Conditional Statements in Bash

Conditional statements in Bash allow you to make decisions in your script based on conditions. These are used to control the flow of the script and make it execute different commands depending on whether a condition is true or false.

Types of Conditional Statements in Bash

1. if Statement
2. if-else Statement
3. if-elif-else Statement
4. case Statement

1. if Statement

The simplest conditional structure, used to execute a block of code if a condition is true.

```
if [ condition ]
then
  # code to execute if the condition is true
fi
```

Example:

```
#!/bin/bash
age=20
if [ $age -ge 18 ]
then
  echo "You are an adult."
fi
```

2. if-else Statement

Used when you want to execute one block of code if the condition is true, and another block if the condition is false.

```
if [ condition ]
then
  # code to execute if the condition is true
else
  # code to execute if the condition is false
fi
```

Example:

```
#!/bin/bash
age=16
if [ $age -ge 18 ]
then
  echo "You are an adult."
else
  echo "You are a minor."
fi
```

3. if-elif-else Statement

Used when you have multiple conditions to check. The script checks the conditions in order and executes the corresponding block of code for the first condition that evaluates to true.

```
if [ condition1 ]
then
  # code to execute if condition1 is true
elif [ condition2 ]
then
  # code to execute if condition2 is true
else
  # code to execute if no conditions are true
fi
```

Example:

```
#!/bin/bash
day="Monday"
if [ "$day" == "Monday" ]
then
  echo "Start of the workweek!"
elif [ "$day" == "Friday" ]
then
  echo "End of the workweek!"
else
  echo "It's a normal day."
fi
```

4. case Statement

The case statement is an alternative to **if-elif-else** for matching a value against multiple options. It’s useful when you have many possible conditions.

```
case "$variable" in
  pattern1)
    # code to execute if variable matches pattern1
    ;;
  pattern2)
    # code to execute if variable matches pattern2
    ;;
  *)
    # default code to execute if no patterns match
    ;;
esac
```

Example:

```
#!/bin/bash
day="Friday"
case "$day" in
  "Monday")
    echo "Start of the workweek!"
    ;;
  "Friday")
    echo "End of the workweek!"
    ;;
  *)
    echo "It's a normal day."
    ;;
esac
```

Test Operators for Conditions

To test conditions, you can use comparison operators inside [ ] (the test command). Here are common operators:

Operator	Description	Example
`-eq`	Equal to	`[ $a -eq $b ]`
`-ne`	Not equal to	`[ $a -ne $b ]`
`-lt`	Less than	`[ $a -lt $b ]`
`-le`	Less than or equal to	`[ $a -le $b ]`
`-gt`	Greater than	`[ $a -gt $b ]`
`-ge`	Greater than or equal to	`[ $a -ge $b ]`
`-z`	String is empty	`[ -z "$string" ]`
`-n`	String is not empty	`[ -n "$string" ]`
`==`	String equality	`[ "$string1" == "$string2" ]`
`!=`	String inequality	`[ "$string1" != "$string2" ]`