AWK command in Unix/Linux Documented by Praveen

Awk is a scripting language used for manipulating data and generating reports.The awk command programming language requires no compiling, and allows the user to use variables, numeric functions, string functions, and logical operators.

Awk is a utility that enables a programmer to write tiny but effective programs in the form of statements that define text patterns that are to be searched for in each line of a document and the action that is to be taken when a match is found within a line. 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 performs the associated actions.

WHAT CAN WE DO WITH AWK

1. **AWK Operations:**
   (a) Scans a file line by line
   (b) Splits each input line into fields
   (c) Compares input line/fields to pattern
   (d) Performs action(s) on matched lines

2. **Useful For:**
   (a) Transform data files
   (b) Produce formatted reports

3. **Programming Constructs:**
   (a) Format output lines
   (b) Arithmetic and string operations
   (c) Conditionals and loops

Syntax:

```
awk options 'selection _criteria {action }' input-file > output-file
```

Options:

```
-f program-file : Reads the AWK program source from the file
                  program-file, instead of from the
                  first command line argument.
-F fs           : Use fs for the input field separator
```

Sample Commands

Example:
Consider the following text file as the input file for all cases below.

```
root@ip-172-31-30-133:~# cat > file1.txt
venkatl clerk account 25000
phani manager sales 50000
sanjay manager account 47000
raju peon sales 15000
praveen clerk sales 23000
harish peon sales 13000
akhil director purchase 80000
```

1. **Default behavior of Awk :** By default Awk prints every line of data from the specified file.

```
root@ip-172-31-30-133:~# awk '{print}' file1.txt
venkatl clerk account 25000
phani manager sales 50000
sanjay manager account 47000
raju peon sales 15000
praveen clerk sales 23000
harish peon sales 13000
akhil director purchase 80000
```

In the above example, no pattern is given. So the actions are applicable to all the lines. Action print without any argument prints the whole line by default, so it prints all the lines of the file without failure.

2. **Print the lines which matches with the given.**

```
root@ip-172-31-30-133:~#  awk '/manager/ {print}' file1.txt
phani manager sales 50000
sanjay manager account 47000
```

In the above example, the awk command prints all the line which matches with the 'manager'.

3. **Spliting a Line Into Fields :** For each record i.e line, the awk command 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 respectively. Also, $0 represents the whole line.

```
root@ip-172-31-30-133:~# awk '{print $1,$4}' file1.txt
venkatl 25000
phani 50000
sanjay 47000
raju 15000
praveen 23000
harish 13000
akhil 80000
```

In the above example, $1 and $4 represents Name and Salary fields respectively.

Built In Variables In Awk

Awk's built-in variables include the field variables—$1, $2, $3, and so on ($0 is the entire line) — that break a line of text into individual words or pieces called fields.

NR: NR command keeps a current count of the number of input records. Remember that records are usually lines. Awk command performs the pattern/action statements once for each record in a file.
NF: NF command keeps a count of the number of fields within the current input record.
FS: FS command contains the field separator character which is used to divide fields on the input line. The default is "white space", meaning space and tab characters. FS can be reassigned to another character (typically in BEGIN) to change the field separator.
RS: RS command stores the current record separator character. Since, by default, an input line is the input record, the default record separator character is a newline.
OFS: OFS command stores the output field separator, which separates the fields when Awk prints them. The default is a blank space. Whenever print has several parameters separated with commas, it will print the value of OFS in between each parameter.
ORS: ORS command stores the output record separator, which separates the output lines when Awk prints them. The default is a newline character. print automatically outputs the contents of ORS at the end of whatever it is given to print.

Examples:

Use of NR built-in variables (Display Line Number)

```
root@ip-172-31-30-133:~# awk '{print NR,$0}' file1.txt
1 venkatl clerk account 25000
2 phani manager sales 50000
3 sanjay manager account 47000
4 raju peon sales 15000
5 praveen clerk sales 23000
6 harish peon sales 13000
7 akhil director purchase 80000
8
```

In the above example, the awk command with NR prints all the lines along with the line number.

Use of NF built-in variables (Display Last Field)

```
root@ip-172-31-30-133:~# awk '{print $1,$NF}' file1.txt
venkatl 25000
phani 50000
sanjay 47000
raju 15000
praveen 23000
harish 13000
akhil 80000
```

In the above example $1 represents Name and $NF represents Salary. We can get the Salary using $NF , where $NF represents last field.

Another use of NR built-in variables (Display Line From 3 to 6)

```
root@ip-172-31-30-133:~# awk 'NR==3, NR==6 {print NR,$0}' file1.txt
3 sanjay manager account 47000
4 raju peon sales 15000
5 praveen clerk sales 23000
6 harish peon sales 13000
```

More Examples

For the given text file:

```
root@ip-172-31-30-133:~# cat  file2
A    B    C
ajay    A12    1
venkat    B6    2
Praveen    M42    3
```

1) **To print the first item along with the row number(NR) separated with ” – “ from each line in geeksforgeeks.txt:**

```
root@ip-172-31-30-133:~# awk '{print NR "- " $1 }' file2
1- A
2- ajay
3- venkat
4- Praveen
5-
```

2) **To return the second row/item from geeksforgeeks.txt:**

```
root@ip-172-31-30-133:~# awk '{print $2}' file2
B
A12
B6
M42
```

3) **To print any non empty line if present**

```
root@ip-172-31-30-133:~# awk 'NF > 0' file2
A    B    C
ajay    A12    1
venkat    B6    2
Praveen    M42    3
```

4) **To find the length of the longest line present in the file:**

```
root@ip-172-31-30-133:~# awk '{ if (length($0) > max) max = length($0) } END { print max }' file2
19
```

5) To count the lines in a file:

```
root@ip-172-31-30-133:~# awk 'END { print NR }' file2
5
```

6) Printing lines with more than 10 characters:

```
root@ip-172-31-30-133:~# awk 'length($0) > 10' file2
A    B    C
ajay    A12    1
venkat    B6    2
Praveen    M42    3
```

7) To find/check for any string in any column:

```
$ awk '{ if($3 == "B6") print $0;}' file2
```

8) To print the squares of first numbers from 1 to n say 6:

```
root@ip-172-31-30-133:~# awk '{ if($3 == "B6") print $0;}' file2
root@ip-172-31-30-133:~# awk 'BEGIN { for(i=1;i<=6;i++) print "square of", i, "is",i*i; }'
square of 1 is 1
square of 2 is 4
square of 3 is 9
square of 4 is 16
square of 5 is 25
square of 6 is 36
root@ip-172-31-30-133:~# awk 'BEGIN { for(i=1;i<=6;i++) print "square of", i, "is",i*i; }' file2
square of 1 is 1
square of 2 is 4
square of 3 is 9
square of 4 is 16
square of 5 is 25
square of 6 is 36
```