Μια σύντομη εισαγωγή στις κανονικές εκφράσεις

Βασίλειος Καρακόιδας

bkarak@aueb.gr

Φεβρουάριος 2004

Περιεχόμενα

- Γενικά περί κανονικών εκφράσεων
- Κατηγορίες κανονικών εκφράσεων
- Συντακτικοί κανόνες
- Ένα ολοκληρωμένο παράδειγμα

Γενικά περί κανονικών εκφράσεων

- [a-zA-Z]*(abra|cadabra)+[0-9]?
- Χρησιμοποιούνται για την έρευση προτύπων σε κείμενα
- Swiss Army Knife

Που υπάρχουν;

- awk, vim, emacs και γενικότερα σε πολλές κλασσικές εφάρμογές των Unix
- Java - JDK 1.4.x, Jakarta και ORO
- Perl, Python, PHP, C/C++
- Compilers
- Apache (mod_rewrite)
- MySQL (SQL select)

Δηλαδή ;

```
import java.util.regex.*;
[...]

public void checkIPAddress(){
    String data = "193.92.177.2";

    Pattern pat = Pattern.compile("[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+");

    Matcher mat = pat.matcher(data);

    if(mat.matches()){
        System.out.println(data + " is a valid IP address");
    }
}
```

Import the regex library

The input data

Initialize the Pattern Object

Initialize the matcher Object

Perform a full length match

Πως είπατε;

```
#!/usr/bin/perl

while(</cdrom/*rpm>) {

    if(/(.*-[0-9a-zA-Z\.]*)-[0-9]*.*\.i386\.rpm/){
        print $1." // ".$_."\n";
    }
}
```

For all the rpm files in /cdrom

Perform a match with the regex

Κατηγορίες κανονικών εκφράσεων

- Traditional Unix Regular Expressions
- POSIX modern Regular Expressions
- Perl Compatible Regular Expressions (PCRE)

Traditional Unix & POSIX

- Οι Traditional Unix Regular Expressions εμφανίστηκαν σε μια υλοποίηση του Ken Thompson των κειμενογράφων qed και ed
- Υπάρχουν επίσης στις εφαρμογές expr, grep, emacs, vim, lex κτλ
- Το πρότυπο POSIX επέκτεινε το συντακτικό με επιπλέον τελεστές (?,+,|), καθώς και με κλάσεις χαρακτήρων

Perl Compatible Regular Expressions (PCRE)

- Practical Exctraction and Report Language
- Αποτελεί μέρος του συντακτικού της γλώσσας (τελεστές, ειδικές μεταβλητές)
- Επέκταση του συντακτικού του POSIX
- Ταίριασμα, π.χ. `m/[a-zA-Z0-9_]*/` ή `m/\w*/`
- Αντικατάσταση, π.χ. `s/abra/cadabra/`

Συντακτικοί κανόνες (POSIX)

`.`	Οποιοσδήποτε χαρακτήρας εκτός του newline	`a.*b`
`^`	Αρχή γραμμής	`^abc`
`$`	Τέλος γραμμής	`abc$`
`[s]`	Όποιοσδήποτε χαρακτήρας στο s	`[abc]`
`[^s]`	Όποιοσδήποτε χαρακτήρας εκτός s	`[^abc]`
`r*`	Μηδέν ή παραπάνω από r	`a*`
`r+`	Ένα ή παραπάνω από r	`a+`
`r?`	Μηδέν ή ένα από r	`a?`
`r{m,n}`	Από m εώς n εμφανίσεις του r	`a{1,5}`
r_1r_2	r_1 ακολουθώντας το r_2	`ab`
$r_1\|r_2$	r_1 ή r_2	$a\|b$
`(r)`	`r`	$(a\|b)$
r_1/r_2	r_1 ακολουθούμενο από το r_2	`abc/123`

Παραδείγματα

- a*b → b, ab, aab, aaab κτλ.
- [a-zA-Z0-9 -]+ → πχ Hawaii 5 - 0
- [0-9]{1,3} → Ένα ως τρείς αριθμούς
- [1-9]{1} poulakia? kath(e|on)tai → Αναζητεί όλες τις συμβολοσειρές 1 poulaki kathetai . . . 9 poulakia kathontai

Ένα πλήρες παράδειγμα

- Έστω ότι θέλουμε να ανιχνεύσουμε όλες τις IP διευθύνσεις σε ένα κείμενο
- Μορφή των διευθύνσεων:
  $0 \rightarrow 255.0 \rightarrow 255.0 \rightarrow 255.0 \rightarrow 255$

Ένα πλήρες παράδειγμα (2)

- $1^{η}$ προσέγγιση: [0-9]+\.[0-9]+\.[0-9]+\.[0-9]+
- Η IP *2394029402.20349.329403.3498432* θεωρείται σωστή

Ένα πλήρες παράδειγμα (3)

- $2^{η}$ προσέγγιση:
  [12]?[0-9]{0,2}\.[12]?[0-9]{0,2}\.[12]?[0-9]{0,2}\.[12]?[0-9]{0,2}

- Η IP 299.299.299.299 θεωρείται σωστή

- $3^{η}$ προσέγγιση:
  ([12]?[0-9]{0,2}){3}[12]?[0-9]{0,2}

Ένα πλήρες παράδειγμα (4)

Ο Θείος J.E.F Friedl (Mastering Regular Expressions 2^{nd} Edition) προτείνει:

```
([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.
([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.
([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5])\.
([01]?[0-9][0-9]?|2[0-4][0-9]|25[0-5])
```

Αναφορές

- http://jakarta.apache.org/ (Regular Expression Engine)
- http://java.sun.com/ (Regular Expression Engine)
- http://regex.info/ (Mastering Regular Expression official site)
- http://tusker.disorder.com.au/ (Benchmarking for Java)
- http://en2.wikipedia.org/wiki/Regular_Expressions (Wiki Encyclopedia)

Αναφόρες (2)

- Perl Regular Expressions perlre, perltut, perlquick, perlre, perlfaq6 (man pages)
- http://www.gnu.org/directory/text/
- http://www.pcre.org/