Chapter 05 – AWK Scripting

Introduction

- What is AWK?
  - Interpreted Language
  - Really awesome for
    - Text processing
    - Generating reports
    - Performing string/arithmetic ops

Introduction

- Why learn AWK?
  - Save you a bunch of time when analyzing large text files
  - When simple grep or searching isn't enough
  - Extremely useful when you have table-like text file (columns separated by space, commas, etc.)
- Variants of AWK – AWK, NAWK, GAWK

Running AWK

- First let's create a sample text file to process:
  - Wayne Batman Gotham 123456
    Diana WonderWoman Themyscira 67890
    Kent Superman Metropolis 567364
    Allen Flash Central 768392
    Jordan GreenLantern Coast 61234

Running AWK – Command Line

Running AWK from the command line

Simply put the awk script within quotes and specify the text file to process

```
awk '{print}' sample
```

This simple script {print} will print each line of sample

Running AWK – Script

Running AWK from a script

Create a script called myawk.awk

Put the script {print} in this file

Run the script with

awk -f myawk.awk sample

Notice the -f option passed to awk in this case

Running AWK – Script (#!)

Add the #! line in myawk.awk

```
#!/bin/awk -f
{print}
```

chmod 700 myawk.awk
Run the script with

```
./myawk.awk sample
```

Notice the -f option passed to #! in this case

AWK Syntax

- How to think about an AWK script?
- Think of AWK scripts as an implicit loop
- AWK automatically goes through the text file, line-by-line and executes the statements within {} for each line
- Example

  {print}
- Executes print one-by-one for each line of text file

AWK Syntax

- The print syntax

  print – will print the entire line

  print “ “ - will print the string in “ ”

  print $var – will print the value of $var

  print “mysting” $var – will join “mystring” and value of $var, and print it

AWK Syntax – Built-in Variables

- Built-in variables

  $1, $2, $3, … - variables for columns

  $0 – variable with the entire line

  FS – holds the field separator

- Try:

```
{print $0}
{print $1}
{print “A Line “ $0}
```

AWK Syntax – Pattern Match

- One of the coolest features of AWK
- Before the commands for all lines of text, specify a search term (pattern)
- The commands are performed only for those lines where the search term exists somewhere in them
- Example

  ```
  /man/{print $3}
  ```

  will only print the third column of lines where the string man appears anywhere in the line

AWK Syntax – BEGIN and END

- Optional blocks of code
- Statements in BEGIN are executed once before starting to process the text file
  Statements in END are executed once after starting to process the text file
- Overall structure
  BEGIN{ AWK statements to execute once at beginning }
  { AWK statements to execute once per line }
  END{ AWK statements to execute once before ending}

AWK Syntax – BEGIN and END

- Example:

```
BEGIN{print “Name Alias City SSN”}
/man/{print}
END{print “ ----------------------”}
```

AWK Syntax – Arithmetic

- =, +, -, *, /, %, ++, -- are all supported

- Example:

```
/man/{a=10; b=20; c=a+b; print c $2}
```

AWK Syntax – Accumulators, etc.

- Arithmetic in tandem with BEGIN and END lets us do some cool things!
- BEGIN {count=0}
  /man/{count++; print $0}
  END {print count}

Moving Forward

- This was your introduction to AWK
- AWK has other cool features such as conditional statements, functions, etc.
- Continue practicing AWK and you will find daily instances where AWK comes to your rescue