Reinald Cronauer
7. Februar 2001

Reguläre Ausdrücke

1 Einführung

Reguläre Ausdrücke, kurz *Regex* genannt, erlauben das Suchen von Zeichenfolgen in große Texte. Ein regulärer Ausdruck ist aber flexibler als nur das Suchen eines einfachen Strings. Ein Suchen nach einem „Meier“ in verschiedene Variationen kann mit einem regulären Ausdruck leicht realisiert werden. Auch ein Suchen nach Geldbeträgen, Zahlen, Kundennummern u.s.w. ist möglich. Aus diesem Grund nennt man einen Regulären Ausdruck manchmal auch *Suchmuster.* (engl. *Pattern*)

Es gibt viele Programme, die mit regulären Ausdrücken arbeiten:

- grep und egrep
- awk
- sed
- ex, ed, vi und emacs
- Perl
- csplit
- tcl
- Python
- flex

Und viele weitere hier nicht aufgezählte Programme. Man sieht schon: Es lohnt sich, mit regulären Ausdrücken näher zu beschäftigen. Leider gibt es dadurch auch einen Nachteil: Es gibt zwischen den verschiedene Programmen auch Dialekte, die beim Formulieren eines regulären Ausdruck beachtet werden muß. In einer Tabelle am Schluß des Scriptums werde ich die gängigste Dialekte zusammenfassen.

2 Reguläre Ausdrücke

2.1 Einfache Zeichenketten

Für unsere Übungen benutzen wir egrep und später auch Perl. Die Übungen sind teilweise so aufgebaut, daß zum Lösen der Aufgaben auch das Wissen von vorangegangene Kapiteln benötigt wird. Die Lösungen der Übungen findet Ihr auf meiner Homepage[1]. Später kommt das natürlich auch auf dem LUGA-Server[2]. Die Lösungen stelle ich erst einige Tage, nachdem Ihr die Hausaufgaben gemacht habt, in den Server rein.

Egrep-Aufrufsyntax: `egrep -v -i` *'Muster' Datei*

Egrep gibt jede Zeile, die das Muster enthält, aus. Sofern im Musterausdruck keine spezielle Zeichen (Klammern, Fragezeichen etc.) enthalten sind, wird exakt nach der angegebene Zeichenfolge durchsucht. Falls nach spezielle Sonderzeichen gesucht werden soll, muß ein Backslash („\“) davorgesetzt werden. Z. B. wird eine geöffnete eckige Klammer mit „\[“ formuliert. Es gibt aber einige spezielle Ausnahmen, wo ein vorangestelltes Backslash die Funktion des nachfolgenden Zeichens ändert. Bei der Angabe von Wortgrenzen trifft das beispielsweise zu. (Siehe Kapitel 2.3.2) Ein Leerzeichen wird auch als Teil des Musters interpretiert! In diesem Scriptum stelle ich ein Leerzeichen als ␣ dar.

Das Muster fassen wir in das Zeichen ' ein, um eine Bearbeitung des Musters durch die Shell zu verhindern. Gerade bei einige Sonderzeichen ist das notwendig. Auch bei einem Muster, das ein Leerzeichen enthält, muß die Einfassung mit ' vorgenommen werden. Der Schalter *-i* läßt egrep die Unterscheidung zwischen Groß- und Kleinschreibung ignorieren. Ohne diesen Schalter ist egrep etwas pingeliger. *-v* negiert die Suche. Das heißt, egrep sucht nach Zeilen, die **nicht** den Suchausdruck enthält. Weitere Egrep-Schalter erklärt die Egrep-Manpage[7].

Übung 1: Suchen Sie mit dem egrep-Befehl nach dem Benutzernamen „root“ in der Datei /etc/passwd. Wie lautet der entsprechende egrep-Aufruf? Diese Übung sollte eigentlich jeder schaffen :-) Probieren Sie das auch mit andere Suchwörtern aus.

2.2 Mehrere Suchmuster durch Alternativen

Durch Verwendung von runde Klammern und einem Pipesymbol kann nach mehrere Strings gleichzeitig gesucht werden. Der Syntax dazu lautet:

(*Muster1*|*Muster2*)

Der reguläre Ausdruck sucht sowohl nach *„Muster1“* als auch *„Muster2“*. Wird einer der beiden Muster gefunden, gibt egrep die Zeile aus, die das Muster enthält. Die Alternativangabe kann auch Teil eines größeren Musters sein. Z. B. sucht *M(ay|ei)er* sowohl nach „Mayer“ als auch nach „Meier“.

Die runde Klammern haben noch weitere Funktionen, die in den Kapiteln 2.7.2 und 2.5 beschrieben sind.

Beispiel: *egrep '(Meier|Maier)' namen* sucht nach „Meier“ und „Maier“.

Übung 2: Dasselbe wie die letzte Übung. Suchen Sie zusätzlich noch den Namen „gast“ mit!
Übung 3: Formulieren Sie unter Verwendung von runde Klammern einen regulären Ausdruck, der nach „Aufsteigen“ und „Absteigen“ sucht.
Übung 4: Was würde egrep suchen, wenn man als Muster `(From|Subject):␣` angibt? (Das Zeichen ␣ wird muß als Leerzeichen eingegeben werden.)

2.3 Anker

Mit einem Anker kann der Anfang oder Ende eines Suchmusters an bestimmte Stellen „festnageln“. Einige Programme wie z. B. Perl oder einige egrep-Versionen kennen auch Wortanker, mit dem man den Anfang und das Ende eines Worts definiert.

2.3.1 Zeilenanker

Für alle regex-Dialekte gibt es einen Zeilenanfangs- und Zeilenende-Anker. Der Anker wird durch ein bestimmtes Zeichen repräsentiert.

^ Anker für den Zeilenanfang
$ Anker für das Zeilenende

Beispiel: *^# Kommentar* gibt alle Zeilen aus, die ein # am Zeilenanfang haben.

Übung 5: Geben Sie einen Suchbegriff an, der das Wort „LUGA“ in einer eigene Zeile suchen läßt.
Übung 6: Nach welchem String würde das Muster „^$“ suchen?

2.3.2 Wortanker

Eine Suche nach einem Wort kann durch Setzen von Leerzeichen vor und nach dem Suchbegriff durchgeführt werden. Allerdings wird ein Wortende, was am Ende eines Satzes steht, nicht erkannt, da am Wortende bekanntlich ein Punkt steht. (Das gilt auch für ein Wort am Zeilenanfang.) Ein Wortanker dagegen ist da wesentlich flexibler. Der Wortanker kennzeichnet den Anfang bzw. das Ende eines Worts. Satzzeichen und Interpunktionszeichen werden auch als Wortgrenzen erkannt.
Leider beherrschen nicht alle Regex-Programme Wortanker. Für Details muß ich auf die entsprechende Programmdokumentation verweisen. Perl und egrep kennen Wortanker.

\< Anker für den Wortanfang.
\> Anker für das Wortende.

2.4 Zeichenklassen

Eine Zeichenklasse definiert eine Menge von Zeichen, die an einer angegebene Stelle eines regulären Ausdrucks vorkommen darf. Die vom Benutzer anzugebende Menge muß in eckige Klammern eingeschlossen werden. Eine Ausnahme bildet das im nächsten Kapitel beschriebene Punkt, die nicht in eckige Klammern eingeschlossen wird.

2.4.1 Ein beliebiges Zeichen

Der Punkt in einem regulären Ausdruck bedeutet „beliebiges Zeichen". Das gilt auch für Leerzeichen. Das funktioniert ähnlich wie das „?" in der Shell. Soll direkt nach einem Punkt gesucht werden, muß der Punkt nach einem Backslash stehen.

Beispiel:
ab. sucht nach „ab" und irgendein Zeichen dahinter.
ab\. sucht nach „ab."

2.4.2 Ausgewählte Zeichen

Eine Zeichenklasse gibt alternative Zeichen oder einen Bereich von Zeichen an, die in einem regulären Ausdruck vorkommt. Eine Zeichenklasse wird immer in eckige Klammern eingefaßt. Alles, was zwischen den eckige Klammern steht, wird als **ein** Zeichen aufgefaßt! Wird innerhalb des Zeichenklassenausdrucks das Zeichen ˆ als 1. Zeichen angegeben, bedeutet dies eine Negierung der Zeichenklasse. Das heißt, daß **nicht** nach den Zeichen gesucht wird. Steht ˆ irgendwo in dem Zeichenklassenausdruck, wird das als literales[1] Zeichen aufgefaßt.

[x] Ein einzelnes Zeichen „x".
[abc] Ein einzelnes Zeichen, das „a", „b" oder „c" enthält.
[a-z] Ein Zeichenbereich von „a" bis „z".
[a-zA-Z] Ein Zeichen aus dem Bereich von „a" bis „z" und „A" bis „Z".
[ˆ0-9] Alle Zeichen, aber **nicht** die Ziffern.
[ˆx] Alle Zeichen, aber **nicht** „x".
[abˆ] Ein einzelnes Zeichen, das „a", „b" oder „ˆ" enthält.

Beispiele:
M[ae][iy]er sucht nach folgende Zeichenketten: Maier, Meier, Mayer, Meyer.
[0-9] sucht nach Ziffern.
ab[ˆc] sucht alle Zeichenketten, die "ab„ enthält, aber **nicht** "abc„.

Übung 7: Suchen Sie alle Zeilen aus einer beliebigen Textdatei alle Zeilen, die **nicht** den Buchstaben „E" enthält. Das gilt sowohl für das groß- als auch für das kleingeschriebene „E". Die Benutzung des egrep-Schalters *-i* dazu ist verboten!

[1] Mit einem literalen Zeichen ist genau das zu suchende Zeichen gemeint. Das Zeichen wird nicht irgendwie interpretiert.

2.4.3 Spezielle Zeichen und -klassen

Soll ein Zeichen, was sonst in einem regulären Ausdruck eine spezielle Funktion hat, als normaler Teil des zu suchenden Text interpretiert werden, muß ein Backslash davorgesetzt werden. Zum Beispiel sucht „\.“ nach einem Punkt und die Sonderbedeutung des Punkts wird damit aufgehoben.

Einige Programme wie z. B. Perl sind in der Lage, auch nach Steuerzeichen zu suchen. Dazu werden Macros verwendet. Nachfolgend die Macros, die beim Perl verwendet werden:

a Piep Piep
n Zeilenvorschub (Newline)
r Wagenrücklauf (Carriage Return)
t Tabulator
f Seitenvorschub (Form-Feed)
e Escape
d Ziffer
D Nicht-Ziffer
w Wort-Zeichen [a-zA-Z_0-9]
W Nicht-Wortzeichen
s White-Space [␣\t\n\r\f]
S Nicht-White-Space

2.4.4 POSIX-Klammerausdrücke

Vordefinierte Zeichenklassen werden durch nachfolgend aufgelistete Klammerausdrücke definiert.

[**:alnum:**] Alphanumerische Zeichen. (Buchstaben und Ziffern)
[**:alpha:**] Buchstaben
[**:blank:**] Leerzeichen und Tabulator
[**:cntrl:**] Steuerzeichen (Codes 1-31)
[**:digit:**] Ziffern
[**:graph:**] „schwarze“ Zeichen (nicht: Leerzeichen und Tabulator)
[**:lower:**] Kleinbuchstaben
[**:print:**] wie [:graph:], aber mit Leerzeichen
[**:punct:**] Interpunktionszeichen
[**:space:**] „weiße“ Zeichen (Leerzeichen, Newline, Tabulator)
[**:upper:**] Großbuchstaben
[**:xdigit:**] Hexadezimale Ziffern

Übung 8: Wie lautet der reguläre Ausdruck für die Suche nach einer 4stelligen Hexadezimalzahl?

2.5 Quantifier

Jetzt wird es interessant: Mit den nachfolgend beschriebene Sonderzeichen kann man nach beliebig viele Zeichen suchen. Das Sonderzeichen bezieht sich nur auf das unmittelbar vorangegangenen Atom[2].

Ein paar Worte zum Verhalten des Regex-Automaten: Die Regex-Maschine nimmt aus dem durchzusuchenden Text so viele Zeichen wie möglich. Aus diesem Grund nennt man dieses Verhalten auch „gierig". Anhand von Beispielen wird das gierige Verhalten genauer erläutert.

Bei einige Programmen wie z. B. Perl kann die Gierigkeit durch spezielle Formen des Quantifiers ausgeschaltet werden. In diesem Falle wird nur so wenige Zeichen wie möglich aus dem Suchtext herangezogen, um einen Treffer zu erlangen.

2.5.1 0 oder 1 Atom: Fragezeichen

Wenn ein Zeichen oder Atom auch mal fehlen darf, wird dazu das Fragezeichen herangezogen. Ist das Atom vorhanden, wird das ebenfalls als Treffer gewertet. Aufpassen: Das Fragezeichen funktioniert anders als ein Fragezeichen in der Shell.

Beispiele:
LUGA? Sucht nach „LUG" und "LUGA„.
ABC.? Sucht nach „ABC" und zusätzlich nach einem String, was mit ABC beginnt und mit einem beliebigen Zeichen endet.
ABC(XYZ)? Findet sowohl ABC als auch ABCXYZ.

Übung 9: Welche Treffermäglichkeiten gibt es beim `␣LUG[AL]?␣` ?
Übung 10: Formulieren Sie einen Regex, die nach „Maier" und „Mair" sucht.

2.5.2 1 und ∞ Atome: Pluszeichen

Das Pluszeichen erlaubt das Suchen von mindestens einem oder beliebig viele Atome. Hier tritt die Gierigkeit besonders n Erscheinung.

Beispiele: **^#.+** sucht nach Zeilen, die mit # beginnen und etwas dahinter haben.

Übung 11: Mal etwas Kniffligeres. Formulieren Sie einen regulären Ausdruck, was nach einem Wort oder Satz, die in doppelte Gänsefüßchen eingefaßt sind, sucht.

[2]Mit Atom wird ein minimales Element bezeichnet. Das kann ein einzelnes Zeichen, ein Klammerausdruck oder eine Zeichenklasse sein.

2.5.3 0 bis ∞ Atome: Stern

Ein Sternchen arbeitet ähnlich wie ein Stern bei Dateiangaben in der Shell. Um einen Treffer zu erlangen, reichen Null bis beliebig viele Atome an der Stelle des Sternchens.

Beispiele:
ABC* Sucht nach einer Zeichenfolge, die AB enthält.
ABC.*XYZ Sucht nach einer Zeichenfolge, die ABC und XYZ enthält, wobei dazwischen evtl. weitere Zeichen stehen.
ABC(XYZ)* Findet ABC und ABCXYZ. Es wird auch ABCXYZXYZ gefunden. Die Anzahl vom „XYZ“ ist egal.

Übung 12: Worin besteht der Unterschied zwischen *ABC*XYZ* und *ABC.*XYZ*?
Übung 13: Warum findet der reguläre Ausdruck *[0-9]** alle Zeilen, und nicht nur Zeilen, die Zahlen enthält? Welcher regulärer Ausdruck führt zum gewünschten Ziel?

2.6 Intervalle

Damit kann die Anzahl von Wiederholungen des unmittelbar vorangegangenen Elements festgelegt werden. Ein Intervall wird in geschweifte Klammern angegeben. Manche Dialekte verlangen noch einen Backslash unmittelbar vor dem Klammernzeichen.

Beispiele:
(ABC){n} verlangt n-mal hintereinander die Zeichenfolge ABC.
(ABC){n,m} verlangt mindestens n-mal ABC, aber höchstens m-mal ABC.
(ABC){n,} verlangt mindestens n-mal ABC. Das ABC kann aber auch mehr als n-mal auftauchen.

Übung 14: Definieren Sie einen regulären Ausdruck, was nach 4stellige Zahlen sucht.
Übung 14: Wie sieht ein regulärer Ausdruck aus, was nach einem Geldbetrag in der Form von „125.34 DM“ sucht?

2.7 Bezug zum früheren Match

Einige Programme können nicht nur nach regulären Ausdrücken suchen, sondern auch den gesuchten Ausdruck ersetzen. Mit Hilfe von runde Klammern kann ein Muster oder Teile davon „eingefangen“ werden. Der eingefangene Teilausdruck wird dann Teil eines Ersetzungstextes. Die Sache geht noch weiter: Ein eingefangener Ausdruck kann wiederum Teil des Suchmusters werden.

2.7.1 Kurzer Ausflug zum Perl

Für die nachfolgende Kapitel verwenden wir Perl. Keine Angst, wir schreiben keine Perl-Scripte, sondern Perl-Einzeiler. Die Perl-Einzeiler wird gleich beim

Shell-Prompt eingegeben. Damit das Bearbeiten einer Datei mit Perl funktioniert, übergeben wir einige Optionen an Perl. Der Aufrufsyntax sieht so aus:

`perl -p -iext -e 's/Muster/Ersetzung/Optionen' datei`

Der Schalter *-p* sorgt dafür, daß die übergebene Dateinamen bearbeitet werden. *-i* erledigt die Ausführung der Dateioperationen auf Standard-Ein- und Standard-Ausgabe. Wird „.ext" dazugehängt, wird an den Dateinamen des Originals die Zeichenfolge „.ext" angehängt und das Original wird mit der Ersetzungsoperation bearbeitet. Ohne Zusatz wird die angegebene Datei direkt bearbeitet und die vorherige Version geht verloren. *-e* sagt dem Perl, daß das nachfolgende Argument Perl-Code darstellt. Genauere Details zu den Schaltern wird in der Manpage[6] beschrieben.

Der reguläre Ausdruck wird hinter dem s in Schrägstrichen eingeschlossen. Das s am Anfang heisst für Perl: „Suche das Muster *Muster* und ersetze den durch *Ersetzung*." *Optionen* steht für diverse Schalter, mit dem man z. B. die Unterscheidung von Groß- und Kleinschreibung ausschalten kann. Im Rahmen des Vortrags interessieren uns folgende Optionen:

i Ignoriert Groß- und Kleinschreibung.
g Ersetze global.

Es gibt noch weitere Operatoren, die den Rahmen des Vortrags sprengen. Wer sich näher dafür interessiert, kann ja die zugehörige Manpage[5] einsehen oder ins entsprechende Buch[3] reinschauen.

Tip: Wird Perl mit `perl -p -e 's/Muster/Ersetzung/Optionen'` aufgerufen, arbeitet Perl mit stdin und stdout[3]. Damit kann der zu bearbeitende Text zum Experimentieren direkt eingetippt werden. Aussteigen ist mit <strg-C> möglich.

Übung 15: Wie würde der Perl-Aufruf lauten, um in einer HTML-Datei alle
-Tags nach Großbuchstaben umzuwandeln?

2.7.2 Bezug zum früheren Match

Mit dieser Funktion kann bei Such- und Ersetzoperationen ein Match gespeichert und als Ersetzungstext eingesetzt werden. Für die Speicherung von Mustern oder Teile davon werden runde Klammern verwendet. Jede öffnende Klammer wird von links nach rechts durchgezählt und die entsprechende Zahl wird im Ersetzungstext eingesetzt. Das Bearbeitungsprogramm ersetzt diesen Ersetzungsmacro durch den tatsächlichen Match. Ein Ersetzungsmacro wird beim Perl mit einem Dollarzeichen und einer nachfolgende Zahl definiert. (z. B. $1) Andere Programme verwenden statt des Dollarzeichens den Backslash.

Beispiel:
s/␣(Rind|Fleisch)␣/␣BSE-$1␣/g ersetzt das Wort „Fleisch" bzw. „Rind" durch „BSE-Fleisch" bzw. "BSE-Rind".

[3] In der Regel ist `stdin` die Tastatur, `stdout` der Bildschirm.

Literatur

[1] **WWW:**`http://home.diedorf.net/dh2mbm/regex/`

[2] **WWW:**`http://www.luga.de`

[3] **Buch:***Programmieren mit Perl*, O'Reilly-Verlag, Seite 78

[4] **Buch:***Reguläre Ausdrücke*, O'Reilly-Verlag, Seite 67

[5] **Manpage:***Regex beim Perl*, `man perlre`

[6] **Manpage:***Kommandozeilenschalter beim Perl*, `man perlrun`

[7] **Manpage:***Grep und egrep*, `man grep` und `man egrep`

A Regex-Dialekte

Wie schon am Anfang des Vortrags erwähnt, gibt es leider verschiedene Dialekte bei den einzelnen Programmen. Die nachfolgende Tabelle stellt die Elemente eines regulären Ausdrucks im entsprechenden Dialekt dar. Allerdings gibt diese Tabelle nur einen groben Überblick. Zum Beispiel verhält sich die Wortgrenze (\w und \W) bei alle drei Programme etwas unterschiedlich. Genaue Details steht in dem Regex-Buch[4] aus dem ich auch die untenstehende Tabelle entnommen habe.

grep	egrep	awk	emacs	perl	Tcl	vi	Bemerkung
\?	?	?	?	?	?	\?	0 bis 1 Atom
\+	+	+	+	+	+	\+	1 bis n Atome
\\|	|	|	\\|	|	|		Alternativen
\(...\)	(...)	(...)	\(...\)	(...)	(...)	\(...\)	Gruppenbildung
	\< \>		\< \> \b \B	\b \B		\< \>	Wortgrenzen
	\w \W		\w \W	\w \W		\< \>	Wortzeichen
•			•	•		•	Rückwärtsreferenzen