Test Command

A test command is a command that is used to test the validity of a command. It checks whether the command/expression is true or false. It is used to check the type of file and the permissions related to a file. Test command returns 0 as a successful exit status if the command/expression is true, and returns 1 if the command/expression is false.

Syntax:

```
test [expression]
```

Example:

```
test "variable1' operator "variable2"
```

EXAMPLE

```
#!/bin/bash
# Example to check if two numbers are equal
# or not

# first number
a=20

# second number
b=20

# using test command to check if numbers
# are equal
if test "$a" -eq "$b"
then
   echo "a is equal to b"
else
   echo "a is not equal to b"
fi
```

SHELL DECISION STATEMENTS

The if...fi statement

The **if...fi** statement is the fundamental control statement that allows Shell to make decisions and execute statements conditionally.

Syntax

```
if [ expression ]
then
   Statement(s) to be executed if expression is true
fi
```

EXAMPLE

```
a=10
b=20

if [ $a == $b ]
then
   echo "a is equal to b"
fi

if [ $a != $b ]
then
   echo "a is not equal to b"
fi
```

The if...else...fi statement

The **if...else...fi** statement is the next form of control statement that allows Shell to execute statements in a controlled way and make the right choice.

Syntax

```
if [ expression ]
then
   Statement(s) to be executed if expression is true
else
   Statement(s) to be executed if expression is not true
fi
```

EXAMPLE

```
a=10
b=20

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

The if...elif...fi statement

The **if...elif...fi** statement is the one level advance form of control statement that allows Shell to make correct decision out of several conditions.

Syntax

```
if [ expression 1 ]
then
   Statement(s) to be executed if expression 1 is true
elif [ expression 2 ]
then
   Statement(s) to be executed if expression 2 is true
elif [ expression 3 ]
then
   Statement(s) to be executed if expression 3 is true
else
```

```
  Statement(s) to be executed if no expression is true
fi
```

EXAMPLE

```
a=10
b=20

if [ $a == $b ]
then
  echo "a is equal to b"
elif [ $a -gt $b ]
then
  echo "a is greater than b"
elif [ $a -lt $b ]
then
  echo "a is less than b"
else
  echo "None of the condition met"
fi
```

The case...esac Statement

Shell supports case...esac statement which handles exactly this situation, and it does so more efficiently than repeated if...elif statements.

Syntax

```
case word in
  pattern1)
    Statement(s) to be executed if pattern1 matches
    ;;
  pattern2)
    Statement(s) to be executed if pattern2 matches
    ;;
```

```
  pattern3)
    Statement(s) to be executed if pattern3 matches
    ;;
  *)
    Default condition to be executed
    ;;
esac
```

EXAMPLE

```
FRUIT="kiwi"
case "$FRUIT" in
  "apple") echo "Apple pie is quite tasty."
  ;;
  "banana") echo "I like banana nut bread."
  ;;
  "kiwi") echo "New Zealand is famous for kiwi."
  ;;
esac
```