Doc 15 Linux AWK commands

## What is AWK Command in Linux?

AWK is a programming language and tool that is used for text processing and data manipulation. It is typically used for extracting and manipulating data from text files, but it can also be used for more complex tasks such as generating reports, analyzing data and more. AWK is available on almost all Unix-based systems and is often used in conjunction with other command-line tools such as sed and grep.

## AWK Command Syntax

The AWK command has a simple syntax that consists of a series of patterns and actions. Patterns are used to match specific lines of input, while actions are used to specify what should be done with those lines. Here is an example of basic syntax of AWK command ?

awk 'pattern { action }' input\_file

The pattern is enclosed in single quotes and specifies which lines of input file should be processed. action is enclosed in curly braces and specifies what should be done with those lines. Multiple patterns and actions can be combined to create more complex scripts.

## AWK Command Functions

AWK has a variety of built-in functions that can be used to manipulate and process data. Here are some of most commonly used functions ?

* **print** ? This function is used to print text to console or a file.
* **printf** ? This function is used to format text and print it to console or a file.
* **length** ? This function is used to determine length of a string.
* **substr** ? This function is used to extract a substring from a string.
* **split** ? This function is used to split a string into an array based on a specified delimiter.
* **getline** ? This function is used to read next line of input.

## AWK Command Examples

Now that we understand basic syntax and functions of AWK command, let's look at some examples of how it can be used.

### Example 1: Print Contents of a File

The simplest use of AWK command is to print contents of a file to console. Here's how to do it ?

awk '{ print }' input\_file

This script will match every line in input file and print it to console.

### Example 2: Print Specific Columns of a File

One of most common uses of AWK is to extract specific columns from a file. Here's how to extract first and third columns of a file ?

awk '{ print $1, $3 }' input\_file

This script will match every line in input file and print first and third columns to console.

### Example 3: Filter Lines Based on a Condition

AWK can also be used to filter lines based on a condition. Here's how to print all lines in a file that contain word "error" ?

awk '/error/ { print }' input\_file

This script will match every line in input file that contains word "error" and print it to console.

### Example 4: Sum Values in a Column

AWK can also be used to perform mathematical operations on data. Here's how to sum values in second column of a file ?

awk '{ sum += $2 } END { print sum }' input\_file

This script will add up all values in second column of input file and print total to console.

### Example 5: Extract Substring from a Column

Another common task is to extract a substring from a column in a file. Here's how to extract first three characters from first column of a file ?

awk '{ print substr($1, 1, 3) }' input\_file

This script will extract first three characters from first column of input file and print them to console.

### Example 6: Join Two Files Based on a Common Column

AWK can also be used to join two files based on a common column. Here's how to join two files based on value of first column ?

awk 'NR==FNR{a[$1]=$2;next}{ print $0, a[$1] }' file1 file2

This script will read first file into memory and create an array with values in second column. It will then read second file and print each line, followed by corresponding value from first file.

### Example 7: Extract First Column of a File

To extract first column of a file using AWK command, use following syntax ?

$ awk '{ print $1 }' filename

Here, $1 refers to first column of input file. print command is used to display output.

### Example 8: Calculate Average of a Column

To calculate average of a column using AWK command, use following syntax ?

$ awk '{ sum += $1 } END { print sum / NR }' filename

Here, sum is a variable that stores sum of values in first column. NR is a built-in variable that stores number of records (lines) processed by AWK command. END pattern matches end of input file.

### Example 9: Replace a String in a File

To replace a string in a file using AWK command, use following syntax ?

$ awk '{ gsub("oldstring", "newstring") } 1' filename

Here, gsub is a function that replaces all occurrences of oldstring with newstring. 1 pattern matches all lines of input file.

### Example 10: Display Last Field of a File

To display last field of a file using AWK command, use following syntax ?

$ awk '{ print $NF }' filename

Here, $NF refers to last field of input file. print command is used to display output.

### Example 11: Using Regular Expressions

Regular expressions are a powerful feature of AWK command that allows users to search for patterns in data. Here's an example that demonstrates use of regular expressions ?

$ awk '/pattern/ { print }' filename

Here, /pattern/ is a regular expression that matches any line that contains pattern.

### Example 12: Using Variables

Variables can be used in AWK command to store values that can be used in actions. Here's an example that demonstrates use of variables ?

$ awk '{ total += $1 } END { print "Total: ", total }' filename

Here, variable total is used to store sum of values in first column.

### Example 13: Using Built-in Variables

The AWK command has several built-in variables that can be used to perform various tasks. Here are some examples ?

* **FS** ? field separator variable is used to specify delimiter used to separate fields in input file. By default, it is set to whitespace.
* **RS** ? record separator variable is used to specify delimiter used to separate records in input file. By default, it is set to a newline character.
* **OFS** ? output field separator variable is used to specify delimiter used to separate fields in output. By default, it is set to whitespace.
* **ORS** ? output record separator variable is used to specify delimiter used to separate records in output. By default, it is set to a newline character.

### Example 14: Using Control Statements

Control statements such as if-else and while loops can be used in AWK command to perform conditional operations. Here's an example that demonstrates use of if-else statements ?

$ awk '{ if ($1 > 50) { print "Pass" } else { print "Fail" } }' filename

Here, if value in first column is greater than 50, output will be "Pass," otherwise it will be "Fail."

### Example 15: Using Functions

Functions can be defined and used in AWK command to perform complex operations. Here's an example that demonstrates use of functions ?

$ awk 'function square(x) { return x\*x } { print square($1) }' filename

Here, function square is defined and used to calculate square of value in first column