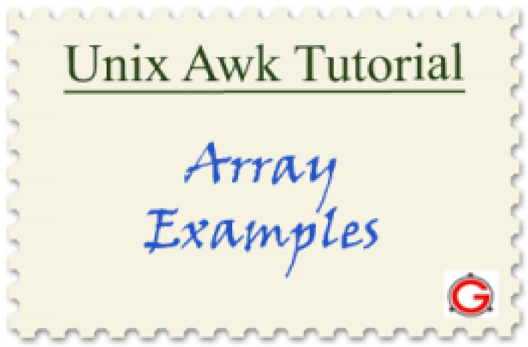
AWK Arrays Explained with 5 Practical Examples

by SASIKALA *on* MARCH 10, 2010



Awk programming language supports arrays. As part of our on-going awk examples series, we have seen awk user defined variables and awk built-in variables. Arrays are an extension of variables. Arrays are variable that hold more than one value. Similar to variables, arrays also has names. In some programming languages, arrays has to be declared, so that memory will be allocated for the arrays. Also, array indexes are typically integer, like array[1],array[2] etc.,

Awk Associative Array

Awk supports only associative array. Associative arrays are like traditional arrays except they uses strings as their indexes rather than numbers. When using an associative array, you can mimic traditional array by using numeric string as index.

```
Syntax:


arrayname[string]=value
```

- **arrayname** is the name of the array.
- **string** is the index of an array.
- **value** is any value assigning to the element of the array.

Accessing elements of the AWK array

If you want to access a particular element in an array, you can access through its index — arrayname[index], which gives you the value assigned in that index.

If you want to access all the array elements, you can use a loop to go through all the indexes of an array as shown below.

```
Syntax:


for (var in arrayname)
actions
```

In the above awk syntax:

- **var** is any variable name
- **in** is a keyword
- **arrayname** is the name of the array.
- **actions** are list of statements to be performed. If you want to perform more than one action, it has to be enclosed within braces.

This loop executes list of actions for each different value which was used as an index in array with the variable var set to that index.

Removing an element from the AWK array

If you want to remove an element in a particular index of an array, use awk delete statement. Once you deleted an element from an awk array, you can no longer obtain that value.

```
Syntax:


delete arrayname[index];
```

The loop command below removes all elements from an array. There is no single statement to remove all the elements from an array. You have to go through the loop and delete each array element using awk delete statement.

```
for (var in array)
     delete array[var]
```

5 Practical Awk Array Examples

All the examples given below uses the Iplogs.txt file shown below. This sample text file contains list of ip address requested by the gateway server. This sample Iplogs.txt file contains data in the following format:

```
[date] [time] [ip-address] [number-of-websites-accessed]
```

```
$ cat Iplogs.txt
180607 093423   123.12.23.122 133
180607 121234   125.25.45.221 153
190607 084849   202.178.23.4 44
190607 084859   164.78.22.64 12
200607 012312   202.188.3.2 13
210607 084849   202.178.23.4 34
210607 121435   202.178.23.4 32
```

Example 1. List all unique IP addresses and number of times it was requested

```
$ awk '{
> Ip[$3]++;
> }
> END{
> for (var in Ip)
> print var, "access", Ip[var]," times"
> }
> ' Iplogs.txt
125.25.45.221 access 1  times
123.12.23.122 access 1  times
164.78.22.64 access 1  times
202.188.3.2 access 2  times
202.178.23.4 access 3  times
```

In the above script:

- Third field ($3) is an ip address. This is used as an index of an array called Ip.
- For each line, it increments the value of the corresponding ip address index.
- Finally in the END section, all the index will be the list of unique IP address and its corresponding values are the occurrence count.

Example 2. List all the IP address and calculate how many sites it accessed

The last field in the Iplogs.txt is the number of sites each IP address accessed on a particular date and time. The below script generates the report which has list of IP address and how many times it requested gateway and total number of sites it accessed.

```
$cat ex2.awk
BEGIN {
print "IP Address\tAccess Count\tNumber of sites";
}
```

```
END{
for (var in Ip)
    print var,"\t",Ip[var],"\t\t",count[var];
}

$ awk -f ex2.awk Iplogs.txt
IP Address  Access Count    Number of sites
125.25.45.221    1        153
123.12.23.122    1        133
164.78.22.64     1        12
202.188.3.2      2        180
202.178.23.4     3        110
```

In the above example:

- It has two arrays. The index for both the arrays are same — which is the IP address (third field).
- The first array named “Ip” has list of unique IP address and its occurrence count. The second array called “count” has the IP address as an index and its value will be the last field (number of sites), so whenever the IP address comes it just keeps on adding the last field.
- In the END section, it goes through all the IP address and prints the Ip address and access count from the array called Ip and number of sites from the array count.

Example 3. Identify maximum access day

```
$ cat ex3.awk
{
date[$1]++;
}
END{
for (count in date)
{
    if ( max < date[count] ) {
        max = date[count];
print "Maximum access is on", maxdate;
}


$ awk -f ex3.awk Iplogs.txt
Maximum access is on 210607
```

In this example:

- array named “date” has date as an index and occurrence count as the value of the array.
- max is a variable which has the count value and used to find out the date which has max count.
- maxdate is a variable which has the date for which the count is maximum.

Example 4. Reverse the order of lines in a file

```
$ awk '{ a[i++] = $0 } END { for (j=i-1; j>=0;) print a[j--] }' Iplogs.tx
210607 132423   202.188.3.2 167
210607 121435   202.178.23.4 32
210607 084849   202.178.23.4 34
200607 012312   202.188.3.2 13
190607 084859   164.78.22.64 12
190607 084849   202.178.23.4 44
180607 121234   125.25.45.221 153
180607 093423   123.12.23.122 133
```

In this example,

- It starts by recording all the lines in the array ‘a’.
- When the program has finished processing all lines, Awk executes the END { } block.
- The END block loops over the elements in the array ‘a’ and prints the recorded lines in reverse manner.

Example 5. Remove duplicate and nonconsecutive lines using awk

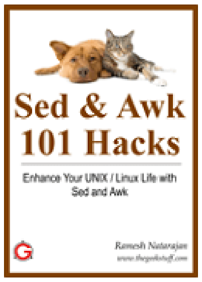
```
foo
baz
bar

$ awk '!($0 in array) { array[$0]; print }' temp
foo
bar
baz
```

In this example:

- Awk reads every line from the file “temp”, and using “in” operator it checks if the current line exist in the array “a”.
- If it does not exist, it stores and prints the current line.

Recommended Reading



Tweet பிடித்திருக்கிற Add your comment

If you enjoyed this article, you might also like..

1. 50 Linux Sysadmin Tutorials
2. 50 Most Frequently Used Linux Commands (With Examples)

- Awk Introduction – 7 Awk Print Examples
- Advanced Sed Substitution Examples