Use variables in Shell

A variable is a string of characters from which we assign a value. The assigned value can be a number, text, file name, device or any other type of data.

A variable is a string of characters from which we assign a value. The assigned value can be a number, text, file name, device or any other type of data.

A variable is nothing but a pointer to real data. Shell allows you to create, assign, and delete variables.

Variable names in Unix / Linux

The name of a variable can contain only characters (a to z or from A to Z), numbers (from 0 to 9), or underscores (_).

By convention, Unix variables will have their names in capital letters.

The following example is about a valid variable name.

```
_ALI TOKEN_A VAR_1 VAR_2
```

The following is an example of invalid variable names:

```
2 _VAR - VARIABLE VAR1 - VAR2 VAR_A !
```

The reason is that you cannot use other characters like!, *, Or -, because these characters have special meaning for the shell.

Definition of variables in Unix / Linux

The variables are defined as follows:

```
variable_name = variable_value
```

For example:

```
NAME = "Zara Ali"
```

For example, define the Name variable and assign Zara Ali value to it. Variables of this type are called scala variables. A scalar variable can only hold one value at a time.

Shell allows you to keep any value you want in a variable. For example:

```
VAR1 = "Zara Ali" VAR2 = 100
```

Access values ??in Unix / Linux

To access the values ??held in a variable, precede the variable name with the symbol ($):

For example, the following script will access the value of the defined NAME variable and print it on STDOUT:

```
#! / bin / sh NAME = "Zara Ali" echo $ NAME
```

It will produce the following result:

```
Zara Ali
```

The variables are read-only (read-only) in Unix / Linux

The shell provides a way to mark variables as read-only using the **readonly** command. After a variable is read-only, its value cannot be changed.

For example, the following script will fail while trying to change the value of NAME variable.

```
#! / bin / sh NAME = "Zara Ali" readonly NAME NAME = "Qadiri"
```

It will produce the following result:

```
/ bin / sh : NAME : bi?n này ???c ??c ch? .
```

Delete a variable in Unix / Linux

Delete a variable for the shell to remove that variable from the list of variables it tracks. Once you delete a variable, you will not be able to access the value stored in that variable.

Here is the syntax to delete a defined variable using the **unset** command:

```
unset variable_name
```

The above command will delete the value of a defined variable. Here is a simple example:

```
#! / bin / sh NAME = "Zara Ali" unset NAME echo $ NAME
```

The above command will not print anything. You cannot use the unset command to delete variables that are read-only.

Variable types in Unix / Linux

When a Shell is running, there are three types of variables that exist:

Local Variables : An internal variable is a variable that exists in the shell's execution process. It is not available in programs that were started by Shell. They are set at the command prompt.

Environment Variables : An environment variable is a variable available in any shell subprocess. Some programs need environment variables to execute functions correctly. Usually a Shell script only defines the environment variables it needs by programs when it runs.

Shell Variables : A shell variable is a special variable that is set by Shell and is required by Shell to execute functions correctly. Some of these variables are environment variables, while others are internal variables.

According to Tutorialspoint

Previous article: What is Shell?

Next article: Special variables in Unix / Linux