International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org, editorijaiem@gmail.com
Volume 2, Issue 2, February 2013 ISSN 2319 - 4847

Review of AWK Programming

[1]Jitendra Singh Kushwah, [2]Abhishek Chaturvedi, [3]Rajeev Sharma

[1]Assistant Professor, School of Computer Application, ITM University Gwalior (M.P)

[2]Assistant Professor, Department of Computer Application
Shri Rawatpura Sarkar Snatak Mahavidhyalaya, Datia (M.P), India

[3]Assistant Professor, Department of Computer Application
Maharaja Institute of Management & Technology, Gwalior(M.P), India

ABSTRACT

The word "AWK" is derived from the initials of the language's three developers: A. Aho, B. W. Kernighan and P. Weinberger. Awk is an extremely versatile programming language for working on files. It is an excellent filter and report writer. AWK is an excellent tool for processing these rows and columns, and is easier to use AWK than most conventional programming languages. AWK also has string manipulation functions, so it can search for particular strings and modify the output. There are few definitions of AWK:

1. "Awk is a programming language whose basic operation is to search a set of files for patterns, and to perform specified actions upon lines or fields of lines which contain instances of those patterns."

2. "Awk is a programming language that make it possible to handle simple, mechanical data manipulation tasks with very short programs, often only one or two lines long"

3. "An awk program is a sequence of patterns and actions that tell what to look for in the input data and what to do when it's found."

Keywords: BEGIN, END, Pattern, Action, Operators, Branching, Looping, String

1. INTRODUCTION

The essential organization of an AWK program follows the form:
pattern { action }
The pattern specifies when the action is performed.That is, the pattern specifies a test that is performed with each line read as input. If the condition is true, then the action is taken. Two other important patterns are specified by the keywords "BEGIN" and "END". BEGIN Specifies procedures that take place before the first input line is processed. END Specifies procedures that take place after the last input record is read. In AWK script, no need to declared of variables using data type because we have only two type of data (Integer & string). AWK is also an interpretor. We can create a in AWK using VI editor with .awk extension name.
$vi csi.awk → Create a script
Script for csi.awk is as follow:

```
BEGIN { print "CSI Communications" }
   { print "Wel Come You"}
END { print " - Thank You -" }
```

Note: END statement execute after press ctrl+d.
We can execute a AWK script using interpreter:
$awk -f csi.awk → Execute a script
Here "-f" option specifies the AWK file containing the instructions.

When we execute AWK script then statements are executed again and again which are enclosed by curly braces between BEGIN and END patterns untill we press ctrl+d. Actually this will work as a looping statement. Only two types of data in AWK: numbers and strings of characters.

- Awk reads one line at a time and splits each line into fields.
- A field is a sequence of characters that doesn't contain any blanks or tabs.
- First field in current input line is called $1, the second $2, and so forth.
- The entire line is called $0.
- The number of fields can vary from line to line.

AWK has a number of built in variables:

- FS - Field separator
- NF - Number of fields in current record
- NR - Number of current record
- RS - Record separator
- $0 - Entire input record
- $n - nth field in current record

Examples:
$awk '{print $1, $3}'
$awk '{print $0}' OR $awk '{print}'
$awk '{print NF, $1, $NF}'
awk '{print $1, $2 * $3}' '{print NR, $0}'
Here is an example:
$awk -f ":" '/arun/ {print $1 " " $3}' tt.txt
-f → AWK executable
: → Field seperator
/arun/ → Pattern to search
tt.txt → The file to operate upon
{Print $1 " " $3} →Action to be perform on line if pattern is match.
Here $1 and $3 are columns of file tt.txt and awk is the command to search pattern /arun/ and perform action to print column1 and column3 from a file. A program snippet will speak better than my description:
For example, A file emp.dat contains name, pay rate, number of hours worked, one employee record per line.
$ awk '$3 > 0 { print $1, $2 * $3 }' emp.dat
This statement print the name and pay for everyone who worked more than zero hours. At the time of execution of AWK script, we can pass some values to the script. For example:
$awk -f add.awk 4 5
Here, value 4 treated as $1 column and Value 5 treated as $2 column.

2. SELECTION

Awk patterns are good for selecting interesting lines from the input for further processing.

2.1 Selection by Comparison

Task: A comparison pattern to select the records of employees who earn $5.00 or more per hour.
$awk '$2 >= 5' emp.data

2.2 Selection by Computation

Task: Print the pay of those employees whose total pay exceeds $50.
$awk '$2*$3 > 50 {printf("$%.2f for %s \n", $2*$3, $1)}'

2.3 Selection by Text Content

Task: Print all lines in which the first field is Susie
$awk '$1 = ="Jitendra"'

2.4 Combinations of Patterns

Patterns can be combined with parentheses and the logical operators &&, ||, and !, which stand for AND, OR, and NOT, respectively.
Task: Print lines where $2 is at least 4 or $3 is at least 20.
$awk '$2 >= 4 || $3 >= 20' emp.data

3. DATA VALIDATION

Awk is an excellent tool for checking that data has reasonable values and that it is in the right format.
Task: Use comparison patterns to apply five plausibility tests to each line of emp.data.
$awk 'NF !=3 {print $0, "number of fields is not equal to 3"}'
$awk '$2 <3.35 {print $0, "rate is below minimum wage"}'
$awk '$2 > 10 {print $0, "rate exceeds $10 per hour"}'
$awk '$3 <0 {print $0, "negative hours worked"}'
$awk '$3 >60 {print $0, "too many hours worked"}'

4. FANCIER OUTPUT

• print statement is meant for quick and easy output.
• use printf statement to format the output exactly the way you want it.
printf statement format is as follows:

printf (format, value1, value2, …, valuen)
where format is a string that contains text to be printed with specification of how each of the values is to be printed. A specification is a % followed by a few characters that control the format of a value. For example:
Example1: Use prinf to print the total pay for every employee-
$awk '{printf ("total pay for %s is $%.2f \n", $1, $2 * $3)}' emp.dat
Above example1,
- contains two % specifications
%s print the first value $1, as a string of characters
%.2f print the second value, $2*$3, as a number with 2 digits after the decimal point
- no blanks or new lines are produced automatically; you must create them yourself. Don't forget the \n.
Example2: Print each employee's name and pay-
$awk '{printf ("%-8s $%6.2f \n", $1, $2 * $3)}' emp.dat
Above example2,
- contains two % specifications
%-8s print the first value $1, as a left justify string of 8 characters with trailing left spaces
%6.2f print the second value, $2*$3, as a number with 6 digits before decimal point and 2 digits after the decimal point
Example3: Print all data for each employee, along with his or her pay, sorted in order of increasing pay.
$awk '{printf ("%6.2f %s \n", $2 * $3, $0)}' emp.data | sort
Above examp3,
- Pipes the output of awk into the sort command.

5. OPERATORS

Different operators are used in AWK programming.

5.1 Arithmetic Operator

+ Arithmetic Addition 3+4 7
- Arithmetic Subtraction 4-3 1
* Arithmetic Multiplication 4*3 12
/ Arithmetic Division 3/4 0.75
% Arithmetic Modulo 4%3 1
<space> String Concatenation 3 4 34

5.2 Unary Arithmetic Operator

The "+" and "-" operators can be used before variables and numbers. If X equals 4, then the statement: print -x; will print "-4."

5.3 Autoincrement, Autodecrement Operator

AWK also supports the "++" and "--" operators of C. Both increment or decrement the variables by one. The operator can only be used with a single variable, and can be before or after the variable. if X has the value of 3, then the AWK statement:
print x++, " ", ++x;
would print the numbers 3 and 5. These operators are also assignment operators, and can be used by themselves on a line:
x++; --y;

5.4 Assignment Operator

Variables can be assigned new values with the assignment operators. The other assignment statement is simply:
variable = arithmetic_expression
The statement x=1+2*3 4; is the same as
x = (1 + (2 * 3)) "4";
Both print out "74."
AWK, like C, has special assignment operators,
Operator Meaning
+= Add result to variable
-= Subtract result from variable
*= Multiply variable by result
/= Divide variable by result
%= Apply modulo to variable

5.5 Conditional Operator

Operator	Meaning
==	Is equal
!=	Is not equal to
>	Is greater than
>=	Is greater than or equal to
<	Is less than
<=	Is less than or equal to

5.6 Logical Operators

Operator	Meaning
AND	&&
OR	||
NOT	!

6. CONTROL-FLOW STATEMENTS

IF Statement

```
BEGIN {print "If statement start"}
{if ($1 > 1000)
gs=$1 + $2 + $3
print "Gross salary is",gs
}
END{print "finish"}
$awk –f posneg.awk 5000 30 400
Result: Gross salary is 5430
where $1 is 5000, $2 is 30 and $3 is 400
```

IF-ELSE Statement

```
BEGIN {print "If-Else statement start"}
{if ($1 > 0)
        print "it is positive number"
else
        print "it is negative number"
}
END{print "finish"}
$awk –f posneg.awk 5
Result: it is positive number (where $1 is 5)
```

7. LOOPING STATEMENTS

While Statement

```
BEGIN {print "While statement start"}
{ n=$1;
        while (n>0){
   r=n%10;
   print r;
   n=n/10;}
}
END{print "Finish"}
$awk –f rev.awk 123
Result: 321
```

For Statement

```
BEGIN {print "For statement start"}
{ sum=0;
        for(i=1;i<=10;i++)
        {sum=sum+i;}
}
```

END{print "Sum of 10 numbers is ",sum}
$awk –f rev.awk
Result: Sum of 10 numbers is 55

8. ARRAYS IN AWK

AWK has arrays with elements subscripted with numbers or strings (associative arrays) Arrays in AWK are associative. This means that each array is a collection of pairs: an index, and its corresponding array element value.

Example:

Element 1 value 2
Element 2 value "foo"
Element "cat" value "chicken"

Assign arrays in one of two ways:

- Name them in an assignment statement
  myArray[i]=n++
- Use the split() function
  n=split(input, words, " ")

The function "split" splits the string "input" into elements of array words[1], …, words[n], at each occurrence of blank space, " ", and returns n.

9. AWK FUNCTIONS

9.1 AWK Numerical Functions

Name	Function
cos	cosine
exp	Exponent
int	Integer
log	Logarithm
sin	Sine
sqrt	Square Root
delete	delete index
break	exit from loop
continue	continue loop

9.2 AWK String Functions

length(string)	Find length of a string
tolower(string)	Convert alphabets in lower case
toupper(string)	Convert alphabets in upper case
match(s,r)	Match string s in r, return position otherwise 0
>	Directs the output to file overwriting its previous contents.
>>	Appends the output to a file, preserving its previous contents.

AUTHOR'S PROFILE:

Jitendra Singh Kushwah did M.Tech. from RGPV Bhopal (M.P) and MCA from Jiwaji University, Gwalior (M.P) after B.Sc. Mathematics from Govt. P.G.college Morena (M.P). He has seven years teaching experience from academic and two years industry experience. He published a book entitled "Digital Principles" in Shree sai Publications, Meerut (U.P) and also published many research papers in international, national journals and conferences. He organized many national level workshops and conferences and also presented papers in conferences. He is also member of Computer Society of India (CSI).

Abhishek Chaturvedi did M.C.A from Guru Jambheshwar University of Science & Technology, Hisar (Haryana) in the year 2008, M.Sc.(cs) from M.C.R.P University Bhopal(M.P) in the year 2006 and B.C.A from M.C.R.P. University Bhopal(M.P.) in the year 2003. He has Four years teaching experience. He is also published many research papers in international, national journals and conferences. He organized many national level workshops and conferences and also presented papers in conferences.

Rajeev Sharma did M.C.A from RGTU Bhopal (M.P). He has Five years teaching experience. He is also published many research papers in international, national journals and conferences. He organized many national level workshops and conferences and also presented papers in conferences. He is also worked as a system administrator in The CITY college, Gwalior (M.P).

REFERENCES

[1] Arnold Robbins, "Effective awk Programmin", Third Edition, O'Reilly Media
[2] Alfred V. Aho, Brian W. Kernighan, Peter J. Weinberger, "The AWK Programming Language"
[3] Dale Dougherty, Arnold Robbins, "Sed & AWK", 2nd Edition, O'Reilly Media
[4] Arnold Robbins, "sed and awk Pocket Reference", 2nd Edition
[5] Tim Sherwood, "AWK: The Duct Tape of Computer Science Research"