

Conditional Statements in Bash Programming

Learn about the different conditional statements available in Bash programming and how they can be used to control the flow of your scripts.

If Statement

The if statement allows you to execute a block of code if a specified condition is true.

```
if [ expression ]
then
statement
fi
```

If-Else Statement

The if-else statement provides an alternative block of code to be executed if the specified condition in the if part is not true.

```
if [ expression ]
then
  statement1
else
  statement2
fi
```



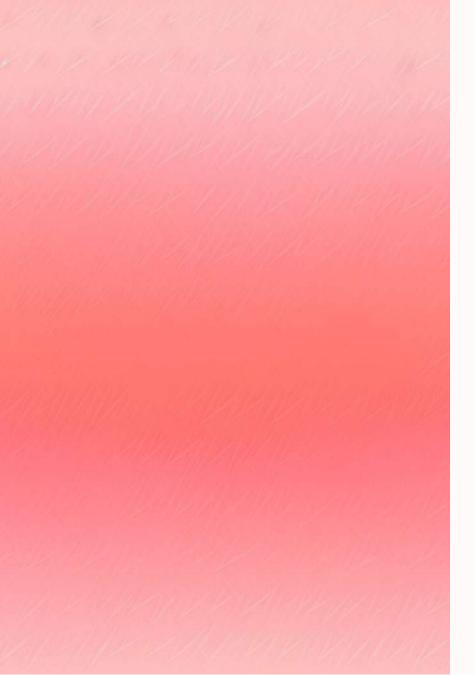
If..Elif..Else..Fi Statement

The if..elif..else..fi statement, also known as the "Else If ladder," allows you to test multiple conditions and execute different blocks of code based on the first condition that is true.

```
if [ expression1 ]
then
 statement1
 statement2
elif [expression2]
then
 statement3
 statement4
else
 statement5
```

If..Then..Else..If..Then..Fi..Fi Statement

The if..then..else..if..then..fi..fi statement, also known as nested if, allows you to have multiple levels of conditional statements within your script.



Switch Statement

The switch statement provides a way to perform different actions based on the value of a variable or an expression.

```
case in
Pattern 1) Statement 1;;
Pattern n) Statement n;;
esac
```

Checking If a File Exists

Use the -f option in the if statement to check if a file exists.

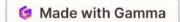
if [-f "File.txt"]; then

echo "File is exist"

else

echo "File is not exist"

f



Checking If a Directory Exists

Use the -d option in the if statement to check if a directory exists.

```
if [[ -d "GFG_dir" ]];
then
echo "Directory is exist"
else
echo "Directory is not exist"
fi
```

