Introduction to Sed and Awk scripting

Andy Spencer

February 2, 2013

San Fernando Valley Linux Users Group

One tool for one job?

Those days are dead and gone and the eulogy was delivered by ~~Perl~~ Python.” -Rob Pike

When to use sed and awk?

When to use sed?

- When you are “editing” a file
- Short programs 1-5 lines

When to use awk?

- When you need to “summarize” a file
- Slightly longer programs 5-50 lines

When not to use them?

- For longer program, use a programming language
  - Awk lacks type checking, etc
- For some very simple things
  - grep, tr, seq
  - Unix shell expansion ${var/from/to}
  - Fancy editor commands (vim, emacs)

Sed - simple commands

Example

commands

s	regular expression find/replace
y	translate characters (like tr)
p, P	print out the current line
i, a	insert or append text
c	change line
n, N	read the next line of input
q, Q	exit/quit
r, R, w, W	read/write files
b, t, T	branch to label
d, D	delete pattern space
h, H, g, G, x	work with the pattern and hold spaces

addresses and ranges

number	match a line number
$	the last line
/rege/	match a regular expression (not the same as s///)
first step	match every step'th line starting with first
0,addr	match every step'th line starting with first
addr,+N	address addr and N following lines
addr, N	match addr and lines up to a multiple of N

branching

- Can be used for loops
- Skipping when a pattern fails
- I rarely use these

the hold space

- Can be used for loops
- Skipping when a pattern fails
- Can be used like a variable
- Also rarely used

fizz buzz

Example

Awk - simple commands

Example

Command structure

Basic awk scripts consist of patterns and actions
When a pattern matches, the action is executed
Some special patterns: BEGIN, END, BEGINFILE, ENDFILE

```
PATTERN  {  ACTION  }
PATTERN  {  ACTION  }
```

Records and fields

Text is split twice when reading the file
Patterns are matched against records
Fields can be accessed using $N

Records by default, a line, changed using RS

Fields by default, a word ina line, changed using FS

Command line arguments

- -v Set a variable (var=val)
- -F Change field separator, can also be done from BEGIN

```
awk -v 'var=$VAR' '{ script }
```

Awk quirks

- default values are 0, no need for declaration
- string concatonation is implicit

vpaste examples - get param

Example

vpaste examples - cut file

Example

vpaste examples - stat

Example

vpaste examples - head

Example

vpaste examples - cowlife

Example

awk as a programming language

- built-in functions
- user defined functions
- variables
- file i/o
- networking

Variables and arrays

Statements

- if (condition) body [else body]
- while (condition) body
- for (i=0; i<10; i++) body
- for (i in array) body
- break, continue
- delete var, delete array[index]
- swtich (expr) case value: body; default: body

Builtin functions

Math	atan2, cos, exp, int, log, rand, sin, sqrt, srand
Strings	asort, gsub, length, match, split, sprintf
Time	mktime, strftime, systime
Bitwise	add, or, xor, etc

I/O and networking

- next, nextfile, print, printf
- system, fflush

Reading from files and commands

- getline [var] [<file]
- command | getline [var]
- command |& getline [var]

Writing to files and commands

- print .. > > file
- print .. | command
- print .. |& command

User defined functions

```
function sayhi(name,          str) {
    str = "Hello, " name "!'
    print str
}
```

GNU extensions

Example