CSI6203 Scripting Languages

Module 9

AWK

Contents

- History of AWK
- Using AWK for text manipulation
- AWK Variables
- AWK conditional statements

Learning Objectives

After finishing this module, you should be able to:

- Understand and execute scripts that use AWK
- Process streams using either sed or AWK or a combination of the two

Introduction to AWK

AWK

- awk is a command that, similar to sed, is designed to manipulate text streams

- Unlike sed, awk is actually an entire programming language built around this concept

AWK History

- AWK was first developed by bell labs in the 1970s
- It was named after the three programmers who originally designed it Alfred Aho, Peter Weinberger and Brian Kernighan (AWK)
- Over the years there have been many different implementations of awk, but the two most common forms in use today are:
  - gawk (used in linux based systems)
  - BWK (used in bsd and MacOS based systems)

Awk as a programming language

- Although awk is often used as a stream editor in bash scripts, it is a complete language in its own right.

```
#!/bin/bash
awk 'BEGIN {print "hello world"}'
```

```
Hello World
```

Using AWK

Processing text from files

- Awk can be used to filter content from files in a powerful and customisable way

```
#!/bin/bash
awk '{ print $0 }' info.txt
```

- Within the awk command, $0 refers to the entire line of text
- $1 $2 $3 etc. refer to each field (usually a word) of the text from left to right

Processing text from files

Info.txt

```
name: robot price: $2000
name: computer price: $1500
name: disk price: $50
```

- This can be used to parse specific input from each line and format it

```
awk '{ print "the price for", $2, "is", $4 }' ./info.txt
```

```
The price for a robot is $2000
The price for a computer is $1500
The price for a disk is $50
```

Field Separators

- By default, the fields $1 and $2 etc. are delimited by a space as in the previous example.
- However, the separator can be set to any delimiter/separator required. (similar to setting the $IFS variable)
- The field separator is often set in the “BEGIN” portion of an awk command to ensure it occurs before processing the text

Processing text from files

Info.csv

```
robot,$2000
computer,$1500
disk,$50
```

```
awk 'BEGIN {FS=","}
{ print "the price for", $1, "is", $2 }' ./info.csv
```

```
The price for a robot is $2000
The price for a computer is $1500
The price for a disk is $50
```

Awk code blocks

- BEGIN blocks happen once at the start of the awk command
- Then the main body block happens for each line of the text
- END blocks happen once at the end
- Each command within a block can be separated by a ;

Processing text from files

Info.csv

```
robot,$2000
computer,$1500
disk,$50
```

```
awk 'BEGIN {FS=","; print "outputting prices"}
{ print "the price for", $1, "is", $2 }
END{print "goodbye!"}' ./info.csv
```

```
Outputting prices
The price for a robot is $2000
The price for a computer is $1500
The price for a disk is $50
Goodbye
```

AWK Variables

AWK Variables

- Awk has many built in variables that can be useful
- $0 $1 $2 $3 for each field we have already learned
- FS is the field separator
  - default: “ ” to separate words
- RS is the record separator
  - default: “\n” to separate lines
- FILENAME is the name of the file being processed
- NR is the total number of records in the file

Record Separators

- By default, each record processed by awk is separated by a new line
- However, the separator can be set to any delimiter/separator required.
- The record separator is often also set in the “BEGIN” portion of an awk command to ensure it occurs before processing the text

Processing text from files

Info.txt

```
robot
$2000

computer
$1500

disk
$50
```

```
awk 'BEGIN {FS="\n"; RS=""}
{ print "the price for", $1, "is", $2 }
END{print "goodbye!"}' ./info.csv
```

```
The price for a robot is $2000
The price for a computer is $1500
The price for a disk is $50
Goodbye
```

User defined variables

- Just like in bash, awk can make use of variables

```
awk 'BEGIN {FS=","; currency="$" }
{ print "the price for", $1, "is", currency $2 }
END{print "goodbye!"}' ./info.txt
```

- Unlike in bash, user defined variables do not need to have a $ prepended to them.

AWK Formatting

printf

- The print command in awk works well for most situations, however, it doesn't have much functionality for more complex formatting
- Printf can be used to add more advanced formatting through the use of format codes

```
awk '{ printf "the result is %.2f", $1 }'
./results.txt
```

printf format codes

Code	Result
%s	String
%5s	String with a minimum of 5 characters
%f	Number (floating point)
%5f	Number with a minimum of 5 characters
%5.2f	Number with a minimum of 5 characters and 2 decimal places
%d	Whole number (Decimal Integer)
%c	Single Character

Processing text from files using printf

Info.csv

```
robot,$2000
computer,$1500
disk,$50
```

```
awk 'BEGIN {FS=","; currency="$"; print "Name | Price" }
{ printf "%-10s| %c%07.2f\n", $1, currency, $2 }
END{printf "there are %d items total!\n", NR}' ./info.txt
```

```
Name      | Price
robot     |$2000.00
computer  |$1500.00
disk      |$0050.00
there are 3 items total!
```

Summary

- Terms to review and know include:
  - awk
  - formatting
  - parsing
  - fields
  - records
  - printf

References and Further Reading

- Ebrahim, M. and Mallet, A. (2018) Mastering Linux Based Scripting (2nd Ed) Chapter 10, pp 175-193
- http://gnu.org/software/gawk/manual/gawk
- https://likegeeks.com/awk-command