Sed

sed (von stream editor) ist ein nicht-interaktiver Texteditor für die Verwendung auf der Kommandozeile oder in Skripten. sed zählt zu den “Urgesteinen” in der Unix- / Linux-Welt und ist quasi in jeder Linux-Installation (auch Minimalinstallationen) enthalten.

Grundlagen sed

Sed Syntax

```
# sed [Optionen] sed-Skript [Textdatei...]
```

oder

```
# sed [Optionen] [-e sed-Skript | -f Skriptdatei] [Textdatei...]
```

Das Ergebnis wird auf der Standardausgabe ausgegeben. Wird keine Datei angegeben, so wird die Standardeingabe verwendet. Die Syntax von sed-Skripten findet man in der info-Seite von sed oder über die Links am Ende des Artikels. sed-Skripte verwenden reguläre Ausdrücke ähnlich denen von grep. Es ist empfehlenswert, sed-Skripte durch Hochkommas (') zu schützen, damit die Shell Sonderzeichen nicht selbst auswertet.

Sed Parameter

Diese Parameterliste ist unvollständig. Weiteres findet sich in der man-page von sed.

Kurzform	Langform	Beschreibung
-n	–quiet, –silent	Verhindert das automatische Ausgeben des Ergebnisses. Ausgaben erfolgen nur über das Kommando p.
-e Skript	–expression=Skript	Angeben eines sed-Skriptes, mehrere sed-Skripte sind möglich. Bei nur einem sed-Skript kann -e weggelassen werden.
-f Skriptdatei	–file=Skriptdatei	Das sed-Skript steht in der Skriptdatei, nicht auf der Kommandozeile.
-i	–in-place	Die Textdatei wird verändert, anstatt das Ergebnis auf Standardausgabe auszugeben.

Bei der Ausgabe in eine Datei darf die Zieldatei nicht mit der Quelldatei identisch sein, denn bei der Ausführung eines einfachen Redirektors würde „sed“ die Datei zuerst löschen und dann neu anlegen, die Datei wäre leer. Mit der Option [-i] –in-place wird keine Ausgabe erzeugt, sondern gleich die Quelldatei bearbeitet.

Sed Beispiele

Ersetzen von Text in einer Datei

- Jedes Auftreten von “Werner” wird durch “Urs” ersetzt (aber auch “Wernerius” wird zu “Ursius”). Wird **g (global)** weggelassen, wird nur das erste Auftreten in einer Zeile ersetzt.

```
# sed s/Werner/Urs/g Textdatei
```

- Alle Wörter “Werner” werden durch “Urs” ersetzt (nicht “Wernerius”), aber nur in Zeilen, die “Name” enthalten.

```
# sed /Name/s/\bWerner\b/Urs/g Textdatei
```

- Ersetzt alle “Werner” durch “Urs” und gibt nur die betroffenen Zeilen aus.

```
# sed -n s/Werner/Urs/gp Textdatei
```

Entfernen von Zeilen

Zeilen die mit # anfangen, werden entfernt.

```
# sed '/^#/d' Textdatei
```

Zeilen einfügen

- Vor der dritten Zeile wird “Neue Zeile” eingefügt.

```
# sed '3iNeue Zeile' Textdatei
```

- Hier wird eine “Neue Zeile” nach der vierten Zeile eingefügt.

```
# sed '4aNeue Zeile' Textdatei
```

- Hier wird eine “Neue Zeile” nach der letzten Zeile eingefügt.

```
# sed '$aNeue Zeile' Textdatei
```

Reguläre Ausdrücke

Alle Zeilen, die mit “E-Mail:” anfangen, werden ersetzt.

```
# sed 's/^E-Mail:.*$/E-Mail-Adresse ist privat/' Textdatei
```

Bearbeiten von Dateinamen

Normalerweise wird “/” als Trennzeichen verwendet. Es lässt sich aber beliebig austauschen, was beim Bearbeiten von Dateinamen nützlich ist.

```
# sed 's!/home/werner/!/home/urs/!' Textdatei
```

Spaces mit Underlinie ersetzen

```
# echo "This is just a test" | sed -e 's/ /_/g'
```

```
This_is_just_a_test
```

Surprised when sed replacing a token with a URI

As I was replacing tokens with secrets in an init container for a Kubernetes Pod, I was not getting it to work correctly. Not until I actually took a look at the resulting configuration file, I was getting some hints as to what was wrong. I had used sed before when replacing tokens with URI’s, but apparently not URI’s with multiple query-string parameters which are joined by ampersands (&).

Let’s replace the token %link% with a URI

```
$ URI='https://example.com/?num=7'
$ echo 'my_link = %link%' | sed "s|%link%|${URI}|"
my_link = https://example.com/?num=7
```

Seems good. Let’s add another parameter to the URI

```
$ URI='https://example.com/?num=42&type=magic'
$ echo 'my_link = %link%' | sed "s|%link%|${URI}|"
my_link = https://example.com/?num=42%link%type=magic
```

Surprise. What I expected the link to look like was https://example.com/?num=42&type=magic

```
A quick search revealed the culprit;
```

The REPLACEMENT can contain \N (N being a number from 1 to 9, inclusive) references, which refer to the portion of the match which is contained between the Nth \(and its matching \). Also, the REPLACEMENT can contain unescaped & characters which reference the whole matched portion of the pattern space

What I could do here is to first sanitise the URL by backslashing all occurences of & before using it as the replacement

```
URI='https://example.com/?num=42&type=magic'
$ echo 'my_link = %link%' | sed "s|%link%|$(echo "$URI" | sed 's|&|\\&|g')|"
my_link = https://example.com/?num=42&type=magic
```

But to me, that seems clunky. I would rather switch to a different replacer to avoid corner cases like this and excessive code in the init containers. Trying with a perl script instead

```
URI='https://example.com/?num=42&type=magic'
$ echo 'my_link = %link%' | perl -p -e "s|%link%|${URI}|"
my_link = https://example.com/?num=42&type=magic
```

This seems to be okay, but let's add login credentials and see what happens

```
URI='https://user:pass@example.com/?num=42&type=magic'
$ echo 'my_link = %link%' | perl -p -e "s|%link%|${URI}|"
my_link = https://user:pass.com/?num=42&type=magic
```

So this time around perl uses @ for named capture groups. In which case I first have to escape them like \@. So I'm where I started out again.

Let's try with awk

```
URI='https://user:pass@example.com/?num=42&type=magic'
$ echo 'my_link = %link%' | awk "{ gsub(/%link%/, \"$URI\"); print }"
my_link = https://user:pass@example.com/?num=42%link%type=magic
```

Same problem as with sed in that it replaces & with the matching string.

I yield. I found a nice script written by Ed Morton which sanitises both the search and the replacement string before running sed one last time. I added an option to do in-place file replacements.

```
#!/bin/sh
# Modified version of Ed Morton's sedstr:
# https://stackoverflow.com/a/29626460/90674
# stigok, 2019

# In-place option
opts=''
if [ "$1" = '-i' ]; then
  opts='-i ' # note trailing space
  shift
fi

old="$1"
new="$2"
file="${3:--}"
escOld=$(sed 's/[^^]/[&]/g; s/\^/\\^/g' <<< "$old")
escNew=$(sed 's/[&/\]/\\&/g' <<< "$new")
sed ${opts}"s/$escOld/$escNew/g" "$file"
```

This is finally a working solution for me with all the possible special characters of the URI in question, without breaking or triggering unwanted features and variable expansions in sed and bash.

Weiteres

- https://www.linux-community.de/ausgaben/linuxuser/2002/11/sed/
- https://www.computerhope.com/unix/used.htm
- http://kreativgarten.bplaced.net/doku.php?id=sed

Last update: **2019/05/27 16:10**