Use array in Shell

A shell variable is able to hold a single value. This type of variables is called scalar variables.

A shell variable is able to hold a single value. This type of variables is called scalar variables.

The shell supports different types of variables called an array variable that can hold multiple values ??at the same time. Arrays provide a method for grouping a set of variables. Instead of creating a new name for each required variable, you can use an array variable to store all other variables.

All name rules are discussed in chapter Shell variables that will be applied during array naming.

Define array values ??in Unix / Linux

The difference between an array variable and a scalar variable can be explained as follows:

When you are trying to represent the names of different students as a set of variables. Each single variable is a scalar variable as follows:

```
NAME01 = "Zara" NAME02 = "Qadir" NAME03 = "Mahnaz" NAME04 = "Ayan" NAME05 = "Da
```

We use a single array to keep all the names mentioned above. The following is a simple method for creating an array variable to assign a value to one of them. This is described as follows:

```
array_name [ index ] = value
```

Here array_name is the name of the array, index is the index of the items in the array you want to set and value is the value you want to set for that item.

Here is an example, the following commands:

```
NAME [ 0 ] = "Zara" NAME [ 1 ] = "Qadir" NAME [ 2 ] = "Mahnaz" NAME [ 3 ] = "Ay
```

If you are using **ksh** shell, then the syntax of the array is:

```
set - A array_name value1 value2 . valuen
```

If you are using a **bash** shell, here the array syntax is:

```
array_name = ( value1 . valuen )
```

Access to array values ??in Unix / Linux

After you have set any array value, you can access it as follows:

Here array_name is the array name, and index is the index of the values ??to be accessed. Here is a simple example:

```
#! / bin / sh NAME [ 0 ] = "Zara" NAME [ 1 ] = "Qadir" NAME [ 2 ] = "Mahnaz" NA
```

It will produce the following result:

```
$ ./ test .  sh First Index : Zara Second Index : Qadir
```

You can access all items in an array in one of the following ways:

```
$ { array_name [*]} $ { array_name [@]}
```

Here array_name is the name of the array you are interested in. Here is a simple example:

```
#! / bin / sh NAME [ 0 ] = "Zara" NAME [ 1 ] = "Qadir" NAME [ 2 ] = "Mahnaz" NA
```

It will produce the following result:

```
$ ./ test .  sh First Method : Zara Qadir Mahnaz Ayan Daisy Second Method : Za
```

According to Tutorialspoint

Previous article: Special variables in Unix / Linux

Next lesson: Basic Shell operators