

# Bash Scripting examples- Beginner Level

By Coding-Bootcamps.com

## Outline

- 1- Using While Loop
- 2- Using For Loop
- 3- Get User Input
- 4- Using if statement
- 5- Using if statement with AND logic
- 6- Using if statement with OR logic
- 7- Using else if statement
- 8- Using Case Statement
- 9- Create Function
- 10- Create function with Parameters
- 11- Pass Return Value from Function
- 12- Get Parse Current Date

### 1- Using While Loop

Create a bash file with the name, '**while\_example.sh**', to know the use of **while** loop. In the example, **while** loop will iterate for **5** times. The value of **count** variable will increment by **1** in each step. When the value of **count** variable is 5 then the **while** loop will terminate.

```
#!/bin/bash
valid=true
count=1
while [ $valid ]
do
    echo $count
    if [ $count -eq 5 ]; then
        break
    fi
    ((count++))
done
```

Run the file with bash command.

```
$ bash while_example.sh
```

### 2- Using For Loop

The basic **for** loop declaration is shown in the following example. Create a file named '**for\_example.sh**' and add the following script using **for** loop. Here, **for** loop will iterate for **10** times and print all values of the variable, **counter** in single line.

```
#!/bin/bash
for (( counter=10; counter>0; counter-- ))
do
    echo -n "$counter "
done
printf "\n"
```

Run the file with bash command.

```
$ bash for_example.sh
```

Other forms of for loops:

```
for i in 1 2 3 4 5
do
    echo "Welcome $i times"
done
```

```
for i in $(seq 1 2 20)
do
    echo "Welcome $i times"
done
```

### 3- Get User Input

'**read**' command is used to take input from user in bash. Create a file named '**user\_input.sh**' and add the following script for taking input from the user. Here, one string value will be taken from the user and display the value by combining other string value.

```
#!/bin/bash
echo "Enter Your Name"
read name
echo "Welcome $name to Intro to Bash Scripting"
```

Run the file with bash command.

```
$ bash user_input.sh
```

#### 4- Using if statement

You can use if condition with single or multiple conditions. Starting and ending block of this statement is define by 'if' and 'fi'. Create a file named 'simple\_if.sh' with the following script to know the use if statement in bash. Here, 10 is assigned to the variable, n. if the value of \$n is less than 10 then the output will be "It is a one digit number", otherwise the output will be "It is a two digit number". For comparison, '-' is used here. For comparison, you can also use '-eq' for equality, '-ne' for not equality and '-gt' for greater than in bash script.

```
#!/bin/bash
n=10
if [ $n -lt 10 ]; then
    echo "It is a one digit number"
else
    echo "It is a two digit number"
fi
```

Run the file with bash command.

```
$ bash simple_if.sh
```

#### 5- Using if statement with AND logic

Different types of logical conditions can be used in if statement with two or more conditions. How you can define multiple conditions in if statement using AND logic is shown in the following example. '&&' is used to apply AND logic of if statement. Create a file named 'if\_with\_AND.sh' to check the following code. Here, the value of username and password variables will be taken from the user and compared with 'admin' and 'secret'. If both values match then the output will be "valid user", otherwise the output will be "invalid user".

```
#!/bin/bash

echo "Enter username"
read username
echo "Enter password"
read password

if [[ ( $username == "admin" ) && ( $password == "secret" ) ]];then
    echo "valid user"
else
    echo "invalid user"
fi
```

Run the file with bash command.

```
$ bash if_with_AND.sh
```

## 6- Using if statement with OR logic

'||' is used to define **OR** logic in **if** condition. Create a file named '**if\_with\_OR.sh**' with the following code to check the use of **OR** logic of **if** statement. Here, the value of **n** will be taken from the user. If the value is equal to **15** or **45** then the output will be "**You won the game**", otherwise the output will be "**You lost the game**".

```
#!/bin/bash

echo "Enter any number"
read n

if [[ ( $n -eq 15 ) || ( $n -eq 45 ) ]]; then
    echo "You won the game"
else
    echo "You lost the game"
fi
```

Run the file with bash command.

```
$ bash if_with_OR.sh
```

## 7- Using else if statement

The use of **else if** condition is little different in bash than other programming language. '**elif**' is used to define **else if** condition in bash. Create a file named, '**elseif\_example.sh**' and add the following script to check how **else if** is defined in bash script.

```
#!/bin/bash

echo "Enter your lucky number"
read n

if [ $n -eq 101 ]; then
    echo "You got 1st prize"
```

```

elif [ $n -eq 510 ]; then
    echo "You got 2nd prize"
elif [ $n -eq 999 ]; then
    echo "You got 3rd prize"

else
    echo "Sorry, try for the next time"
fi

```

Run the file with bash command.

```
$ bash elseif_example.sh
```

## 8- Using Case Statement

**Case** statement is used as the alternative of **if-elseif-else** statement. The starting and ending block of this statement is defined by '**case**' and '**esac**'. Create a new file named, '**case\_example.sh**' and add the following script. The output of the following script will be same to the previous **else if** example.

```

#!/bin/bash

echo "Enter your lucky number"
read n
case $n in
101)
    echo "You got 1st prize" ;;
510)
    echo "You got 2nd prize" ;;
999)
    echo "You got 3rd prize" ;;
*)
    echo "Sorry, try for the next time" ;;
esac

```

Run the file with bash command.

```
$ bash case_example.sh
```

## 9- Create Function

How you can create a simple function and call the function is shown in the following script. Create a file named '**function\_example.sh**' and add the following code. You can call any function by name only without using any bracket in bash script.

```
#!/bin/bash
function F1() {
    echo 'I like bash programming'
}

F1
```

Run the file with bash command.

```
$ bash function_example.sh
```

## 10- Create function with Parameters

Bash can't declare function parameter or arguments at the time of function declaration. But you can use parameters in function by using other variable. If two values are passed at the time of function calling then \$1 and \$2 variable are used for reading the values. Create a file named '**function\_parameter.sh**' and add the following code. Here, the function, '**Rectangle\_Area**' will calculate the area of a rectangle based on the parameter values.

```
#!/bin/bash

Rectangle_Area() {
    area=$(( $1 * $2 ))
    echo "Area is : $area"
}

Rectangle_Area 10 20
```

Run the file with bash command.

```
$ bash function_parameter.sh
```

## 11- Pass Return Value from Function

Bash function can pass both numeric and string values. How you can pass a string value from the function is shown in the following example. Create a file named, **'function\_return.sh'** and add the following code. The function, **greeting()** returns a string value into the variable, **val** which prints later by combining with other string.

```
#!/bin/bash
function greeting() {
    str="Hello, $name"
    echo $str
}

echo "Enter your name"
read name

val=$(greeting)
echo "Return value of the function is $val"
```

Run the file with bash command.

```
$ bash function_return.sh
```

## 12- Get Parse Current Date

You can get the current system date and time value using **`date`** command. Every part of date and time value can be parsed using **'Y', 'm', 'd', 'H', 'M'** and **'S'**. Create a new file named **'date\_parse.sh'** and add the following code to separate day, month, year, hour, minute and second values.

```
#!/bin/bash
Year=`date +%Y`
Month=`date +%m`
Day=`date +%d`
Hour=`date +%H`
Minute=`date +%M`
Second=`date +%S`
echo `date`
echo "Current Date is: $Day-$Month-$Year"
echo "Current Time is: $Hour:$Minute:$Second"
```

Run the file with bash command.

```
$ bash date_parse.sh
```