Regular Expressions (+ GREP&SED)

Jan Caspar, Aktualisiert 26. Juni 2017, v 1.0.0

Anker

^	Zeilenbeginn
\A	Stringbeginn
$	Zeilenende
\b	Wortgrenze „nur ganze Wörter“ z.B. \bword\b
\<	Beginn eines Wortes
\>	Ende eines Wortes

Quantifizierer

*	0 oder mehr
+	1 oder mehr
?	0 oder 1
{3}	genau 3
{3,}	3 oder mehr
{3,5}	3,4 oder 5

Strings

$‘	vor dem gesuchten String
$’	nach dem gesuchten String
$+	das letzte Vorkommen des gesuchten String
$&	der gesamte gefundene String
$n	nte Gruppe z.B.
$2	„-xyz-“ in /^(abc-(xy-z))$/

Buchstaben

Zeichen-Gruppen

\c	Steuerzeichen
\s	Leerzeichen
\S	kein Leerzeichen [^\s]
\d	Zahl [0-9]
\D	keine Zahl [^0-9]
\w	Wort
\W	kein Wort [^\w]
\x	hexadezimal Zahl
\O	oktal Zahl

Spezielle

\n	neue Zeile
\r	Carriage Return
\t	Tabulator
\v	vertiakler Tabulator
\f	Form Feed
\xxx	Octal Zahl xxx
\xhh	Hex Zahl hh
\	Escapen von Steuerzeichen

Gruppen & Bereiche

.	beliebiges Zeichen außer NewLine
(a|b)	a oder b
(...)	Gruppe
[abc]	Bereich (a oder b oder c)
[^abc]	Nicht a oder b oder c
[a-q]	Bereich von a - q
[A-Q]	Bereich von A - Q
[0-7]	Zahl von 0 bis 7
[a-zA-Z0-9]	Groß- und Kleinbuchstaben und Ziffern von 0-9

Modifizierer

g	Global
i	case insensitive
m	Multiline
s	Singleline
e	evaluate Replacement
U	Ungreedy

Posix

[:upper:]	Großbuchstaben
[:lower:]	Kleinbuchstaben
[:alpha:]	Alle Buchstaben
[:alnum:]	alphanumerisch
[:digit:]	Ziffern
[:xdigit:]	Hexadezimal Ziffern
[:punct:]	Satzzeichen
[:blank:]	Leerzeichen oder Tab
[:space:]	Leerzeichen zwischen Zeichen
[:cntrl:]	Steuerzeichen
[:graph:]	darstellbare Zeichen
[:print:]	dargestellte Zeichen und Leerzeichen
[:word:]	Ziffern, Buchstaben und Underscore

Häufig benötigt

Email (Simpel)

```
[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,4}
```

Datum (TT.MM.JJJJ)

```
(\d{1,2})\.(\d{1,2})\.(\d{4})
```

Web-Addresse (ganzer Pfad)

```
(www\.|http\:\/\/|https\:\/\/)
[a-zA-Z0-9.\/\?\=\&]*
```

Hex Farbe (3 oder 6)

```
\#([a-fA-F0-9]{6}|[a-fA-F0-9]{3})
```

Leerzeichen vorne und hinten

```
(^[\s]+|[\s]+$)
```

Alle Linebreaks

```
(\r?\n|\r)
```

Mehr als 2 Spaces

```
([ ]{2,})
```

Uhrzeit

```
^(?:\d|[01]\d|2[0-3]):[0-5]\d
```

Telefonnummer (Simpel)

```
([01][- .])?(\(\d{3}\)|\d{3})[- .]?\d{3}
[- .]\d{4}
```

Regex mit GREP

GREP unterstützt die Standard PERL Implmentierung von Regex als auch die Extended Implementierung (ERE inkl. POSIX, Metazeichen werden ohne \ akzeptiert. Einfache Anführungszeichen nicht vergessen!
Wird der Ausdruck nicht gefunden, erfolgt keine Ausgabe. Der **Returncode** $? des GREP Kommandos ist dann **1**.

Syntax

grep [-Optionen] regex [dateien]

Optionen

-c	(count) gibt nur die Anzahl passender Zeilen aus
-h	(hide) unterdrückt die Ausgabe der Dateinamen
-i	(ignore) keine Unterscheidung zwischen Groß - und Kleinschreibung
-l	(list) gibt nur die Namen der Dateien aus, die passende Zeilen enthalten
-n	(number) stellt jeder Ausgabezeile die Zeilennummer aus der entsprechenden Eingabedatei voraus.
-o	(only) gibt jeden Match in einer eigenen Zeile aus
-s	(supress) unterdrückt Fehlermeldungen über nicht existierende Dateien
-v	(vice versa) gibt alle Zeilen aus, die nicht zu regex passen
(-E|-P)	extended regular expressions / perl regex

Regex mit SED

Der Suchbereich von SED wird anhand von Adressen und (Regex)Pattern angegeben und erfolgt immer für eine bestimmte Aktion.

Syntax

sed [Argumente] Pattern/Flags Eingabedatei [>> Out]

Argumente

-r	Extended Regular Expressions
-e	Eigentlich ist die Angabe der Option ‘-e’ nicht zwingend notwendig. Wenn diese Option weggelassen wird, dann wird das erste Argument als Script und alle folgenden Argumente als Dateinamen interpretiert.
-n	(implizite Filterung) (oder --quiet oder --silent) Ausgegeben wird dann nur noch, was mit dem ausdrücklichen Print-Kommando (p) angegeben wird.

Flags

g	(global) Ersetze alle gefunden Textpassagen, die zu REGEX passen, mit REPLACEMENT (ohne g wird immer nur die erste in einer Zeile ersetzt)
n	Kann beliebige Zahl sein, ersetzt n-tes Vorkommen von REGEX mit REPLACEMENT.
p	Falls eine Ersetzung durchgeführt wurde, wird die geänderte Zeile ausgegeben
i	Case Insensitive

Adressen

Den meisten Kommandos kann man eine Adresse voranstellen. Eine solche Adresse bestimmt, welche Zeilen mit dem betreffenden Kommando zu bearbeiten sind

sed -e ’1d’ datei.txt
Gibt die erste Zeile von datei.txt **nicht** mit aus.

sed -e ’1,10d’ datei.txt
Gibt die ersten 10 Zeilen von datei.txt **nicht** aus.

sed -e ’10~2d’ datei.txt
jede zweite Zeile, ausgehend von der 10 Zeile wird **nicht** ausgegeben.

sed -e ’/^#.*/d’ irgendwas.config
Zeigt nur nicht auskommentierte (nicht mit # beginnende) Zeilen

Beispiele

Zeilen löschen

Entferne alle Leerzeilen aus einem Dokument
sed -r ’/^$/d’ mydocument
Entferne die erste Zeile eines Dokuments bzw. lösche den Zeilenbereich 7 bis 9.
sed –r ’1d’ mydocument
sed –r ’7,9d’ filename.txt

Suchen und Ersetzen

Entferne alle HTML-Tags aus einem HTML File
sed –r ’s/<[^>]*>//g’ myhtml | sed -r’/^$//d’
1. Löschen alle HTML Tags 2. Lösche alle entstandenen Leerzeilen

Suche alle Zeilen, die einen Bindestrich enthalten und entferne alle Zeichen nach dem Bindestrich bis zum Zeilenende
sed –r ’/-/s/-.*//g’ mydata

Suche im Zeilenbereich 1-4 nach Linux und ersetze Linux durch LINUX
sed -r ’1,4 s/Linux/LINUX/g’ mydata

Ersetze das 2.Vorkommen des Wortes Linux in Linux-Unix
sed –r ’s/Linux/Linux-Unix/2’ mydata

Ersetze alle Vorkommen des Wortes Linux in Linux-Unix
sed –r ’s/Linux/Linux-Unix/g’ mydata