AWK - ARRAYS

AWK has associative arrays and one of the best thing about it is - index need not to be continuous set of number; we can use either string or number as a array index. Also there is no need to declare the size of array in advance - array can expand/shrink at runtime. This tutorial describe the AWK arrays with suitable examples.

Below is the syntax of the array:

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

Where **array_name** is the name of array, **index** is the array index and the **value** is any value assigning to the element of the array.

Creation

To gain insights about array let us create and access the elements of the array.

```
[jerry]$ awk 'BEGIN {
fruits["mango"]="yellow";
fruits["orange"]="orange"
print fruits["orange"] "\n" fruits["mango"]
}'
```

On executing the above code, you get the following result:

```
orange
yellow
```

In above example we have declared array namely **fruits** whose index is fruit name and value is colour of the fruit. To access array element we have used **array_name[index]** format.

Deletion

For insertion we used assignment operator. Similarly we can use **delete** statement to remove an element from the array. Below is the syntax of delete statement:

```
delete array_name[index]
```

Below example deletes element **orange** hence command does not show any output.

```
[jerry]$ awk 'BEGIN {
fruits["mango"]="yellow";
fruits["orange"]="orange";
delete fruits["orange"];
print fruits["orange"]
}'
```

Multi-dimensional arrays

Awk only supports single dimensional arrays. But we can easily simulate a multi-dimensional array using the single dimensional array itself.

For instance below is 3x3 three-dimensional array:

```
100 200 300
400 500 600
700 800 900
```

In above example array[0][0] stores 100, array[0][1] stores 200 and so on. To store 100 at array location [0][0] we can use following syntax:

```
array["0,0"] = 100
```

Though we have given **0,0** as index, these are not two indexes. In reality it's just one index with the string **0,0**.

Below simple example simulates 2-D array:

```
[jerry]$ awk 'BEGIN {
array["0,0"] = 100;
array["0,1"] = 200;
array["0,2"] = 300;
array["1,0"] = 400;
array["1,1"] = 500;
array["1,2"] = 600;
# print array elements
print "array[0,0] = " array["0,0"];
print "array[0,1] = " array["0,1"];
print "array[0,2] = " array["0,2"];
print "array[1,0] = " array["1,0"];
print "array[1,1] = " array["1,1"];
print "array[1,2] = " array["1,2"];
}'
```

On executing the above code, you get the following result:

```
array[0,0] = 100
array[0,1] = 200
array[0,2] = 300
array[1,0] = 400
array[1,1] = 500
array[1,2] = 600
```

We can also perform variety of operations on the array- like sorting its elements/indexes. For that purpose we can use AWK's **asort** and **asorti** function. We will see usage of these functions in latter chapter.