unix / linux

Standardkanäle und Filterkommandos

Skript Seite 109

© 2007...2009 u. heuer

Pipelines

Ziele:

- Die Ein- und Ausgabe-Umlenkung der Shell beherrschen
- Die wichtigsten Filterkommandos kennen und anwenden können

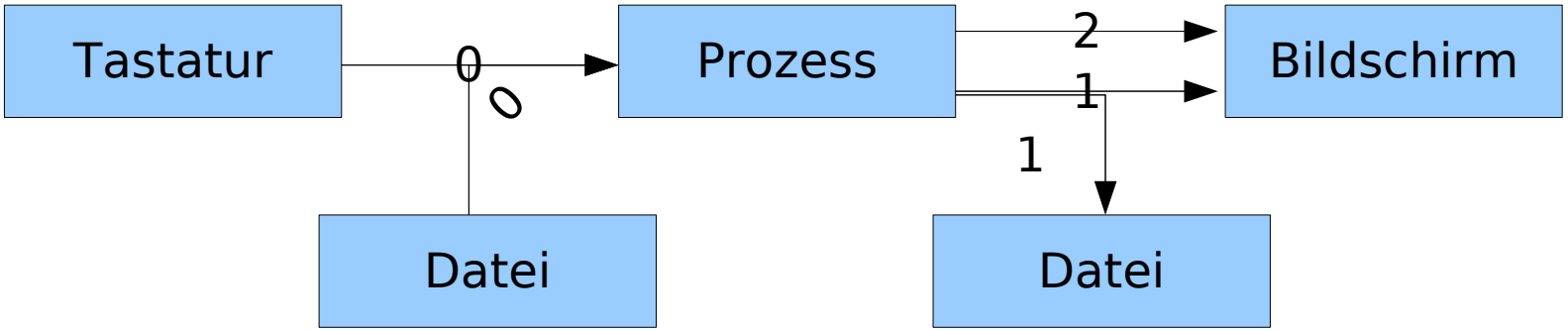
Pipelines

Unter UNIX ist alles eine Datei. Die Tastatur ist eine Datei, der Bildschirm ist eine Datei, die Harddisk ist eine Datei ...

Ein Programm liest die Befehle von der Tastatur und schreibt die Ausgaben auf den Bildschirm. Ein Programm verwendet einen speziellen Kanal um Fehlermeldungen auf dem Bildschirm zu schreiben.

Kürzel	Kanal	Device	Zweck
stdin	0	Keyboard	Programm erhalten Eingaben
stdout	1	Terminal	Programm sendet Ausgaben
stderr	2	Terminal	Programm sendet Fehlermeldungen

Pipelines



Wird nun beispielsweise der Kanal 1 anstatt auf den Bildschirm in eine Datei umgeleitet, so kann die Ausgabe vom Programm später beliebig oft angesehen werden.
Nicht nur die Ausgabe kann umgelenkt werden, sondern auch die Eingabe kann anstatt von der Tastatur auch aus einer Datei gelesen werden.

Ausgabeumleitung

Diese Umleitungen können in der Shell mit den spitzen Klammern '>', '<' angegeben werden.

Ausgabeumleitung

$ **`ls -l > inhalt`**

Die Shell erstellt die Datei "inhalt". Existiert bereits eine Datei mit dem gleichen Namen, so wird diese überschrieben. Danach startet die Shell den Befehl **`ls -la`** und verbindet den Ausgabekanal (stdout) vom neuen Prozess mit der Datei "inhalt".
Ist der Prozess beendet, so wird die Datei "inhalt" geschlossen.

☞ Die Datei "inhalt" wird **vor** dem Prozess-Aufruf angelegt.

Ausgabeumleitung

```
$ ls -l
total 168
-rw-r--r-- 1 heuer staff  2048 2007-04-17 18:18 bar.txt
-rw-r--r-- 1 heuer staff 17408 2007-04-17 18:19 bla.txt
-rw-r--r-- 1 heuer staff  7168 2007-04-17 18:19 foo.txt
-rw-r--r-- 1 heuer staff 67584 2007-04-16 23:34 liste1
-rw-r--r-- 1 heuer staff 67584 2007-04-16 23:34 liste2
$ ls -l > inhalt
$ cat inhalt
total 168
-rw-r--r-- 1 heuer staff  2048 2007-04-17 18:18 bar.txt
-rw-r--r-- 1 heuer staff 17408 2007-04-17 18:19 bla.txt
-rw-r--r-- 1 heuer staff  7168 2007-04-17 18:19 foo.txt
-rw-r--r-- 1 heuer staff     0 2007-04-17 18:20 inhalt
-rw-r--r-- 1 heuer staff 67584 2007-04-16 23:34 liste1
-rw-r--r-- 1 heuer staff 67584 2007-04-16 23:34 liste2
```

Ausgabeumleitung

Will man die Ausgabe an die Datei anhängen, so muss der Operator ">>" verwendet werden.

$ **ls -l >> inhalt**

Existiert die Datei "inhalt", so wird das File geöffnet und der Filepointer ans Ende der Datei gestellt. Existiert die Datei nicht, so wird eine neue - leere - Datei erstellt.

```
$ date >> inhalt
$ cat inhalt
total 168
-rw-r--r-- 1 heuer staff  2048 2007-04-17 18:18 bar.txt
-rw-r--r-- 1 heuer staff 17408 2007-04-17 18:19 bla.txt
-rw-r--r-- 1 heuer staff  7168 2007-04-17 18:19 foo.txt
-rw-r--r-- 1 heuer staff     0 2007-04-17 18:20 inhalt
-rw-r--r-- 1 heuer staff 67584 2007-04-16 23:34 liste1
-rw-r--r-- 1 heuer staff 67584 2007-04-16 23:34 liste2
```

Ausgabeumleitung

Will man den `stderr`-Kanal umleiten, so muss man die Kanalnummer angeben.

Umleiten der Fehlerausgaben in die Datei "fehler"

```
$ ls -l 2> fehler
```

anhängen der Fehlerausgaben in die Datei "fehler"

```
$ ls -l 2>> fehler
```

Umleiten der Ausgabe in die Datei inhalt und der Fehlerausgaben in die Datei "fehler"

```
$ ls -l > inhalt 2> fehler
```

Umleiten der Ausgabe und Fehlermeldung in die Datei "inhalt"

```
$ ls -l > inhalt 2>&1
```

Eingabeumleitung

stdin kann mit dem Operator < umgeleitet werden

Zählen der Wörter in der Datei frosch.txt

```
$ wc -w < frosch.txt
```

Sollen mehrere Dateien als Eingabe verwendet werden so muss der Befehl **`cat datei1 datei2 datei`**$_n$ **`| command`** verwendet werden.

Der Operator << wird für here-Dokumente (oder auch Inline-Daten) verwendet.

```
$ grep Linux <<ENDE
Rosen sind rot,
Veilchen sind blau,
Linux ist super,
das weiss ich genau.
ENDE
```

ergibt dann auf dem Terminal folgende Wahrheit ;)

```
Linux ist super,
```

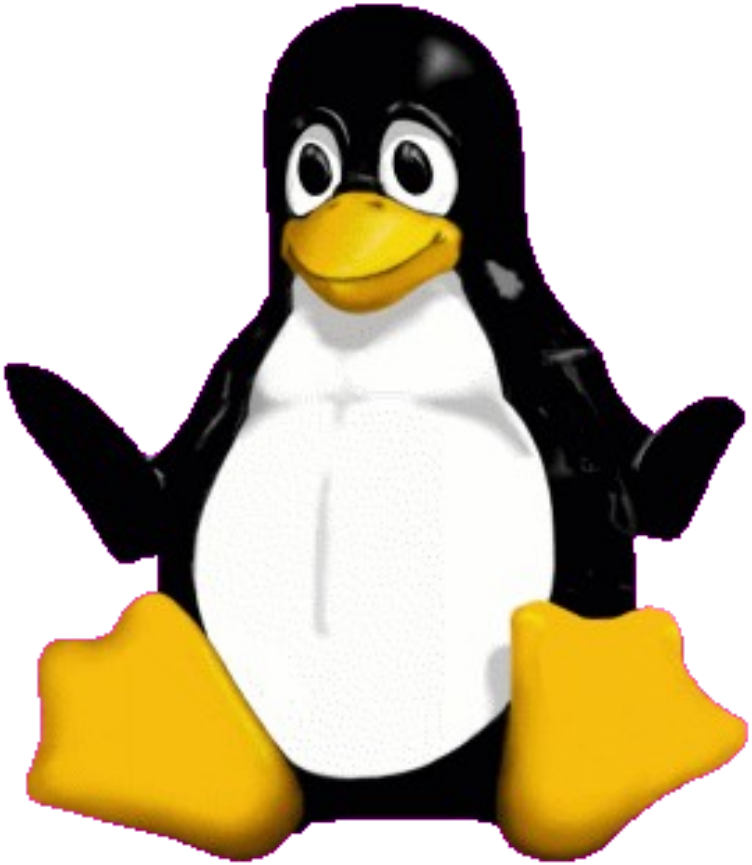
Kombinierte Umleitung

Eingabe- und Ausgabe-Umleitungen können auch kombiniert werden:

Zählen der Wörter in der Datei frosch.txt

```
$ wc -w < frosch.txt > wortzahl
$ cat wortzahl
1255

$ wc -w < frosch.txt >> wortzahl
$ cat wortzahl
1255
1255
```

Fragen ?



Übungen

6.1 Mit der Option `-U` können Sie `ls` dazu bringen, die Verzeichniseinträge unsortiert auszugeben. Trotzdem hat nach dem Kommando "`ls -laU >inhalt`" den Directory-Eintrag für die Datei "inhalt" in der Ausgabedatei immer noch die Länge Null. Woran könnte das liegen?

6.2 Vergleichen Sie die Ausgabe der Kommandos "`ls /tmp`" und "`ls /tmp >ls-tmp.txt`" (wobei wir mit "Ausgabe" im zweiten Fall den Inhalt der Datei `ls-tmp.txt` meinen). Fällt Ihnen etwas auf? Falls ja, wie könnte man das Phänomen erklären?

6.3 Warum ist es nicht möglich, eine Datei in einem einzigen Schritt durch eine neue Version zu ersetzen, etwa mit "`grep xyz datei >datei`"?

6.4 Und was ist das Problem mit "`cat bla >>bla`" (eine nicht leere Datei bla vorausgesetzt)?

6.5 Wie würden Sie in der Shell eine Fehlermeldung so ausgeben, dass sie auf der Standard-Fehlerausgabe landet?

Pipelines

Oft kommt es vor, dass die Ausgabe eines Befehls die Eingabe vom nächsten Befehl ist. Anstatt, das man

```
$ cmd1 > file1
$ cmd2 < file1 > file2; rm file1
$ cmd3 < file2; rm file2
```

eingibt kann man das viel eleganter lösen:

```
$ cmd1 | cmd2 | cmd3
```

Die 2. Variante ist sogar schneller, da das System keine temporären Files (`file1`, `file2`) erstellen und löschen muss und die die Befehle gleichzeitig als Prozesse laufen.

Pipelines

Der senkrechte Strich "|" verbindet die Ausgabe vom Befehl links mit der Eingabe des Befehls rechts (drehen sie den Strich um 90°).
Entsprechend der Verwendung vom senkrechten Strich wird dieser auch als Pipe bezeichnet.

```
$ cmd1 | cmd2 | cmd3
```

Es können praktisch beliebig viele Pipes hintereinander gehängt werden.

☞ **Pipes sind ein wichtiges Fundament von UNIX dar. Damit können aus den einfachen universellen Befehlen komplexe, massgeschneiderte Befehle bauen.**

$ tee

Will man innerhalb einer Befehls-Pipeline die Ausgabe vom Prozess zusätzlich in einer Datei gespeichert haben, kann man den Befehl **tee** (wie ein T-Rohr) dazwischen schalten.

```
$ cmd1 | cmd2 | tee datei1 | cmd3
```

tee kopiert `stdin` auf `stdout` und gleichzeitig in die `datei1`

tee erstellt / überschreibt die Zieldatei. Wird die Option `-a` oder `--append` angegeben, so werden die Daten am Ende der Datei angehängt.

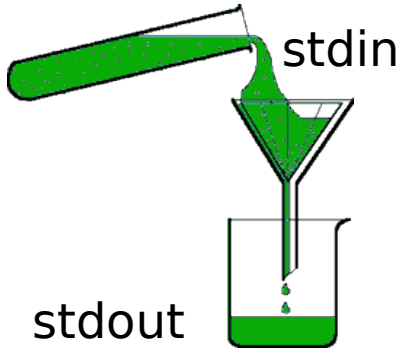
Werden mehrere Dateien als Argument angegeben, so wird `stdin` von `tee` in alle angegeben Dateien kopiert.

Filterkommandos

Unix wird oft als Baukastensystem bezeichnet. Dies weil im "Kasten" (UNIX) viele universelle "Bausteine" (Kommandos), die mit dem geeigneten Mitteln (Optionen) angepasst und zusammen gesetzt (Pipe) werden können, so dass ein neuer, komfortablerer Befehl entsteht.

Beispielsweise kann der Befehl **`man`** aus einer Kombination aus **`find`**, **`nroff`** und **`less`** zusammen bauen (Aktuell verwendet man `nroff` um die Seiten zu formatieren und `less` um das Resultat dann seitenweise zu präsentieren).

Programme, die die Eingabe von stdin lesen und die Ausgabe auf stdout schreiben nennt man auch **Filter-Programme**.



Filterkommandos

Filter-Programme ohne Eingabeumleitung lesen von der Tastatur. Um die Eingabe zu beenden, kann die Tastenkombination `[ctrl][d]` verwendet werden.

Programme die einen Dateinamen als Ausgabeziel erwarten bzw verlangen, können oft mit der Angabe von - (minus) zur Ausgabe auf `stdout` bewegt werden.

Mit `grep` haben sie schon einen Filter Befehl kennen gelernt. Wird bei `grep` keine zu durchsuchende Datei angegeben, so durchsucht `grep` stdin.

$ cat

Dateien zusammen hängen und ausgeben ist mit dem Befehl **cat** (engl. concatenate, verketten) möglich. **cat** liest eine Datei nach der Andern und gibt den Inhalt auf `stdout` aus.

cat kennt verschiedene Optionen:

`-b, --number-nonblank`	Nummeriert alle nicht leeren Zeilen
`-E, --show-ends`	Zeigt ein $ als Zeilenende
`-n, --number`	Nummeriert die Zeilen
`-s, --squeeze-blank`	Zeigt nie mehr als eine Le Leerzeile an

Alle möglichen Optionen können sie unter `cat(1)` nachlesen.

$ cat

Verwenden Sie `cat` nur, wenn sie mehrere Dateien zusammenhängen müssen. Die meisten Filterbefehle akzeptieren auch mehrere Dateinamen als Eingabe.

Beispielsweise:

```
cat datei | grep 'suche etwas'
```

wird besser und effizienter wie folgt geschrieben:

```
grep 'suche etwas' datei
```

$ tac

Der Befehl **tac** ist **cat** rückwärts geschrieben und so gibt **tac** die Dateien zeilenweise rückwärts aus.

tac kennt verschiedene Optionen:

`-b, --before` Der Separater wird vorher anstatt nach der Zeile ausgegeben

`-r, --regex` Der Separater wird als **regex(7)** interpretiert

`-s, --separator=STRING` verwendet STRING als Separater

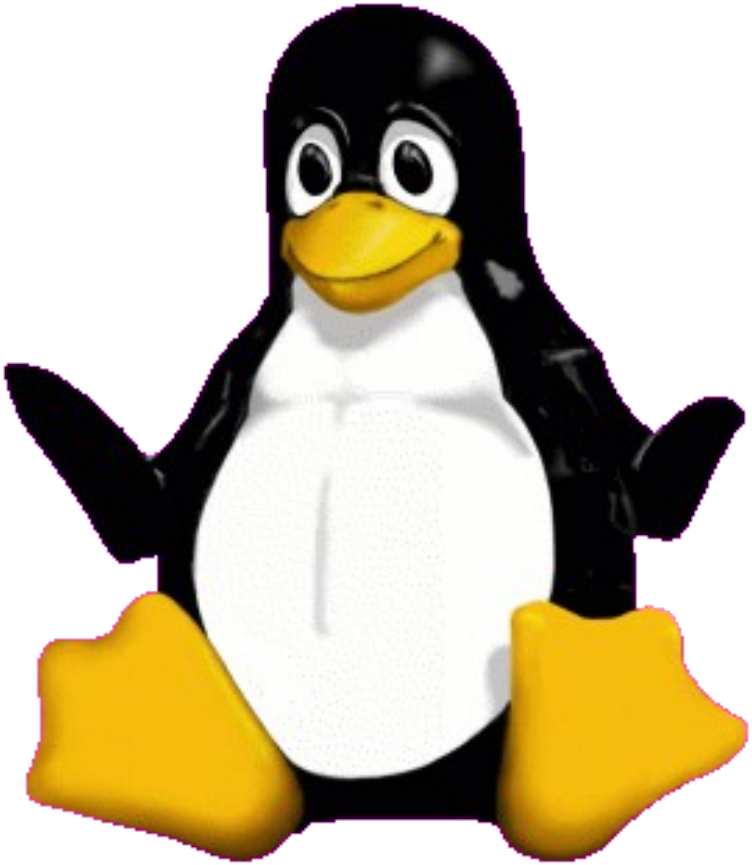
Alle Optionen können unter **tac(1)** nachgelesen werden.

$ tac

```
$ cat /etc/motd
Linux guybrush 2.6.21.1 #2 SMP PREEMPT Sun Apr 29 17:52:32 CEST 2007 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
```

```
$ tac /etc/motd
permitted by applicable law.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent

individual files in /usr/share/doc/*/copyright.
the exact distribution terms for each program are described in the
The programs included with the Debian GNU/Linux system are free software;

Linux guybrush 2.6.21.1 #2 SMP PREEMPT Sun Apr 29 17:52:32 CEST 2007 i686
```

Fragen ?



Übungen

6.6 Wie würden Sie dasselbe Zwischenergebnis gleichzeitig in mehrere Dateien schreiben?

6.7 Wie können Sie prüfen, ob in einem Verzeichnis Dateien mit "merk-würdigen" Namen enthalten sind, etwa solche mit Leerzeichen am Schluss oder mit unsichtbaren Steuerzeichen in der Mitte?

Anfang, Ende von Dateien

Oft interessieren Sie sich für den Anfang oder die letzten paar Zeilen einer Datei. Die Befehle **`head`** (kopf) und **`tail`** (Schwanz) sind da genau die Richtigen.

Optionen:
`-c, --bytes=N` zeigt die ersten / letzten N Bytes an
`-n, --lines=N` zeigt die ersten / letzten N Linien an
Ist `N` negativ, so zeigt **`head`** alle bis auf die N letzten Bytes / Zeilen angezeigt.
nur **`tail`**
`-f, --follow[={name|descriptor}]` tail versucht dem Namen der Datei oder der Inode Nummer der Datei zu folgen.

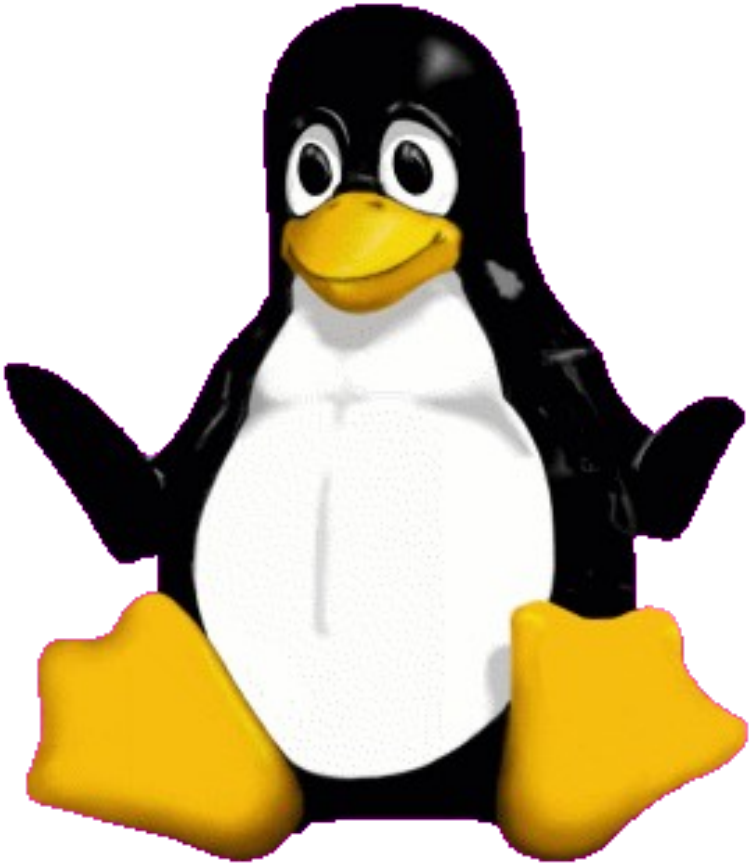
Alle Optionen können unter `head(1)` bzw `head(1)` nachgelesen werden.

$ head; tail

```
$ head -n -3 /etc/motd
Linux guybrush 2.6.21.1 #2 SMP PREEMPT Sun Apr 29 17:52:32 CEST 2007 i686

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
$
```

```
$ tail -n 3 /etc/motd

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
$
```

Fragen ?



Übungen

6.8 Wie würden Sie gezielt nur die 13. Zeile der Eingabe ausgeben?

6.9 Probieren Sie "`tail -f`" aus:
Legen Sie eine Datei an und rufen Sie "`tail -f`" für diese Datei auf. Hängen Sie dann in einem anderen Fenster bzw. auf einer anderen virtuellen Konsole z. B. mit "`echo >>...`" etwas an die Datei an und beobachten Sie die tail-Ausgabe.
Wie sieht das ganze aus, wenn `tail` mehrere Dateien gleichzeitig beobachtet?

6.10 Was passiert mit "`tail -f`", wenn die beobachtete Datei kürzer wird?

6.11 Erklären Sie die Ausgabe der folgenden Kommandos

```
$ echo Hallo >/tmp/hallo
$ echo "Huhu Welt" >/tmp/hallo
```

wenn Sie nach dem ersten echo in einem anderen Fenster das Kommando

```
$ tail -f /tmp/hallo
```

gestartet haben.

Octal-, Hex-Dump

Die Befehle `cat`, `tac`, `head` und `tail` sind, vor allem bei Textdateien anwendbar.

Wenn man binäre Dateien genauer anschauen will so kann man mit dem Befehl **od** (octal dump) der Inhalt der Datei in verschiedenen Formaten angezeigt werden.

Alle Optionen können unter `od(1)`nachgelesen werden. Die wichtigsten sind auf der folgenden Folie aufgelistet:

Octal-, Hex-Dump

Die wichtigsten Optionen von **od**:

`-A r, --address-radix=RADIX`
Basis der Positionsangabe am Zeilenanfang.

`-j o, --skip-bytes=BYTES`
Am Anfang der Eingabe werden o Bytes übersprungen und erst dann mit der Ausgabe begonnen.

`-N n, --read-bytes=BYTES`
Maximal n Bytes ausgeben.

`-t t, --format=TYPE`
Details hierfür finden sich in od(1).

`-v, --output-duplicates`
Gibt auch mehrfach vorkommende Zeilen komplett aus.

`-w w, --width[=BYTES]`
Gibt w Bytes pro Zeile aus; Standardwert ist 16.

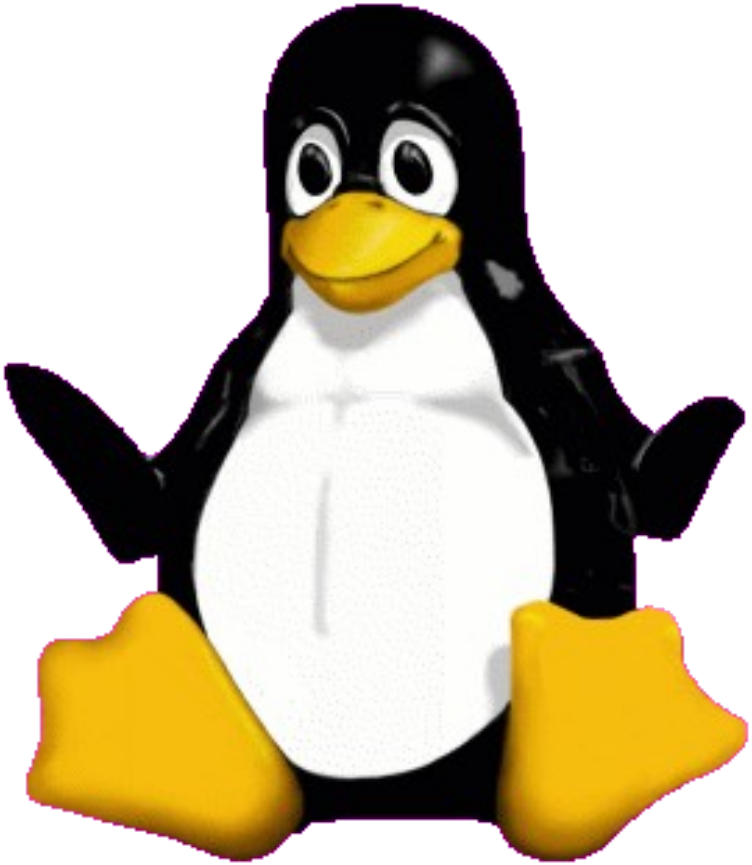
Octal-, Hex-Dump

```
$ od -txC /etc/motd | head -n 2
0000000 4c 69 6e 75 78 20 67 75 79 62 72 75 73 68 20 32
0000020 2e 36 2e 32 31 2e 31 20 23 32 20 53 4d 50 20 50
```

Offset Daten entsprechend der -t Option formatiert

```
$ od -txz /etc/motd | head -n 2
0000000 756e694c 75672078 75726279 32206873  >Linux guybrush 2<
0000020 322e362e 20312e31 53203223 5020504d  >.6.21.1 #2 SMP P<

$ od -Ad -tx2z /etc/motd | head -n 2
0000000 694c 756e 2078 7567 6279 7572 6873 3220  >Linux guybrush 2<
0000016 362e 322e 2e31 2031 3223 5320 504d 5020  >.6.21.1 #2 SMP P<

$ od -Ax -N 64  -tx /dev/zero
000000 00000000 00000000 00000000 00000000
*
000040
```

Fragen ?



Übungen

6.12 Was ist der Unterschied zwischen den **od**-Typangaben "a" und "c"?

6.13 Das Device `/dev/random` liefert zufällige Bytes (siehe Abschnitt 8.4). Verwenden Sie **od** mit `/dev/random`, um der Shell-Variable r eine dezimale Zufallszahl zwischen 0 und 65535 (einschliesslich) zuzuweisen.

$ tr

Um innerhalb von einem Text Zeichen zu ersetzen oder zu löschen kann der Befehl **tr** (translate) verwendet werden.

tr [OPTION]... SET1 [SET2]

tr wird normalerweise mit 2 Zeichenarrays aufgerufen.
tr ersetzt das Zeichen aus dem SET1[n] mit dem entsprechenden Zeichen aus dem SET2[n].

```
$ tr Aa aA < datei; # ersetzt alle A zu a und alle a zu A
```

Ist das SET1 länger, als SET2, so wird für die überzähligen Buchstaben vom SET1 der letzte Buchstabe aus SET2 verwendet

```
$ tr Aa a < datei; # ersetzt alle A oder a zu a
```

$ tr

`tr` kennt folgende Optionen:

`-c, -C, --complement`
ersetzt alle Zeichen, die nicht in (s1) stehen, durch Zeichen aus (s2)

`-d, --delete`
entfernt alle Zeichen, die in (s1) stehen, ohne Ersatz

`-s, --squeeze-repeats`
mehrere gleiche, aufeinander folgende Zeichen aus (s2) werden zusammengefasst

$ tr

Bereiche können zusammen gefasst werden.

anstatt
```
tr ABCDEFGHIJKLMNOPQRSTXYZ NOPQRSTXYZABCDEFGHIJKLM
```

kann man folgenden Befehl verwenden
```
tr A-Z N-ZA-M
```

$ tr

Bei den SET1 bzw SET2 können Sonderzeichen wie folgt angegeben werden:

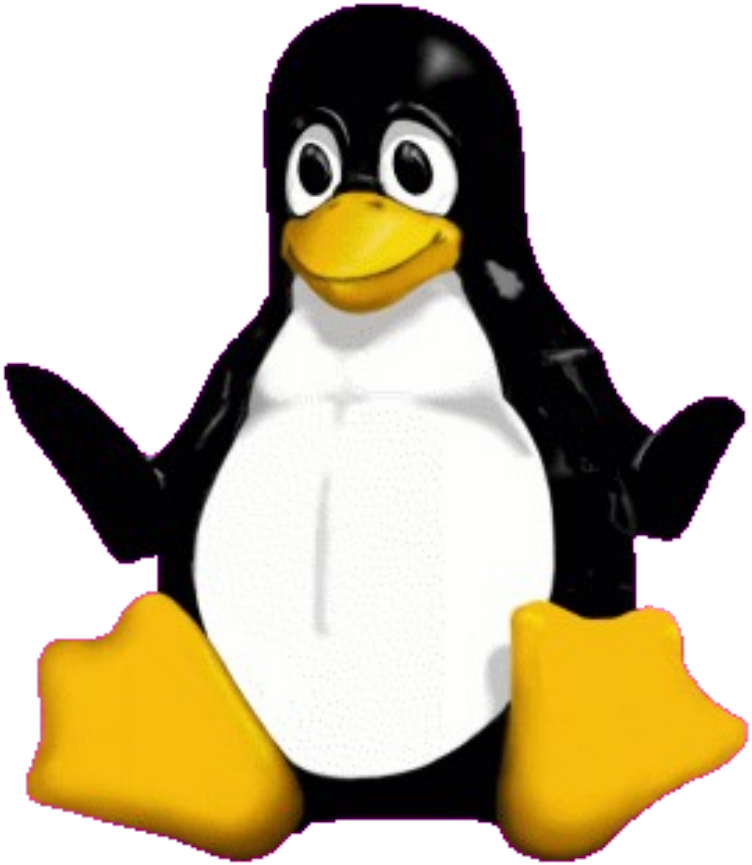
\a	Control-G (ASCII 7), Glockenton
\b	Control-H (ASCII 8), "backspace"
\f	Control-L (ASCII 12), "formfeed"
\n	Control-J (ASCII 10), "linefeed"
\r	Control-M (ASCII 13), "carriage return"
\t	Control-I (ASCII 9), Tabulatorzeichen
\v	Control-K (ASCII 11), vertikaler Tabulator
\kkk	das durch die drei Oktalziffern kkk benannte Zeichen
\\	ein Backslash
[c*n]	in (s2): das Zeichen c n-mal
[c*]	in (s2): das Zeichen c so oft, dass (s2) so lang ist wie (s1)

$ tr

`[:alpha:]`	alle Buchstaben
`[:blank:]`	aller horizontaler Freiplatz
`[:cntrl:]`	alle Kontrollzeichen
`[:digit:]`	alle Ziffern
`[:graph:]`	alle druckbaren Zeichen
`[:lower:]`	alle Kleinbuchstaben
`[:print:]`	alle druckbaren Zeichen
`[:punct:]`	alle Satzzeichen
`[:space:]`	aller horizontaler oder vertikaler Freiplatz
`[:upper:]`	alle Großbuchstaben
`[:xdigit:]`	alle Hexadezimalziffern (0–9, A–F, a–f)
`[:c:]`	alle Zeichen, die äquivalent zu c sind (im Moment nur c selbst)

Beachten sie auch tr(1) wenn sie Klassen verwenden!

Fragen ?



Übungen

6.14 Der berühmte römische Feldherr Julius Caesar verwendete angeblich für die Übermittlung geheimer Botschaften das folgende Verschlüsselungsverfahren:
Der Buchstabe "A" wurde ersetzt durch "D", "B" durch "E" und so weiter; "X" entsprechend durch "A", "Y" durch "B" und "Z" durch "C" (wenn wir mal die heute üblichen 26 Buchstaben zugrundelegen und den Umstand ignorieren, dass die alten Römer kein J, kein K, kein W und kein Y hatten). Stellen Sie Sich vor, Sie sind ein Programmierer in Caesars Legion. Wie lauten die tr-Aufrufe, um eine Botschaft des Feldherrn zu verschlüsseln und wieder zu entschlüsseln?

6.15 Mit welchem tr-Kommando können Sie alle Vokale in einem Text durch einen gegebenen einzigen ersetzen? Denken Sie an das Kinderlied:
DREI CHINESEN MIT DEM KONTRABASS
DRAA CHANASAN MAT DAM KANTRABASS

Übungen

6.16 Wie würden Sie eine Textdatei so umformen, dass alle Satzzeichen entfernt und jedes Wort auf eine einzelne Zeile gestellt wird?

6.17 Geben Sie ein tr-Kommando an, mit dem die Zeichen "a", "z", und "-" aus der Standardeingabe entfernt werden können.

6.18 Wie würden Sie sich davon überzeugen, dass `unexpand` tatsächlich Leer- und Tabulatorzeichen durch eine "optimale" Folge ersetzt?

$ fmt

fmt (format) kann einen Text auf eine beliebige Zeilenlänge (-w) umbrechen.

```
$ fmt -c -w 40 frosch0.txt | head -7
Der Froschkönig oder der eiserne
Heinrich

  In alten Zeiten, als das Wünschen
noch geholfen hat, lebte einmal
ein König, der hatte wunderschöne
Töchter. Die jüngste von ihnen war
```

Die Beispieldateien frosch.txt bzw frosch0.txt können unter https://heuer.org/TS/.49/unix/dateien herunter geladen werden.

$ pr

pr (print) kann einen Text so vorbereiten, dass dieser auf einem Drucker gedruckt werden kann. Der eigentliche Druckjob wird mit `lp` oder `lpr` ausgelöst.

```
$ fmt -c -w 30  frosch0.txt  | pr -n -2

2007-05-20 16:28                                        Page 1

 1   Der Froschkönig oder der
 2   eiserne Heinrich
 3
 4     In alten Zeiten, als das
 5   Wünschen noch geholfen hat,
 6   lebte einmal ein König,
 7   der hatte wunderschöne
 8   Töchter. Die jüngste von
 9   ihnen war so schön, daß die
10   Sonne selber, die doch so
11   vieles schon gesehen hat,
12   sich verwunderte, sooft sie
13   ihr ins Gesicht schien.
14
57   dem Wasser streckte. »Ach,
58   du bist's, alter
59   Wasserpatscher«, sagte
60   sie. »Ich weine über meine
61   goldene Kugel, die mir in d
62   Brunnen hinabgefallen ist.«
63
64     »Sei still und weine
65   nicht«, antwortete der
66   Frosch, »ich kann wohl Rat
67   schaffen. Aber was gibst du
68   mir, wenn ich dein Spielzeug
69   wieder heraufhole?«
70
```

$ pr Optionen

- ***-n*** ergibt eine Darstellung in ***n*** Spalten (nur Zahlen von 2 bis 5 ergeben sinnvolle Ausgaben)
- `-h t` (header) Gibt statt des Dateinamens den Text t in jedem Seitenkopf aus
- `-l n` (length) Legt die Anzahl der Zeilen pro Seite fest, voreingestellt sind 66
- `-n` (number) Versieht jede Zeile mit einer fünfstelligen Zeilennummer, die durch ein Tabulatorzeichen abgetrennt ist
- `-o n` (offset) Rückt den Text n Zeichen vom linken Rand ein
- `-t` (omit) unterdrückt die je fünf Kopf- und Fußzeilen
- `-w n` (width) Legt die Anzahl von Zeichen pro Zeile fest, voreingestellt ist 72

$ nl

nl (number line) nummeriert die nicht leeren Zeilen der Eingabe (stdin oder der Argumente) fortlaufend.

```
$ nl frosch0.txt
     1  Der Froschkönig oder der eiserne Heinrich

     2    In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
     3  König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so
     4  schön, dass die Sonne selber, die doch so vieles schon gesehen hat,
     5  sich verwunderte, sooft sie ihr ins Gesicht schien.
```

`cat -b frosch0.txt` ergibt das gleiche ...

$ nl

... **nl** kann aber sehr viel mehr:

```
$ nl -b a -n rz -w 5 -v 1000 -i 10 frosch.txt
01000 Der Froschkönig oder der eiserne Heinrich
01010
01020 In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
01030 König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so
01040 schön, dass die Sonne selber, die doch so vieles schon gesehen hat,
01050 sich verwunderte, sooft sie ihr ins Gesicht schien.
01060
...
```

`-b a`	Nummeriert alle Zeilen
`-n rz`	Nummeriert rechtsbündig mit führenden Nullen,
`-w 5`	Nummeriert 5stellig
`-v 1000`	Die Zahlen beginnen bei 1000
`-i 10`	Die Zeilennummern werden jeweils um 10 erhöht

$ nl Optionen

`-b s`	(body style) Nummeriert die Zeilen des Körpers gemäss s. Mögliche Werte für s sind a (alle Zeilen nummerieren), t (nur nicht leere Zeilen nummerieren), [default] n (gar keine Zeilen nummerieren) und p(Ausdruck) (nur die Zeilen nummerieren, die auf den regulären Ausdruck (Ausdruck) passen).
`-d p[q]`	(delimiter) Verwendet die beiden Zeichen pq statt »\:« in Trennzeilen. Wurde nur p angegeben, so ist q weiterhin »:«.
`-f s`	(footer style) Formatiert die Fußzeilen gemäss s. Die möglichen Werte für s entsprechen denen bei -b; Standardwert ist n.
`-h s`	(header style) Analog zu -f, für Kopfzeilen.

$ nl Optionen

`-i n`	(increment) Erhöht die Zeilennummer pro Zeile um n.
`-n f`	(number format) Bestimmt das Format für die Zeilennummern. Mögliche Werte für f : **ln** (linksbündig ohne führende Nullen), **rn** (rechtsbündig ohne führende Nullen), **rz** (rechtsbündig mit führenden Nullen).
`-p`	(page) Setzt die Zeilennummer zwischen logischen Seiten nicht auf den Anfangswert zurück.
`-v n`	Beginnt die Nummerierung bei Zeilennummer n.
`-w n`	(width) Die Zeilennummer wird n Zeichen breit ausgegeben.

$ nl

... **nl** kann noch mehr:

In der Datei können die Kopf-Zeilen und Fuss-Zeilen wie folgt gekennzeichnet werden:

```
\:\:\:
Kopf-Zeile
\:\:
Text
\:
Fuss-Zeile
[EOF]
```

Headertext und Footertext werden dann nicht nummeriert.
Die Delimiter `\:` können mit der Option `-d` geändert werden.

$ wc

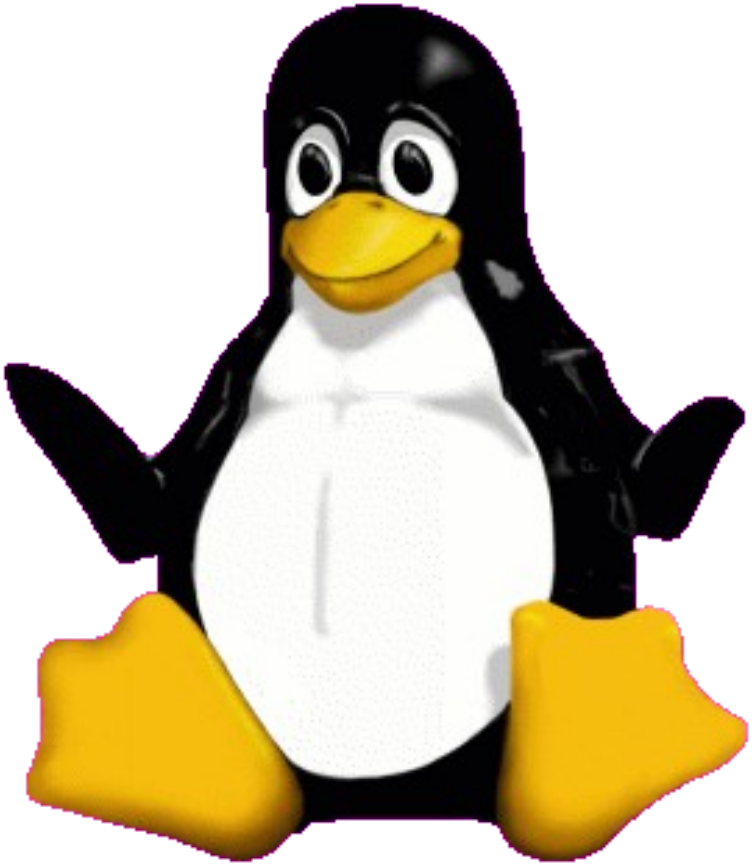
wc (word count) kann neben Wörter auch Zeichen und Linien in Dateien zählen.

```
$ wc frosch.txt
159 1255 7406 frosch.txt
```

wc kennt folgende Optionen

`-l`	(lines)	gibt Anzahl der Zeilen aus
`-w`	(words)	gibt Anzahl der Wörter aus
`-c`	(characters)	gibt Anzahl der Zeichen aus

Fragen ?



Übungen

6.19 Numerieren Sie die Zeilen der Datei frosch.txt in Zweierschritten beginnend bei 100.

6.20 Wie können Sie die Zeilen einer Datei in umgekehrter Reihenfolge numerieren, also etwa so:

```
159 Der Froschkönig oder der eiserne Heinrich
158
157 In alten Zeiten, als das Wünschen noch geholfen hat, lebte einmal ein
156 König, der hatte wunderschöne Töchter. Die jüngste von ihnen war so
```

(Tipp: Zweimal umgedreht gibt wieder das Original).

6.21 Wie unterscheidet sich die Ausgabe des Kommandos »wc a.txt b.txt c.txt" von der des Kommandos »cat a.txt b.txt c.txt | wc"?

$ sort

sort sortiert Dateien zeilenweise anhand dem ASCII Werte der Zeichen.

sort sortiert ohne Optionen lexikographisch über die ganze Zeile

sort kann auch gezielt nach Spalten / Feldern einer Tabelle sortieren. Die Option `-k n,m` sortiert die Datei anhand der Felder n bis m. Die Option `-k` kann mehrmals angegeben werden.

Die Option `-t` definiert das Trennzeichen der Felder (default ist [space]).

$ sort Optionen

`-b`	(blank)ignoriert führende Leerzeichen im Feldinhalt
`-d`	(dictionary) sortiert nach Wörterbuch-Kriterien, d. h. nur Buchstaben, Ziffern und Leerzeichen werden berücksichtigt
`-f`	(fold) keine Unterscheidung von Gross- und Kleinbuchstaben
`-i`	(ignore) nicht druckbare Zeichen werden ignoriert
`-k` (Feld)[,(Feld')] (key)	Sortiere gemäss (Feld) (bis einschliesslich (Feld'))
`-n`	(numeric) betrachtet Feldinhalt als Zahl und sortiert nach dem numerischen Wert, führende Leerzeichen werden ignoriert

$ sort Optionen

`-o datei`	(output) schreibt das Arbeitsergebnis in eine Datei, deren Name hier mit der Ursprungsdatei übereinstimmen darf!
`-r`	(reverse) sortiert absteigend, also von Z nach A
`-t(Zeichen)`	(terminate) das (Zeichen) dient als Feldtrennzeichen
`-u`	(unique) gibt nur die erste einer Folge von identischen Zeilen aus

$ sort Beispiele

Sortieren der Teilnehmer Nachnamen

```
$ sort teilnehmer.dat
```

Fluschinsky	Käthe	Sportfreunde Renntal	57	Damen
Flöttmann	**Fritz**	**Sportfreunde Renntal**	**217**	**Herren**
Langbein	Leni	SV Schnaufenberg	55	Damen
Rasbichel	Katja	TSV Lahmhausen	119	Damen
Rasbichel	Ulla	TSV Lahmhausen	117	Damen
Schleicher	Detlef	TSV Lahmhausen	13	Herren
Schulz	Hugo	SV Schnaufenberg	123	Herren
Schwitz	Sieglinde	Sportfreunde Renntal	93	Damen
Springinsfeld	Karlheinz	TV Jahnstein	154	Herren
Zielinger	Hannes	TV Jahnstein	45	Herren
von Traben	**Gesine**	**TV Jahnstein**	**26**	**Damen**

$ sort Beispiele

Sortieren der Teilnehmer Nachnamen, Umlaute werden berücksichtig

```
$ LC_COLLATE=de_CH sort teilnehmer.dat
Flöttmann       Fritz       Sportfreunde Renntal    217 Herren
Fluschinsky     Käthe       Sportfreunde Renntal    57  Damen
Langbein        Leni        SV Schnaufenberg        55  Damen
Rasbichel       Katja       TSV Lahmhausen          119 Damen
Rasbichel       Ulla        TSV Lahmhausen          117 Damen
Schleicher      Detlef      TSV Lahmhausen          13  Herren
Schulz          Hugo        SV Schnaufenberg        123 Herren
Schwitz         Sieglinde   Sportfreunde Renntal    93  Damen
Springinsfeld   Karlheinz   TV Jahnstein            154 Herren
von Traben      Gesine      TV Jahnstein            26  Damen
Zielinger       Hannes      TV Jahnstein            45  Herren
```

$ sort Beispiele

Sortieren der Teilnehmer Vornamen

```
$ sort -k2,2 teilnehmer.dat
```

Schulz	Hugo	SV Schnaufenberg	123 Herren
Schwitz	Sieglinde	Sportfreunde Renntal	93 Damen
Langbein	Leni	SV Schnaufenberg	55 Damen
Flöttmann	**Fritz**	**Sportfreunde Renntal**	**217 Herren**
Zielinger	**Hannes**	**TV Jahnstein**	**45 Herren**
Rasbichel	**Katja**	**TSV Lahmhausen**	**119 Damen**
Rasbichel	**Ulla**	**TSV Lahmhausen**	**117 Damen**
Schleicher	Detlef	TSV Lahmhausen	13 Herren
Fluschinsky	Käthe	Sportfreunde Renntal	57 Damen
Springinsfeld	Karlheinz	TV Jahnstein	154 Herren
von Traben	Gesine	TV Jahnstein	26 Damen

Als Trennzeichen wird [space] verwendet. Zwischen den Namen und Vornamen sind jedoch mehrere Leerzeichen vorhanden! Darum sind nur die grün markierten Namen richtig sortiert. Die Option -b (ignore blank) kann Abhilfe schaffen

$ sort Beispiele

Sortieren der Teilnehmer Vornamen

```
$ sort -b -k2,2 teilnehmer.dat
Schleicher      Detlef      TSV Lahmhausen          13  Herren
Flöttmann       Fritz       Sportfreunde Renntal    217 Herren
Zielinger       Hannes      TV Jahnstein            45  Herren
Schulz          Hugo        SV Schnaufenberg        123 Herren
Springinsfeld   Karlheinz   TV Jahnstein            154 Herren
Rasbichel       Katja       TSV Lahmhausen          119 Damen
Fluschinsky     Käthe       Sportfreunde Renntal    57  Damen
Langbein        Leni        SV Schnaufenberg        55  Damen
Schwitz         Sieglinde   Sportfreunde Renntal    93  Damen
von Traben      Gesine      TV Jahnstein            26  Damen
Rasbichel       Ulla        TSV Lahmhausen          117 Damen
```

Der rot markierte Eintrag ist falsch einsortiert!
Korrigieren kann man das nur, indem man ein anderes Trennzeichen verwendet, das unter keinen Umständen in den einzelnen Feldern vorkommt - beispielsweise einen Doppelpunkt ':'

$ sort Beispiele

Sortieren der Teilnehmer Vornamen, mit Doppelpunkten als Feldtrenner

```
$ LC_COLLATE=de_CH sort -t: -k2,2 teilnehmer0.dat
Schleicher:Detlef:TSV Lahmhausen:13:Herren
Flöttmann:Fritz:Sportfreunde Renntal:217:Herren
von Traben:Gesine:TV Jahnstein:26:Damen
Zielinger:Hannes:TV Jahnstein:45:Herren
Schulz:Hugo:SV Schnaufenberg:123:Herren
Springinsfeld:Karlheinz:TV Jahnstein:154:Herren
Fluschinsky:Käthe:Sportfreunde Renntal:57:Damen
Rasbichel:Katja:TSV Lahmhausen:119:Damen
Langbein:Leni:SV Schnaufenberg:55:Damen
Schwitz:Sieglinde:Sportfreunde Renntal:93:Damen
Rasbichel:Ulla:TSV Lahmhausen:117:Damen
```

Der Nachteil ist, dass die Datei nicht mehr so einfach zu lesen ist.

$ sort Beispiele

Sortieren der Teilnehmer nach Verein und Nachnamen

```
$ LC_COLLATE=de_CH sort -t: -k3,3 -k1,1 teilnehmer0.dat
Flöttmann:Fritz:Sportfreunde Renntal:217:Herren
Fluschinsky:Käthe:Sportfreunde Renntal:57:Damen
Schwitz:Sieglinde:Sportfreunde Renntal:93:Damen
Langbein:Leni:SV Schnaufenberg:55:Damen
Schulz:Hugo:SV Schnaufenberg:123:Herren
Rasbichel:Katja:TSV Lahmhausen:119:Damen
Rasbichel:Ulla:TSV Lahmhausen:117:Damen
Schleicher:Detlef:TSV Lahmhausen:13:Herren
Springinsfeld:Karlheinz:TV Jahnstein:154:Herren
von Traben:Gesine:TV Jahnstein:26:Damen
Zielinger:Hannes:TV Jahnstein:45:Herren
```

$ sort Beispiele

Sortieren nach der Startnummer (alphabetisch)

```
$ sort -t: -k4,4 teilnehmer0.dat
Rasbichel:Ulla:TSV Lahmhausen:117:Damen
Rasbichel:Katja:TSV Lahmhausen:119:Damen
Schulz:Hugo:SV Schnaufenberg:123:Herren
Schleicher:Detlef:TSV Lahmhausen:13:Herren
Springinsfeld:Karlheinz:TV Jahnstein:154:Herren
Flöttmann:Fritz:Sportfreunde Renntal:217:Herren
von Traben:Gesine:TV Jahnstein:26:Damen
Zielinger:Hannes:TV Jahnstein:45:Herren
Langbein:Leni:SV Schnaufenberg:55:Damen
Fluschinsky:Käthe:Sportfreunde Renntal:57:Damen
Schwitz:Sieglinde:Sportfreunde Renntal:93:Damen
```

Sollte **sort** nummerisch sortieren muss die Option -n angegeben werden.

$ sort Beispiele

Sortieren nach der Startnummer (nummerisch)

```
$ sort -t: -k4,4n teilnehmer0.dat
Schleicher:Detlef:TSV Lahmhausen:13:Herren
von Traben:Gesine:TV Jahnstein:26:Damen
Zielinger:Hannes:TV Jahnstein:45:Herren
Langbein:Leni:SV Schnaufenberg:55:Damen
Fluschinsky:Käthe:Sportfreunde Renntal:57:Damen
Schwitz:Sieglinde:Sportfreunde Renntal:93:Damen
Rasbichel:Ulla:TSV Lahmhausen:117:Damen
Rasbichel:Katja:TSV Lahmhausen:119:Damen
Schulz:Hugo:SV Schnaufenberg:123:Herren
Springinsfeld:Karlheinz:TV Jahnstein:154:Herren
Flöttmann:Fritz:Sportfreunde Renntal:217:Herren
```

Die Option `-n` und `-r` kann einerseits global angegeben werden, andereseits kann sie für jede `-k` Option getrennt angegeben werden.

$ sort Beispiele

Sortieren nach Geschlecht absteigend und Nachname

```
$ LC_COLLATE=de_CH sort -t: -k5r -k1,1 teilnehmer0.dat
Flöttmann:Fritz:Sportfreunde Renntal:217:Herren
Schleicher:Detlef:TSV Lahmhausen:13:Herren
Schulz:Hugo:SV Schnaufenberg:123:Herren
Springinsfeld:Karlheinz:TV Jahnstein:154:Herren
Zielinger:Hannes:TV Jahnstein:45:Herren
Fluschinsky:Käthe:Sportfreunde Renntal:57:Damen
Langbein:Leni:SV Schnaufenberg:55:Damen
Rasbichel:Katja:TSV Lahmhausen:119:Damen
Rasbichel:Ulla:TSV Lahmhausen:117:Damen
Schwitz:Sieglinde:Sportfreunde Renntal:93:Damen
von Traben:Gesine:TV Jahnstein:26:Damen
```

Die Option `-k` kann mehrmals angegeben werden.

$ sort versus uniq

Sortieren und nur die unterschiedlichen Zeilen ausgeben

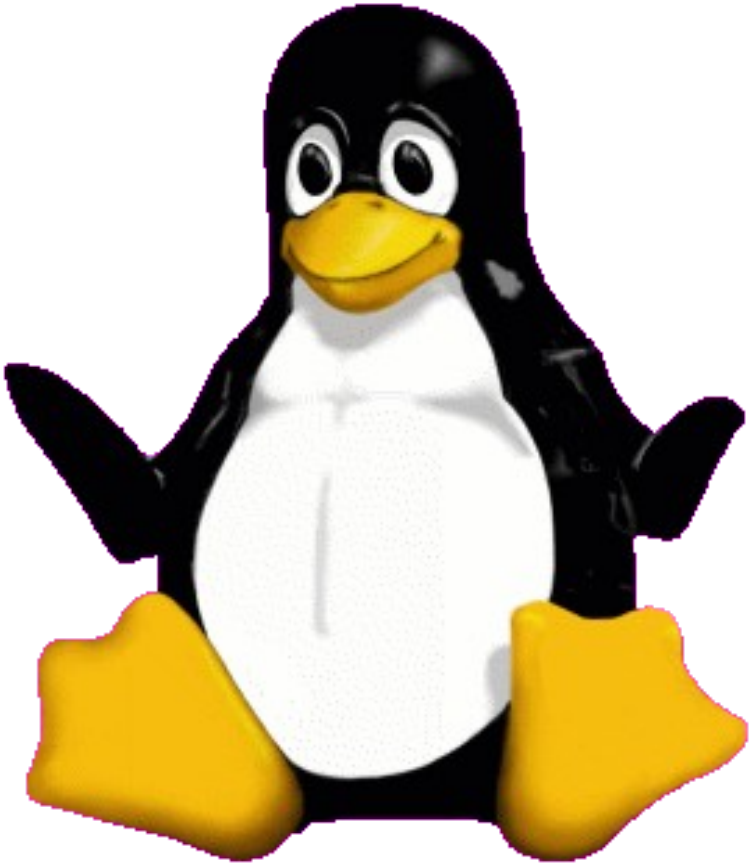
```
$ sort -u uniq-test
Hipp
Hopp
```

der Befehl **uniq** findet nur direkt aufeinander folgende gleiche Zeilen

```
$ uniq uniq-test
Hipp
Hopp
Hipp
Hopp
```

```
$ cat uniq-test
Hipp
Hopp
Hipp
Hipp
Hopp
Hopp
$
```

Fragen ?



Übungen

6.22 Sortieren Sie die Teilnehmerliste in teilnehmer0.dat (der Datei mit Doppelpunkten als Feldtrenner) nach den Vereinsnamen und innerhalb der Vereine nach den Nach- und Vornamen der Spieler (in dieser Reihenfolge).

6.23 Wie können Sie die Teilnehmerliste aufsteigend nach den Vereinsnamen und innerhalb der Vereine absteigend nach der Startnummer sortieren? (Tipp: Dokumentation lesen!)

6.24 Was ist der "Schönheitsfehler", von dem in den Beispielen die Rede ist, und warum tritt er auf?

Übungen

6.25 Ein Verzeichnis enthält Dateien mit den folgenden Namen:
01-2002.txt 01-2003.txt 02-2002.txt 02-2003.txt
03-2002.txt 03-2003.txt 04-2002.txt 04-2003.txt
...
11-2002.txt 11-2003.txt 12-2002.txt 12-2003.txt

Geben Sie ein sort-Kommando an, mit dem Sie die Ausgabe von `ls` in die "chronologisch richtige" Reihenfolge
01-2002.txt
02-2002.txt
...
12-2002.txt
01-2003.txt
...
12-2003.txt
bringen können.

Übungen

6.26 Wie können Sie eine sortierte Liste aller Wörter in einer Textdatei aufstellen? Jedes Wort soll in der Liste nur einmal erscheinen.
(Tipp: Übung 6.16)

$ cut

cut schneidet entsprechende den Optionen Text 'vertikal' aus Dateien oder stdin.

```
cut [option] [file …]
-d, --delimiter          → Trennzeichen der Felder
-c, --characters Bereich → Zeichen
-f, --fields Bereich     → Felder
-b, --bytes Bereich      → Bytes
```

Bereich:

N → das N-te Feld der Zeile
N,M → das N-te und M-te Feld der Zeile
N- → vom N-tem Feldbis ans Ende der Zeile
N-M → vom N-ten bis und mit dem M-ten Feld der Zeile
-M → vom Anfang bis und mit dem M-ten Feld der Zeile

Feld kann auch Zeichen oder Byte sein. Je nachdem welche Option (-c, -f, -b) verwendet wurde

$ cut

Schneidet das 15 und 1 bis 5te Zeichen pro Zeile aus der Datei teilnehmer.dat heraus

```
$ cut -c 15,1-5 teilnehmer.dat
Schulz          Hugo        SV Schnaufenberg        123 Herren
Schleicher      Detlef      TSV Lahmhausen          13  Herren
Flöttmann       Fritz       Sportfreunde Renntal    217 Herren
Springinsfeld   Karlheinz   TV Jahnstein            154 Herren
von Traben      Gesine      TV Jahnstein            26  Damen
...

$ cut -c 15,1-5 teilnehmer.dat
SchulH
SchleD
FlöttF
SprinK
von TG
...
```

$ cut

Schneidet das 15 und 1 bis 5te Zeichen pro Zeile aus der Datei teilnehmer.dat heraus

```
$ cut -c 15,1-5 teilnehmer.dat
SchulH
SchleD
FlöttF
SprinK
von TG
...
```

Die Bereiche werden immer in der Reihenfolge ausgegeben, wie sie in der Datei vorkommen!

$ paste

Mit **paste** können mehrere Dateien zeilenweise zusammen gefügt werden. Die Option **-d** gibt an, welches Trennzeichen eingefügt werden soll (default ist ein Tabulator eingestellt)

```
$ cut -d: -f4 teilnehmer0.dat > startnummer.dat
$ cut -d: -f1-3,5 teilnehmer0.dat |\
 paste -d: startnummer.dat -
123:Schulz:Hugo:SV Schnaufenberg:Herren
13:Schleicher:Detlef:TSV Lahmhausen:Herren
217:Flöttmann:Fritz:Sportfreunde Renntal:Herren
154:Springinsfeld:Karlheinz:TV Jahnstein:Herren
26:von Traben:Gesine:TV Jahnstein:Damen
117:Rasbichel:Ulla:TSV Lahmhausen:Damen
93:Schwitz:Sieglinde:Sportfreunde Renntal:Damen
119:Rasbichel:Katja:TSV Lahmhausen:Damen
55:Langbein:Leni:SV Schnaufenberg:Damen
45:Zielinger:Hannes:TV Jahnstein:Herren
57:Fluschinsky:Käthe:Sportfreunde Renntal:Damen
```

$ paste

Die Option `-s` (seriell) bewirkt, dass die Daten 'horizontal' zusammen gesetzt werden.

```
$ cat liste1
Hund
Katze
Maus
$ cat liste2
Ei
Blut
Kakao
$ paste -s liste*
Hund    Katze   Maus
Ei      Blut    Kakao
```

$ join

join verbindet zwei Dateien mittels eines Schlüsselfeldes (ähnlich wie bei Datenbanken). **join** setzt voraus, dass bei beiden Dateien das Schlüsselfeld sortiert ist.

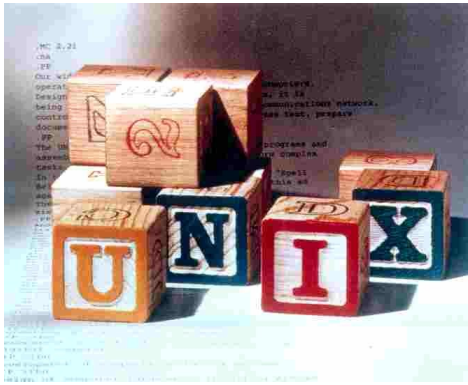
```
$ sort -n zeiten.dat > zeiten-s.dat
$ sort -n tn-startnr.dat |\
  join -t: zeiten-s.dat -
13:8832:Schleicher:Detlef:TSV Lahmhausen:Herren
26:9129:von Traben:Gesine:TV Jahnstein:Damen
45:8445:Zielinger:Hannes:TV Jahnstein:Herren
57:9111:Fluschinsky:Käthe:Sportfreunde Renntal:Damen
93:8641:Schwitz:Sieglinde:Sportfreunde Renntal:Damen
117:8954:Rasbichel:Ulla:TSV Lahmhausen:Damen
```

Die Rangliste ist fast perfekt, Es muss einzig noch nach den Zeiten sortiert werden:

$ join

```
$ sort -n tn-startnr.dat |\
  join -t: zeiten-s.dat - |\
  sort -t: -k2,2n
45:8445:Zielinger:Hannes:TV Jahnstein:Herren
123:8517:Schulz:Hugo:SV Schnaufenberg:Herren
217:8533:Flöttmann:Fritz:Sportfreunde Renntal:Herren
93:8641:Schwitz:Sieglinde:Sportfreunde Renntal:Damen
154:8772:Springinsfeld:Karlheinz:TV Jahnstein:Herren
119:8830:Rasbichel:Katja:TSV Lahmhausen:Damen
13:8832:Schleicher:Detlef:TSV Lahmhausen:Herren
117:8954:Rasbichel:Ulla:TSV Lahmhausen:Damen
57:9111:Fluschinsky:Käthe:Sportfreunde Renntal:Damen
26:9129:von Traben:Gesine:TV Jahnstein:Damen
```

Mit wenige Befehlen geschickt kombiniert können komplizierte Aufgaben gelöste werden. Diese Art von Befehlsverknüpfen ist typisch für UNIX.



Fragen ?

Übungen

6.27 Generieren Sie eine neue Version der Datei teilnehmer.dat (der mit der festen Spaltenbreite), in der die Startnummer und die Vereinszugehörigkeit nicht auftauchen.

6.28 Generieren Sie eine neue Version der Datei teilnehmer0.dat (der mit den durch Doppelpunkt getrennten Feldern), in der die Startnummer und die Vereinszugehörigkeit nicht auftauchen.

6.29 Erzeugen Sie eine Version der Datei teilnehmer0.dat, bei der die Felder nicht durch Doppelpunkte, sondern durch die Zeichenkette ", " (Komma gefolgt von einem Leerzeichen) getrennt sind.

6.30 Wie viele verschiedene Gruppen werden von Benutzern auf Ihrem System als primäre Gruppen benutzt? (Die primäre Gruppe eines Benutzers ist das vierte Feld in der Datei /etc/passwd.)