Persistent

Bash Shell Scripting

AWK

Persistent University

Contents

AWK introduction

- AWK – pattern scanning and processing language
- Key features of AWK:
  - Text processing
  - Produce formatted text reports
  - Performs arithmetic operations
  - Perform string operations
- Comparison with Perl:
  - Awk is simple.
  - Awk syntax is far more regular.
  - Awk can be smaller, thus quicker to execute.
  - Like shell, awk variables doesn’t have $ in front of them.

AWK introduction

- Basic functions of AWK:
  - Search files for lines that contain certain patterns
  - When line matches, perform action on that line
  - Keep searching pattern till end of input files
- Awk programs are data-driven.
- Awk program:
  - *pattern* { *action* }
    - *pattern* { *action* }
    - ...
    - awk '*program input-file1 input-file2* ...
    - awk -f *program-file input-file1 input-file2* ...

Running AWK program

- Running long programs:

BEGIN

pattern {action}

pattern {action}

….

END

- $ **cat test.awk**
- BEGIN { print “Hello !!!!" }

- $ **awk -f test.awk**

- Executable awk program :
- $ **cat test.awk**
- #!/bin/awk -f
- BEGIN { print “Hello !!!!" }

- $ **test.awk**

BEGIN & END construct

- AWK program constructs -
- BEGIN - For processing to execute prior to reading input
- pattern - processing for input data
- END - for processing after end of input data
- **AWK Program :-**

BEGIN { initializations> }
<search pattern 1> {<program actions>}
<search pattern 2> {<program actions>}
...
END {<final actions>}

Comments in AWK program

- A comment is some text that is included in a program for the sake of readers.
- It is not really an executable part of the program. Comments can explain what the program does and how it works
  - #!/bin/awk –f
  - # This program prints a nice friendly message.
  - BEGIN { print “Hello !!!!" } # prints Hello
  - # This program ends here
- In one-shot throwaway program, don’t put an apostrophe in a comment
  - $ awk 'BEGIN { print "hello" } # let's be cute'

Variables in AWK

- AWK variables are dynamic
- User defined variable
  - $ awk 'BEGIN { l1="line one"; l2="line two"; print l1"\n"l2; }‘
  - line one
  - line two
- Positional variable:
  - $ awk 'BEGIN { for (i = 0; i < ARGC; i++) print ARGV[i] }' One Two count=3
  - awk
  - One
  - Two
  - count=3

Arithmetic operators

- Addition:

  $ awk 'BEGIN { a = 50; b = 20; print "(a + b) = ", (a + b) }‘

  (a + b) = 70

- Subtraction:

  $ awk 'BEGIN { a = 50; b = 20; print "(a - b) = ", (a - b) }‘

  (a - b) = 30

- Multiplication:

  $ awk 'BEGIN { a = 50; b = 20; print "(a * b) = ", (a * b) }‘

  (a * b) = 1000

- Division:

  $ awk 'BEGIN { a = 50; b = 20; print "(a / b) = ", (a / b) }'

  (a / b) = 2.5

Arithmetic operators

- Modulus

  $ awk 'BEGIN { a = 50; b = 20; print "(a % b) = ", (a % b) }‘

  (a % b) = 10

- Pre-increment, Post-increment :

  awk 'BEGIN { a = 10; b = ++a; printf "a = %d, b = %d\n", a, b }‘

  a = 11, b = 11

  $ awk 'BEGIN { a = 10; b = a++; printf "a = %d, b = %d\n", a, b }‘

  a = 11, b = 10

- Pre-decrement, Post-decrement :

  awk 'BEGIN { a = 10; b = --a; printf "a = %d, b = %d\n", a, b }‘

  a = 9, b = 9

  $ awk 'BEGIN { a = 10; b = a--; printf "a = %d, b = %d\n", a, b }‘

  a = 9, b = 10

- Short-hand operations : +=, -+, *=, /=, %=, ^=, **=

  $ awk 'BEGIN { cnt=10; cnt += 10; print "Counter =", cnt }'

Relational operators

- Equal to:
  - $ awk 'BEGIN { a = 10; b = 10; if (a == b) print "a == b" }‘
  - a == b
- Not equal to:
  - $ awk 'BEGIN { a = 10; b = 20; if (a != b) print "a != b" }‘
  - a != b
- Less than:
  - $ awk 'BEGIN { a = 10; b = 20; if (a < b) print "a < b" }‘
  - a < b
- Less than or equal to:
  - $ awk 'BEGIN { a = 10; b = 10; if (a <= b) print "a <= b" }‘
  - a <= b

Relational operators

- Greater than:
  - $ awk 'BEGIN { a = 10; b = 20; if (b > a) print "b > a" }‘
  - b > a
- Greater than or equal to:
  - $ awk 'BEGIN { a = 10; b = 10; if (a >= b) print "a >= b" }‘
  - b >= a

String concatenation operator in AWK

- Space in string concatenation operator which merge two strings. Below simple example illustrates this:
  - awk 'BEGIN { str1="Hello, "; str2="World"; str3 = str1 str2; print str3 }'
- On executing the above code, you get the following result:
  - Hello, World
- Similarly AWK supports below string functions –
  - index(string,search) Will search for specific characters inside a string
  - length(string) Calculates length of the string
  - split(string,array,separator) Used to split a string
  - substr(string,position) Used to extract a portion of a string.
  - substr(string,position,max) Used to extract a portion of a string

AWK Built in variables

- **FS** – The input field separator variable
- **OFS** – The output field separator variable
- **NF** – The number of fields variable
- **NR** – the number of records variable
- **RS** – The record separator variable
- **ORS** – The output record separator variable
- **FILENAME** - The current filename variable

Quiz

- **AWK** is an interpreted programming language designed for _____and typically used as a data extraction and reporting.
- When AWK search files for lines that contain certain patterns and when line matches it performs _____
- A _____ is some text that is included in a program for the sake of readers.
- AWK variables are **dynamic**
- State TRUE or FALSE –
- AWK has arithmetic and string operators

Quiz Answers

- AWK is an interpreted programming language designed for ***text processing*** and typically used as a data extraction and reporting.
- When AWK search files for lines that contain certain patterns and when line matches it performs ***action***
- A ***comment*** is some text that is included in a program for the sake of readers.
- AWK variables are ***dynamic***
- State **TRUE** or FALSE –
- AWK has arithmetic and string operators

Summary

- With this we have come to an end of this session, where we discussed about Grep and Regular Expressions
- Now, you should be able to answer following questions:
  - What is AWK?.
  - The syntax of AWK and ways to run AWK programs.
  - Standard numeric and string functions used in AWK.
  - What are AWK built in variables?.

Reference Material

- http://www.tutorialspoint.com/awk/
- http://www.grymoire.com/Unix/Awk.html
- http://www.thegeekstuff.com/2010/01/awk-introduction-tutorial-7-awk-print-examples/

Key contacts

- **Persistent Interactive**

  Bhimashankar Gavkare

  bhimashankar_gavkare@persistent.com

Persistent

Persistent

Thank You !!!

Persistent University