AWK command in Unix/Linux with examples

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 perform the associated actions.

Awk is abbreviated from the names of the developers – Aho, Weinberger, and Kernighan.

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

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:

```
$cat > employee.txt
```

```
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik director purchase 80000
```

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

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

Output:

```
ajay manager account 45000
sunil clerk account 25000
varun manager sales 50000
amit manager account 47000
tarun peon sales 15000
deepak clerk sales 23000
sunil peon sales 13000
satvik 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 match the given pattern.

```
$ awk '/manager/ {print}' employee.txt
```

Output:

```
ajay manager account 45000
varun manager sales 50000
amit manager account 47000
```

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

3. Splitting a Line Into Fields : For each record i.e line, the awk command splits the line has 4 words, it will be stored in $1, $2, $3 and $4 respectively. Also, $0 represents the whole line.

```
$ awk '{print $1,$4}' employee.txt
```

Output:

```
ajay 45000
sunil 25000
varun 50000
amit 47000
tarun 15000
deepak 23000
sunil 13000
satvik 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:

```
$ awk '{print NR,$0}' employee.txt
```

Output:

```
1 ajay manager account 45000
2 sunil clerk account 25000
3 varun manager sales 50000
4 amit manager account 47000
5 tarun peon sales 15000
6 deepak clerk sales 23000
7 sunil peon sales 13000
8 satvik director purchase 80000
```

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)

```
$ awk '{print $1,$NF}' employee.txt
```

Output:

```
ajay 45000
sunil 25000
varun 50000
amit 47000
tarun 15000
deepak 23000
sunil 13000
satvik 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)

```
$ awk 'NR==3, NR==6 {print NR,$0}' employee.txt
```

Output:

```
3 varun manager sales 50000
```

```
5 tarun peon sales 15000
6 deepak clerk sales 23000
```

More Examples

For the given text file:

```
$cat > geeksforgeeks.txt


A    B    C
Tarun    A12    1
Man    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:

```
$ awk '{print NR "- " $1 }' geeksforgeeks.txt


1 - A
2 - Tarun
3 – Manav
4 - Praveen
```

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

The question should be:- To return the second column/item from geeksforgeeks.txt:

```
$ awk '{print $2}' geeksforgeeks.txt


B
A12
B6
M42
```

3) To print any non empty line if present

```
$ awk 'NF < 0' geeksforgeeks.txt
```

here NF should be 0 not less than and the user have to print the line number also:

```
awk ‘NF <= 0 {print NR}’  geeksforgeeks.txt
```

```
0
```

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

```
$ awk '{ if (length($0) > max) max = length($0) } END { print max }' geeksforgeeks.txt
```

```
13
```

5) To count the lines in a file:

```
$ awk 'END { print NR }' geeksforgeeks.txt
```

```
3
```

6) Printing lines with more than 10 characters:

```
$ awk 'length($0) > 10' geeksforgeeks.txt
```

```
Tarun    A12    1
Praveen    M42    3
```

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

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

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

```
$ 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
```

This article is contributed by **Anshika Goyal** and **Praveen Negi**. If you like GeeksforGeeks or mail your article to review-team@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please write comments if you find anything incorrect, or you want to share more information about the topic discussed above.

Last Updated : 17 Jun, 2022

208

Previous

Next

Article Contributed By :

GeeksforGeeks

Vote for difficulty

Current difficulty : Easy

Easy Normal Medium Hard Expert

Article Tags : linux-command, Linux-text-processing-commands, Linux-Unix

Improve Article Report Issue

GeeksforGeeks
A-143, 9th Floor, Sovereign Corporate Tower,
Sector-136, Noida, Uttar Pradesh - 201305

feedback@geeksforgeeks.org

Company

About Us

Careers

In Media

Languages

Python

Java

C++

Terms and Conditions

Privacy Policy

Copyright Policy

Third-Party Copyright Notices

Advertise with us

Data Structures

Array

String

Linked List

Stack

Queue

Tree

Graph

Web Development

HTML

CSS

JavaScript

Bootstrap

ReactJS

AngularJS

NodeJS

Computer Science

GATE CS Notes

Operating Systems

Computer Network

Database Management System

Software Engineering

Digital Logic Design

Engineering Maths

Interview Corner

Company Preparation

Preparation for SDE

SQL

R Language

Android Tutorial

Algorithms

Sorting

Searching

Greedy

Dynamic Programming

Pattern Searching

Recursion

Backtracking

Write & Earn

Write an Article

Improve an Article

Pick Topics to Write

Write Interview Experience

Internships

Video Internship

Data Science & ML

Data Science With Python

Data Science For Beginner

Machine Learning Tutorial

Maths For Machine Learning

Pandas Tutorial

NumPy Tutorial

NLP Tutorial

Python

Python Tutorial

Python Programming Examples

Experienced Interview

Internship Interview

Competitive Programming

Aptitude

Python Projects

Python Tkinter

OpenCV Python Tutorial

GfG School

CBSE Notes for Class 8

CBSE Notes for Class 9

CBSE Notes for Class 10

CBSE Notes for Class 11

CBSE Notes for Class 12

English Grammar

UPSC/SSC/BANKING

SSC CGL Syllabus

SBI PO Syllabus

IBPS PO Syllabus

UPSC Ethics Notes

UPSC Economics Notes

UPSC History Notes