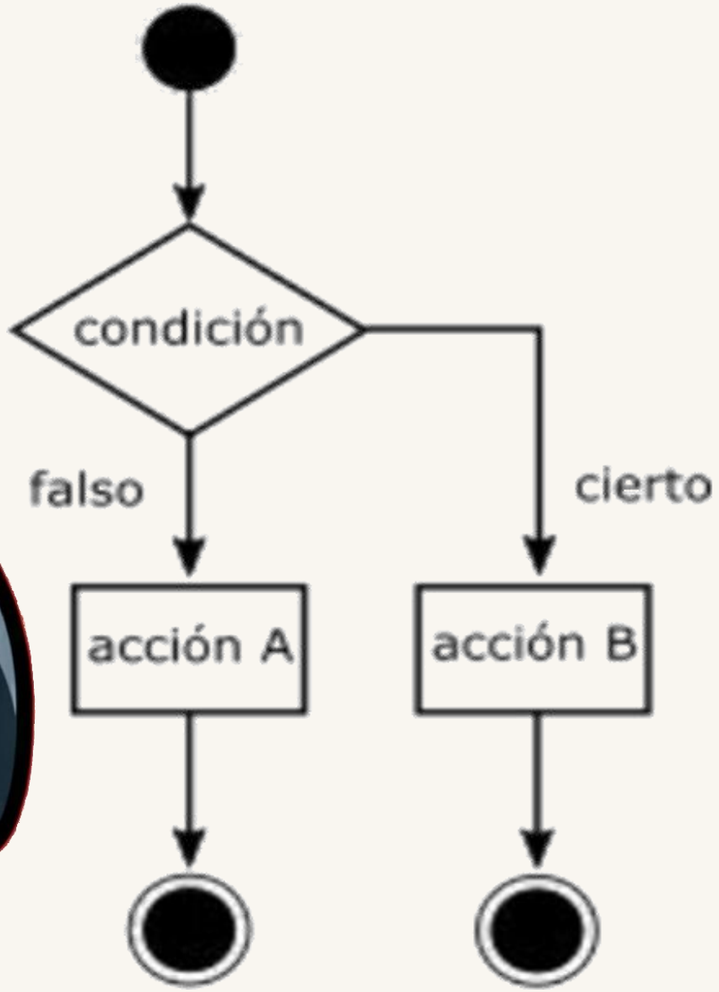
Condicional IF

Sintaxis básica

```
if [[ CONDICIÓN ]];
then
  acción B si se cumple la condición
fi
```

¿Y si no es así?

```
if [[ CONDICIÓN ]];
then
  acción B si se cumple la condición
else
  acción A si no se cumple la condición
fi
```



Condicional: IF encadenadas

Condición 1

Si la condición 1 es verdadera, se ejecuta el comando 1.

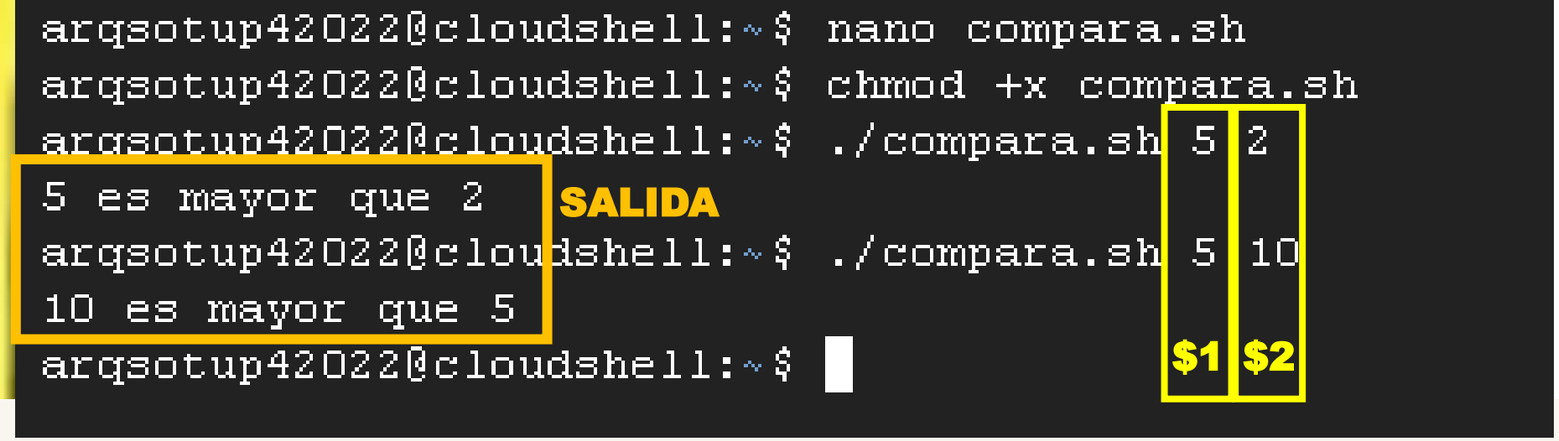
Condición 2

Si la condición 1 es falsa, se evalúa la condición 2. Si es verdadera, se ejecuta el comando 2.

Condición final

Si ninguna de las condiciones anteriores es verdadera, se ejecuta el comando 3.

```
if [[ CONDICIÓN 1 ]];
then
  COMANDO 1
elif [[ CONDICIÓN2 ]];
then
  COMANDO 2
else
  COMANDO 3
fi
```



Condicional IF: Números

Comparación

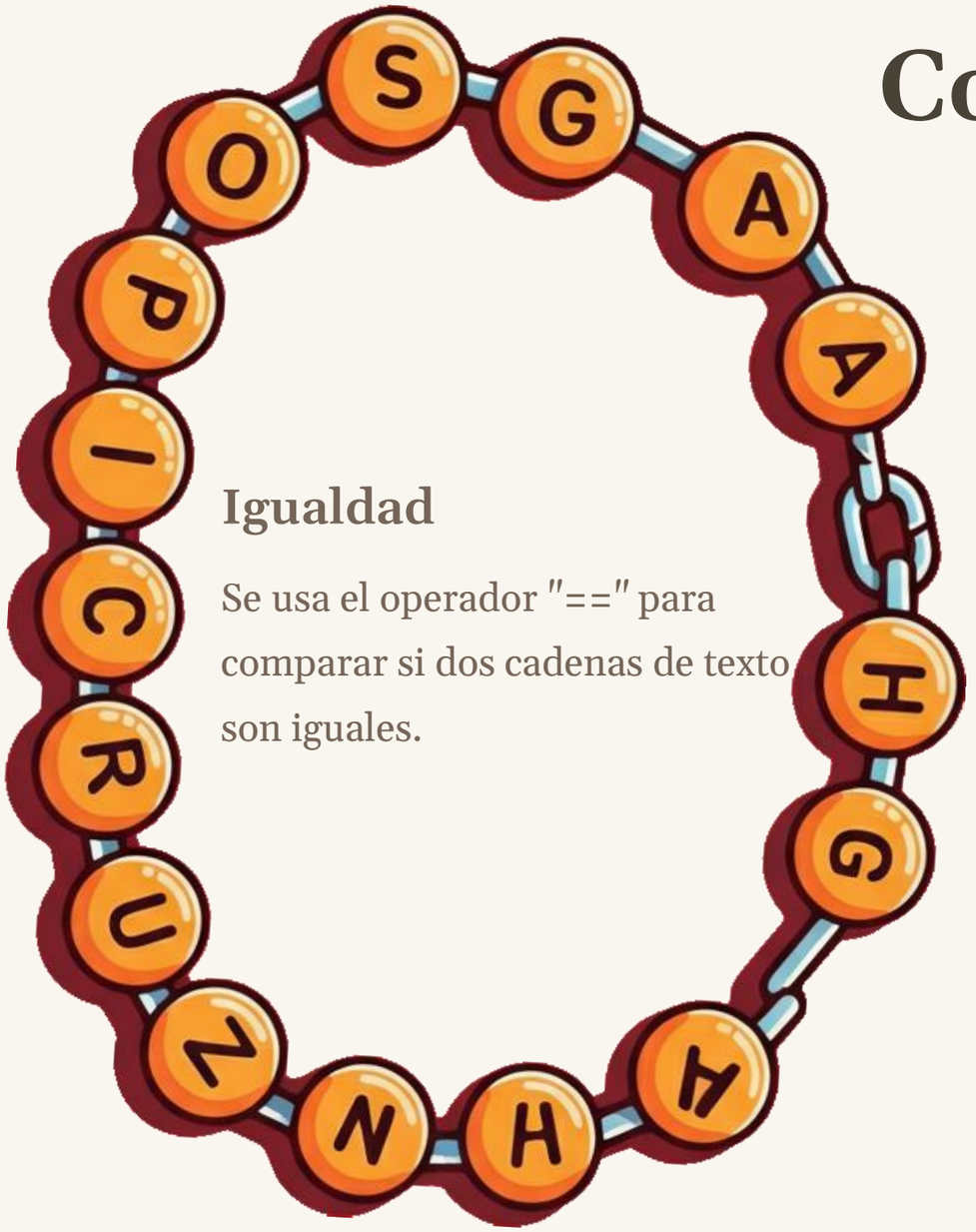
- lt -> menor que (<)
- gt -> mayor que (>)
- le -> menor o igual que (<=)
- ge -> mayor o igual que (>=)
- eq -> igual (==)
- ne -> no igual (!=)

Ejemplo

Modo de uso del script: **./compara.sh num1 num2**

```
#!/bin/bash
num1=$1 # la variable toma el 1er parámetro
num2=$2 # la variable toma el 2do parámetro
if [[ $num1 -gt $num2 ]];
then
  echo $num1 es mayor que $num2
else
  echo $num2 es mayor que $num1
fi
```

Condicional IF: Cadenas



Igualdad

Se usa el operador "==" para comparar si dos cadenas de texto son iguales.

Desigualdad

Se usa el operador "!=" para comparar si dos cadenas de texto son diferentes.

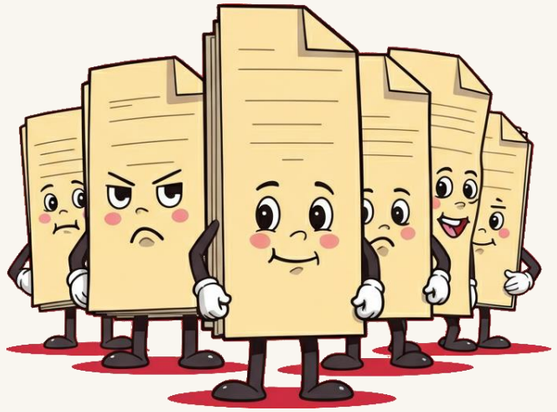
Orden alfabético

Se usan los operadores "<" y ">" para comparar el orden alfabético de las cadenas.

Condicional IF: Archivos

-d: Verifica si el archivo es un directorio.

```
if [[ -d "nombre_de_archivo" ]]; then
  echo "Es un directorio."
else
  echo "No es un directorio."
fi
```



-e: Verifica si el archivo o directorio existe.

```
if [[ -e "nombre_de_archivo" ]]; then
  echo "El archivo existe."
else
  echo "El archivo no existe."
fi
```

-f: Verifica que sea un archivo regular.

```
if [[ -f "nombre_de_archivo" ]]; then
  echo "Es un archivo regular."
else
  echo "No es un archivo regular."
fi
```

-s: Verifica si el tamaño del archivo es mayor que cero.

```
if [[ -s "nombre_de_archivo" ]]; then
  echo "El archivo no está vacío."
else
  echo "El archivo está vacío."
fi
```

El Bucle FOR

Sintaxis

El bucle FOR tiene una sintaxis básica:

```
for VARIABLE in LISTA_VALORES;
do
   COMANDO 1
   COMANDO 2
   ...
   COMANDO N
done
```

Ejemplos de Listas

- Rango numérico:

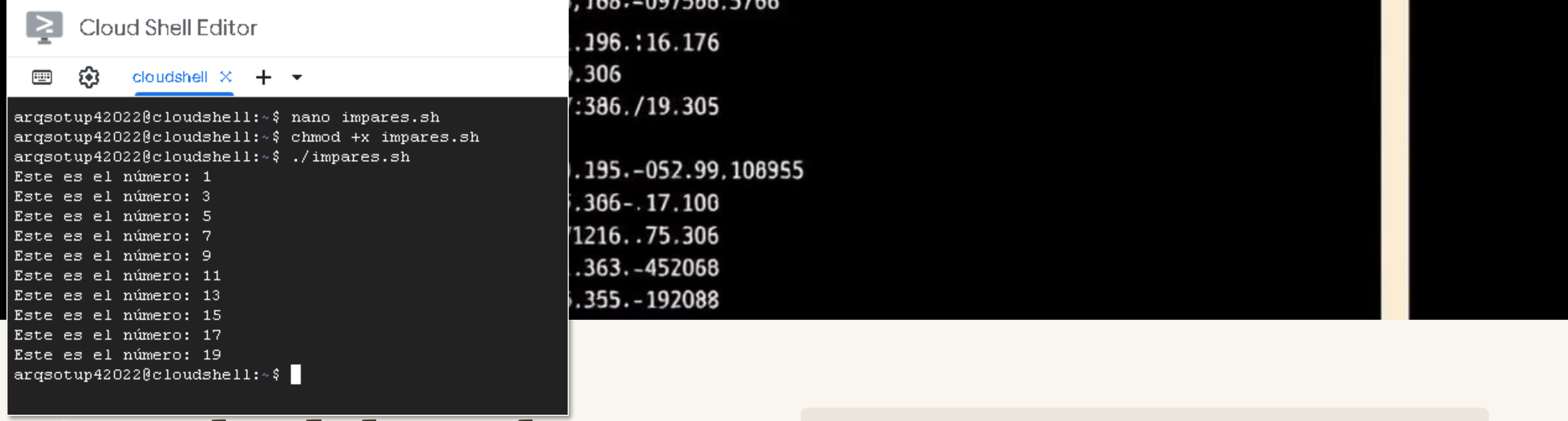
  for VARIABLE in {1..10};

- Serie de valores:

  for VARIABLE in file1 file2 file3;

- Salida de un comando:

  for VARIABLE in $(ls /bin | grep -E 'c.\[aeiou\]');



Ejemplo del Bucle FOR

```
#!/bin/bash

for numero in {1..20..2};
do
  echo Este es el número: $numero
done
```

numero Variable del bucle

in {1..20..2} Define secuencia (1, 3, 5,...19).

{1..20}: Rango del 1 al 20.

..2: Incremento de 2.

do...done Bloque de código.

echo Imprime texto.

$numero Valor actual de la variable

WHILE y UNTIL. Diferencias con FOR

Bucle	Condición	¿Cuándo se usa?
for	Implícita en la lista de elementos	Para iterar sobre una colección de elementos.
while	Se evalúa al inicio, se repite mientras sea verdadera	Cuando no se sabe cuántas veces se debe repetir el bucle.
until	Se evalúa al inicio, se repite hasta que la condición sea verdadera.	Similar a while, pero con la condición invertida.

Bucles WHILE y UNTIL

WHILE (Mientras que..)

Se ejecuta mientras la condición **sea cierta.**

```
#!/bin/bash
contador=0
termina=10
while [ $termina -ge $contador ]; do
  echo $contador
  let contador=$contador+1
done
```

UNTIL (Hasta que...)

Se ejecuta mientras la condición **sea falsa.**

```
#!/bin/bash
contador=0
termina=10
until [ $termina -ge $contador ]; do
  echo $contador
  let contador=$contador+1
done
```