Alternately, define an array as a set of index-value pairs as follows:

```
array_var[0]="test1"
array_var[1]="test2"
array_var[2]="test3"
array_var[3]="test4"
array_var[4]="test5"
array_var[5]="test6"
```

2. Print the contents of an array at a given index using the following commands:

```
echo ${array_var[0]}
test1
index=5
echo ${array_var[$index]}
test6
```

3. Print all of the values in an array as a list using the following commands:

```
$ echo ${array_var[*]}
test1 test2 test3 test4 test5 test6
```

Alternately, you could use:

```
$ echo ${array_var[@]}
test1 test2 test3 test4 test5 test6
```

4. Print the length of an array (the number of elements in an array) as follows:

```
$ echo ${#array_var[*]}
6
```

There's more...

Associative arrays have been introduced to Bash from Version 4.0 and they are useful entities to solve many problems using the hashing technique. Let us go into more detail.

Defining associative arrays

In an associative array, we can use any text data as an array index. Initially, a declaration statement is required to declare a variable name as an associative array. This can be done as follows:

```
$ declare -A ass_array
```

After the declaration, elements can be added to the associative array using two methods as follows:

- By using inline index-value list method, we can provide a list of index-value pairs:

```
$ ass_array=([index1]=val1 [index2]=val2)
```

- Alternately, you could use separate index-value assignments:

```
$ ass_array[index1]=val1
$ ass_array'index2]=val2
```

For example, consider the assignment of price for fruits using an associative array:

```
$ declare -A fruits_value
$ fruits_value=([apple]='100dollars' [orange]='150 dollars')
```

Display the content of an array as follows:

```
$ echo "Apple costs ${fruits_value[apple]}"
Apple costs 100 dollars
```

Listing of array indexes

Arrays have indexes for indexing each of the elements. Ordinary and associative arrays differ in terms of index type. We can obtain the list of indexes in an array as follows:

```
$ echo ${!array_var[*]}
```

Or, we can also use:

```
$ echo ${!array_var[@]
```

In the previous `fruits_value` array example, consider the following command:

```
$ echo ${!fruits_value[*]}
orange apple
```

This will work for ordinary arrays too.

Visiting aliases

An **alias** is basically a shortcut that takes the place of typing a long-command sequence. In this recipe, we will see how to create aliases using the `alias` command.