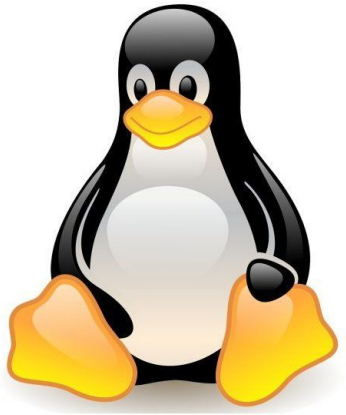
Linux Plus

for

AWS and DevOps

Session - 6



CLARUSWAY

WAY TO REINVENT YOURSELF

Table of Contents

If Statements

Unix Shell supports conditional statements that are used to perform different actions on the basis of different conditions.

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

The following string operators are supported by BASH Shell.

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" ]]
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

There are a few operators that can be used to test various properties associated with a Linux file.

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 [[ -r file ]]
then
  echo "file is a readable file"
fi

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

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

if [[ -x $0 ]]
then
  echo "$0 is an executable file "
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 Elif 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

If statements can be nested. Let's see the nested structure on the following example.

```
#!/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 it's an odd number."
 else
   echo "And it's 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
```

Boolean Operations

The Boolean operators below are supported by the Bourne Shell.

Operator	Description
!	NEGATION
&&	AND
||	OR

- `!` inverts a true condition into false and vice versa.
- `&&` is logical AND. If both the operands are true, then the condition becomes true otherwise false.
- `||` is logical OR. If one of the operands is true, then the condition becomes true.

Boolean Operations

```
#!/bin/bash

read -p "Input your name: " name
read -sp "Input your password: "
password

if [[ $name = $(whoami) ]] && [[
$password = Aa1234 ]]
then
  echo -e "\nWelcome $(whoami)"
else
  echo -e "\nIt's wrong account"
fi
```

Output:

```
Input your name: ec2-user
Input your password:
Welcome ec2-user

+++++++++
Input your name: root
Input your password:
It's wrong account
```

Case Statements

The case statement is good alternative to Multilevel if-then-else-fi statement. It enable you to match several values against one variable. Its easier to read and write.

Syntax:

```
case $variable-name in
    pattern1)  command
               ...
               ..
               command;;
    pattern2)  command
               ...
               ..
               command;;
    patternN)  command
               ...
               ..
               command;;
    *)         command
               ...
               ..
               command;;
esac
```

```
#!/bin/bash

read -p "Input first number: " first_number
read -p "Input second number: " second_number
read -p "Select an math operation
1 - addition
2 - subtraction
3 - multiplication
4 - division
" operation

case $operation in
  "1")
     echo "result= $(( $first_number +
$second_number))"
  ;;
  "2")
     echo "result= $(( $first_number -
$second_number))"
  ;;
  "3")
     echo "result= $(( $first_number *
$second_number))"
     ;;
  "4")
     echo "result= $(( $first_number /
$second_number))"
  ;;
  *)
     echo "Wrong choice  "
```

HomeWork

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 reach ALE.” # (X = ale - age)
      else:
         # beep sound # echo -ne '\007'
         “!!! Life Expectancy Reached !!!”
         # wait 1 sec.
         “!!! Life Expectancy Reached !!!”
         # wait 2 secs.
         “!!! Life Expectancy Reached !!!”

THANKS!

Any questions?

You can find me at:

- @sumod
- sumod@clarusway.com