grep Tutorial

update 06.11.2002

Gesamtindex:

- Startseite
- Forum Musik und Recht
- Juristische Links
- Sonstige Links
- Mail
- Privat

Die Volltext-Suche mit regulären Ausdrücken
- eine Anleitung zur Nachbildung der boolschen Operatoren am Beispiel von `grep`
Version 0.3

ToDo:

- Beispielverzeichnis für Rechercheübungen
- erweiterte Erläuterung von "(wort1 ([^]*) {0,2}wort2"

`Befehle und systeminterne Begriffe sind in im Text so dargestellt.` *<Kommentierungen auf diese Weise.>*

Mit dieser Frage fing alles an:
Wie suche ich nach einem Wort in allen meinen Textdateien?

Hat man (beispielsweise als Jurist) eine Sammlung von Textdateien (*.doc, *.htm, *.html, *.txt, *.rtf usw.) die man komplett nach bestimmten Begriffen durchsuchen will, stößt man unter Windows schnell an die Grenzen der im Betriebssystem implementierten Suchfunktion. Anders unter Linux. Hier gibt es `grep`.

Dieses kleine Tutorial will zur Verwendung von `grep` anleiten. Es kann nicht auf jede Frage eine Antwort geben und keinesfalls Selbstbildung durch das Lesen von `manpages` erübrigen.

"Grep" steht für "get regular expression", der Befehl soll also das Suchen nach regulären Ausdrücken ermöglichen (vgl. Szenario 5). Dieses mächtige Werkzeug kann man sich im einzelnen wie folgt zu Nutze machen.

Inhalt:
I. Zur Einführung - Beispiele und Erläuterungen

Szenario 1 - Suche nach `wort1`
Szenario 2 - Suche nach `wort1 ODER wort2`
Szenario 3 - Suche nach `wort1 UND wort2`
Szenario 4 - Suche nach `wort1 UND wort2 UND wort3`

II. Exkurs

Szenario 5 - Komplexe Suchen und `grep`: die Operatoren UND, ODER, UND NICHT und NAHE BEI mit Beispielen vorgestellt.

Szenario 1:
Es soll in allen Dateien des Typs '.doc' gesucht werden, die das wort1 enthalten.

Anleitung:
Zuerst ins entsprechende Verzeichnis wechseln - zum Beispiel das Verzeichnis /home/user/jura/BGH_ZR

Wie wechsele ich das Verzeichnis?
Gewechselt wird wie unter DOS auch:
- im Verzeichnisbaum nach unten mit '`cd Verzeichnisname`'
< `cd` *steht für change directory>;*
kennt man den kompletten Pfad kann man natürlich auch diesen angeben
- also: '`cd /verzeichnisname/unterverzeichnisname`'.
- im Verzeichnisbaum nach oben mit '`cd ..`'

In welchem Verzeichnis bin ich?
Um festzustellen, wo man sich im Verzeichnisbaum befindet ist der Befehl `pwd` hilfreich. Er gibt das aktuelle Verzeichnis auf der Konsole aus.

Und wie suche ich nun?
Gesucht wird dann mit folgendem Befehl:

```
grep -l -i wort1 *.doc
```

Natürlich gibt es auch Varianten wie einen Listbefehel <`ls`> der - hier ohne Beschränkung auf einen zu durchsuchenden Dokumententyp - mittels einer pipe "`|`" auf das "`grep`" - Kommando geleitet wird:

```
ls | grep wort1
```

<Das Zeichen für die pipe "`|`" ist auf der Tastatur neben dem Y (mit AltGr). Die Ausgabe des Befehls "`ls`" wird verwendet um "`grep`" mitzuteilen, was durchsucht werden soll. Grep durchsucht also alles was ls ihm zuschiebt.>

Was machen die Zusätze `-l` und `-i` ?
Es handelt sich um Parameter die dem `grep` - Befehl mitgegeben werden, um ein genaueres Kommando zu geben.
Der Befehl `-i` sorgt dafür, daß Groß- oder Kleinschreibung ignoriert werden. Der Befehl `-l` sorgt dafür, daß die Dateien aufgelistet (und nicht Zeilen angezeigt) werden, die die Suchkriterien erfüllen.

Wie speichere ich das Ergebnis der Suche?
Zum Abspeichern des Ergebnisses kann man die Ausgabe in eine Datei umleiten und zwar mit einem Zusatz zum eigentlichen Befehl '`> dateiname.dateityp `'. Dies hat zur Folge, daß die Ausgabe, die eigentlich auf der Konsole erscheinen würde, im Verzeichnis als Textfile gespeichert wird. Im Beispiel wird die Ausgabe des Suchkommandos in die Textdatei '`ergebnis.txt `' umgeleitet. Mehr dazu hier.

Beispiel:

```
grep -l -i wort1 *.doc > ergebnis.txt
```

zum Inhalt

Szenario 2
Es wird nach allen Dateien des Typs '.doc' gesucht, die das wort1 ODER das wort2 enthalten.

Gesucht wird (wort1 ODER wort2)
Lösung:
Das Vorgehen erfolgt wie in Szenario 1, wird aber wie folgt abgeändert:

```
grep -i -l -E "(wort1|wort2)" *.doc
```

Der Befehl -E steht für extended grep. Daher kann der Befehl auch so ausgedrückt werden:
`egrep -l -i "(wort1|wort2)" *.doc`
zum Inhalt

Szenario3
Es wird nach allen Dateien des Typs '.doc' im aktuellen Verzeichnis gesucht, die das wort1 UND das wort2 enthalten.

Gesucht wird (wort1 UND wort2)
Der Befehl lautet:
`grep -il wort1 *.doc | xargs grep -il wort2`
Ein Beispiel mit wort1=rüge und wort 2=präklusion
`grep -il rüge *.doc | xargs grep -il präklusion`
Warum haben sich die Parameter verändert?
Hier wurde **`-i` und `-l` zusammengefasst zu `-il`**.
Der Befehl ist auch auf weitere Wörter erweiterbar wie Szenario 4 zeigt.

Was ist `xargs`?
Mit diesem Befehl wird eine Argumentliste aufgebaut um ein Kommando (hier `grep`) auszuführen. Es wird zuerst nach wort1 gesucht, die gefundenen Dateien werden daraufhin nach wort2 durchsucht. Dabei ist die pipe wieder hilfreich.

zum Inhalt

Szenario 4
Es wird nach allen Dateien des Typs '.doc' im aktuellen Verzeichnis gesucht, die das wort1 UND das wort2 UND das wort3 enthalten.

Gesucht wird (wort1 UND wort2 UND wort3)
Der Befehl lautet (1 Zeile):
`grep -il wort1 *.doc | xargs grep -il wort2` **`| xargs grep -il wort3`**

zum Inhalt

Szenario 5

Übergabe komplexer Suchmuster an `grep` mittels regulärer Ausdrücke.
Beipiel für eine komplexe Suche:

(((wort1 ODER wort2) UND ((wort4 NAHE BEI wort5) ODER wort6)))

< wobei in der Lösung NAHE BEI (= maximal 2 Worte Abstand) >

Lösung:
```
(
(wort1|wort2) .*
(wort4 ([^ ]* ) {0,2} wort5|wort5 ([^ ]* )
{0,2}wort4)
```

```
)
|
(
(wort4 ([^ ]* ) {0,2} wort5|wort5 ([^ ]* ) {0,2}wort4
|wort6) .*(wort1|wort2)
)
```

Lösung mit Kommentierung:

```
(
(wort1|wort2) .*
```

<wort1 ODER wort2, vgl. unten>

```
(wort4 ([^ ]* ) {0,2} wort5|wort5 ([^ ]* )
{0,2}wort4)
)
```

< wort 4 NAHE BEI wort5, vgl. unten>

```
|
```

< die pipe und die vertauschte Wiederholung ist notwendig um das boolsche UND zu basteln, vgl. unten >

```
(
(wort4 ([^ ]* ) {0,2} wort5|wort5 ([^ ]* ) {0,2}wort4
|wort6) .*(wort1|wort2)
)
```

Eine komplexe Suche wie die obige kann verständlicherweise nicht mehr mit einem einfachen Suchbefehl in der Art 'Suche mir Wort1' erfolgen. Bevor der Suchvorgang gestartet werden kann, ist die Beschreibung des zu suchenden Textmusters erforderlich.
Ein regulärer Ausdruck wie der obige soll komplexe Muster beschreiben. Wenn er dann abgearbeitet wird, erfolgt der Vergleich des Textmusters mit den Zeichenketten der zu durchsuchenden Dateien.

Beschreibung von Textmustern
- Die Elemente eines regulären Ausdrucks
Man kann sich schrittweise an die Elemente eines solchen Ausdrucks herantasten. Im einzelnen wird kurz vorgestellt:

- UND
- ODER
- UND NICHT
- Kombination UND NICHT und UND
- Problemstellung Zeilenumbruch
- Lösung des Zeilenproblems
- NAHE BEI
- GEFOLGT VON

1. Das boolsche UND
Beispiel: (wort1 UND wort2) werden in allen .doc Dateien gesucht.
Lösung:

```
grep -E 'wort1.*wort2|wort2.*wort1' *.doc
```

< (wort1 UND wort2) entspricht (wort2 UND wort1) >

Oder auch (vgl. oben)
```
grep -il wort1 *.doc | xargs grep -il wort2
```

2. Das boolsche ODER
Beispiel: (wort1 ODER wort2) werden in allen *.doc Dateien gesucht.
Lösung:
```
grep -E 'wort1|wort2' *.doc
```
< Die pipe '|' ist von sich aus kommutativ, sie bringt quasi ein ODER bereits mit>

3. Das boolsche UND NICHT
Beispiel: Gesucht wird (wort1 UND NICHT wort2) in allen Dateien vom Typ ".doc"
<umzuformen in (NICHT wort2 UND wort1)>

Lösung:
```
grep -v wort2 *.doc | grep wort1
```
< `-v` gibt nur die Dateien aus, die das ausgeschlossene wort2 nicht enthalten, die Suche wird quasi invertiert. Die Ausgabe wird dann nach wort1 durchsucht>

4. Einfache Kombination von UND und UND NICHT
Das Beispiel: ((wort1 UND wort2) UND NICHT wort3)
entspricht ((NICHT wort3) UND wort1 UND wort2)

Der Befehl muss lauten:
```
grep -v wort3 datei.txt | grep -E
'wort1.*wort2|wort2.*wort1'
```
< Hier wird zunächst wieder mit -v wort3 ausgeschlossen, dann erfolgt eine Suche wie in Szenario 3.>

5. Das boolsche NAHE BEI - der Stellungsoperator NEAR

5.1 Notwendige Vorarbeit - die Lösung des Zeilenproblems

Zeichenbstände - Praktikabel?
Einzelne Zeichenabstände lassen sich genau durch Punkte definieren, da sie abgesehen vom Leerzeichen beliebige Symbole repräsentieren.

Beispiel:
"Donaudampfschiffskapitän"
wird gefunden durch Suche nach
"Donau.....schiffskapitän"

Das ist auf Wortebene aber nicht praktikabel. Worte sind durch Leerzeichen getrennt. Diese lassen sich durch Punkte aber nicht darstellen. Auf Zeilenumbrüche ist man bei der Suche nicht angewiesen. Damit ergibt sich ein anderer praktikabler Ansatz um der NEAR-Option nahezukommen, und ein auch ein weiteres Problem.

Zunächst stellt sich das Problem, dass Zeilen die kleinsten Suchelemente für `grep` sind. Will man über den gesamten Text nach Abständen suchen, so muss man die Zeilumbrüche entfernen.

Wie lassen sich Zeilenumbrüche entfernen?
Hierzu kann folgendes Kommando verwendet werden:
`tr -d \\012 < beispieldatei.txt`
<hierbei steht `tr` *für translate und* `-d` *für delete,* `\\012` *ist das ASCII-Symbol für den Zeilenumbruch und das '<' hat die Funktion, klarzustellen, in welcher Datei die Zeilenumbrüche entfernt werden sollen.>*

5.2 Anwendung des Kommandos zur Zeilenumbruchentfernung

Das in "datei.txt" zu Suchende Beispielmuster sei "(wort1 UND wort2)"

Versagt aufgrund eines Zeilenumbruchs die Suche mit diesem Befehl:
`grep -E 'wort1.*wort2|wort2.*wort1' datei.txt`

könnte dieser Suchbefehl zu Ergebnissen führen:
```
cat datei.txt | tr -d \\012 | grep -E
'wort1.*wort2|wort2.*wort1'
```

< '`cat`' *entspricht ungefähr dem Kommando* '`type`' *unter DOS, es gibt den Inhalt einer Datei zeilenweise auf der Konsole aus, das ist bei großen Dateien unbefriedigend. Daher gibt es* '`cat`', '`more`' *und* '`less`', *sie bilden eine Trias. ´*'`more`' *und* '`less`' *zeigen Dateiinhalte seitenweise an - eine sinnvolle Ergänzung zu* '`cat`'.
Funktion:

- `cat dateiname` *zeigt die Datei an*
- `cat dateiname | more` *zeigt die Datei seitenweise an und erlaubt Blättern vorwärts mit den Tasten* `Return` *und* `Space`*, beenden mit '*`q`*'.*
- `less dateiname` *zeigt zeigt die Datei seitenweise an und erlaubt Blättern in beide Richtungen mit BILD, beenden mit '*`q`*'; less benötigt aber eine optimal eingestellte Konsole. >*

Da häufig alle Dateien des Verzeichnisses durchsucht werden sollen bietet sich für das Beispielmuster (wort1 UND wort2) folgendes an:
u = dateiname.dateityp

```
for u in 'ls':
do
echo $u:;
cat $u |tr -d \\012|grep - E
'wort1.*wort2|wort2.*wort1';
done | tee ergebnis.txt
```

<Ziel ist die gleichzeitige Ausgabe auf dem Bildschirm und das Speichern des Ergebnisses in einer Datei. Deshalb wird das Kommando `tee` *verwendet. Es veranlaßt den Rechner zur Erfüllung dieser Aufgabe.>*

5.3 Beispiele für die Stellungsoperatoren NAHE BEI und GEFOLGT VON

5.3.1 Beispiel für den Stellungsoperator NAHE BEI

Das folgende Beispiel ist für die Suche des Musters (wort1 NAHE BEI wort2) mit maximal 2 Worten Abstand innerhalb datei.txt ausformuliert:

```
grep (wort1 ([^ ]* ) {0,2}wort2|wort2 ([^ ]* ) {0,2} wort1) datei.txt
```

*< ^ verneint Leerzeichen ; * erlaubt n Vorkommen >*

5.3.2 Verfeinerung des Operators NAHE BEI zu GEFOLGT VON

Ist es nicht egal in welcher Reihenfolge die Worte auftauchen, darf natürlich der Befehl nicht wie im obigen Beispiel 5.3.1 kommutativ sein. Vielmehr muss das wort1, dass in einem bestimmten Abstand von einem wort2 gefolgt werden soll auch in der Suche vorne stehen.

Man kann bei der Suche in datei.txt statt den Operator NAHE BEI den Operator GEFOLGT VON verwenden.
Wird das Muster (wort1 GEFOLGT VON wort2) gesucht, wobei maximal 2 Worten Abstand zwischen den Worten sein soll, muss der Suchbefehl lauten:

```
grep (wort1 ([^ ]* ) {0,2}wort2 datei.txt
```

*< ^ verneint Leerzeichen ; * erlaubt n Vorkommen >*

6. Ergebnis

Man kann mit `grep` eine ganze Menge machen, auch komplexe Suchanfragen lassen sich realisieren. Für diese wäre aber ein grafisches Frontend, welches zumindest die hier vorgestellten Operatoren umsetzt, eine wesentliche Erleichterung. Leider bin ich nicht in der Lage ein solches Projekt in Angriff zu nehmen.
Im Hausgebrauch (1 user, 1 Rechner, Daten auf HD) ist grep auch bei größeren Datenmengen und komplexen Suchmustern sicher performant genug. Ob das in einem größeren Umfang (viele user greifen über LAN zu, Daten auf HD oder CD-ROM des servers) immer noch der Fall wäre entzieht sich aber meiner Kenntnis.

Bitte senden Sie mir Ihre Anmerkungen und Verbesserungsvorschläge.

Danke:
Ohne die engagierte Mithilfe von Andreas Sigg würde es diese Seite so nicht geben, er hat wesentlich dazu beigetragen. Herzlichen Dank!

zum Inhalt