

T-Time mit Udo

Vorträge/Bash/T-Time mit Udo;-

[Vorträge](#), [Vortrag](#), [BASH](#), [T-Time mit Udo](#)

T-Time (Terminal-Time) sind Udos kleine Einführungen und Übungen in BASH und Terminal - aka Konsole, xterm, term ...

Viel Spaß bei den einzelnen Übungen! 😊

2020-05-27: RegEx ist kein Dinosaurier!

Regular Expression;- RegEx;- BASH/Befehle/grep;- BASH/Befehle/egrep;-

Was ist RegEx?

Regular Expression (regex, RegEx) sind Teil der theoretischen Informatik.

Wozu braucht man RegEx?

RegEx wird gebraucht um nach (mathematischen) Mustern in Dateien zu suchen und die gefundenen Zeilen auszugeben. RegEx ist Bestandteil vieler Programmiersprachen(z.B. Perl, Python, Tcl,...)

Einführung

Wie schon am Anfang geschrieben ist RegEx das Suchen nach 'Mustern' (Ausdrücken), und nicht nach „Namen“. Die Syntax der Ausdrücke ist in vielen Programmen ähnlich in den einfachen Anweisungen, weicht aber erheblich in den komplexeren ab und ist daher eher vergleichbar mit einer Sprache. (S.a.: [WP: Regulärer Ausdruck](#))

Wir behandeln hier nur ein paar Basics um den Einstieg zu meistern.

Die Ausdrücke bestehen aus Zeichen mit Sonderbedeutung, von denen wir uns folgende näher anschauen:

- Der Zeilenanfang wird mit dem Circonflex (auch [Zirkumflex](#)) '^' maskiert. Beispiel: '^[AS]' sucht nach Wörtern am Zeilenanfang die mit (Groß-)A oder -S beginnen.
- Das Dollar-Zeichen '\$' steht für Zeilenende. Beispiel: '*.txt\$' (!) In einigen Sprachen muss explizit angegeben werden, ob sich '^|\$)' auf Zeilen- oder Datei-Anfang/Ende bezieht.
- Das Zeichen '<' zeigt den Wortanfang. Beispiel: '<Col'
- Das Wortende wird mit '>' symbolisiert. Beispiel: 'ger>'
- Die Eckigen Klammern '[' bilden Aufzählungen und Gruppen von Zeichen. Beispiel: '[0-9a-zA-Z\.-]', '[RFSW-Y]'
- Runde Klammern '(' enthalten Alternativen, das '|' (Pipe)-Zeichen ergibt das 'oder'. Beispiel: '(Colombo|Arnol|anco)'
- Geschweifte Klammern '{}' enthalten, wie oft der vorstehende Ausdruck in {min,max} vorkommt. Beispiel: '[a-c]{2,4}' a|b|c kommt 2-4 Mal vor.
- Weiter gibt es so genannte Quantoren: Sie geben die Anzahl der Zeichen in Bezug auf den unmittelbar vorstehenden Ausdruck oder Zeichen an:

- '?' 'optional' null oder einmal Beispiel : 'July?' Hier ist das 'y' optional
- '+' mindestens einmal oder mehr
- '*' einmal, mehr oder keinmal
- Der Punkt '.' steht für ein beliebiges Zeichen.
 - Der Punkt '.' gefolgt vom Sternchen steht somit für ein, oder kein, beliebiges Zeichen in beliebiger Anzahl. Beispiel: '.*warz.*'

Siehe auch: man regex (7)

Beispiele

- 'Arno' # alle 'Arno'
- '[SCZ]+' # mindestens ein (S|C|Z) muss enthalten sein
- '(rno|ick|tes)' # enthält 'rno', 'ick' oder 'tes'
- '\\<19[0-9][0-9]\\>' # alle Jahre aus 19xx
- '\\<19[0-9][0-9]\\>{2,4}' # muss 2-4 Mal 19xx enthalten
- '[0-9]{4}-[0-9]{2}-[0-9]{2}' # Datum in ISO
- '[0-9]{1,2}\\.[0-9]{1,2}\\.(19|20)[0-9]{2}' # Datum in DE Schreibweise, nur die Jahrhunderte 19, 20
- '^\\[s]*\\#' # Zeile beginnt vorn mit Kommentarzeichen
- '[^D-F]' # Nicht (Gross-) D-F

Übungen

Die Übungen beziehen sich auf unten stehende Beispieldaten. In denen ist gelistet, wer aus welchem Land in welchem Jahr [Mr. Olympia](#) wurde:

- Suche Olivia
- Suche alle die mindestens drei mal in 1900er vorkommen
- Suche alle die nicht USA sind
- Suche alle die Frank heissen
- Wer hat bis 2007 mindestens drei mal im zwanzigsten Jahrhundert gewonnen?

Zum ausprobieren eigenen sich die Befehle: grep, egrep oder sed. Mehr über die Befehle erfahrt ihr wie immer mit: man \$befehl.

Zum experimentieren und herumspielen die Datei 'mrolympia.dat' mit den Beispieldaten herunterladen und ordentlich RegEx'en. (Zum speichern ggf.: Rechtsklick und Speichern_Unter)

[mrolympia.dat](#)

```
Larry Scott USA 1965 1966
Sergio Oliva USA 1967 1968 1969
Arnold Schwarzenegger Österreich 1970 1971 1972 1973 1974 1975 1980
Franco Columbu Italien 1976 1981
Frank Zane USA 1977 1978 1979
Chris Dickerson USA 1982
Samir Bannout Libanon 1983
Lee Haney USA 1984 1985 1986 1987 1988 1989 1990 1991
Dorian Yates Grossbritannien 1992 1993 1994 1995 1996 1997
Ronnie Coleman USA 1998 1999 2000 2001 2002 2003 2004 2005
Jay Cutler USA 2006 2007 2009 2010
Dexter Jackson USA 2008
Phil Heath USA 2011 2012 2013 2014 2015 2016 2017
Shawn Rhoden USA 2018
```

Brandon Curry USA 2019

(Daten anlehndend an Wikipedia)

24.05.2020

2020-05-24

Mehr RegEx gibt es u.a. auf: <https://www.regular-expressions.info>

Viel Spaß beim Suchen nach Mustern! 😊

2020-05-20: tar und less

BASH/Befehle/tar;- BASH/Befehle/less;- BASH/Befehle/find;-

Was wir heute machen wollen:

1. mkdir Uebung im directory bs_lug/
2. cd Uebung/
3. file001-100 erstellen
4. Alle Dateien aus /etc die NICHT mit [aeiou] anfangen nach ~/bs_lug/Uebung/ kopieren
5. Eine 100M-große Datei erzeugen
6. zurück zu bs_lug/
7. Einfaches Archive mit „tar“ erzeugen
8. Komprimiertes Archive mit tgz-endung erzeugen
9. Komprimiertes Archive mit bz2-endung erzeugen
10. ls -lh (Wie stark komprimiert?)
11. Inhalt der komprimierten Archive lesen mit less
12. Inhalt lesen ohne less (tar tvf Uebung...)
13. Ordner „Uebung“ löschen und wieder herstellen(tar xvf
14. find /home/udo/bs_lug/Uebung/ -empty -type f -exec rm -rf {} \;
15. ls -lh (Prüfen ob 0-Byte-files raus)

(!) Wir arbeiten immer im Verzeichnis bs_lug damit wir nicht aus versehen was anderes löschen. Also erst dieses anlegen, und hinein wechseln:

```
mkdir bs_lug
cd bs_lug
```

Und hier die zugehörigen Kommandos:

```
mkdir Uebung
cd Uebung/
touch file{001..100}
cp -rv /etc/[^aeiou]* .
dd if=/dev/zero of=100M-file bs=1M count=100
cd ..
tar -cvf Uebung.tar Uebung/ # Archive von Ordner "Uebung" erzeugen
tar -cvzf Uebung.tgz Uebung/
tar -cvjf Uebung.tar.bz2 Uebung/
less Uebung.tar
less Uebung.tar.bz2 # ggf. mit zless oder zcat lesen (müssen u.U. nachinstalliert werden)
tar -tvf Uebung.tar
```

```
rm -rvf Uebung/  
tar -xvf Uebung.tar.bz2  
find Uebung/ -empty -type f exec rm -rf {} \; # alle 0-Byte-files löschen  
ls -lh Uebung/ # prüfen ob alle 0-Byte-files gelöscht sind
```

From:

<https://bs-lug.de/> - **BS-LUG**

Permanent link:

<https://bs-lug.de/vortraege/bash/t-time/start?rev=1590667351>Last update: **2020-05-28 14:02**