IT 240
Shell Scripting for Administrators

Chapter 7
Processing Text with AWK

Stan J. Senesy
IT Program/CCS
New Jersey Institute of Technology

awk

- Similar in function to sed, awk is used to perform text processing on data, either through a pipe or file operation
- *awk* was named after the initials of it's creators: Aho, Weinberger & Kernighan
- Other versions of awk exist including:
  - *gawk* (gnu awk)
  - *nawk* (new awk)
  - *oawk* (old awk)

awk Basics

- Finding out your awk version:
  - *awk --version*
  - *awk -W versions*
- *awk* looks through the text supplied, looking for similar patterns to what was specified in the search and performs an action on the text
- Both *sed* and *awk* use regular expressions to process text
- One of the principal differences, is that the action in *awk* is enclosed in braces { }

Running *awk* scripts

- We can run scripts as before by entering an awk command:
  - *$awk '{ print "Hi Mom" }'*
- Note the use of punctuation and spacing
- Nothing will happen until we press the enter key as we did not specify a file to process

Running *awk* Scripts

- We can also convert our command line script to a file:
  - { *print "Hi Mom"* }
- Which we'll save as *hello.awk*
- The script may be executed by:
  - *$ awk -f hello.awk*

Creating An *awk* Program

- We can convert the previous scripts into a self contained program:
  - *#! /usr/bin/awk -f*
  - { *print "Hi Mom"* }
- Once we've set the execute bit, we can run the program using ./

awk Commands

- *print*
    - Print allows us to output a subset of the data in a file based on some search criteria
    - For instance, specifying { *print $0* } prints all the columns of data, but { *print $5, $6* } will print only the 5th and 6th columns of data in a file

awk Commands

- *print*
  - We can use print to mix string data with file data:
    - *awk ' { print "This is a string",$1,$2 } '*
  - This will print the text "This is a string" followed by the contents of the first and second columns for each row of data in an input file

awk Commands

- Field Separators
  - By default, awk uses the space char as a field separator
- This behavior may be changed with the -F flag to specify a new field separator
  - *# changes separator to a comma*
  - *$awk -F,*

awk Commands

- *printf*
  - The printf command produces formatted output
  - It allows you to embed file data easily in the output by specifying the data type
  - Page 242-3

awk Commands

- *sprintf*
    - Almost identical to *printf*, *sprintf* allows you to assign the output of the print statement to a variable
    - This has potential uses if you wish to ‘store’ the print output for use later
    - The variable can be printed by simply using the *print* command

Variables

- Variables in awk have a few naming rules:
  - They must not start with a digit
  - The are case sensitive
  - The name may consist of alphanumeric characters and the underscore symbol
  - The variable name must not be a reserved word
- Variable definitions should be placed in the *BEGIN* block of your script

Built-in Variables

- *FS*
  - This variable holds the current field separator
- *NR*
  - This variable increments automatically each time a line of data is processed
- Page 248

Control Statements

- These statements control the flow of execution of your script
  - *if then-action else-action*
    - Allows action based upon a test condition
    - Comparison operators are shown on page 250

Arithmetic Functions

- awk is able to perform all of the standard arithmetic operations that are typically used with numbers
- These functions include:
  - + - * / ++ --
- awk also supports assignment shortcuts:
  - *myvar* += *1*
- Some of the comparison operators do double duty with output redirection

More Control Structures

- *while* loops
  - The *while* loop allows continuous execution until an exit condition is met
  - The loop is specified by a condition and an action:
    - *awk ‘BEGIN { while (++myvar <= 10) print myvar }’*

More Control Structures

- *for* loops
  - *for* loops are defined by an entry condition, exit condition and increment action
    - *awk ‘BEGIN { for (myvar = 1; myvar <= 10; myvar++) print myvar }’*

Functions

- There are a number of functions built into awk to make programming a bit easier
- Functions are used in the same manner as other programming languages like c++ and java
- A partial listing of functions are shown on page 255