AWK COMMAND EXAMPLES

awk is one of the most powerful utilities used in the unix world. Whenever it comes to text parsing, sed and awk do some unbelievable things. In this first article on awk, we will see the basic usage of awk.

The syntax of awk is:

awk 'pattern{action}' file

where the pattern indicates the pattern or the condition on which the action is to be executed for every line matching the pattern. In case of a pattern not being present, the action will be executed for every line of the file. In case of the action part not being present, the default action of printing the line will be done. Let us see some examples:

Assume a file, say file1, with the following content:

```
$ cat file1

Name Domain

Deepak Banking

Neha Telecom

Vijay Finance

Guru Migration
```

This file has 2 fields in it. The first field indicates the name of a person, and the second field denoting their expertise, the first line being the header record.

1. To **print only the names** present in the file:

```
$ awk '{print $1}' file1

Name

Deepak

Neha

Vijay
```

```
Guru
```

The above awk command does not have any pattern or condition. Hence, the action will be executed on every line of the file. The action statement reads "print $1". awk, while reading a file, splits the different columns into $1, $2, $3 and so on. And hence the first column is accessible using $1, second using $2, etc. And hence the above command prints all the names which happens to be first column in the file.

2. Similarly, to **print the second column** of the file:

```
$ awk '{print $2}' file1

Domain

Banking

Telecom

Finance

Migration
```

3. In the first example, the list of names got printed along with the header record. **How to omit the header record** and get only the names printed?

```
$ awk 'NR!=1{print $1}' file1

Deepak

Neha

Vijay

Guru
```

The above awk command uses a special variable NR. NR denotes line number ranging from 1 to the actual line count. The conditon 'NR!=1' indicates not to execute the action part for the first line of the file, and hence the header record gets skipped.

4. How do we **print the entire file contents**?

```
$ awk '{print $0}' file1

Name Domain

Deepak Banking

Neha Telecom

Vijay Finance

Guru Migration
```

$0 stands for the entire line. And hence when we do "print $0", the whole line gets printed.

5. How do we get the **entire file content printed in other way**?

```
$ awk '1' file1

Name Domain

Deepak Banking

Neha Telecom

Vijay Finance

Guru Migration
```

The above awk command has only the pattern or condition part, no action part. The '1' in the pattern indicates "true" which means true for every line. As said above, no action part denotes just to print which is the default when no action statement is given, and hence the entire file contents get printed.

Let us now **consider a file with a delimiter**. The delimiter used here is a comma. The comma separated file is called csv file. Assuming the file contents to be:

```
$ cat file1

Name,Domain,Expertise

Deepak,Banking,MQ Series

Neha,Telecom,Power Builder
```

```
Vijay,Finance,CRM Expert

Guru,Migration,Unix
```

This file contains 3 fields. The new field being the expertise of the respective person.

6. Let us try to **print the first column of this csv file** using the same method as mentioned in Point 1.

```
$ awk '{print $1}' file1

Name,Domain,Expertise

Deepak,Banking,MQ

Neha,Telecom,Power

Vijay,Finance,CRM

Guru,Migration,Unix
```

The output looks weird. Isnt it? We expected only the first column to get printed, but it printed little more and that too not a definitive one. If you notice carefully, it printed every line till the first space is encountered. awk, by default, uses the white space as the delimiter which could be a single space, tab space or a series of spaces. And hence our original file was split into fields depending on space.

Since our requirement now involves dealing with a file which is comma separated, we need to specify the delimiter.

```
$ awk -F","  '{print $1}' file1

Name

Deepak

Neha

Vijay

Guru
```

awk has a command line option "-F' with which we can specify the delimiter. Once the delimiter is specified, awk splits the file on the basis of the delimiter specified, and hence we got the names by printing the first column $1.

7. awk has a **special variable called "FS"** which stands for field separator. In place of the command line option "-F', we can also use the "FS".

```
$ awk  '{print $1,$3}' FS="," file1

Name Expertise

Deepak MQ Series

Neha Power Builder

Vijay CRM Expert

Guru Unix
```

8. Similarly, to **print the second column**:

```
$ awk -F, '{print $2}' file1

Domain

Banking

Telecom

Finance

Migration
```

9. To **print the first and third columns,** ie., the name and the expertise:

```
$ awk -F"," '{print $1, $3}' file1

Name Expertise

Deepak MQ Series
```

```
Neha Power Builder

Vijay CRM Expert

Guru Unix
```

10. The output shown above is not easily readable since the third column has more than one word. It would have been better had the fields being displayed are present with a delimiter. Say, lets **use comma to separate the output**. Also, lets discard the header record.

```
$ awk -F"," 'NR!=1{print $1,$3}' OFS="," file1

Deepak,MQ Series

Neha,Power Builder

Vijay,CRM Expert

Guru,Unix
```

OFS is another awk special variable. Just like how FS is used to separate the input fields, OFS (Output field separator) is used to separate the output fields.