Shell functions

Functions allow you to streamline an overall feature of a script into a smaller and more logical part that can perform the same function whenever it is needed through the function call.

Functions allow you to streamline an overall feature of a script into a smaller and more logical part that can perform the same function whenever it is needed through the function call.

Using functions to perform repetitive tasks is a smart way to create reuse of code. Reuse of code is an important part of modern object-oriented program rules.

Shell functions are similar to subroutines, methods, and functions of other programs.

Create functions in Unix / Linux

To publish a function, simply use the following syntax:

```
function_name () { list of commands }
```

Your function name is **function_name** , and it is what you will use to call it from anywhere in your script. The function name must be followed by **parentheses** , which are followed by a list of surrounded **commands** in **curly braces** ({}).

For example:

The following is a simple example of using functions:

```
#! / bin / sh # Define your function here Hello () { echo "Hello World" } # Inv
```

When you run the script above, it will produce the following result:

```
$ ./ test .  sh Hello World $
```

Pass parameters to a function in Unix / Linux

You can define a function that takes parameters while calling those functions. These parameters will probably be represented by $ 1, $ 2 and .

Here is an example where we pass the two parameters Zara and Ali and then we catch and print these parameters in the function.

```
#! / bin / sh # Define your function here Hello () { echo "Hello World $ 1 $ 2
```

This code will produce the following result:

```
$ ./ test .  sh Hello World Zara Ali $
```

Returns values ??from a function in Unix / Linux

If you run an **exit** command from within a function, its effect is to not only end the function of the function but also the Shell program.

If you want to end the operation of the function, there is a way to get rid of a defined function.

Based on the situation you can return any value from your function using **the return command** with the following syntax:

```
return code
```

Here the code can be anything you choose, but obviously you should choose something that is meaningful or useful in your script's overall context.

For example:

The following function returns the value 1.

```
#! / bin / sh # Define your function here Hello () { echo "Hello World $ 1 $ 2
```

The above code produces the following result:

```
$ ./ test .  sh Hello World Zara Ali Return value is 10 $
```

The functions are nested within Unix / Linux

One of the interesting features of functions is that they can either call the function itself or call other functions. A callable function itself is known as a recursive function.

Below is an example that explains the interlocking of two functions.

```
#! / bin / sh # Calling a function from another number_one () { echo "This is
```

It will produce the following result:

```
?ây là câu l?nh ??u tiên . này hi?n th?i là hai function Speaking .
```

Call the function from the command prompt in Unix / Linux

You can set the definition of commonly used functions within a .profile so that they are available every time you log in and you can use them whenever at the command prompt.

You can also choose to group these function definitions into a file, called test.sh, and then run the file on the current Shell by typing from the keyboard.

```
$ .  test .  sh
```

This can be done when the functions in this file can be read and it is defined in the current Shell as follows:

```
$ number_one This is the first function speaking . This is now the second funct
```

To remove a function definition from the current Shell, you can use the unset command with the function.**f** . It is similar to the command you use to remove a variable definition from Shell.

```
$ unset .  f function_name
```

According to Tutorialspoint

Previous article: Navigation IO in Unix / Linux

Next article: ManPage Help in Unix