



**NORTHERN  
UNIVERSITY**

Knowledge for Innovation and Change

**COURSE TITLE: OPERATING SYSTEM LAB WORK**

**COURSE CODE: CSE 3373**

**REPORT ON: IMPLEMENTATION OF  
CONDITIONAL STATEMENTS IN BASH**

*SUBMITTED TO,*  
**NIZIA NAHYAN**  
**LECTURER, NUB(CSE)**

*SUBMITTED BY,*  
**ZUBAER AHMED ZIHAD**  
**ID: 41230100877**  
**SECTION: 7A**

**DATE OF SUBMISSION: 13 April 2025**

# **Lab Report: Implementation of Conditional Statements in Bash with Relational Operators**

## **Introduction**

Conditional statements in Bash allow us to make decisions in shell scripts. They help us control the flow of a program by executing different commands based on certain conditions. In this lab, we will learn how to use `if-fi` and `if-else` statements along with relational operators to compare values.

## **Section 1: Conditional Statements and Relational Operators**

### **1. if-fi Statement**

- **Function:** Executes commands if a condition is true.

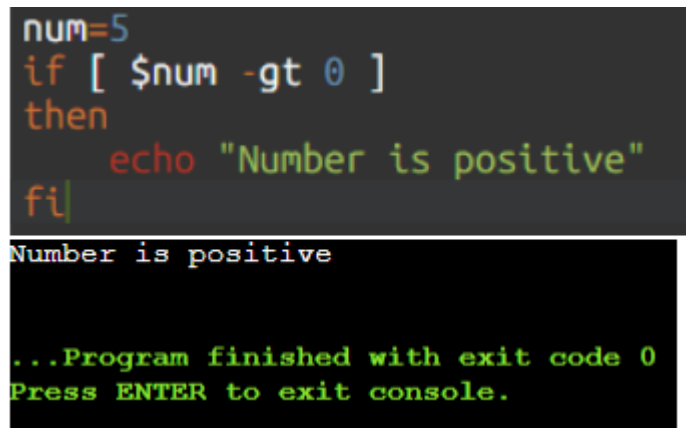
**Description:** This statement checks a condition. If the condition is true, it runs the commands inside the block. Syntax:

```
if [ condition ]  
then  
    command  
fi
```

Example:

```
if [ $a -eq $b ]  
then  
    echo "a is equal to b"
```

- fi



```
num=5  
if [ $num -gt 0 ]  
then  
    echo "Number is positive"  
fi  
Number is positive  
...Program finished with exit code 0  
Press ENTER to exit console.
```

### **2. if-else Statement**

- **Function:** Executes one set of commands if the condition is true and another if false.

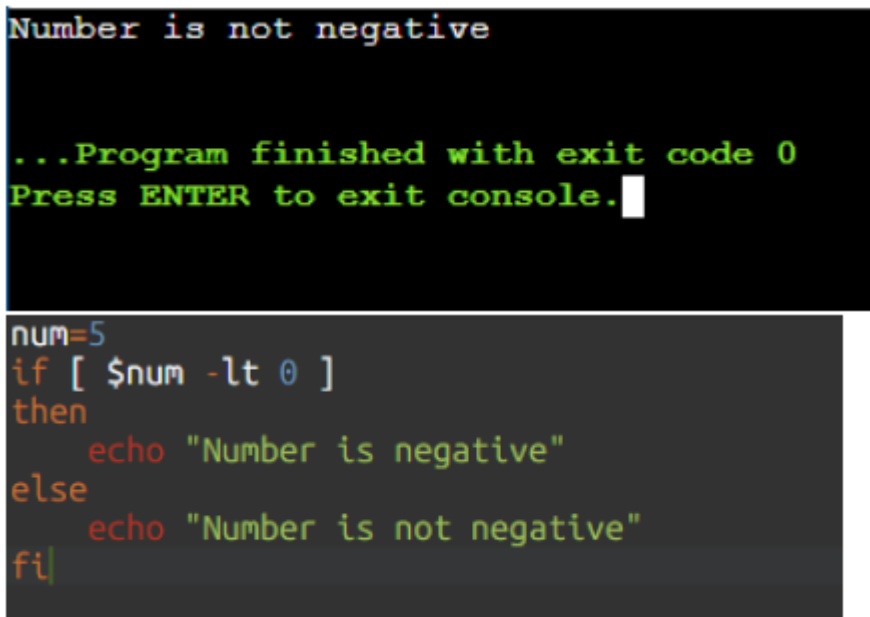
**Description:** This is an extension of the `if` statement with an `else` block. Syntax:

```
if [ condition ]
then
    command1
else
    command2
fi
```

Example:

```
if [ $a -gt $b ]
then
    echo "a is greater than b"
else
    echo "a is not greater than b"
```

- `fi`



```
Number is not negative

...Program finished with exit code 0
Press ENTER to exit console.

num=5
if [ $num -lt 0 ]
then
    echo "Number is negative"
else
    echo "Number is not negative"
fi
```

### 3. Relational Operators in Bash

Relational operators are used to compare two values inside the conditional statements. Here are the main operators:

#### **-eq (Equal To)**

- **Function:** Checks if two numbers are equal.

**Example:**

```
if [ $a -eq $b ]
then
```

echo "a is equal to b"

- fi

```
a=10
b=10
if [ $a -eq $b ]
then
    echo "a is equal to b"
else
    echo "a is not equal to b"
fi|

a is equal to b

...Program finished with exit code 0
Press ENTER to exit console.
```

### -ne (Not Equal To)

- **Function:** Checks if two numbers are not equal.

#### Example:

```
if [ $a -ne $b ]
then
    echo "a is not equal to b"
```

```
a=10
b=20
if [ $a -ne $b ]
then
    echo "a is not equal to b"
else
    echo "a is equal to b"
fi|

a is not equal to b

...Program finished with exit code 0
Press ENTER to exit console.
```

### -lt (Less Than)

- **Function:** Checks if one number is less than another.

#### Example:

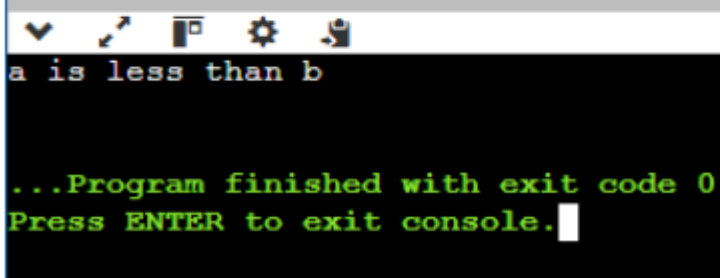
```
if [ $a -lt $b ]
```

then

echo "a is less than b"

- fi

```
a=5
b=10
if [ $a -lt $b ]
then
    echo "a is less than b"
else
    echo "a is not less than b"
fi
```



### -gt (Greater Than)

- **Function:** Checks if one number is greater than another.

**Example:**

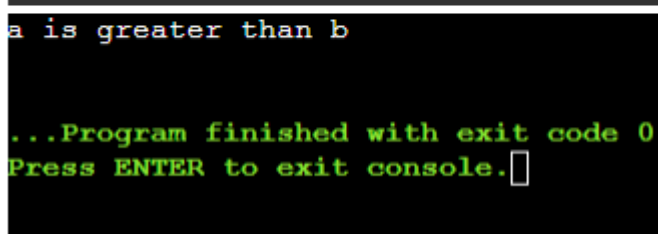
if [ \$a -gt \$b ]

then

echo "a is greater than b"

- fi

```
a=15
b=10
if [ $a -gt $b ]
then
    echo "a is greater than b"
else
    echo "a is not greater than b"
fi
```



### **-ge (Greater Than or Equal To)**

- **Function:** Checks if one number is greater than or equal to another.

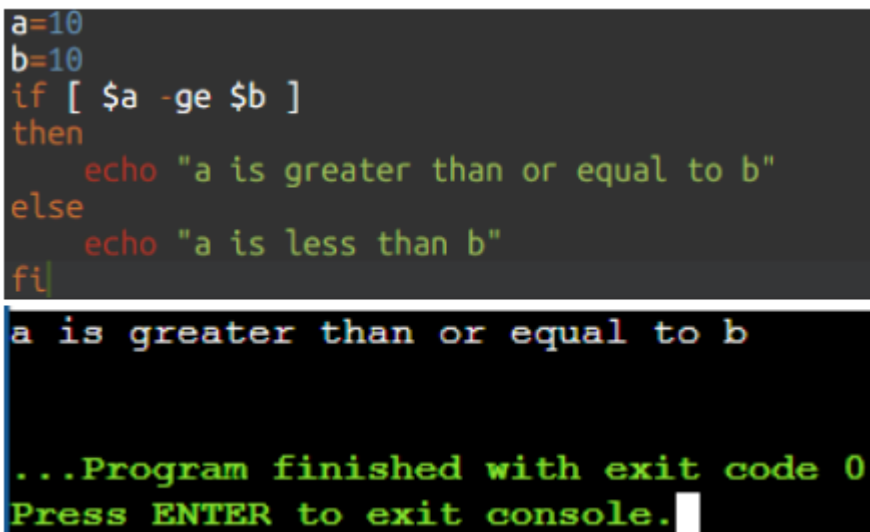
#### **Example:**

```
if [ $a -ge $b ]
```

```
then
```

```
    echo "a is greater than or equal to b"
```

- ```
fi
```



```
a=10
b=10
if [ $a -ge $b ]
then
    echo "a is greater than or equal to b"
else
    echo "a is less than b"
fi

a is greater than or equal to b

...Program finished with exit code 0
Press ENTER to exit console.
```

### **-le (Less Than or Equal To)**

- **Function:** Checks if one number is less than or equal to another.

#### **Example:**

```
if [ $a -le $b ]
```

```
then
```

```
    echo "a is less than or equal to b"
```

- ```
fi
```

```
a=5
b=10
if [ $a -le $b ]
then
    echo "a is less than or equal to b"
else
    echo "a is greater than b"
fi
```

```
a is less than or equal to b
```

```
...Program finished with exit code 0
Press ENTER to exit console.□
```

## Discussion

Conditional statements make Bash scripts smarter and more flexible. By using `if-fi` and `if-else` blocks, we can control what happens depending on the values of variables. Relational operators help compare numbers in these conditions. Understanding and using these tools properly is very important for writing useful Bash scripts.

## Conclusion

In this lab, we learned how to implement conditional statements in Bash using `if-fi`, `if-else`, and various relational operators. These commands help us make decisions in scripts and are a key part of automation and system scripting in Linux.