**Awk Introduction and Printing Operations**

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 reads from a file or from its standard input, and outputs to its standard output. Awk does not get along with non-text files.

Syntax:

awk '/search pattern1/ {Actions}

/search pattern2/ {Actions}' file

In the above awk syntax:

* search pattern is a regular expression.
* Actions – statement(s) to be performed.
* several patterns and actions are possible in Awk.
* file – Input file.
* Single quotes around program is to avoid shell not to interpret any of its special characters.

**Awk Working 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.

Let us create employee.txt file which has the following content, which will be used in the  
examples mentioned below.

$cat employee.txt

100 Thomas Manager Sales $5,000

200 Jason Developer Technology $5,500

300 Sanjay Sysadmin Technology $7,000

400 Nisha Manager Marketing $9,500

500 Randy DBA Technology $6,000

**Awk Example 1. Default behavior of Awk**

By default Awk prints every line from the file.

$ awk '{print;}' employee.txt

100 Thomas Manager Sales $5,000

200 Jason Developer Technology $5,500

300 Sanjay Sysadmin Technology $7,000

400 Nisha Manager Marketing $9,500

500 Randy DBA Technology $6,000

In the above example pattern is not given. So the actions are applicable to all the lines.  
Action print with out any argument prints the whole line by default. So it prints all the  
lines of the file with out fail. Actions has to be enclosed with in the braces.

**Awk Example 2. Print the lines which matches with the pattern.**

$ awk '/Thomas/

> /Nisha/' employee.txt

100 Thomas Manager Sales $5,000

400 Nisha Manager Marketing $9,500

In the above example it prints all the line which matches with the ‘Thomas’ or ‘Nisha’. It has two patterns. Awk accepts any number of patterns, but each set (patterns and its corresponding actions) has to be separated by newline.

**Awk Example 3. Print only specific field.**

Awk has number of built in variables. For each record i.e line, it splits the record delimited by whitespace character by default and stores it in the $n variables. 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.

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

Thomas $5,000

Jason $5,500

Sanjay $7,000

Nisha $9,500

Randy $6,000

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

Thomas $5,000

Jason $5,500

Sanjay $7,000

Nisha $9,500

Randy $6,000

In the above example $2 and $5 represents Name and Salary respectively. We can get the Salary using  $NF also, where $NF represents last field. In the print statement ‘,’ is a concatenator.

**Awk Example 4. Initialization and Final Action**

Awk has two important patterns which are specified by the keyword called BEGIN and END.

Syntax:

BEGIN { Actions}

{ACTION} # Action for everyline in a file

END { Actions }

# is for comments in Awk

Actions specified in the BEGIN section will be executed before starts reading the lines from the input.  
END actions will be performed after completing the reading and processing the lines from the input.

$ awk 'BEGIN {print "Name\tDesignation\tDepartment\tSalary";}

> {print $2,"\t",$3,"\t",$4,"\t",$NF;}

> END{print "Report Generated\n--------------";

> }' employee.txt

Name Designation Department Salary

Thomas Manager Sales $5,000

Jason Developer Technology $5,500

Sanjay Sysadmin Technology $7,000

Nisha Manager Marketing $9,500

Randy DBA Technology $6,000

Report Generated

--------------

In the above example, it prints headline and last file for the reports.

**Awk Example 5. Find the employees who has employee id greater than 200**

$ awk '$1 >200' employee.txt

300 Sanjay Sysadmin Technology $7,000

400 Nisha Manager Marketing $9,500

500 Randy DBA Technology $6,000

In the above example, 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 Example 6. Print the list of employees in Technology department**

Now department name is available as a fourth field, so need to check if $4 matches with the string “Technology”, if yes print the line.

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

200 Jason Developer Technology $5,500

300 Sanjay Sysadmin Technology $7,000

500 Randy DBA Technology $6,000

Operator ~ is for comparing with the regular expressions. If it matches the default action i.e print whole line will be  performed.

**Awk Example 7. Print number of employees in Technology department**

The below example, checks if the department is Technology, if it is yes, in the Action, just increment the count variable, which was initialized with zero in the BEGIN section.

$ awk 'BEGIN { count=0;}

$4 ~ /Technology/ { count++; }

END { print "Number of employees in Technology Dept =",count;}' employee.txt

Number of employees in Tehcnology Dept = 3

Then at the end of the process, just print the value of count which gives you the number of employees in Technology department.

**Print every line after erasing the 2nd field**

awk '{$2 = ""; print}' file

**Print hi 48 times**

yes | head -48 | awk '{ print "hi" }'

**Print hi.0010 to hi.0099**

yes | head -90 | awk '{printf("hi00%2.0f n", NR+9)}'

**print when column 3 is < 1900 (in myfile.txt). If a value is not numeric, it doesn't complain**

**cat myfile.txt| awk '{if ($3 < 1900) print $3, " ",$5,$7,$8}'**

**Count number of lines where col 3 > col 1**

**awk '$3 > $1 {print i + "1"; i++}' file**

**print only lines of less than 65 characters**

**awk 'length < 64'**