# awk

# awk

Awk is a programming language which allows easy manipulation of structured data and the generation of formatted reports. Awk stands for the names of its authors “**A**ho, **W**einberger, and **K**ernighan”

The Awk is mostly used for pattern scanning and processing.

It searches one or more files to see if they contain lines that matches with the specified patterns and then perform associated actions.

Some of the key features of Awk are:

- Awk views a text file as records and fields.
- Like common programming language, Awk has variables, conditionals and loops
- Awk has arithmetic and string operators.
- Awk can generate formatted reports

# Awk Methodology

1. Awk reads the input files one line at a time.
2. For each line, it matches with given pattern in the given order, if matches performs the corresponding action.
3. If no pattern matches, no action will be performed.
4. In the above syntax, either search pattern or action are optional, But not both.
5. If the search pattern is not given, then Awk performs the given actions for each line of the input.
6. If the action is not given, print all that lines that matches with the given patterns which is the default action.
7. Empty braces with out any action does nothing. It wont perform default printing operation.
8. Each statement in Actions should be delimited by semicolon.

$cat employee.txt

```
120  Suraj   Manager     Sales    $5,000
140  Anil    Developer   Technology  $5,500
280  Rajiv   Sysadmin    Technology  $7,000
430  Pooja   Manager     Marketing   $9,500
510  Rajesh  DBA         Technology  $6,000
```

Example 1:

```
$ awk '{print;}' employee.txt
```

Example 2

```
$ awk '/Thomas/
> /Nisha/' employee.txt
```

Note: prints all the line which matches with the ‘Thomas’ or ‘Nisha’. It has two patterns.

Example 3

**$ awk '{print $2,$5;}' employee.txt**

**$ awk '{print $2,$NF;}' employee.txt**

If the line has 4 words, it will be stored in $1, $2, $3 and $4. $0 represents whole line. NF is a built in variable which represents total number of fields in a record.

# Initialization and Final Action

Syntax:

```
BEGIN { Actions}
{ACTION} # Action for everyline in a file
END { Actions }
```

# is for comments in Awk

```
$ awk 'BEGIN {print "Name\tDesignation\tDepartment\tSalary";}
> {print $2,"\t",$3,"\t",$4,"\t",$NF;}
> END{print "Report Generated\n--------------";
> }' employee.txt
```

```
$ awk '$1 >200' employee.txt
```

first field ($1) is employee id. So if $1 is greater than 200, then just do the default print action to print the whole line.

```
$ awk '$4 ~/Technology/' employee.txt
```

Operator ~ is for comparing with the regular expressions.

```
$ awk 'BEGIN { count=0;}
$4 ~ /Technology/ { count++; }
END { print "Number of employees in Technology Dept =",count;}' employee.txt
```