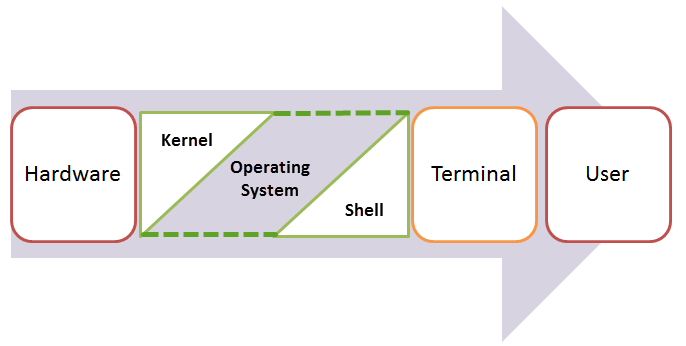
**Shell Scripting**



A shell in a Linux operating system takes input from you in the form of commands, processes it, and then gives an output. It is the interface through which a user works on the programs, commands, and scripts. A shell is accessed by a terminal which runs it.

When you run the terminal, the Shell issues **a command prompt (usually $),** where you can type your input, which is then executed when you hit the Enter key. The output or the result is thereafter displayed on the terminal.

The Shell wraps around the delicate interior of an Operating system protecting it from accidental damage. Hence the name **Shell**.

**Types of Shell**

There are two main shells in Linux:

**1**. The **Bourne Shell**: The prompt for this shell is $ and its derivatives are listed below:

* POSIX shell also is known as sh
* Korn Shell also knew as sh
* **B**ourne **A**gain **SH**ell also knew as bash (most popular)

**2.** **The C shell**: The prompt for this shell is %, and its subcategories are:

* C shell also is known as csh
* Tops C shell also is known as tcsh

## What Is Shell Scripting?

**SHELL SCRIPTING** is writing a series of commands for the shell to execute. It can combine lengthy and repetitive sequences of commands into a single and simple script, which can be stored and executed anytime. This reduces the effort required by the end user.

Let us understand the steps in creating a Shell Script

1. **Create a file** **using** a **vi** editor(or any other editor).  Name  script file with **extension .sh**
2. **Start** the script with **#! /bin/sh**
3. Write some code.
4. Save the script file as filename.sh
5. For **executing** the script type **bash filename.sh**

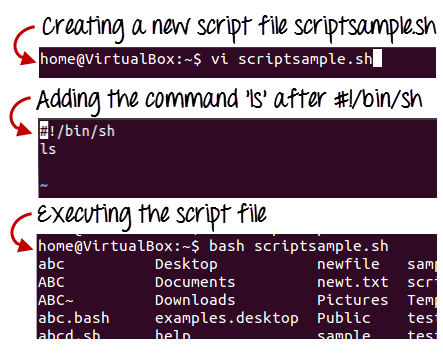
"#!" is an operator called shebang which directs the script to the interpreter location. So, if we use"#! /bin/sh" the script gets directed to the bourne-shell.

Let's create a small script -

#!/bin/sh

ls

Let's see the steps to create it -

[](https://www.guru99.com/images/vi_scriptsample(2).png)

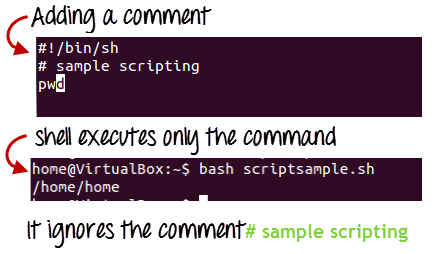
Command 'ls' is executed when we execute the scrip sample.sh file.

**Adding shell comments**

Commenting is important in any program. In Shell programming, the syntax to add a comment is

#comment

Let understand this with an example.

[](https://www.guru99.com/images/adding_comment.png)

## What are Shell Variables?

As discussed earlier, Variables store data in the form of characters and numbers. Similarly, Shell variables are used to store information and they can by the shell only.

For example, the following creates a shell variable and then prints it:

variable ="Hello"

echo $variable

Below is a small script which will use a variable.

#!/bin/sh

echo "what is your name?"

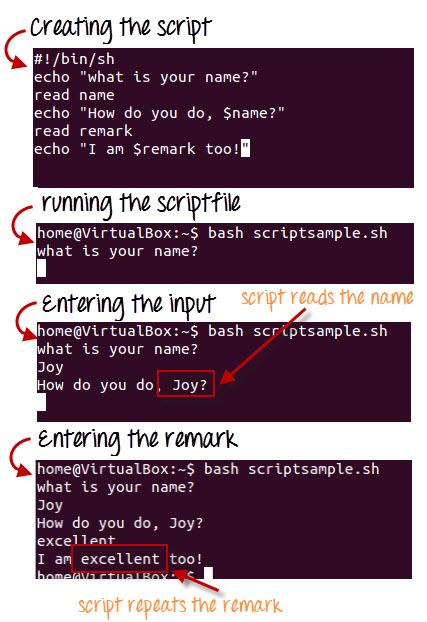
read name

echo "How do you do, $name?"

read remark

echo "I am $remark too!"

Let's understand,  the steps to create and execute the script

[](https://www.guru99.com/images/program.jpg)

As you see, the program picked the value of the variable 'name' as Joy and 'remark' as excellent.

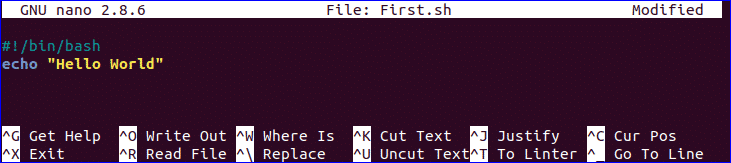
**Some more examples**

**1)** Open any editor to create a bash file. Here, **nano** editor is used to create the file and filename is set as ‘**First.sh’**

*$*nano First.sh

Add the following bash script to the file and save the file.

*#!/bin/bash*  
echo "Hello World"



You can run bash file by two ways. One way is by using bash command and another is by setting execute permission to bash file and run the file. Both ways are shown here.

*$*bash First.sh

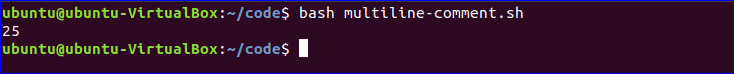
### Use of Multi-line comment:

You can use multi line comment in bash in various ways. A simple way is shown in the following example. Create a new bash named, **‘multiline-comment.sh’** and add the following script. Here, **‘:’** and **“ ’ ”** symbols are used to add multiline comment in bash script. This following script will calculate the square of 5.

*#!/bin/bash*  
: '  
The following script calculates  
the square value of the number, 5.  
'  
((area=5\*5))  
echo $area

Run the file with bash command.

*$*bash multiline-comment.sh



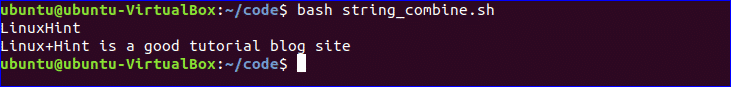
### 3. Combine String variables:

You can easily combine string variables in bash. Create a file named “**string\_combine.sh**” and add the following script to check how you can combine string variables in bash by placing variables together or using **‘+’** operator.

*#!/bin/bash*  
  
string1="Linux"  
string2="Hint"  
echo "$string1$string2"  
string3=$string1+$string2  
string3+=" is a good tutorial blog site"  
echo $string3

Run the file with bash command.

*$*bash string\_combine.sh



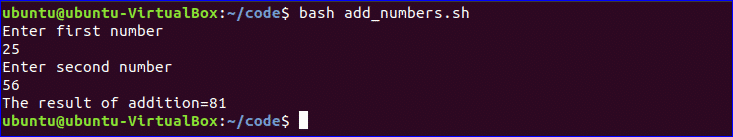
### 4. Add Two Numbers:

You can do the arithmetical operations in bash in different ways. How you can add two integer numbers in bash using double brackets is shown in the following script. Create a file named ‘**add\_numbers.sh**’ with the following code. Two integer values will be taken from the user and printed the result of addition.

*#!/bin/bash*  
echo "Enter first number"  
read x  
echo "Enter second number"  
read y  
(( sum=x+y ))  
echo "The result of addition=$sum"

Run the file with bash command.

*$*bash add\_numbers.sh



### 5. 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, **‘-lt’** 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

