Ruby als sed- und awk-Ersatz

Knut Franke

Wie man aus Quoting-Hölle und Regexp-Dschungel entkommt

Motivation

- **Unterschiede grep/egrep/sed/awk Regexp Syntax**
- **unregelmäßige Syntax bash/grep/sed/awk/find/ (Perl ist besser, aber auch nicht immer intuitiv)**
- **Quoting Hell**
- **begrenzter Funktionsumfang der Standard Unix-Tools**

Ruby

- **Entwickelt ab 1993 von Yukihiro “Matz” Matsumoto**
- **Bekannt geworden durch Ruby on Rails (2004)**
- **Konsequent objektorientierte Skriptsprache**
- **“help every programmer in the world to be productive, and to enjoy programming, and to be happy”**

Aufruf

- `ruby -ne` verarbeitet auf der Kommandozeile genannte Dateien oder STDIN zeilenweise, ähnlich wie `grep` oder `sed -n`
- `ruby -pe` wie `ruby -ne`, aber gibt die verarbeiteten Zeilen automatisch aus (ähnlich wie `sed` ohne die Option `-n`)
- `ruby -ane` bzw. `ruby -ape` trennt jede Zeile in Felder (ähnlich wie `awk`), die über das Array `$F` zur Verfügung stehen

sed-Ersatz

`sed 's/pattern/string/'`	`ruby -pe '$_.sub! /pattern/, “string”'`
`sed 's/pattern/string/g'`	`ruby -pe '$_.gsub! /pattern/, “string”'`
`sed '/rocks/s/Ruby/Linux/'`	`ruby -pe '$_.gsub! /Ruby/, “Linux” if /rocks/' input.txt`

awk-Ersatz

`df | awk '`
`BEGIN { sum = 0 }`
`/^\/dev\// { sum += $2 }`
`END { print sum/1024/1024 }'`
	`df | ruby -nae '`
	`BEGIN { $sum = 0 }`
	`$sum += $F[1].to_i if %r(^/dev/)`
	`END { puts $sum/1024**2 }'`

`awk '{ print FILENAME ":" FNR "\t" }' input1.txt input2.txt`	`ruby -pe 'print $<.filename, ":", $<.file.lineno, "\t"' input1.txt input2.txt`

`awk '{ counts[$0] += 1 }`
`END { for (v in counts) print v }'`
	`ruby -ne '`
	`BEGIN { $counts = Hash.new(0) }`
	`$counts[$_] += 1`
	`END { puts $counts.keys }'`

Auswertung CSV-Daten

```
dept1,host1,0,0,100.0,40.0,80.0,0,0
dept2,host2,100,100,100,0,100,100,0


ruby -F, -ane '
    availability = $F[2..-1].inject(0.0) {|a,x| a + x.to_f} \
        / ($F.size - 2)
    puts "#{$F[1]} #{availability.round}%" \
        if %w(dept1 dept3).include? $F[0] and \
        availability < 80.0
' report.csv
```

Warum Ruby?

- **skaliert von Einzeilern bis >700,000 SLOC (Metasploit)**
- **/Regexp/-Syntax, `Backtick`-Notation**
- **übersichtliche Syntax (alles ist ein Objekt/eine Methode, Code-Blöcke, Statements sind Ausdrücke)**
- **Attribut-Zugriff *standardmäßig* über Getter/Setter**

Danke fürs Zuhören!