Unix / Linux Shell - 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
```

The Shell *expression* is evaluated in the above syntax. If the resulting value is *true*, given *statement(s)* are executed. If the *expression* is *false*, then no statement will be executed.

Example

The above example can also be written using the *if...else* statement as follows −

Live Demo

```
#!/bin/sh
```

```
a=10

b=20


if [ $a == $b ]

then

   echo "a is equal to b"

else

   echo "a is not equal to b"

fi
```

Upon execution, you will receive the following result −

```
a is not equal to b
```

Relational Operators

Operator	Description	Example
-eq	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a -eq $b] is not true.
-ne	Checks if the value of two operands are equal or not; if values are not equal, then the condition becomes true.	[$a -ne $b] is true.
-gt	Checks if the value of left operand is greater than the value of right operand; if yes, then the condition becomes true.	[$a -gt $b] is not true.
-lt	Checks if the value of left operand is less than the value of right operand; if yes, then the condition becomes true.	[$a -lt $b] is true.
-ge	Checks if the value of left operand is greater than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -ge $b] is not true.

-le	Checks if the value of left operand is less than or equal to the value of right operand; if yes, then the condition becomes true.	[$a -le $b] is true.

It is very important to understand that all the conditional expressions should be placed inside square braces with spaces around them. For example, **[$a <= $b]** is correct whereas, **[$a <= $b]** is incorrect.

Example

Here is an example which uses all the relational operators −

Live Demo

```sh
#!/bin/sh


a=10

b=20


if [ $a -eq $b ]

then

   echo "$a -eq $b : a is equal to b"

else

   echo "$a -eq $b: a is not equal to b"
```

```bash
fi


if [ $a -ne $b ]
then
    echo "$a -ne $b: a is not equal to b"
else
    echo "$a -ne $b : a is equal to b"
fi


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


if [ $a -lt $b ]
then
    echo "$a -lt $b: a is less than b"
```

```
else
    echo "$a -lt $b: a is not less than b"
fi


if [ $a -ge $b ]
then
    echo "$a -ge $b: a is greater or  equal to b"
else
    echo "$a -ge $b: a is not greater or equal to b"
fi


if [ $a -le $b ]
then
    echo "$a -le $b: a is less or  equal to b"
else
    echo "$a -le $b: a is not less or equal to b"
fi
```

Boolean Operators

The following Boolean operators are supported by the Bourne Shell.

Assume variable **a** holds 10 and variable **b** holds 20 then −

Show Examples

Operator	Description	Example
!	This is logical negation. This inverts a true condition into false and vice versa.	[! false] is true.
-o	This is logical **OR**. If one of the operands is true, then the condition becomes true.	[$a -lt 20 -o $b -gt 100] is true.
-a	This is logical **AND**. If both the operands are true, then the condition becomes true otherwise false.	[$a -lt 20 -a $b -gt 100] is false.

```sh
#!/bin/sh


a=10

b=20


if [ $a != $b ]

then

    echo "$a != $b : a is not equal to b"

else

    echo "$a != $b: a is equal to b"

fi


if [ $a -lt 100 -a $b -gt 15 ]

then

    echo "$a -lt 100 -a $b -gt 15 : returns true"

else

    echo "$a -lt 100 -a $b -gt 15 : returns false"

fi
```

```
if [ $a -lt 100 -o $b -gt 100 ]

then

    echo "$a -lt 100 -o $b -gt 100 : returns true"

else

    echo "$a -lt 100 -o $b -gt 100 : returns false"

fi


if [ $a -lt 5 -o $b -gt 100 ]

then

    echo "$a -lt 100 -o $b -gt 100 : returns true"

else

    echo "$a -lt 100 -o $b -gt 100 : returns false"

fi
```

String Operators

The following string operators are supported by Bourne Shell.

Assume variable **a** holds "abc" and variable **b** holds "efg" then −

Show Examples

Operator	Description	Example
=	Checks if the value of two operands are equal or not; if yes, then the condition becomes true.	[$a = $b] is not true.
!=	Checks if the value of two operands are equal or not; if values are not equal then the condition becomes true.	[$a != $b] is true.
-z	Checks if the given string operand size is zero; if it is zero length, then it returns true.	[-z $a] is not true.
-n	Checks if the given string operand size is non-zero; if it is nonzero length, then it returns true.	[-n $a] is not false.

str	Checks if **str** is not the empty string; if it is empty, then it returns false.	[$a] is not false.

```sh
#!/bin/sh

a="abc"

b="efg"


if [ $a = $b ]

then

   echo "$a = $b : a is equal to b"

else

   echo "$a = $b: a is not equal to b"

fi


if [ $a != $b ]

then

   echo "$a != $b : a is not equal to b"

else
```

```bash
    echo "$a != $b: a is equal to b"
fi


if [ -z $a ]
then
    echo "-z $a : string length is zero"
else
    echo "-z $a : string length is not zero"
fi


if [ -n $a ]
then
    echo "-n $a : string length is not zero"
else
    echo "-n $a : string length is zero"
fi


if [ $a ]
then
```

```
   echo "$a : string is not empty"
else
   echo "$a : string is empty"
fi
```

File Test Operators

We have a few operators that can be used to test various properties associated with a Unix file.

Assume a variable **file** holds an existing file name "test" the size of which is 100 bytes and has **read**, **write** and **execute** permission on −

We have a few operators that can be used to test various properties associated with a Unix file.

Assume a variable **file** holds an existing file name "test" the size of which is 100 bytes and has **read**, **write** and **execute** permission on −

Operator	Description	Example
-b file	Checks if file is a block special file; if yes, then the condition becomes true.	[-b $file] is false.

-c file	Checks if file is a character special file; if yes, then the condition becomes true.	[-c $file] is false.
-d file	Checks if file is a directory; if yes, then the condition becomes true.	[-d $file] is not true.
-f file	Checks if file is an ordinary file as opposed to a directory or special file; if yes, then the condition becomes true.	[-f $file] is true.
-g file	Checks if file has its set group ID (SGID) bit set; if yes, then the condition becomes true.	[-g $file] is false.
-k file	Checks if file has its sticky bit set; if yes, then the condition becomes true.	[-k $file] is false.
-p file	Checks if file is a named pipe; if yes, then the condition becomes true.	[-p $file] is false.
-t file	Checks if file descriptor is open and associated with a terminal; if yes, then the condition becomes true.	[-t $file] is false.

-u file	Checks if file has its Set User ID (SUID) bit set; if yes, then the condition becomes true.	[-u $file] is false.
-r file	Checks if file is readable; if yes, then the condition becomes true.	[-r $file] is true.
-w file	Checks if file is writable; if yes, then the condition becomes true.	[-w $file] is true.
-x file	Checks if file is executable; if yes, then the condition becomes true.	[-x $file] is true.
-s file	Checks if file has size greater than 0; if yes, then condition becomes true.	[-s $file] is true.
-e file	Checks if file exists; is true even if file is a directory but exists.	[-e $file] is true.

Example

The following example uses all the **file test** operators −

Assume a variable file holds an existing file name **"/var/www/tutorialspoint/unix/test.sh"** the size of which is 100 bytes and has **read**, **write** and **execute** permission −

```sh
#!/bin/sh



file="/var/www/tutorialspoint/unix/test.sh"



if [ -r $file ]

then

   echo "File has read access"

els e

   echo "File does not have read access"

fi



if [ -w $file ]
```

```bash
then

    echo "File has write permission"

else

    echo "File does not have write permission"

fi


if [ -x $file ]

then

    echo "File has execute permission"

else

    echo "File does not have execute permission"

fi


if [ -f $file ]

then

    echo "File is an ordinary file"

else

    echo "This is sepcial file"

fi
```

```bash
if [ -d $file ]

then

    echo "File is a directory"

else

    echo "This is not a directory"

fi


if [ -s $file ]

then

    echo "File size is zero"

else

    echo "File size is not zero"

fi


if [ -e $file ]

then

    echo "File exists"

else
```

```
    echo "File does not exist"

fi
```