Standard– und Programmiertools für Unix

Stephan Thesing

0. Organisatorisches

Ehemaliger 'Unix-Kurs' ist umorganisiert worden.

WS: Einsteigerthemen

SS: Oft benötigte Tools

Infos: `http://www.cs.uni-sb.de/~thesing/Unix-Kurs`

Email: `thesing@cs.uni-sb.de`

Übungen:

Tag	Zeit	Ort
Montag	14-16Uhr	R105
Mittwoch	14-16Uhr	R105
Donnerstag	9-11Uhr	R106
Freitag	14-16Uhr	R106

Übungen werden im Web bereitgestellt.

Programm

19.04.	Tips & Tricks, `sed`
26.04.	`awk`
03.05.	`perl`
10.05.	`perl`
17.05.	??
24.05.	Pfingsten
31.05.	`make`
07.06.	`make`
14.06.	LaTeX
21.06.	LaTeX
28.06.	LaTeX
05.07.	`texinfo`
12.07.	(`texinfo`)

1. Tips & Tricks

1.1. Libraries

Libraries sind Sammlungen häufig benutzten Codes

Sie werden beim Linken eines Programmes zu dessen Code hinzugefügt.

Standard-Libraries findet man unter `/usr/lib` und `usr/local/lib`

Es gibt zwei Varianten:

1. Statische Libraries
2. Dynamische Libraries

Generierung statischer Libraries:

Mittels `ar` und `ranlib`, z.B.:

```
ar qv libfoo.a init.o funcs.o search.o
ranlib libfoo.a
```

Dann beim Linken: `-lfoo`, z.B.

```
gcc -o prog prog.o -lfoo
```

Dynamische Libraries:

Häufig benutzte Libraries sollten nur einmal geladen werden.

Referenzen 'hängen' im Programm: Müssen aufgelöst werden

Verschiedene Versionen unterstützbar (Update einfach).

Suchen und Laden des Library-Codes erst zur Laufzeit.

Namen: `libc.so.1.9`

Problem: Finden zur Laufzeit

Suchen von Libraries (statisch + dynamisch)

- 'Standard'-Plätze: `/usr/lib`
- Kommandozeile beim Linken ('`-L/usr/local/lib`')
- Anhand der Pfade in der Umgebungsvariablen `LD_LIBRARY_PATH`

`LD_LIBRARY_PATH` enthält durch Doppelpunkte getrennte Directories, in denen auch zur Laufzeit nach Libraries gesucht wird.

Etwa: `setenv LD_LIBRARY_PATH "$HOME/lib:/usr/local/lib:/usr/lib"`

Anzeigen der Abhängigkeiten: `ldd`

1.2. Environment Variablen

Lokale vs. globale Variablen

tcsh: `set` *var*=*value*, bzw. `setenv` *var* *value*

sh: *var*=*value* und `export` *var*

Globale Variablen, die nützlich sind:

PATH Programme suchen

MANPATH Manual Pages suchen

LD_LIBRARY_PATH Libraries suchen

INFOPATH Info Einträge suchen

tcsh lokale Variablen

rmstar Warnen bei `rm *`

implicitcd Spart das `cd`

fignore Nützlich bei der Completion

autolist Dito

histdup Nützlich für die History-Liste

1.3. Hilfe

Plätze, an denen man Hilfe bekommen kann:

- `--help` Option von GNU Programmen
- Manual Pages (`man` *`program`*)
- Info Einträge (`info` + Blättern)
- Web, z.B. `http://www.delorie.com/gnu/docs`
- Andere Leute

2. sed

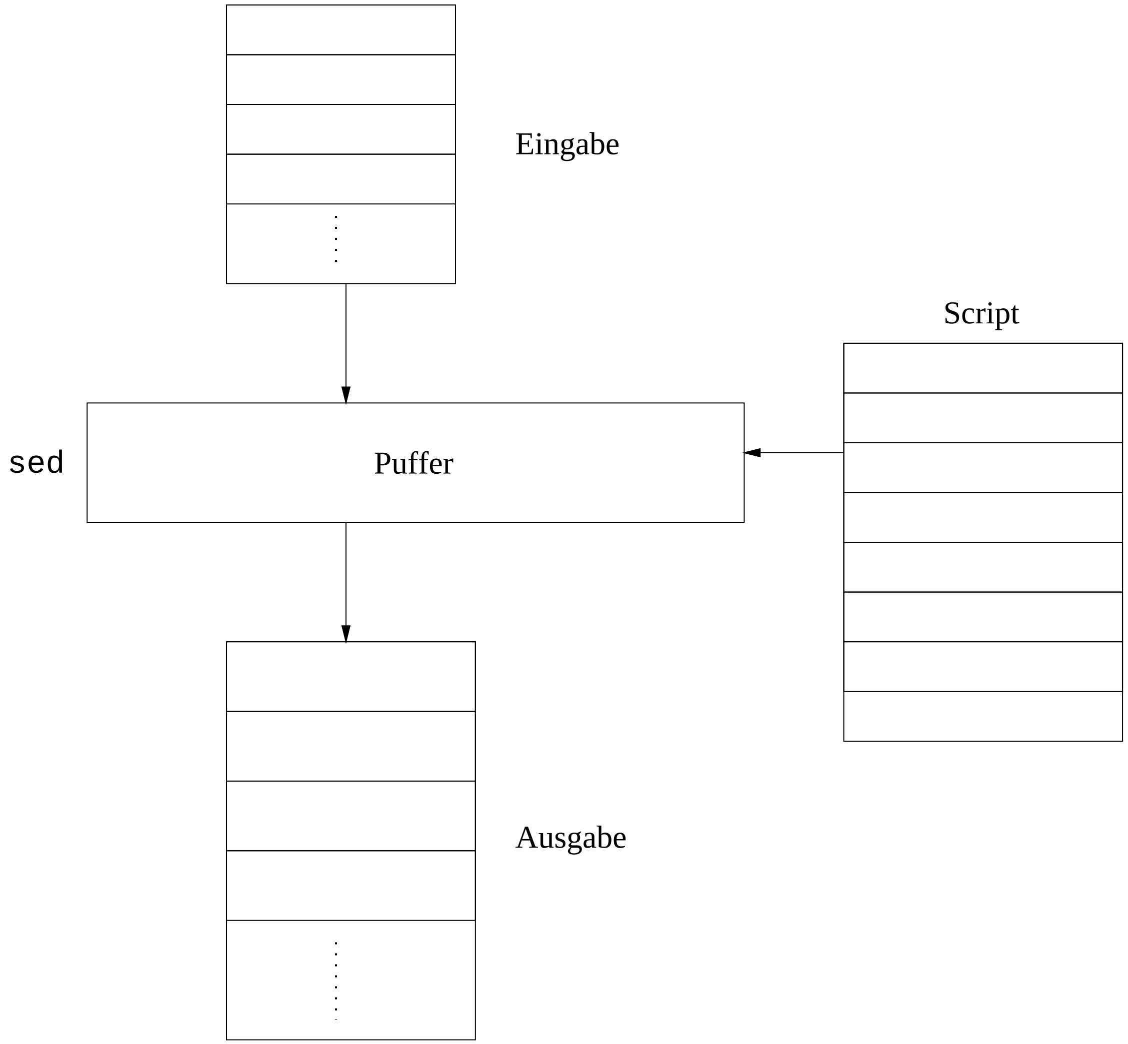
Stream Editor: Batch-Betrieb statt interaktiv

Man gibt Script, das Kommandos enthält.

Selektion zu bearbeitender Zeilen mittels *Adressen* möglich

Aufruf: `sed -e` '*Script*' *[files...]*

`sed -f` *scriptfile* *[files...]*



Option **-n**: Keine automatisches Print.

Z.B. Ersetzung

```
sed -e 's/Uni SB/Uni des Saarlandes/g' Antrag.txt
```

Ersetzung von Text (durch Pattern spezifiziert)

```
sed -e '1,/^$/d' mail
```

Löschen von Text.

Adressen selektieren zu bearbeitende Zeilen.

Zwei wichtige Themen: Pattern und Adressen

2.1. Pattern

`sed` kennt folgende Pattern

`.` Ein beliebiges Zeichen

`[...]` Ein Zeichen aus einer Klasse. Ranges: `[0-9]`, Negation: `[^0-9]`

`*` Wiederholung

`^`,`$` Anfang/Ende des Puffers

`\{`n,m`\}` n- bis m-maliges Auftreten.

Bspe:

```
^[0-9]$
[a-zA-Z][a-zA-Z0-9_]*
```

Pattern werden immer von links auf den längsten Teil der Eingabe gematcht.

Z.B.

`[a-zA-Z]*` matcht im Text `Lokomotiven sind`.

`^.* [a-zA-Z]*` matcht in `Pattern sindnervig`

Problem: `sed` kennt keine Gruppierungen

2.2. Adressen

Dienen zur Selektion von Zeilen.

n Zeilennummern, `$` für die letzte Zeile

/*pattern*/ Ein Pattern

adr,*adr* Ein Bereich

Bspe:

```
1,$
1,/^$/
/^BEGIN/,/^END/
```

2.3. Kommandos

Gruppierung mittels `{, }`:

```
{1,10d
33d
}
```

Delete Kommando (`d`): löscht Zeilen

Print Kommando (`p`): Gibt den Puffer aus, z.B. `sed -n -e '1,10p'`

Substitute Kommando

adr`,`*adr*`s/`*pattern*`/`*repl*`/`*flags*

Ersetzt ersten Match des Patterns durch *repl*.

flags: `g`: Alle Matches werden ersetzt

n: n-ten Match ersetzen

`p`: Nach jeder Substitution den Puffer ausgeben

In Pattern `\(`, `\)`: Gruppen, die im Ersetzungstext als $\backslash n$ verfügbar sind.

`&`: Gesamter Match

Z.B.: `s/\([^ ]*\) *\([^ ]* \)/\1+\2/g`

`s+/*$++`

Append (`a`), Insert (`i`) Kommandos

hängt Text für die Ausgabe hinter/vor dem Puffer an

```
a\
Dies kommt \
dahinter
```

Change (`c`) Kommando: ersetzt Zeile durch Text

```
1,2c\
Statt der ersten beiden Zeilen
```

Aber: Angehängter Text steht nicht im Puffer zur Verfügung.

Next (`n`) Kommando: Liest nächste Zeile

```
/Autor/ {p; n}
s/Thesing/NN/
```

Quit (`q`) Kommando: Beenden

Translate (`y`) Kommando: Umsetzen von Zeichen

```
y/abcdef/ABCDEF/
```

2.4. Sprünge

`:`*label*, `b` *label*, `t` *label*

```
s/a/b/
t weiter
d
:weiter ...
```

2.5. Hold-Space

Zwischenspeicher, kann mit dem Puffer vertauscht werden.

Hold (`h`), Get (`g`) und Exchange (`x`) Kommando.

```
/Titel/ {
h; d
}
/Autor/{
p; g
}
```

2.6. Multi-Line Pattern

Man kann mehrere Zeilen in den Puffer holen.

Newline via (\n) im Substitute Kommando matchbar.

Es gibt verschiedene Kommandos, die auf Multi-Line Puffer arbeiten (siehe Script).

```
s/Uni SB/UdS/g
/Uni$/{ N
s/Uni\nSB/UdS\
/
P
D
}
```

2.7. Beispiele

Instantiierung von Templates: `sed -e 's/@BASDIR@/`*basdir*`/g'`

Extraktion von Daten aus Files: `sed -ne 's/^%Name: \(.*\) $/\1/p'`

Filenamen umformen: `echo` *name* | `sed -e 's+/*$++'`

Namen transformieren: `echo` *name* | sed -e 'y%./-+%___p%'

2.8. Zusammenfassung

`sed` kann zum automatischen Bearbeiten von Files benutzt werden.

Zeilenselektion per Adressen möglich.

Pattern Matching (Adressen, Ersetzung)

Vielfältig verwendbar