
ondia


Linux Plus for
AWS and
DevOps

Table of Contents

If Statements

A simple **if statement** essentially states, if a particular test is true, then perform a specified set of actions. If it's not true, don't take those acts.

```
if [[ <some test> ]]
then
  <commands>
fi
```

```
#!/bin/bash
read -p "Input a number" number

if [[ $number -gt 50 ]]
then
 echo "The number is big."
fi
```

Output:

```
$./if-statement.sh
Input a number: 55
The number is big.
```

Relational Operators

Operator	Description
-eq	equal
-ne	not equal
-gt	greater than
-lt	less than
-ge	greater than or equal
-le	less than or equal

```
#!/bin/bash
read -p "Input a number" number

if [[ $number -gt 50 ]]
then
 echo "The number is big."
fi
```

String Operators

Operator	Description
=	equal
!=	not equal
-z	Empty string
-n	Not empty string

```
#!/bin/bash

if [[ "a" = "a" ]]
then
 echo "They are same"
fi

if [[ "a" != "b" ]] #True
then
 echo "They are not same"
fi

if [[ -z "" ]]
then
 echo "It is empty"
fi

if [[ -n "text" ]]
then
 echo "It is not empty"
fi
```

File Test Operators

Operator	Description
-d file	directory
-e file	exists
-f file	ordinary file
-r file	readable
-s file	size is > 0 bytes
-w file	writable
-x file	executable

```
#!/bin/bash

if [[ -d folder ]]
then
 echo "folder is a directory"
fi

if [[ -f file ]]
then
 echo "file is an ordinary file"
fi

if [[ -w file ]]
then
 echo "file is a writable file"
fi

if [[ -s file ]]
then
 echo "file is > 0 bytes"
fi
```

If Else Statements

If Else Statements execute a block of code if a statement is true, or another block of code if it is false.

```
if [[ <some test> ]]
then
  <commands>
else
  <other commands>
fi
```

```
#!/bin/bash
read -p "Input a number: " number

if [[ $number -ge 10 ]]
then
 echo "The number is bigger than or
equal to 10."
else
 echo "The number is smaller than
10"
fi
```

Output:

```
$./ifelse-statement.sh
Input a number: 27
The number is bigger than or
equal to 10.
$
$./ifelse-statement.sh
Input a number: 5
The number is smaller than 10
```

If Else Statements

```
if [[ <some test> ]]
then
  <commands>
elif [[ <some test> ]]
then
  <different commands>
else
  <other commands>
fi
```

```
#!/bin/bash
read -p "Input a number: " number

if [[ $number -eq 10 ]]
then
 echo "The number is equal to
10."
elif [[ $number -gt 10 ]]
then
 echo "The number is bigger than
10"
else
 echo "The number is smaller than
10"
fi
```

Output:

```
$./elif-statement.sh
Input a number: 15
The number is bigger than 10
$
$./elif-statement.sh
Input a number: 5
The number is smaller than
10
$
$./elif-statement.sh
Input a number: 10
The number is equal to 10
```

Nested If Statements

```
#!/bin/bash

read -p "Input a number: " number

if [[ $number -gt 10 ]]
then
 echo "Number is bigger than 10"

 if (( $number % 2 == 1 ))
 then
   echo "And is an odd number."
 else
   echo "And is an even number"
 fi
else
 echo "It is not bigger than 10"
fi
```

Output:

```
$./nested-if-statement.sh
Input a number: 40
Number is bigger than 10
And is an even number
$
$./nested-if-statement.sh
Input a number: 27
Number is bigger than 10
And is an odd number.
$
$./nested-if-statement.sh
Input a number: 5
It is not bigger than 10
```

Exercise

1. Ask user to enter his/her **name**.
2. Ask user to enter his/her **age**.
3. Ask user **average life expectancy (ale)**.
4. Print user name with one of these messages regarding his/her **age**:
   a. age<18 :
      “Student”
      “At least **X** years to become a worker.” # (X = 18 - age)
   b. 18<=age<65 :
      “Worker”
      “X years to retire.” # (X = 65 - age)
   c. age>=65 :
      if age less than **ale**:
         “Retired”
         “X years to die.” # (X = ale - age)
      else:
            # beep sound # echo -ne '\007'
         “!!! Already died !!!”
            # wait 1 sec.
         “!!! Already died !!!”
            # wait 2 secs.
         “!!! Already died !!!”


Kahoot!

THANKS!

Any questions?