Pipes, Redirection und Regular Expressions

LPI Essentials

Andreas B. Mundt
a.mundt@lehrerfortbildung-bw.de

31. Mai 2022

3.2 Suche und Entnahme von Daten aus Dateien

Gewichtung: 3

Beschreibung: Suche und Entnahme von Daten aus Dateien in den Home-Verzeichnissen.

Hauptwissensgebiete:

- Kommando-Pipelines
- Ein-/Ausgabe Umlenkung
- Grundlegende reguläre Ausdrücke (`.` `[ ]` `*` `?`)

Auszugsweise Liste der verwendeten Dateien, Begriffe und Hilfsprogramme:

- `(find)`
- `grep`
- `less`
- `cat, head, tail`
- `sort`
- `cut`
- `wc`

Aus- und Überblick

1. Standardkanäle
   - Ein- und Ausgaben umlenken
2. Dateien lesen, suchen und finden
   - Dateien lesen: `less` und `more`
   - Dateien ausgeben: `cat`, `head` und `tail`
   - Filterprogramme `sort`, `cut`, `wc`
   - Dateien suchen: `find`
3. Dateien durchsuchen
   - Das Programm `grep`
   - Suchmuster: Regular Expressions
4. Aufgaben und Übungen

Standardkanäle

Standardmäßig kann ein Programm Daten über **drei Kanäle** austauschen:

- Kanal 0: Standard-Eingabe, stdin (Daten einlesen): Tastatur
- Kanal 1: Standard-Ausgabe, stdout (Daten ausgeben): Bildschirm
- Kanal 2: Standard-Fehlerausgabe, stderr (Fehlermeldungen ausgeben): Bildschirm

Standardkanäle

Standardmäßig kann ein Programm Daten über **drei Kanäle** austauschen:

- Kanal 0: Standard-Eingabe, stdin (Daten einlesen): Tastatur
- Kanal 1: Standard-Ausgabe, stdout (Daten ausgeben): Bildschirm
- Kanal 2: Standard-Fehlerausgabe, stderr (Fehlermeldungen ausgeben): Bildschirm

Standardkanäle

Standardmäßig kann ein Programm Daten über **drei Kanäle** austauschen:

- Kanal 0: Standard-Eingabe, stdin (Daten einlesen): Tastatur
- Kanal 1: Standard-Ausgabe, stdout (Daten ausgeben): Bildschirm
- Kanal 2: Standard-Fehlerausgabe, stderr (Fehlermeldungen ausgeben): Bildschirm

Standardkanäle

Standardmäßig kann ein Programm Daten über **drei Kanäle** austauschen:

- Kanal 0: Standard-Eingabe, stdin (Daten einlesen): Tastatur
- Kanal 1: Standard-Ausgabe, stdout (Daten ausgeben): Bildschirm
- Kanal 2: Standard-Fehlerausgabe, stderr (Fehlermeldungen ausgeben): Bildschirm

Eingabe umlenken

Das Programm `wc -w` zählt eingegebene Worte (`STRG+D am Ende`):

```
wc -w
Hallo liebe Leute, wie geht's?
5
```

Die Kanäle können umgelenkt werden, z.B. kann mit < eine Eingabe aus einer Datei gelesen werden:

Zähle stattdessen die Wörter in der Datei `test/File`[1]:

```
wc -w < test/File
9
```

[1] Normalerweise würde man hier einfach den Befehl `wc -w <DATEI>` verwenden.

Eingabe umlenken

Das Programm `wc -w` zählt eingegebene Worte (`STRG+D am Ende`):

```
wc -w
Hallo liebe Leute, wie geht's?
5
```

Die Kanäle können umgelenkt werden, z.B. kann mit < eine Eingabe aus einer Datei gelesen werden:

Zähle stattdessen die Wörter in der Datei `test/File`[1]:

```
wc -w < test/File
9
```

[1] Normalerweise würde man hier einfach den Befehl `wc -w <DATEI>` verwenden.

Ausgabekanäle

```
ls -ld Example/ NoAccess/
drwxr-xr-x 3 andi andi 4096 May 25 19:43 Example/
d--------- 2 andi andi 4096 May 25 19:44 NoAccess/
```

Hier werden `stdout` und `stderr` auf dem Bildschirm ausgegeben:

```
ls -l Example/ NoAccess/
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   95 May 31 10:57 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
ls: cannot open directory 'NoAccess/': Permission denied
```

Umleitung: überschreiben oder anhängen

- > leitet eine Ausgabe[2] in eine Datei und überschreibt diese
- >> hängt eine Ausgabe an eine Datei an

```
echo "Hallo" > Example/File1.txt
cat Example/File1.txt
Hallo
```

```
echo "wie geht's?" >> Example/File1.txt
cat Example/File1.txt
Hallo
wie geht's?
```

```
echo "Alles wieder weg!" > Example/File1.txt
cat Example/File1.txt
Alles wieder weg!
```

[2] Ohne weitere Angabe wird immer die Standardausgabe `stdout` verwendet.

Umleitung: überschreiben oder anhängen

- > leitet eine Ausgabe[2] in eine Datei und überschreibt diese
- >> hängt eine Ausgabe an eine Datei an

```
echo "Hallo" > Example/File1.txt
cat Example/File1.txt
Hallo
```

```
echo "wie geht's?" >> Example/File1.txt
cat Example/File1.txt
Hallo
wie geht's?
```

```
echo "Alles wieder weg!" > Example/File1.txt
cat Example/File1.txt
Alles wieder weg!
```

[2] Ohne weitere Angabe wird immer die Standardausgabe `stdout` verwendet.

Umleitung: überschreiben oder anhängen

- > leitet eine Ausgabe[2] in eine Datei und überschreibt diese
- >> hängt eine Ausgabe an eine Datei an

```
echo "Hallo" > Example/File1.txt
cat Example/File1.txt
Hallo
```

```
echo "wie geht's?" >> Example/File1.txt
cat Example/File1.txt
Hallo
wie geht's?
```

```
echo "Alles wieder weg!" > Example/File1.txt
cat Example/File1.txt
Alles wieder weg!
```

[2] Ohne weitere Angabe wird immer die Standardausgabe `stdout` verwendet.

Umleitung: überschreiben oder anhängen

- > leitet eine Ausgabe[2] in eine Datei und überschreibt diese
- >> hängt eine Ausgabe an eine Datei an

```
echo "Hallo" > Example/File1.txt
cat Example/File1.txt
Hallo
```

```
echo "wie geht's?" >> Example/File1.txt
cat Example/File1.txt
Hallo
wie geht's?
```

```
echo "Alles wieder weg!" > Example/File1.txt
cat Example/File1.txt
Alles wieder weg!
```

[2] Ohne weitere Angabe wird immer die Standardausgabe `stdout` verwendet.

Fehlerausgabe umlenken

Fehlermeldungen mit 2> in eine Datei umlenken:


```
ls -l Example/ NoAccess/ 2> test/ErrorLog
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   18 May 31 11:01 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
```

```
cat test/ErrorLog
ls: cannot open directory 'NoAccess/': Permission denied
```

Fehlerausgabe umlenken

Fehlermeldungen mit 2> in eine Datei umlenken:


```
ls -l Example/ NoAccess/ 2> test/ErrorLog
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   18 May 31 11:01 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
```

```
cat test/ErrorLog
ls: cannot open directory 'NoAccess/': Permission denied
```

Ausgabe und Fehlerausgabe umlenken

Ausgabe in eine Datei, Fehlermeldungen in eine Logdatei schreiben:


```
ls -l Example/ NoAccess/ 2> test/ErrorLog 1> /tmp/StdAusgabe
```

```
cat /tmp/StdAusgabe
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   18 May 31 11:01 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
```

```
cat test/ErrorLog
ls: cannot open directory 'NoAccess/': Permission denied
```

Ausgabe und Fehlerausgabe umlenken

Ausgabe in eine Datei, Fehlermeldungen in eine Logdatei schreiben:


```
ls -l Example/ NoAccess/ 2> test/ErrorLog 1> /tmp/StdAusgabe
```

```
cat /tmp/StdAusgabe
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   18 May 31 11:01 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
```

```
cat test/ErrorLog
ls: cannot open directory 'NoAccess/': Permission denied
```

Ausgabe und Fehlerausgabe umlenken

Ausgabe in eine Datei, Fehlermeldungen in eine Logdatei schreiben:


```
ls -l Example/ NoAccess/ 2> test/ErrorLog 1> /tmp/StdAusgabe
```

```
cat /tmp/StdAusgabe
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   18 May 31 11:01 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
```

```
cat test/ErrorLog
ls: cannot open directory 'NoAccess/': Permission denied
```

Fehlerausgabe an Ausgabe anhängen: 2>&1

Man kann Kanäle auch kombinieren, d.h. zusammenfassen:

```
ls -ld Example/ NoAccess/
drwxr-xr-x 3 andi andi 4096 May 25 19:43 Example/
d--------- 2 andi andi 4096 May 25 19:44 NoAccess/
```

```
ls -l Example/ NoAccess/ > /tmp/StdAusgabe 2>&1
```

```
cat /tmp/StdAusgabe
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   18 May 31 11:01 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
ls: cannot open directory 'NoAccess/': Permission denied
```

Fehlerausgabe an Ausgabe anhängen: 2>&1

Man kann Kanäle auch kombinieren, d.h. zusammenfassen:

```
ls -ld Example/ NoAccess/
drwxr-xr-x 3 andi andi 4096 May 25 19:43 Example/
d--------- 2 andi andi 4096 May 25 19:44 NoAccess/
```

```
ls -l Example/ NoAccess/ > /tmp/StdAusgabe 2>&1
```

```
cat /tmp/StdAusgabe
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   18 May 31 11:01 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
ls: cannot open directory 'NoAccess/': Permission denied
```

Fehlerausgabe an Ausgabe anhängen: 2>&1

Man kann Kanäle auch kombinieren, d.h. zusammenfassen:

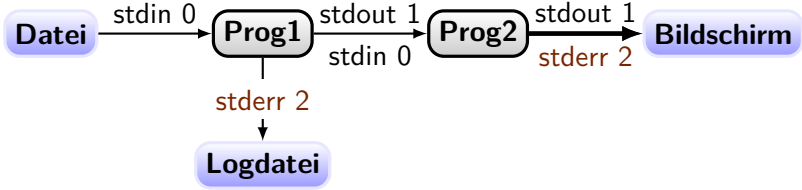
```
ls -ld Example/ NoAccess/
drwxr-xr-x 3 andi andi 4096 May 25 19:43 Example/
d--------- 2 andi andi 4096 May 25 19:44 NoAccess/
```

```
ls -l Example/ NoAccess/ > /tmp/StdAusgabe 2>&1
```

```
cat /tmp/StdAusgabe
Example/:
total 12
drwxr-xr-x 2 andi andi 4096 May 19  2021 Directory
-rw-r--r-- 1 andi andi   18 May 31 11:01 File1.txt
-rwxr-xr-x 1 andi andi   71 May 25 19:43 File2.sh
ls: cannot open directory 'NoAccess/': Permission denied
```

Standard-Ausgabe als Eingabe: Pipe |

Man kann die Ausgabe auch in ein weiteres Programm füttern:



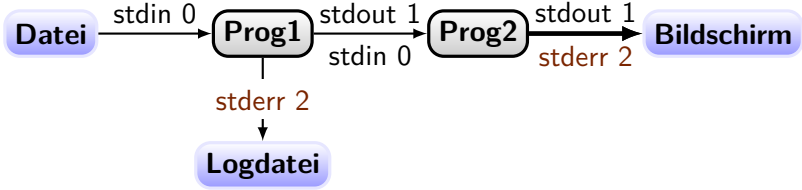

```
grep "exit" /var/log/syslog 2>/tmp/error | wc -l
83
```

```
grep "error" Example/NoAccess/ 2>>/tmp/error | wc -l
0
```

```
cat /tmp/error
grep: Example/NoAccess/: No such file or directory
```

Standard-Ausgabe als Eingabe: Pipe |

Man kann die Ausgabe auch in ein weiteres Programm füttern:



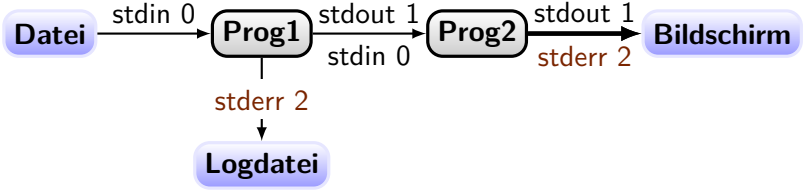

```
grep "exit" /var/log/syslog 2>/tmp/error | wc -l
83
```

```
grep "error" Example/NoAccess/ 2>>/tmp/error | wc -l
0
```

```
cat /tmp/error
grep: Example/NoAccess/: No such file or directory
```

Standard-Ausgabe als Eingabe: Pipe |

Man kann die Ausgabe auch in ein weiteres Programm füttern:




```
grep "exit" /var/log/syslog 2>/tmp/error | wc -l
83
```

```
grep "error" Example/NoAccess/ 2>>/tmp/error | wc -l
0
```

```
cat /tmp/error
grep: Example/NoAccess/: No such file or directory
```

Aus- und Überblick

Dateien lesen: more und less

Der Inhalt von Dateien kann mit den Kommandos `more` oder dem verbesserten[3] `less` dargestellt werden. Einige nützliche Tasten bezgl. `less`:

Tasten	Wirkung
/ <Zeichenkette> Return	Suche abwärts <Zeichenkette>
? <Zeichenkette> Return	Suche aufwärts <Zeichenkette>
n	Nächster Treffer abwärts
N	Nächster Treffer aufwärts
v	Öffnet Editor
F	Verhalten wie `tail -f` (Logfile Analyse)
q	Beenden

[3] `less is more`

Dateien ausgeben: cat

Dateien mit cat auf stdout ausgegeben:

```
echo "== Anfang der Datei ==" > Example/File1.txt
echo "Dies ist die Mitte der Datei," >> Example/File1.txt
echo "mit mehreren Zeilen." >> Example/File1.txt
echo "== Ende der Datei ==" >> Example/File1.txt

cat Example/File1.txt
== Anfang der Datei ==
Dies ist die Mitte der Datei,
mit mehreren Zeilen.
== Ende der Datei ==
```

Werden cat mehrere Dateien übergeben, so werden sie nacheinander ausgegeben (con**cat**enate).

Datei-Anfang ausgeben: head

Anfang von Dateien mit head ausgegeben[4]:

```
head -n 6 /usr/share/hunspell/de_DE.dic
75595
        This is the dictionary file of the de_DE Hunspell dic
        derived from the igerman98 dictionary

        Version: 20161207
```

[4] Option -n: Anzahl der Zeilen, Standardwert 10.

Datei-Ende ausgeben: `tail`

Ende von Dateien mit `tail` ausgeben:

```
tail -n 5 /usr/share/hunspell/de_DE.dic
zynismus/ozm
zypern/Sozm
zypresse/Nozm
zyste/Nozm
zzgl.
```

Anwendung: Logdateien beobachten:

```
tail -n 5 /var/log/syslog
Nov 13 10:30:01 tuxe CRON[5902]: (root) CMD ([ -x /etc/init.d/anacron ] && 
Nov 13 10:33:40 tuxe systemd[1]: Started Run anacron jobs.
Nov 13 10:33:40 tuxe anacron[6015]: Anacron 2.3 started on 2021-11-13
Nov 13 10:33:40 tuxe anacron[6015]: Normal exit (0 jobs run)
Nov 13 10:33:40 tuxe systemd[1]: anacron.service: Succeeded.
```

Mit Option `-f` werden neue Zeilen weiter ausgegeben.

Datei-Ende ausgeben: `tail`

Ende von Dateien mit `tail` ausgeben:

```
tail -n 5 /usr/share/hunspell/de_DE.dic
zynismus/ozm
zypern/Sozm
zypresse/Nozm
zyste/Nozm
zzgl.
```

Anwendung: Logdateien beobachten:

```
tail -n 5 /var/log/syslog
Nov 13 10:30:01 tuxe CRON[5902]: (root) CMD ([ -x /etc/init.d/anacron ] && 
Nov 13 10:33:40 tuxe systemd[1]: Started Run anacron jobs.
Nov 13 10:33:40 tuxe anacron[6015]: Anacron 2.3 started on 2021-11-13
Nov 13 10:33:40 tuxe anacron[6015]: Normal exit (0 jobs run)
Nov 13 10:33:40 tuxe systemd[1]: anacron.service: Succeeded.
```

Mit Option `-f` werden neue Zeilen weiter ausgegeben.

Zeilen sortieren: sort

In /etc/passwd sind alle Benutzer unsortiert aufgeführt:

```
head -n5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

Wir sortieren die Datei:

```
sort /etc/passwd | head -n5
andi:x:1000:1000:Andreas B. Mundt,,,:/home/andi:/bin/bash
apt-cacher-ng:x:113:121::/var/cache/apt-cacher-ng:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:105:113:Avahi autoip daemon,,,:/var/lib/avahi-autoip
avahi:x:116:124:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nol
```

Zeilen sortieren: sort

In /etc/passwd sind alle Benutzer unsortiert aufgeführt:

```
head -n5 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
```

Wir sortieren die Datei:

```
sort /etc/passwd | head -n5
andi:x:1000:1000:Andreas B. Mundt,,,:/home/andi:/bin/bash
apt-cacher-ng:x:113:121::/var/cache/apt-cacher-ng:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
avahi-autoipd:x:105:113:Avahi autoip daemon,,,:/var/lib/avahi-autoip
avahi:x:116:124:Avahi mDNS daemon,,,:/run/avahi-daemon:/usr/sbin/nol
```

Zeilen sortieren: sort

Optionen (Auswahl):

- -g sortiert nach numerischen Werten
- -t<Trenner> Feldseperator
- -k<KEYDEF> im einfachsten Fall die Spalte

```
sort -g -t: -k3 /etc/passwd | head
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
```

Felder extrahieren: cut

Optionen (Auswahl):

- -d<Trenner> Feldseperator
- -f<Felder> Spalten

Uns interessiert nur Login, User-ID und Name:

```
sort /etc/passwd | cut -d: -f1,3,5 | head
andi:1000:Andreas B. Mundt,,,
apt-cacher-ng:113:
_apt:100:
avahi-autoipd:105:Avahi autoip daemon,,,
avahi:116:Avahi mDNS daemon,,,
backup:34:backup
bin:2:bin
colord:118:colord colour management daemon,,,
daemon:1:daemon
Debian-exim:112:
```

Zählen: wc (*word count*)

Zeilen, Wörter und Zeichen zählen:

```
ls -l /etc/passwd
wc /etc/passwd
-rw-r--r-- 1 root root 2347 May 30 19:21 /etc/passwd
  40   67 2347 /etc/passwd
```

Optionen (Auswahl):

- `-l` Zeilen
- `-w` Wörter
- `-m` Zeichen

```
wc -l /etc/passwd; wc -w /etc/passwd; wc -m /etc/passwd
40 /etc/passwd
67 /etc/passwd
2347 /etc/passwd
```

Zählen: `wc` (*word count*)

Zeilen, Wörter und Zeichen zählen:

```
ls -l /etc/passwd
wc /etc/passwd
-rw-r--r-- 1 root root 2347 May 30 19:21 /etc/passwd
  40   67 2347 /etc/passwd
```

Optionen (Auswahl):

- `-l` Zeilen
- `-w` Wörter
- `-m` Zeichen

```
wc -l /etc/passwd; wc -w /etc/passwd; wc -m /etc/passwd
40 /etc/passwd
67 /etc/passwd
2347 /etc/passwd
```

Dateien suchen[5]: `find`

Um nach Dateien oder Verzeichnissen zu suchen verwendet man das Kommando `find`:

find <STARTPFAD> <KRITERIUM>

Einige Suchkriterien:

`-name <NAME>`	hat den Namen <NAME>
`-user <user>`	gehört <USER>
`-group <GROUP>`	gehört <GROUP>
`-type <TYPE>`	Typ <TYPE>, u.a. Datei (d), Verz. (d), symb. Link (l)
`-size <SIZE>`	hat die Größe <SIZE>
`-Xtime <TIME>`	Zugriffszeit, letzte Modifikation, ...
`-perm <XXX>`	nach Zugriffsrechten
`-links <REF>`	Referenzzähler
`-inum <INUM>`	Dateien zur Inode-Nummer <INUM>

Mittels `-exec <COMMAND> '{}'\;` kann man den Befehl <COMMAND> auf alle Treffer anzuwenden.

[5] Nicht mehr im Anforderungskatalog

Aus- und Überblick

Dateien durchsuchen: grep

Mit grep[6] kann man Dateien nach Mustern durchsuchen:

grep <OPTIONEN> <SUCHMUSTER> <DATEI/VERZ.>

Einige Optionen:

`-v`	Treffer invertieren
`-i`	Groß-/Kleinschreibung ignorieren
`-C <ZAHL>`	<ZAHL> Zeilen Kontext ausgeben
`-c`	nur Anzahl Treffer ausgeben
`-n`	Zeile für Treffer ausgeben
`-r, -R`	Verzeichnisse rekursiv durchsuchen
`-E`	Extended Regular Expressions
`-o`	nur den Teil einer Zeile, der passt, ausgeben
`--color=auto`	Treffer farblich markieren (oft Alias)

[6] `man grep`

Dateien durchsuchen: grep

Die Muster unterscheiden sich von den Suchmustern (Globbing) der Shell, wobei zusätzlich verschiedene Implementierungen existieren:

- basic (BRE)
- extended (ERE)
- perl (PCRE)
- python
- ...

Einfachster Fall: Suche nach einer festen Zeichenkette, hier „fff“:

```
grep fff /usr/share/hunspell/de_DE.dic
Rheinschifffahrt/m
Schifffahrt/m
Schifffahrts/hij
griffest/A
rheinschifffahrt/ozm
schifffahrt/ozm
schifffahrts/hke
```

Regular Expressions

Für ***Extended (erweiterte) Regular Expressions*** gelten folgende Regeln:

- Zeichen ohne besondere Bedeutung passen auf sich selbst.
- Eine besondere Bedeutung haben: `. ? * + { } [ ] | \ ^ $`
  - Der Punkt `.` steht für ein beliebiges Zeichen (auch Leerzeichen)
  - Steht hinter einem Element ein `?`, so ist es optional, kann also auftreten, muss es aber nicht.
  - Steht hinter einem Element ein `*`, so kann es beliebig oft, auch gar nicht auftreten.
  - Steht hinter einem Element ein `+`, so muss es mindestens ein mal auftreten.
  - Will man die besondere Bedeutung aufheben, so muss man dem Zeichen im Allgemeinen ein `\` voranstellen: `\. \? \* \+ \{ \} \[ \] \| \\ \^ \$`

Regular Expressions: Wiederholungen

- Mit { } kann man Wiederholungen genau festlegen:
  - {*min*, *max*} Element muss mindestens *min*, höchstens *max* mal vorkommen.
  - {*min*, } Element muss mindestens *min* mal, kann sonst aber bliebig oft vorkommen.
  - {*num*} Element muss genau *num* mal vorkommen.

Ausdruck	passt auf	passt nicht auf
`RE{2,4}G`	`REEG`	`REG`
`RE{2,4}G`	`REEEG`	`REEEEEG`
`RE{2,4}G`	`REEEEG`	
`RE{2,}G`	`REEEEEG`	
`RE{2}G`	`REEG`	`REEEG`

Regular Expressions: Zeichen-Listen

- Mit [ ] kann man eine Liste von Zeichen definieren, ein Minuszeichen definiert dabei einen Bereich. Ein ^ am Anfang der Liste schließt alle folgenden Zeichen aus:

Ausdruck	passt auf	passt nicht auf
`A[BCD]E`	ABE	ABCE
`A[BCD]E`	ACE	ACDE
`A[BCD]E`	ADE	ABCDE
`A[B-Y]Z`	AEZ	AAE
`[A-Za-z]`	alle Buchstaben	Ziffern
`[0-9]`	alle Ziffern	Buchstaben, ...
`[0-9A-Za-z]`	Ziffer/Buchstabe	sonstiges Zeichen
`[^0-9]`	keine Ziffer	Ziffer ...
`[^123]`	4	3
`[123^]`	^	4

Regular Expressions: Zeilenanfang und -ende

- Ein `^` steht für den Zeilenanfang, ein `$` für das Zeilenende:

Ausdruck	passt auf	passt nicht auf
`^Hans`	`Hans` (Zeilenanfang)	`Lieber Hans`
`\.$`	Zeile endet mit Punkt.	

Regular Expressions: Elemente gruppieren, Alternativen

- Mit runden Klammern kann man Elemente gruppieren. Auf das resulierende Element kann dann wieder `? * + { }` angewendet werden:

Ausdruck	passt auf	passt nicht auf
`(ab)+`	`ababab`	`aa bb`
`R(ab){2}G`	`RababG`	RabG

- Mit `|` können Alternativen angegeben werden („oder“):

Ausdruck	passt auf	passt nicht auf
`(ab|xz)`	`ab`	`axby`
`^(#|$)`	`# am Zeilenanfang oder leere Zeile`	

Regular Expressions: Sonstiges[7]

- Für viele Zeichenklassen gibt es Abkürzungen:
  `[:alnum:]`, `[:alpha:]`, `[:cntrl:]`, `[:digit:]`,
  `[:graph:]`, `[:lower:]`, `[:print:]`, `[:punct:]`,
  `[:space:]`, `[:upper:]`, `[:xdigit:]`
  `\w` = `[_[:alnum:]]`, `\W` = `[^_[:alnum:]]`.

Ausdruck	passt auf	passt nicht auf
`[[:alnum:]]`	Ziffern und Buchstaben	sonstige Zeichen

Unterschied Basic/Extended Regular Expressions:

In den *Basic Regular Expressions* verlieren `? + { } | ( )` ihre spezielle Bedeutung, darum muss man dort stattdessen `\? \+ \{ \} \| \( \)` verwenden.

[7] Details findet man unter `man grep` und `man 7 regex`.
Zum ausprobieren: `http://regex101.com/`

Beispiel: Kommentarzeichen

Eine Konfigurationsdatei:

```
wc -l /etc/ssh/sshd_config; head -n 24 /etc/ssh/sshd_config
123 /etc/ssh/sshd_config
#       $OpenBSD: sshd_config,v 1.103 2018/04/09 20:41:22 tj Exp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/bin:/bin:/usr/sbin:/sbin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_ecdsa_key
#HostKey /etc/ssh/ssh_host_ed25519_key

# Ciphers and keying
```

Beispiel: Kommentarzeichen

Welche Zeilen sind relevant?

```
grep -vE "^(#|$)" /etc/ssh/sshd_config
Include /etc/ssh/sshd_config.d/*.conf
ChallengeResponseAuthentication no
UsePAM yes
X11Forwarding yes
PrintMotd no
AcceptEnv LANG LC_*
Subsystem       sftp    /usr/lib/openssh/sftp-server
```

```
grep -v "^\(#\|$\)" /etc/ssh/sshd_config
Include /etc/ssh/sshd_config.d/*.conf
ChallengeResponseAuthentication no
UsePAM yes
X11Forwarding yes
PrintMotd no
AcceptEnv LANG LC_*
Subsystem       sftp    /usr/lib/openssh/sftp-server
```

Beispiel: Kommentarzeichen

Welche Zeilen sind relevant?

```
grep -vE "^(#|$)" /etc/ssh/sshd_config
Include /etc/ssh/sshd_config.d/*.conf
ChallengeResponseAuthentication no
UsePAM yes
X11Forwarding yes
PrintMotd no
AcceptEnv LANG LC_*
Subsystem       sftp    /usr/lib/openssh/sftp-server
```

```
grep -v "^\(#\|$\)" /etc/ssh/sshd_config
Include /etc/ssh/sshd_config.d/*.conf
ChallengeResponseAuthentication no
UsePAM yes
X11Forwarding yes
PrintMotd no
AcceptEnv LANG LC_*
Subsystem       sftp    /usr/lib/openssh/sftp-server
```

Zusammenfassung

1. Standardkanäle
   - Ein- und Ausgaben umlenken
     - Eingabe umlenken
     - Ausgabe und Fehlerausgabe umlenken
     - Fehlerausgabe an Ausgabe anhängen: 2>&1
     - Standard-Ausgabe als Eingabe: Pipe |
2. Dateien lesen, suchen und finden
   - Dateien lesen: `less` und `more`
   - Dateien ausgeben: `cat`, `head` und `tail`
   - Filterprogramme `sort`, `cut`, `wc`
   - Dateien suchen: `find`
3. Dateien durchsuchen
   - Das Programm `grep`
   - Suchmuster: Regular Expressions
4. Aufgaben und Übungen

Aufgaben und Übungen

Standardkanäle und deren Umleitung

1. Probieren Sie die vorgestellten Beispiele und Variationen davon selbst aus.
2. Testen Sie eigene Kommando Konstrukte, die Kanalumlenkung und Pipes verwenden.
3. Starten Sie eine Suche (`find, grep`) die viele Ergebnisse liefert. Nutzen sie die Pipe um die Ausgabe mit `less` angenehm analysieren zu können.

`less` is `more`

1. Machen Sie sich mit den Kommandos `less` und `more` vertraut.
2. Verwenden Sie `less` um ein Logfile (z.B. `/var/log/sys`) zu analysieren. (Suche nach Zeichenkette, beobachten mit F)

Suche mit `find`

1. Machen Sie sich mit dem Kommando `find` und möglichen Suchkriterien vertraut. Probieren Sie verschiedene Suchkriterien aus.

Quellen der Inspiration

- Tobias Heine <tobias.heine@springer-schule.de>
  LINUX ESSENTIALS: Mit Texten arbeiten, Kapitel 10

(22. Mai 2019)