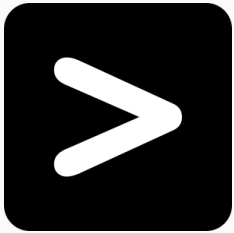
Einführung in Reguläre Ausdrücke

Suchen und Ersetzen für Fortgeschrittene



Dirk Deimeke

11. August 2019

My own IT @ FrOSCon

Dirk Deimeke – d5e.org

Wer weiss gar nicht, was reguläre Ausdrücke sind?

Wer benutzt schon reguläre Ausdrücke?

Wer benutzt einen Texteditor,
der reguläre Ausdrücke beherrscht?

Welchen Editor?

https://twitter.com/garabatokid/status/1147063121678389253

Allgemeines

Für viele sind reguläre Ausdrücke ein Buch mit sieben Siegeln.

Dabei sind sie eines der besten Werkzeuge, um effiziente Abfragen zu erstellen.

Regeln

Bei regulären Ausdrücken, die oft auch als *regular expression* – kurz *Regex* oder *RegExp*, englisch für *regulärer Ausdruck* bezeichnet werden, handelt es sich um **syntaktische Regeln** zur Beschreibung von Mengen oder Untermengen einer Zeichenkette.

Regeln? Wofür?

Mit diesen Regeln können Zeichenketten extrahiert werden (Mustersuche oder *Pattern Matching*), auch wenn deren genaue Abfolge nicht bekannt ist.

Ein weiterer Anwendungszweck ist das Suchen und Ersetzen, das mit regulären Ausdrücken ebenfalls vereinfacht bzw. um viele Anwendungsfälle erweitert werden kann.

Unterschiedliche Implementierungen

Viele Programme verwenden eigene Implementierungen von regulären Ausdrücken, die sich in Umfang und Syntax unterscheiden.

Die folgenden drei großen decken fast alle zu findenden Implementierungen ab:

- **Basic Regular Expressions (BRE)**
  Regex nach dem *POSIX*-Standard.
- **Extended Regular Expressions (ERE)**
  Im *POSIX*-Standard definiert: *Regex* mit erweitertem Funktionsumfang.
- **Perl Compatible Regular Expressions (PCRE)**
  Angelehnt an die Implementierung von *Regex* in der Programmiersprache *Perl*.

Beispiele

Vim benutzt Basic Regular Expressions.

egrep benutzt Extended Regular Expressions.

Perl aber auch **PHP** nutzen Perl Compatible Regular Expressions.

One ring …

grep beherscht alle drei Varianten je nach Aufruf:

- **-G, - -basic-regexp**
- **-E, - -extended-regexp**
- **-P, - -perl-regexp**

Die Implementierungen unterscheiden sich vor allem in der Syntax.

Zeichen und Mengen

BRE	ERE	PCRE	Bedeutung
`a`	`a`	`a`	das einzelnen Zeichen »a«
`.`	`.`	`.`	ein beliebiges Zeichen
`z*`	`z*`	`z*`	beliebig oft »z«, auch **kein** Vorkommen
`x\+`	`x+`	`x+`	mindestens einmal »x«
`y\?`	`y?`	`y?`	einmal oder keinmal »y«
`a\{6\}`	`a{6}`	`a{6}`	genau sechsmal »a«
`a\{3,\}`	`a{3,}`	`a{3,}`	mindestens dreimal »a«
`a\{,5\}`	`a{,5}`	`a{,5}`	höchstens fünfmal »a«
`a\{2,4\}`	`a{2,4}`	`a{2,4}`	zwei- bis viermal »a«

Zeichenbereiche

BRE	ERE	PCRE	Bedeutung
`ab`	`ab`	`ab`	die Zeichenkette »ab« in genau dieser Reihenfolge
`[abc]`	`[abc]`	`[abc]`	ein einzelnes »a«, »b« oder »c«
`[a-z]`	`[a-z]`	`[a-z]`	ein beliebiges Zeichen zwischen »a« und »z«
`[^ab]`	`[^ab]`	`[^ab]`	ein beliebiges Zeichen, ohne »a« und ohne »b«

Charakterklassen

BRE	ERE	PCRE	Bedeutung
`\w`	`\w`	`\w`	alphanumerische Zeichen und Unterstrich (Wortzeichen)
`\W`	`\W`	`\W`	kein Wortzeichen
`\d`	`\d`	`\d`	eine Ziffer
`\D`	`\D`	`\D`	keine Ziffer
`\s`	`\s`	`\s`	Leerzeichen (*Whitespace*)
`\S`	`\S`	`\S`	kein Leerzeichen

Rückbezüge und Positionen

BRE	ERE	PCRE	Bedeutung
`\(...\)`	`(...)`	`(...)`	Gruppierung von Ausdrücken für Rückbezüge
`\1 \2`	`\1 \2`	`\1 \2`	Rückbezug auf Gruppe 1 und 2
`a|b`	`a|b`	`a|b`	entweder »a« oder »b«
`^`	`^`	`^`	Zeilenanfang
`$`	`$`	`$`	Zeilenende
`\<`	`\<`	`\<`	Anfang eines Worts
`\>`	`\>`	`\>`	Wortende

Aufgabe »Meier«

In einem Text sollen alle Menschen, die man »Meier« ruft (!), gefunden werden, das heisst, alle Schreibweisen sind zulässig:

- Meier
- Meyer
- Maier
- Mayer

Wie sieht der reguläre Ausdruck dazu aus?

Bonusfrage: Wie sieht der Ausdruck aus, wenn das »e« optional ist?

Aufgabe Rückbezug

Ein zentrales Element von regulären Ausdrücken sind Rückbezüge. Mit diesen können nicht nur Zeichenketten extrahiert, ein Suchen und Ersetzen durchgeführt, sondern auch Vergleiche vorgenommen werden.

Aufgabe:

> *Sucht nach Begriffen, die aus fünf Buchstaben bestehen und von vorn und hinten gelesen gleich aussehen, wie beispielsweise »rotor«, »maoam«, »kayak«, …*

Wir nehmen an, dass alle Wörter in Kleinbuchstaben geschrieben sind.

Für den Test: `https://www.deimeke.net/transfer/linux.words`
(Datei aus dem Paket »words« in Fedora).

ERE vs. PCRE

Je nach Art des regulären Ausdrucks ist grep mit Extended Regular Expressions um den Faktor 2 bis 10 langsamer als mit Perl Compatible Regular Expressions.

```
$ time grep -P '^(.)(.).\2\1$' /usr/share/dict/linux.words
...
real    0m0.047s
user    0m0.041s
sys     0m0.006s

$ time grep -E '^(.)(.).\2\1$' /usr/share/dict/linux.words
...
real    0m0.373s
user    0m0.368s
sys     0m0.005s
```

Reguläre Ausdrücke in der Bash

Auch Linux-Shells, etwa die *bash* ab Version 3, verstehen reguläre Ausdrücke.

```
#!/bin/bash
FILENAME=dh-20091005-ausgabe-006.ogg
REGEX='^dh-([0-9]{4})([0-9]{2})([0-9]{2})-ausgabe-([0-9]{3}).ogg$'

if [[ $FILENAME =~ $REGEX ]]
then
        jahr=${BASH_REMATCH[1]}
        monat=${BASH_REMATCH[2]}
        tag=${BASH_REMATCH[3]}
        episode=${BASH_REMATCH[4]}
fi
echo "Jahr:␣$jahr,␣Monat:␣$monat,␣Tag:␣$tag,␣Episode:␣$episode"
```

Frage: Um welche Art regulären Ausdruck handelt es sich? Was lässt sich verbessern?

”Für jedes Problem gibt es eine Lösung,
die einfach, klar und falsch ist.”

Henry Louis Mencken im Jahr 1921

(amerikanischer Autor, Satiriker und Journalist)

Die »Gier«

Reguläre Ausdrücke sind gierig, sie entsprechen dem maximalen Ergebnis, auf das das Suchmuster zutrifft.

Im Englischen spricht man von »Greediness«.

Empfehlungen

- Reguläre Ausdrücke so genau wie irgendmöglich formulieren.
- Hohe Komplexität des Ausdrucks führt zu schlechter Wartbarkeit.
- Alleine aus Gründen der Lesbarkeit, BREs nur da einsetzen, wo es nicht anders geht.

Wider den Zwang

Es ist *fast* alles mit regulären Ausdrücken möglich.

Vieles wird durch den Zwang, einen regulären Ausdruck verwenden zu wollen, aber unnötig kompliziert.

Wenn man ein Hammer ist,
sieht jedes Problem aus wie ein Nagel.

Quelle leider unbekannt.

Suche nach IP-Adressen

Aus einem längeren Text sollen alle gültigen IP-Adressen herausgesucht werden.

IP-Adressen folgen dem folgenden Muster:

a.b.c.d – wobei jeder Buchstabe für eine Zahl zwischen 0 und 255 steht.

Wie sieht der reguläre Ausdruck aus?

Validierung von E-Mail-Adressen

RFC 5322 definiert, wie eine E-Mail-Adresse aussehen darf.

`https://www.ietf.org/rfc/rfc5322.txt`

Wie sieht der reguläre Ausdruck aus?

Eine einfache Websuche

- Four Regular Expressions to Check Email Addresses
- Email Address Regular Expression That 99.99% Works. Disagree?
- regex - How to validate an email address using a regular expression?
- Stop Validating Email Addresses With Regex

Ebenfalls komplex

- Zerlegen von CSV
- Zerlegen von XML (oder HTML)
- Zerlegen von JSON

Suchen und Ersetzen mit sed

In einem Text sind verschiedene Zeilen, in denen Begriffe durch » und « getrennt sind.

»Kegel und Kind« oder »Spiele und Brot«.

Mit welchem sed-Kommando kann man das umdrehen?

```
sed -E 's/Suchbegriff/Ersetzung/'
```
(das -E steht für ERE)

Kleinere Tipps mit sed

Das Zeichen nach dem »s« wird für den Trenner zwischen Suchen und Ersetzen benutzt.

`-i` verändert die Datei »inplace«

`-i.bak` macht das gleiche und benennt die Ursprungsdatei in `Dateiname.bak` um.

`47d` löscht Zeile 47 aus der Eingabe.

Was wollt Ihr zerlegen?

Habt Ihr gute (oder schlechte) Beispiele?

Tools

- grep (es gibt übrigens auch »git grep«)
- sed
- awk
- ag (The Silver Searcher)

Editoren (nicht komplett)

- Vim
- Atom
- Visual Studio Code
- Geany
- Gedit

Links – Auswahl, es gibt tausende

- Tutorial Reguläre Ausdrücke
- Regex Tester - Javascript, PCRE, PHP
- Wikipedia (en) Regular Expression
- Die Geschichte des Begriffs „regulärer Ausdruck“
- RegExr: Learn, Build, & Test RegEx
- Debuggex: Online visual regex tester. JavaScript, Python, and PCRE.
- Regex Tutorial, Examples and Reference - Regexp Patterns
- Schritt-für-Schritt zu eigenen Regulären Ausdrücken
- Wikipedia (en) Comparison of regular expression engines
- Wikipedia (de) Regulärer Ausdruck
- Selflinux Reguläre Ausdrücke
- **Online regex tester and debugger: PHP, PCRE, Python, Golang and JavaScript**

Vielen Dank!

Dirk Deimeke, 2019, CC-BY

dirk@deimeke.net

d5e.org – speakerdeck.com/ddeimeke

PDF bei Speakerdeck herunterladen, dann sind die Links klickbar.