Linux/Unix-
Kommandoreferenz

Istud quod tu summum putas, gratus est.
(Was Du für den Gipfel hältst, ist nur eine Stufe)

Seneca

Prof. Dr. Helmut Herold

Inhaltsverzeichnis

1 Alphabetische Kommandoübersicht

Im folgenden werden die wichtigsten Kommandos kurz beschrieben, wobei nur die wichtigsten Optionen dazu angegeben sind. Um sich mehr Informationen zu den einzelnen Kommandos anzeigen zu lassen, muss man in der zugehörigen `man`-Seite nachschlagen oder auch nur das Kommando mit der Option **-?** bzw. **--help** aufrufen.

2 Dateikommandos

2.1 Ausgeben von Dateien

cat – Datei(en) nacheinander ausgeben

cat [*optionen*] [*datei(en)*]
Werden keine *datei(en)* beim Aufruf von **cat** angegeben, so liest **cat** den auszugebenden Text von der Standardeingabe; das Ende des Eingabetexts wird dabei mit der Eingabe des EOF-Zeichens (Strg)-D angezeigt.

-v	Nicht-druckbare Zeichen (außer Tabulator-, Neuezeile- und Seitenvorschub-Zeichen) werden bei der Ausgabe sichtbar gemacht. Steuerzeichen werden dabei mit `^x` und das Zeichen *DEL* mit `^?` angezeigt. Nicht-ASCII-Zeichen (höchstwertige Bit gesetzt) werden mit `M-`*x* ausgegeben, wobei *x* das Zeichen ist, das durch die 7 niederwertigen Bits dargestellt wird.
-t	nur mit Option **-v** erlaubt; Tabulatorzeichen werden mit `^I` und Seitenvorschubzeichen mit `^L` angezeigt.
-e	nur mit Option **-v** erlaubt; am Ende jeder Zeile wird ein `$` ausgegeben.

Beim GNU-**cat** von Linux existieren u.a. die folgenden Optionen:

-b	(**--number-nonblank**) alle nicht-leeren Zeilen mit Zeilennummern ausgeben.
-e	äquivalent zu **-vE**.
-n	(**--number**) alle Zeilen mit Zeilennummern ausgeben.
-s	(**--squeeze-blank**) bei mehreren aufeinanderfolgenden Leerzeilen nur eine Leerzeile ausgeben.
-t	äquivalent zu **-vT**.
-A	äquivalent zu **-vET**.
-E	(**--show-ends**) Am Ende jeder Zeile ein `$` ausgeben.
-T	(**--show-tabs**) Tabulatoren als `^I` anzeigen.

- **cat kap*.tex > buch.tex**
  fügt die einzelnen Dateien `kap1.tex`, `kap2.tex` usw. zu einer Gesamtdatei `buch.tex` zusammen.

Es existiert im übrigen auch ein Kommand **tac**, das die beim Aufruf angegebenen *datei(en)* in umgekehrter Reihenfolge (die letzte Zeile zuerst) ausgibt.

more – Datei(en) seitenweise ausgeben

more [*optionen*] [*datei(en)*]

Wenn keine *datei(en)* angegeben sind, liest **more** von der Standardeingabe. Dies ermöglicht den Einsatz von **more** auf der rechten Seite einer Pipe, wie z.B. **ls –l | more**.

-f	(*no fold*) Zeilen nicht abschneiden, wenn sie länger als eine Bildschirmzeile sind.
-s	(*squeeze*) Für aufeinanderfolgende Leerzeilen nur eine Leerzeile ausgeben.
-w	(*wait*) bewirkt, dass **more** beim Erreichen des Dateiendes auf eine Benutzereingabe wartet; normalerweise beendet sich **more** immer beim Erreichen des Dateiendes.
-zeilen	Für *zeilen* ist eine ganze Zahl anzugeben, die die Größe einer auszugebenden Bildschirmseite festlegt.
+zeilennr	Für *zeilennr* ist eine ganze Zahl anzugeben, die die Nummer der Zeile festlegt, ab der die Ausgabe der entsprechenden *datei(en)* am Bildschirm erfolgen soll.

more-Kommandos müssen immer am unteren Bildschirmrand eingegeben und nicht mit (Enter) abgeschlossen werden. Sie entsprechen weitgehend denen von **less**, wobei **less** leistungsfähiger ist und deswegen auch häufiger benutzt wird.

less – Datei(en) seitenweise ausgeben

less [*optionen*] [*datei(en)*]

Wenn keine *datei(en)* angegeben sind, liest **less** von der Standardeingabe. Dies ermöglicht den Einsatz von **less** auf der rechten Seite einer Pipe, wie z.B. **ls –l | less**.

-M	aktuelle Zeilennummer unten anzeigen.
-I	Groß-/Kleinschreibung beim Suchen ignorieren.
-p *text*	Ausgabe der Datei beginnt an der Stelle, wo *text* gefunden wird.
-s	(*squeeze*) Für aufeinanderfolgende Leerzeilen nur eine Leerzeile ausgeben.
-E	(*exit*) bewirkt, dass **less** beim Erreichen des Dateiendes nicht auf Benutzereingabe wartet, sondern sich sofort beendet.
+zeilnr	Für *zeilnr* ist eine ganze Zahl anzugeben, die die Nummer der Zeile festlegt, ab der die Ausgabe der entsprechenden *datei(en)* am Bildschirm erfolgen soll.

less-Kommandos müssen immer am unteren Bildschirmrand eingegeben und meist nicht mit Drücken der (Enter)-Taste abgeschlossen werden. In der folgenden Liste werden die wichtigsten interaktiven **less**-Kommandos vorgestellt, die nicht mit Drücken der (Enter)-Taste abzuschließen sind.

(Leertaste) bzw. **b**	eine Bildschirmseite vor- bzw. zurückblättern.
(Enter)	eine Zeile weiterblättern.
q	**less** beenden.
=	aktuelle Zeilennummer ausgeben.
v	Editor (über Variable `EDITOR` festgelegt) aufrufen; Voreinstellung ist **vi**.
h	Kurzübersicht der **less**-Kommandos ausgeben.
/text	vorwärts nach Vorkommen von *text* suchen.
n	letzte Textsuche wiederholen.

Auch wird unter Linux oft ein Alias (Kurzform) der folgenden Form in der Datei ~/`.alias` definiert: `alias m='less '` oder `alias m='less -MI '`. Falls **less** Probleme mit deutschen Sonderzeichen hat, sollte man in der Datei `~/.profile` folgende Zeile eintragen: `export LESSCHARSET=latin1`.

tail – Datei ab bestimmter Zeile ausgeben

tail [*optionen*] [*datei*]

Ist keine *datei* angegeben, so liest **tail** den Eingabetext von der Standardeingabe.

+n oder *-n*	Ab der *n*. ten Zeile bzw. die letzten *n* Zeilen ausgeben
*+n***b** oder *-n***b**	Ab dem *n*. ten Block bzw. die letzten *n* Blöcke ausgeben
*+n***c** oder *-n***c**	Ab dem *n*. ten Zeichen bzw. die letzten *n* Zeichen ausgeben
-f	bewirkt, dass **tail** in einer Endlosschleife läuft: Nachdem **tail** die geforderte Ausgabe geliefert hat, wartet es eine Sekunde und prüft dann, ob weitere Zeilen zu einer Datei, die z.B. von einem entfernten Rechner geschickt wird oder aber eine Protokolldatei ist, hinzugefügt wurden. Wenn ja, so gibt es diese aus. Dies setzt sich fort, bis **tail** mit einem (Strg)-C abgebrochen wird.

head – Anfangszeilen von Dateien ausgeben

head [*optionen*] [*datei(en)*]

-n	Ersten *n* Zeilen ausgeben (Voreinstellung ist **-10**)

Sind keine *datei(en)* angegeben, so liest **head** die auszugebenden Zeilen von der Standardeingabe. Sind mehr als eine *datei* angegeben, so wird vor jeder einzelnen Datei der Text ==>*dateiname*<== ausgegeben. Nachfolgend einige typische Verwendungen von **head**:

head -9999 *datei1 datei2* **...** > *dateibibliothek*
head -9999 *datei1 datei2* **...** | **lpr**
head -9999 *datei1 datei2* **...** | **less**

nl – Datei mit Zeilennummern ausgeben

nl [*optionen*] [*datei*]

nl liest Zeilen von der angegebenen *datei* oder von der Standardeingabe, falls keine *datei* angegeben ist, und gibt diese Zeilen mit einer Zeilennumerierung (am linken Rand) wieder auf die Standardausgabe aus. **nl** teilt den Eingabetext in so genannte logische Seiten ein, wobei die Zeilennumerierung am Anfang jeder *logischen Seite* wieder neu beginnt. Eine *logische Seite* setzt sich dabei aus einem Seitenkopf (*header*), einem Seiteninhalt (*body*) und einem Seitenfuß (*footer*) zusammen, wobei diese einzelnen Seitenteile auch leer sein, d.h. weggelassen werden können. Für diese drei Seitenteile sind unterschiedliche Zeilennumerierungen möglich, wie z.B. keine Zeilennumerierung für Kopf- und Fußzeilen, aber eine Zeilennumerierung für den Seiteninhalt. Der Beginn der einzelnen Teile einer logischen Seite kann mit folgenden Angaben im Text angezeigt werden:

`\:\:\:`	für Seitenkopf (*header*)
`\:\:`	für Seiteninhalt (*body*)
`\:`	für Seitenfuß (*footer*)

Fehlen solche Angaben im Eingabetext, so betrachtet **nl** den Eingabetext immer als Seiteninhalt (*body*) einer logischen Seite (kein Seitenkopf und kein Seitenfuß). Einige wichtige Optionen sind:

-b*typ*	(*body*) legt fest, welche Zeilen des Seiteninhalts einer logischen Seite zu numerieren sind. Für *typ* kann dabei angegeben werden: **a** (alle Zeilen numerieren), **t** (nur die Zeilen numerieren, die einen druckbaren Text enthalten) oder **n** (keine Zeilen numerieren. Voreinstellung ist: **-bt**
-h*typ*	(*header*) legt fest, welche Zeilen des Seitenkopfs einer logischen Seite zu numerieren sind; für *typ* siehe Option **–b**. Voreinstellung ist: **-hn**
-f*typ*	(*footer*) legt fest, welche Zeilen des Seitenfusses einer logischen Seite zu numerieren sind; für *typ* siehe Option **–b**. Voreinstellung ist: **-fn**
-p	bewirkt, dass die Zeilennumerierung nicht auf jeder logischen Seite neu gestartet, sondern weitergezählt wird.
-s*string*	Der hier angegebene *string* wird verwendet, um die Zeilennummern von den eigentlichen Textzeilen zu trennen. Voreinstellung ist: Tabulatorzeichen
-w*weite*	Die hier für *weite* angegebene Zahl legt die Anzahl der Stellen fest, die für die Zeilennummern bei der Ausgabe zu verwenden sind. Voreinstellung ist: **-w6**
-n*format*	*format* legt hierbei das Format für die Ausgabe der Zeilennummern fest. Für *format* kann dabei folgendes angegeben werden: **ln** (links justieren und führende Nullen nicht ausgeben), **rn** (rechts justieren und führende Nullen nicht ausgeben) oder **rz** (rechts justieren und führende Nullen ausgeben). Voreinstellung ist: **-nrn**

Der Kommandoaufruf **pr -n** erlaubt auch eine Zeilennumerierung, allerdings ist dabei keine Unterscheidung von Seitenkopf, Seiteninhalt und Seitenfuß möglich.

paste – Mehrere Dateien nebeneinander ausgeben

(1) **paste** *datei(en)* (2) **paste -d"***string***"** *datei(en)* (3) **paste -s** [**-d"***string***"**] *datei(en)*

Ist nur eine *datei* angegeben, wird deren Inhalt – wie bei **cat** – am Bildschirm ausgegeben. Wird anstelle einer *datei* ein Minuszeichen (–) angegeben, so steht dieses für eine Zeile aus Standardeingabe. **paste** kennt drei Aufrufarten:

1. Bei (1) werden die *datei(en)* nebeneinander ausgegeben, wobei die Zeilen der einzelnen Dateien durch ein Tabzeichen voneinander getrennt werden. Für anderes Trennzeichen ist die Aufrufform (2) zu verwenden.

2. Bei (2) werden die nebeneinander ausgegebenen Zeilen der einzelnen Dateien mit den in *string* angegebenen Zeichen voneinander getrennt. Als erstes Trennzeichen wird das erste Zeichen in *string* verwendet, als zweites Trennzeichen das zweite usw. Wurden alle Zeichen aus *string* als Trennzeichen verwendet, wird wieder mit erstem Zeichen begonnen und dann alle Zeichen in *string* durchlaufen. Als Zeichen sind dabei auch `\n` (für Neuezeilezeichen), `\t` (für Tabzeichen), `\0` (für leeres Zeichen) und `\\` (für Backlsash) erlaubt.

3. Bei (3) werden nicht die einzelnen Zeilen der angegebenen *datei(en)* nebeneinander ausgegeben, sondern der Inhalt jeder Datei parallel ausgegeben. Ist Option **-d** nicht angegeben, wird als Trennzeichen für die einzelnen Zeilen das Tabulatorzeichen verwendet. Ist die Option **-d** angegeben, so werden wie bei der vorherigen Aufrufform die im *string* angegebenen Zeichen zum Trennen der parallel auszugebenden Zeilen verwendet.

ls | paste - - - - gibt die Dateien des Working-Directorys vierspaltig aus.

od – Datei oktal- bzw. hexadezimal ausgeben

od [*optionen*] [*datei*] [[+]*offset*] [**.**|**b**|**x**]

Sind keine Optionen angegeben, ist **-o** die Voreinstellung. Ist keine *datei* angegeben, liest **od** von Standardeingabe.

-b	(*bytes*) Bytes oktal ausgeben
-c	(*character*) Bytes als ASCII (wenn möglich), sonst in C-Notation ausgeben
-d	(*decimal*) Ausgabe der Worte (nicht Bytes) als vorzeichenlose Dezimalzahlen
-o	(*octal*) Ausgabe der Worte als Oktalzahlen
-v	(*verbose*) Ausgabe aller Datenbytes. Normalerweise werden gleiche Datenbytes zusammengefaßt, d.h. für einen Bereich von Nullbytes wird nicht gesamter Bereich, sondern nur Hinweis ausgegeben, dass hier ein Nullbyte-Bereich vorliegt.
-x	(*hexadecimal*) Worte hexadezimal ausgeben

Ein angegebenes *offset* legt das Byte-Offset in *datei* fest, ab dem **od** mit der Ausgabe beginnen soll. Normalerweise wird *offset* als Oktalzahl interpretiert. Wenn Punkt (**.**) an diese Zahl angehängt ist, wird *offset* als Dezimalzahl interpretiert. Bei angehängtem **x** wird *offset* als Hexadezimalzahl und bei angehängtem **b** wird die angegebene Zahl als ein Block-Offset interpretiert. Pluszeichen (+) ist nur dann vor *offset* anzugeben, wenn keine *datei* angegeben ist. Die am linken Rand stehenden siebenstelligen Zahlen zeigen die Bytennrn. und sind normalerweise Oktalzahlen. Da das Format der Bytenummern über das *offset* festgelegt wird, ist es möglich, sich Bytenrn. nicht nur oktal, sondern auch dezimal oder hexadezimal ausgeben zu lassen. Um eine ganze Datei auszugeben, ist als Offset nur 0 verwendet anzugeben, wie z.B.:

od -c *datei* **0x** (hexadezimale Ausgabe der Bytenummern)
od -c *datei* **0.** (dezimale Ausgabe der Bytenummern)

hd – Datei hexadezimal bzw. oktal ausgeben

hd [*optionen*] [*datei*]

hd gibt den Inhalt der *datei* in hexadezimaler, oktaler, dezimaler und Zeichen-Form aus. Falls keine *datei* angegeben ist, liest **hd** die auszugebenden Zeichen von der Standardeingabe. Falls keine *format*-Angabe festgelegt wird, so entspricht dies der Angabe **-abx -A**, was bedeutet, dass Adressen und Bytes in hexadezimaler Form und zusätzlich auch die zugehörigen ASCII-Zeichen auszugeben sind (siehe auch unten).

-s *offset*	legt *offset* für *datei* fest, ab dem mit Ausgabe zu beginnen ist. Für *offset* darf eine dezimale Zahl, eine hexadezimale Zahl (muss mit `0x` beginnen) oder eine oktale Zahl (muss mit `0` beginnen) angegeben sein. Nach dieser Zahl darf dabei einer der folgenden Buchstaben angegeben sein: **w** (für *words*), **l** (für *long words*), **b** (für *blocks*) oder **k** (für Kilobytes). Um bei einer hexadezimalen Angabe das Anhängsel **b** von der Hexaziffer `b` unterscheiden zu können, darf immer zwischen der Zahl und dem angehängten Buchstaben ein Stern (*) angegeben werden.
-n *zahl*	*zahl* legt die Anzahl von Bytes fest, die von **hd** auszugeben sind. Für *zahl* darf dabei das gleiche Format verwendet werden wie für *offset* bei der Option **-s**.
-f *format*	Über *format* kann sowohl die Ausgabeform (Adressen, Zeichen, Bytes, *Words*, *Long Words*) als auch die Zahlenbasis festgelegt werden. Die Zahlenbasis gibt dabei an, wie die gewählte Ausgabeform darzustellen ist, wie z.B., dass die Bytes in hexadezimaler oder die *Long Words* in oktaler Form auszugeben sind. Es gibt zwei spezielle Ausgabeformen: **t** (Testausgabe) und **A** (ASCII). Ausgabeform- und Zahlenbasis-Buchstaben können dabei beliebig kombiniert und wiederholt werden, um unterschiedliche Zahlenbasen für verschiedene Ausgabeformen festzulegen.

Ausgabeform:

a	(*address*) für Adresse; wird immer in der ersten Zeile ausgegeben.
c	(*character*) druckbare Zeichen werden normal und andere in C-Notation oder als Zahlenwert ausgegeben.
b	(*byte*) für Byte
w bzw. **l**	für *word* oder *long word*
A	(*Ascii*) druckbare Zeichen werden unverändert und nicht druckbare als Punkt (**.**) ausgegeben. Falls noch zusätzlich ein Buchstabe für die Zahlenbasis angegeben ist, so hat dieser hierbei keine Auswirkung.
t	(*Testausgabe*) vor jeder Zeile wird eine Adresse ausgegeben. Steuerzeichen (`0x00` bis `0x1f`) werden durch `^@` bis `^_` dargestellt. Bytes, bei denen das erste Bit gesetzt ist, werden so ausgegeben, als ob das erste Bit nicht gesetzt ist, wobei aber eine Tilde (`~`) vorangestellt wird. Den Zeichen `^`, `~` und `\` wird ein Backslash (`\`) vorangestellt.

Zahlenbasis bei der Ausgabe: **x** (hexadezimale Ausgabe), **d** (dezimale Ausgabe), **o** (oktale Ausgabe). Falls keine Zahlenbasis vereinbart wird, so entspricht dies der Angabe **xdo**.

2.2 Auflisten und Analysieren von Dateien

ls – Dateien auflisten

ls [*optionen*] [*datei(en)*]

Das Kommando **ls** gibt die Dateinamen des Working-Directorys aus, wenn keine *datei(en)* angegeben sind. Sind *datei(en)* angegeben, so werden deren Namen ausgegeben, wenn sie als einfache Dateien existieren. Wenn bei den *datei(en)* Namen von Directories angegeben sind, so werden die Namen der Dateien ausgegeben, die in diesem Directory vorhanden sind.

Die Namen werden dabei von **ls** immer alphabetisch sortiert ausgegeben.

Namen, die mit **.** (Punkt) beginnen, werden normalerweise nicht aufgelistet, sondern nur, wenn die Option **-a** angegeben ist.

-a	Liste alle Einträge in einem Directory, auch Namen, die mit . (Punkt) beginnen.
-c	Datum des letzten letzten Änderung des I-nodes (bei Kombination mit **-l**) ausgeben bzw. dieses Datum zum Sortieren verwenden (bei Kombination mit **-t**).
-C	Dateinamen nicht untereinander, sondern nebeneinander ausgeben; ist die Voreinstellung.
-d	Bei Directories nur deren Namen und nicht deren Inhalt listen (meist mit **-l** kombiniert, um Zugriffsrechte eines Directorys auszugeben). So listet z.B. **ls –ld ./*** alle Directories (keine Dateien) des Working-Directorys auf.
-F oder **-p**	Hinter jedem Directorynamen einen Slash **/**, hinter jeder ausführbaren Datei einen Stern ***** und hinter jedem symbolischen Link einen Klammeraffen **@** angeben.
-i	vor jedem Dateinamen dessen I-node Nummer ausgeben.
-I*ra*	Die durch den regulären Ausdruck *ra* abgedeckten Dateinamen werden nicht aufgelistet. So bewirkt z.B. der Aufruf **ls –I*.o**, dass Dateien mit der Endung **.o** nicht angezeigt werden.
-l	Datei-Informationen im „Langformat“ ausgeben
-L	Bei symbolischen Links nicht den Link selbst, sondern die Datei bzw. das Directory ausgeben, auf das der Link zeigt.
-m	Dateinamen in einer Zeile (mit Komma getrennt) ausgeben.
-n	wie **-l**, nur dass anstelle des Login-Namens des Eigentümers dessen User-ID und Group-ID ausgegeben wird.
-o	wie **-l**, nur dass die Gruppe nicht ausgegeben wird.
-r	Reihenfolge der Ausgabe umkehren.
-R	Alle Subdirectories und Dateien ab der angegebenen Directory-Ebene rekursiv auflisten
-s	Dateigröße nicht in Bytes, sondern in Blöcken ausgeben.
-S	Dateien nach ihrer Größe auflisten (größte Datei zuerst).
-t	Bei der Ausgabe nach Zeitpunkt der letzten Änderung (zuletzt modifizierte zuerst) und nicht alphabetisch sortieren.
-u	Datum des letzten Zugriffs anstelle der letzten Änderung (bei Kombination mit **-l**) ausgeben bzw. dieses Datum zum Sortieren verwenden (bei Kombination mit **-t**).
-x	Dateinamen nicht untereinander, sondern nebeneinander ausgeben; dabei sind sie horizontal und nicht vertikal sortiert.
-X	Dateinamen nach ihrer Endung (Zeichenkette nach dem letzten Punkt) sortieren.
-1	(*Ziffer 1*) Dateien nicht nebeneinander, sondern untereinander ausgeben.

lsattr – Anzeigen der Attribute einer Datei (unter Linux)

lsattr [*optionen*] *datei(en)*

lsattr zeigt die vom ext2-Dateisystem zusätzlich unterstützten Attribute sowie die Version einer Datei an.

-a	Liste alle Einträge in einem Directory, auch Namen, die mit . (Punkt) beginnen.
-R	Alle Subdirectories und Dateien ab der angegebenen Directory-Ebene rekursiv auflisten

wc – Zeichen, Wörter, Zeilen einer Datei zählen

wc [*optionen*] [*datei(en)*]

Mit dem Kommando **wc** können Zeilen, Wörter und Zeichen eines Textes gezählt werden. Voreinstellung für das Kommando **wc** ist:

- keine Optionen angegeben: es wird alles (Zeilen, Wörter und Zeichen) gezählt.
- keine Dateinamen angegeben: Es wird der Text von der Standardeingabe (bis zur Eingabe von (Strg)-D) ausgewertet.
- Sind mehrere *datei(en)* angegeben, so werden alle einzeln ausgewertet und abschließend wird ein Gesamtergebnis über die Anzahl aller Zeilen, Wörter und Zeichen ausgegeben.

-l	(*lines*) es werden nur die Zeilen gezählt
-w	(*words*) es werden nur die Wörter gezählt
-c	(*characters*) es werden nur die Zeichen gezählt

Diese Optionen können beliebig kombiniert werden. **wc** wird oft rechts von einer Pipe eingesetzt, um die Ausgaben des vorherigen Kommandos auszuwerten, wie z.B.:

- **ls | wc –l**: zählt die Anzahl der Dateien im Working-Directory.
- **find ~ -type f -print | wc –l**: liefert die Gesamtzahl aller regulären Dateien im Home-Directorybaum.

type – Klassifizieren von Kommandos

type *kommandoname(n)*

type klassifiziert alle angegebenen *kommandoname(n)*, indem es u.a. ausgibt, ob es sich dabei um ein Programm, ein in die Shell fest eingebautes Kommando (*builtin*) oder um einen Alias-Kurznamen handelt.

```
$ type cat alias m cd (Enter)
cat is /bin/cat
alias is a shell builtin
m is aliased to 'less '
cd is a shell builtin
$
```

file – Analysieren des Inhalts von Dateien

file [*optionen*] [*datei(en)*]

Das Kommando **file** kann dazu verwendet werden, um die angegebenen *datei(en)* auf ihren Inhalt hin überprüfen zu lassen. So gibt dieses Kommando an, ob die einzelnen *datei(en)* z.B. ein C-Programm, einen ASCII-Text, ein ausführbares Programm usw. enthalten. Um den Inhalt einer Datei zu identifizieren, verwendet **file** die Datei `/etc/magic` bzw. `/usr/share/misc/magic`, die angibt, welche Bytes einer Datei zu untersuchen sind, und welche Bytemuster dann auf den Inhalt dieser Datei schließen lassen.

-f *fdatei*	Die nach **-f** angegebene *fdatei* enthält in diesem Fall die Namen der zu untersuchenden Dateien.
-m *mdatei*	es wird die nach **-m** angegebene *mdatei* (anstelle von `/etc/magic` bzw. `/usr/share/misc/magic`) als *Magic*-Datei verwendet.
-L	bei symbolischen Links keine Information über den Link selbst, sondern über die Zieldatei ausgeben.

cksum / sum – Ermitteln einer Prüfsumme für eine Datei

cksum *datei*
sum [**-r**] *datei*

Eine Möglichkeit zu prüfen, ob eine Datei verändert wurde, ist alle Bytewerte des Dateiinhalts aufzuaddieren und festzustellen, ob die berechnete Summe gleich der früher berechneten Summe ist. Eine solche Summe wird auch Prüfsumme (*check sum*) genannt.

- **cksum** ermittelt eine solche Prüfsumme für die angegebene *datei* und gibt diese Prüfsumme zusammen mit der Länge der Datei (in Bytes) aus.
- **sum** berechnet auch eine solche Prüfsumme für die angegebene *datei* und gibt diese Prüfsumme zusammen mit der Anzahl der Blöcke, die von dieser Datei belegt werden, aus.

-r	bewirkt, dass ein anderer (alternativer) Algorithmus zur Berechnung der Prüfsumme verwendet wird.

cksum und **sum** werden oft verwendet, um schnell festzustellen, ob zwei Dateien identisch sind, wie etwa nach einer Datenübertragung in einem Netzwerk oder bei Verdacht auf Virenbefall. Der Systemverwalter kann **cksum** bzw. **sum** verwenden, um festzustellen, ob sich jemand an den Systemdateien zu schaffen machte. **cksum** ist die neuere und bessere Variante.

```
$ sum add2.c (Enter)
17366 1 add2.c
$ sum add2.c (Enter)
17366 1 add2.c
$ cat >>add2.c (Enter)
/* Ende */ (Enter)
(Strg)-D
$ sum add2.c (Enter)
17998 1 add2.c
$
```

Ein weiteres Kommando zum Ermitteln von Prüfsummen ist **md5sum**.

2.3 Kopieren, Umbenennen und Löschen von Dateien

cp – Dateien kopieren

cp [*optionen*] *datei1 datei2*
cp [*optionen*] *datei(en) directory*

- Die erste Aufrufform kopiert den Inhalt von Datei *datei1* in eine Datei mit Namen *datei2*. Falls die Datei *datei2* bereits existiert, so wird sie überschrieben, wenn dies die Zugriffsrechte dieser Datei zulassen bzw. nicht die Option **–i** angegeben ist. Eigentümer dieser neuen Datei wird der Benutzer, der dieses Kommando angab. Zwar werden die Zugriffsrechte mitkopiert, aber wenn sich der Eigentümer und vielleicht sogar die Gruppe dieser Datei ändert, dann sind diese Zugriffsrechte auf den neuen Eigentümer und die neue Gruppe anzuwenden; wenn z.B. die *datei1* die Zugriffsrechte `rwxr--r--` besitzt, dann kann der Eigentümer von Datei *datei1* – nach dem Kopieren – die neue *datei2* nicht beschreiben, wenn diese mit **cp** durch einen anderen Benutzer kopiert wurde.
- Die zweite Aufrufform kopiert die *datei(en)* in das Directory *directory*, wobei die dort neu angelegten Dateien die Namen der ursprünglichen Dateien erhalten. Als Name wird in das Directory die letzte Komponenente des Pfadnamens der alten Dateien eingetragen; wenn z.B. `/home/egon/uebung1/obst` nach `/home/egon/uebung2` kopiert wird, so würde in `/home/egon/uebung2` der Name `obst` (letzte Komponente des Pfadnamens der ursprünglichen Datei) eingetragen. Auch hier bleiben die Zugriffsrechte der Originaldateien erhalten und beziehen sich dann auf den neuen Eigentümer der Kopien.

-a	(*archive*) behält beim Kopieren möglichst alle Attribute der Dateien bei; identisch zu **–dpR**
-b	(*backup*) überschreibt vorhandene Dateien nicht, sondern benennt diese in Backup-Dateien (Dateiname mit angehängter Tilde ~) um.
-d	(*dereference*) kopiert bei Links nur den Verweis, nicht aber die Datei, auf die der Link zeigt.
-i	(*interactive*) Wenn eine Zieldatei bereits existiert, dann fragt **cp** erst nach, ob diese Zieldatei zu überschreiben ist. Oft wird **alias cp='cp –i '** in `~/.alias` bzw. `~/.profile` eingetragen.
-l	(*link*) erstellt Hard-Links statt die Dateien zu kopieren; entspricht dem Kommando **ln**.
-p	(*preserve*) Eigentümer, Gruppe, Zugriffsrechte und Zeitpunkt der letzten Änderung werden mitkopiert, bleiben also erhalten.
-r	(*recursvie*) ermöglicht das Kopieren ganzer Directorybäume.
-s	(*symbolic link*) erstellt symbolische Links statt die Dateien zu kopieren.
-u	(*update*) kopiert nur die Dateien, zu denen keine gleichnamigen Dateien mit einem neuerem Datum vorhanden sind.

In Unix werden Geräte wie Dateien behandelt. Der Zugriff auf die Geräte könnte somit auch direkt über die Gerätedateien erfolgen. So gibt z.B. das Kommando **cp** *datei* **/dev/tty** den Inhalt von *datei* am Bildschirm aus.

ln – Links auf Dateien erzeugen

Hardlinks:	**ln** [*optionen*] *datei1 datei2*	erzeugt einen Hardlink zur Datei *datei1*
	ln [*optionen*] *datei(en) directory*	erzeugt einen bzw. mehr Hardlinks zu den *datei(en)* im *directory*
Softlinks:	**ln -s** *datei1 datei2*	erzeugt einen Doftlink zur Datei *datei1*
	ln -s *datei(en) directory*	erzeugt einen bzw. mehr Softlinks zu den *datei(en)* im *directory*
	ln -s *dir1 dir2*	erzeugt einen Softlink *dir2* zum Directory *dir1*

-b	(*backup*) benennt vorhandene Dateien in Backup-Dateien (mit angehängtem Tilde ~) um, statt sie zu überschreiben.
-f	(*force*) nicht nachfragen, selbst wenn der Name einer schreibgeschützten Datei verwendet wird.
-n	wenn eine Zieldatei bereits existiert, wird nachgefragt, ob diese zu überschreiben ist; **-f** schaltet diese Option aus.
-s	(*symbolic*) erzeugt einen symbolischen Link.

mv – Dateien umbenennen

1. **mv** [*optionen*] *datei1 datei2*
2. **mv** [*optionen*] *datei(en) directory*
3. **mv** [*optionen*] *directory directory*

1. benennt die Datei *datei1* in *datei2* um. Existiert *datei2* bereits, wird sie überschrieben, wenn dies die Zugriffsrechte dieser Datei zulassen bzw. nicht Option **–i** angegeben ist. Eigentümer dieser neuen Datei wird der Benutzer, der dieses Kommando aufrief. Zwar werden die Zugriffsrechte übernommen, aber wenn sich der Eigentümer und vielleicht sogar die Gruppe dieser Datei ändert, dann sind diese Zugriffsrechte auf den neuen Eigentümer und die neue Gruppe anzuwenden..
2. trägt die *datei(en)* in das *directory* ein, wobei dort die Dateien die Namen der ursprünglichen Dateien erhalten. Als Name wird im Directory letzte Komponenente des Pfadnamens der alten Dateien eingetragen; wenn z.B. `/home/egon/uebung1/obst` nach `/home/egon/uebung2` verschoben wird, würde in `/home/egon/uebung2` der Name `obst` (letzte Komponente des Pfadnamens der ursprünglichen Datei) eingetragen. Danach existieren die ursprünglichen Dateien nicht mehr. Auch hier bleiben die Zugriffsrechte der Originaldateien erhalten und beziehen sich dann auf den neuen Eigentümer der umbenannten Dateien.
3. ist ein Spezialfall der zweiten; sie ermöglicht das Umbenennen eines ganzen Directory.

-b	(*backup*) benennt vorhandene Dateien in Backup-Dateien (Dateiname mit Tilde ~ am Ende) um, statt sie zu überschreiben
-i	(*interactive*) Wenn eine Zieldatei bereits existiert, dann fragt **mv** erst nach, ob diese Zieldatei zu überschreiben ist
-f	(*force*) wenn eine Zieldatei schreibgeschützt ist, wird normalerweise das Zugriffsrechte-Muster ausgegeben, und nachgefragt, ob diese Datei zu überschreiben ist. Ist Option **-f** gesetzt, wird die Zieldatei ohne Rückfrage überschrieben.

Wenn bei einem Aufruf wie **mv** *dir1 dir2* das Zieldirectory *dir2* bereits existiert, verhält sich **mv** wie **cp** und legt *dir1* als Subdirectory von *dir2* an.

rm – Dateien oder ganze Directorybäume löschen

rm [*optionen*] *datei(en)*

Eine Datei kann mehrere Namen (Links) besitzen. Ist eine *datei* ein Link, löscht **rm** nur den Link und nicht die wirkliche Datei. Wird der letzte Link auf eine Datei gelöscht, dann wird auch die Datei selbst gelöscht. Man muss Schreibrechte im entspr. Directory besitzen, um eine Datei darin löschen zu können; für die Datei selbst werden weder Lese- noch Schreibrechte benötigt. Fehlen allerdings die Schreibrechte für eine Datei, wird nachgefragt, ob er diese Datei wirklich löschen möchte. Directories können nur gelöscht werden, wenn die Option **-r** angegeben ist.

-f	löscht Dateien/Directories ohne Rückfrage, sogar wenn die entsprechenden Schreibrechte fehlen.
-r	erlaubt es, Directories zu löschen. Diese Option bewirkt, dass zuerst alle Inhalte eines Directorys und dann das Directory selbst gelöscht werden. Diese Option arbeitet dabei rekursiv, was bedeutet, dass alle Subdirectories, Subsubdirectories usw. zuerst geleert werden, bevor dann das entsprechende Parent-Directory gelöscht wird.
-i	für jede zu löschende Datei wird nachgefragt, ob sie wirklich gelöscht werden soll. Nur wenn auf diese Frage mit der Eingabe **y** geantwortet wird, wird diese Datei dann gelöscht.

Häufig findet man folgende Alias-Definition in ~/.profile bzw. ~/.alias: `alias rm='rm –i '`.Dies mindert die Gefahr des versehentlichen Löschens von Dateien. Ist man sich sicher, dass man Dateien wirklich löschen möchte, muss man nur die Option **–f** angeben, wie z.B. **rm –f *.bak**.

2.4 Suchen in und nach Dateien

grep / fgrep / egrep – Suchen in Dateien

grep [*optionen*] *regulärer-Ausdruck* [*datei(en)*]
egrep [*optionen*] *regulärer-Ausdruck* [*datei(en)*]
fgrep [*optionen*] *string* [*datei(en)*]

- **grep** gibt alle Zeilen aus den angegebenen *datei(en)* aus, die durch den angegebenen *regulären-Ausdruck* abgedeckt werden. Ist mehr als eine *datei* angegeben, wird zu jeder Zeile noch der Name der Datei ausgegeben, aus der diese Zeile stammt. Wird **grep** ohne Angabe von *datei(en)* aufgerufen, so liest es von der Standardeingabe; dies ist sinnvoll für Pipes oder Eingabeumlenkung. **grep** schreibt die gefundenen Zeile auf die Standardausgabe. Um seine Ausgabe also an ein anderes Kommando weiterzuleiten oder aber in eine Datei zu schreiben, muss eine Pipe oder Ausgabeumlenkung verwendet werden.
- **egrep** (*extended grep*) läßt erweiterte und komplexere reguläre Ausdrücke zu, ist aber dafür etwas langsamer. Ein **egrep**-Aufruf entspricht dem Aufruf **grep –E** *regulärer Ausdruck*.
- **fgrep** (*fast grep*) läßt nur die Suche nach einfachen Strings zu, ist aber das schnellste dieser drei Suchkommandos. Ein **fgrep**-Aufruf entspricht dem Aufruf **grep –F** *string*.

-E *ra*	entspricht einem **egrep**-Aufruf, der erweiterte reguläre Ausdrücke zum Suchen zulässt.
-F *string*	entspricht einem **fgrep**-Aufruf, der nur das Suchen nach Strings und nicht nach regulären Ausdrücken zuläßt.
-*n*	gibt zu Zeilen, in denen das Suchmuster gefunden wurde, zusätzlich auch *n* Zeilen unmittelbar davor und danach.
-c	Es wird für jede Datei nur die Anzahl von Zeilen ausgegeben, die durch den *regulären-Ausdruck* abgedeckt sind.
-f *datei*	der zu suchende reguläre Ausdruck befindet sich in der Datei *datei*.
-h	Dateiname wird nicht vor den Zeilen ausgegeben, in denen ein gesuchter String gefunden wurde.
-i	Groß- und Kleinschreibung ist nicht zu unterscheiden.
-l	Nur die Namen der Dateien ausgeben, in denen Zeilen gefunden wurden.
-n	Vor jeder gefundenen Zeile wird die zugehörige Zeilennummer ausgegeben.
-q	keine Ausgabe, sondern liefert nur den Rückgabewert 0 (Suchtext gefunden) oder 1 (nicht gefunden); diese Option ist nützlich, wenn **grep** in Shell-Skripts verwendet wird, wo es nur interessiert, ob Datei Suchtext enthält oder nicht.
-s	Fehlermeldungen über nicht existierende *datei(en)* werden nicht ausgegeben.
-v	Alle Zeilen ausgeben, die nicht durch den angegebenen *regulären-Ausdruck* abgedeckt werden.
-w	nur nach ganzen Wörtern suchen; z.B. wird für den Suchtext. „aus“ nicht das Wort „hausen“ abgedeckt.

- **grep Mueller namliste**: gibt alle Zeilen aus Datei `namliste` aus, in denen der String `Mueller` vorkommt.
- **grep 'M[ea][iy]er' namliste**: gibt alle Zeilen aus der Datei `namliste` aus, in denen einer der folgenden Strings vorkommt: `Meier`, `Maier`, `Meyer` oder `Mayer`.
- **grep –c sleep\(.*\) *.c**: gibt aus, wie oft die Funktion `sleep()` in den C-Programmdateien des Working-Directorys aufgerufen wird.
- **grep Zeichensatz */***: sucht in allen Dateien aller Subdirectories des Working-Directorys nach dem Wort „`Zeichensatz`“ und gibt alle gefundenen Zeilen aus.
- **egrep 'M[ea][iy]er|M(ue|i)ller' namliste**: gibt alle Zeilen aus der Datei `namliste` aus, in denen einer der folgenden Strings vorkommt: `Meier`, `Maier`, `Meyer`, `Mayer`, `Mueller` oder `Miller`.
- **fgrep 'Meier**
  Maier
  Meyer
  Mayer
  Mueller
  Miller' namliste
  gibt das Gleiche wie der vorherige Aufruf aus.

Um nach Strings in komprimierten Dateien zu suchen, steht das Kommando **zgrep** zur Verfügung.

locate – Schnelles Suchen nach Dateien (Linux)

locate [*optionen*] *muster ...*

Unter Linux existiert Kommando **locate**, das wesentlich schneller als **find** ist. Es durchsucht die Datenbank `locatedb` nach Dateien, in denen die angegebenen *muster* im vollständigen Dateinamen (inklusive Pfad) vorkommen. Da **locate** auf eine Datenbank zugreift, ist es im Vergleich zu **find** wesentlich schneller, hat allerdings auch den Nachteil, dass Dateien, die erst nach dem Erstellen der Datenbank erzeugt wurden, oder Dateien, die an einen anderen Platz im Dateibaum verschoben wurden, nicht von **locate** gefunden werden können.
locate setzt voraus, dass `locatedb`-Datenbank zuvor mit **updatedb** eingerichtet wurde. Die Datenbank wird meist in `/var/lib` gespeichert. Um die Datenbank möglichst aktuell zu halten, sollte der Superuser in regelmäßigen Abständen oder aber bei größeren Systemveränderungen **updatedb** aufrufen. Das regelmäßige Aufrufen von **updatedb** läßt sich auch durch einen Eintrag in der Datei `/etc/crontab` automatisieren.
Bei einigen Distributionen wird statt **updatedb** das sichere **slocate** verwendet, das in der Datenbank auch die Zugriffsrechte mit abspeichert. In diesem Fall werden dem Benutzer nur die Dateinamen angezeigt, zu denen er auch entsprechende Zugriffsrechte hat, die er also auch mit **ls** auflisten könnte. **slocate** speichert die Dateinamen und Zugriffsrechte meist in der Datenbankdatei `/var/lib/slocate/slocated.db`.
Das Fehlen von Optionen wie **-size**, **-perm**, **-mtime** usw. macht **locate** zu einem reinen Namens-Suchprogramm, das zwar schneller, aber bei weitem nicht so mächtig wie **find** ist.

- **locate cfg**: listet alle Dateien auf, in deren Pfadname irgendwo der String `cfg` vorkommt. Es ist zu beachten, dass der absolute Pfadname dabei zum Dateinamen zählt. Da dies sehr viele Dateien sein können, empfiehlt sich eine seitenweise Ansicht mit **locate cfg | less**.
- **locate '*cfg'**: listet alle Dateien auf, deren Name mit dem String `cfg` endet.

whereis – Schnelles Suchen nach Dateien

whereis [*optionen*] *name*

durchsucht alle wichtigen Pfade für Binärdateien, man-Dateien und Quellprogramme nach dem angegebenen *namen*. **whereis** ist nicht so allgemein einsetzbar wie **find**, aber dafür wesentlich schneller. Welche Pfade **whereis** durchsucht, kann man sich mit dem Aufruf **man whereis** anzeigen lassen.

```
$ whereis ls (Enter)
ls: /bin/ls /usr/share/man/man1/ls.1.gz
$ whereis file (Enter)
file: /usr/bin/file /usr/share/man/man1/file.1.gz
/usr/share/man/mann/file.n.gz
$ whereis whereis (Enter)
whereis: /usr/bin/whereis /usr/share/man/man1/whereis.1.gz
$
```

which – Schnelles Suchen nach Kommandos

which *kommandoname(n)*

Das Kommando **which** ermöglicht das schnelle Auffinden von Kommandos (Programmen). Dazu durchsucht es die in der Variablen `PATH` angegebenen Directories. Den Inhalt der Variablen `PATH` kann man sich mit **echo $PATH** ausgeben lassen. Kann **which** die angegebenen *kommandoname(n)* in den `PATH`-Directories finden, gibt es den vollen Pfadnamen des jeweiligen Kommandos aus, ansonsten gibt es nichts aus.

```
$ which cal mail (Enter)
/usr/bin/cal
/bin/mail
$
```

find – Suchen nach Dateien

find *pfadname(n) bedingung(en)*

find durchsucht angegebenen *pfadname(n)* nach Dateien, für die angegebene *bedingung(en)* erfüllt sind. Dabei wird für jeden angegebenen *pfadname(n)* der ganze zugehörige Directorybaum nach Dateien durchsucht, auf die die angegebenen *bedingung(en)* zutreffen. Bei den nachfolgend vorgestellten *bedingung(en)* steht *n* für eine ganze Zahl; für *n* kann dabei angegeben werden: *n* (genau *n*), *+n* (mehr als *n*) oder *–n* (weniger als *n*).

-print	immer erfüllt; gibt zu allen gefundenen Dateien die Namen auf der Standardausgabe aus.
-name *dateiname*	erfüllt, wenn eine Datei mit dem Namen *dateiname* gefunden wird.
-path *suchmuster*	erfüllt, wenn Dateien gefunden werden, die dem *suchmuster* entsprechen. Anders als bei **–name** bezieht sich das *suchmuster* nicht nur auf den Dateinamen, sondern auf den ganzen Pfadnamen. Anders als bei den zuerst anzugebenden *pfadname(n)* deckt hier das Jokerzeichen `*` auch das Zeichen `/` ab.
-perm [**-**]*oktalzahl*	erfüllt, wenn eine Datei gefunden wird, deren Zugriffsrechte der angegebenen *oktalzahl* entsprechen; ist vor *oktalzahl* ein – (Minuszeichen) angegeben, werden nur die mit *oktalzahl* spezifizierten Zugriffsrechte überprüft und die restlichen sind dabei nicht von Bedeutung.
-type *c*	erfüllt, wenn eine Datei gefunden wird, deren Dateiart *c* ist; für *c* kann dabei **b** (blockorientierte Gerätedatei), **c** (zeichenorientierte Gerätedatei), **d** (directory), **l** (symbolischer Link), **p** (named pipe) oder **f** (einfache Datei) angegeben werden.
-links *n*	erfüllt, wenn eine Datei gefunden wird, die *n* Links besitzt.
-user *kennung* **-nouser** *kennung*	erfüllt, wenn Datei gefunden wird, die Benutzer mit der angegebenen *kennung* gehört bzw. nicht gehört; für *kennung* kann dabei entweder der Login-Name oder die User-ID eines Benutzers angegeben werden.
-group *kennung* **-nogroup** *kennung*	erfüllt, wenn eine Datei gefunden wird, die der Gruppe mit der angegebenen *kennung* gehört bzw. nicht gehört; für *kennung* kann dabei entweder der Name oder die Group-ID einer Gruppe angegeben werden.
-inum *n*	erfüllt, wenn eine Datei die inode-Nummer *n* hat. Nützlich, um alle Links zu einer Datei zu finden. Links haben immer die gleiche inode-Nummer.
-size *n*[**c**|**k**]	erfüllt, wenn eine Datei gefunden wird, deren Größe *n* Blöcke bzw. *n* Bytes (bei der Angabe *n***c**) oder *n* Kilobytes (bei der Angabe *n***k**) ist.
-atime *n*	erfüllt, wenn Datei gefunden wird, auf die vor *n* Tagen das letztemal zugegriffen wurde; ein Durchsuchen mit **find** wird auch als Zugriff gewertet, jedoch erst nachdem vorherige Zugriffszeit ausgewertet wurde.
-mtime *n*	erfüllt, wenn eine Datei gefunden wird, die vor *n* Tagen das letztemal modifiziert wurde.
-ctime *n*	erfüllt, wenn eine Datei gefunden wird, deren i-node vor *n* Tagen das letztemal geändert wurde.
-exec *kdo*	erfüllt, wenn das Kommando *kdo*, das zunächst ausgeführt wird, erfolgreich ablief (Exit-Status 0). Wird `{}` in der Kommandozeile *kdo* angegeben, so wird hierfür immer der aktuelle Pfadnamen eingesetzt. Das Ende von *kdo* muss immer mit `\;` angezeigt werden; z.B. würde **-exec rm {} \;** bedeuten "*Lösche alle gefundenen Dateien, für die die vorher angegebenen Bedingungen zutreffen*". Falls am Ende von *kdo* ein Pluszeichen (`+`) angegeben ist, sammelt es zunächst alle betroffenen Dateinamen, bevor es *kdo* aufruft.
-ok *kdo*	wie **-exec**, außer dass die entsprechende Kommandozeile immer zuerst mit einem Fragezeichen ausgegeben wird, und nur dann ausgeführt wird, wenn der Benutzer auf diese Frage mit **y** antwortet.
-newer *dateiname*	erfüllt, wenn Datei gefunden wird, deren Modifikationsdatum jünger als das von der Datei *dateiname* ist.
-depth	immer erfüllt; bewirkt, dass im Directorybaum immer zuerst zu den Blättern „abgestiegen" wird. Dies hat zur Folge, dass alle Einträge in einem Directory bearbeitet werden, bevor auf das Directory selbst zugegriffen wird. Dies kann nützlich bei der Kombination mit dem Kommando **cpio** eingesetzt werden, wenn es erforderlich ist, Dateien zu übertragen, die sich in Directories ohne Schreiberlaubnis befinden.
-mount	immer erfüllt; begrenzt die Suche auf das lokale Dateisystem.
-local	erfüllt, wenn eine gefundene Datei sich auf dem lokalen System befindet.
-follow	verfolgt symbolische Links; **-follow** sollte nicht zusammen mit **-type l** benutzt werden.
-prune	Hiermit lassen sich Subdirectories von der Suche ausschließen, die durch einen dem **-prune** vorangehenden Ausdruck näher bestimmt werden. So würde z.B. mit folgendem Aufruf nach allen Dateien im Directorybaum `/home/egon` gesucht, die größer als 30 MByte sind, außer im Subdirectory `sicherung`: **find /home/egon -path '*sicherung*' -prune -o -size +30000k**
-fstype *typ*	erfüllt, wenn eine Datei sich in einem Dateisystem vom Typ *typ* befindet.

Die *bedingung(en)* können auch noch mit folgenden booleschen Operatoren (Reihenfolge der Angabe entspricht abnehmender Priorität) verknüpft werden:

!	entspricht NOT: **! -name '*.[ch]'** bedeutet z.B.: „*Alle Dateinamen, die nicht mit* `.c` *oder* `.h` *enden*“
keine Angabe	entspricht AND: **-size +1000c -name '*.txt'** bedeutet z.B.: „*Alle Dateinamen, die mit .txt enden und mehr als 1000 Bytes belegen*“
-o	entspricht OR: **-size +2000c -o -name '*.bak'** bedeutet z.B.: „*Alle Dateien, die entweder mehr als 2000 Bytes belegen oder deren Name mit* `.bak` *endet*“

- **find . –print**
  gibt alle Namen der Dateien des Working-Directorys sowie der, die sich in den Subdirectories des Working-Directorys befinden.
- **find /usr -name dir.h -print**
  Suche im Directorybaum `/usr` alle Vorkommen der Datei `dir.h`.
- **find / -user egon -print**
  Suche im Directorybaum `/` alle Dateien, die *egon* gehören.
- **find .. -type d -print**
  Gib alle Directories aus, die sich im Directorybaum zum Parent-Directory befinden.
- **find . -print -name add1.c**
  Dieser Aufruf gibt alle Dateien des Working-Directorys und dessen Subdirectories aus, da die Bedingungen von links nach rechts ausgewertet werden, und **-print** ist immer erfüllt.
- **find . -name add1.c -print**
  Suche alle Vorkommen von `add1.c` im Working-Directorybaum.
- **find /usr -links +10 -type d -print**
  Suche im Directorybaum `/usr` alle Directories mit mehr als 10 Links.
- **find / -size +200k -print**
  Suche im Root-Directorybaum alle Dateien, die mehr als 200 Kilobytes enthalten.
- **find /usr -type f -newer add2.c -print**
  Suche im Directorybaum zu `/usr` alle einfachen Dateien, deren Modifikationsdatum jünger als das von Datei `add2.c` ist
- **find / \(-name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;**
  Lösche alle Dateien, deren Name `a.out` ist oder aber mit `.o` endet, wenn auf diese seit einer Woche nicht mehr zugegriffen wurde.
- **head `find . –name '*.tex' -print`**
  gibt zu allen Dateien des Working-Directorys, die mit `.tex` enden, die ersten zehn Zeilen aus.
- **find . -print | wc –l:**
  ermittelt die Anzahl der Dateien, die sich im Working-Directorybaum befinden:

Unter Linux muss kein Pfadname angegeben werden. Fehlt der Pfadname, wird Working-Directory angenommen. Zudem kann unter Linux die Angabe **–print** entfallen. So kann z.B. unter Linux statt **find . –print** auch nur **find** aufgerufen werden.

2.5 Vergleichen, Mischen und Sortieren von Dateien

diff – Vergleichen zweier (un)sortierter Textdateien

diff [*optionen*] *datei1 datei2*

Das Kommando **diff** erlaubt den Vergleich von zwei Textdateien, die nicht sortiert sein müssen, und gibt die Änderungen am Bildschirm aus. Wird für *datei1* oder *datei2* ein Minuszeichen (`-`) angegeben, so wird hierfür (anstelle aus einer Datei) von der Standardeingabe gelesen. Wenn *datei1* ein Directory und *datei2* eine einfache Datei ist, dann wird *datei2* mit einer Datei gleichen Namens im Directory *datei1*, also mit *datei1/datei2* verglichen. Das gleiche gilt, wenn *datei2* ein Directory ist. **diff** verwendet folgendes Ausgabeformat, um anzuzeigen, dass entsprechende Zeilen einzufügen, zu löschen oder zu ändern sind. Im nachfolgenden steht *n* für eine Zeilennummer und *zeile* für eine Zeile von Text.

Einfügen	*n11***a***n21, n22* >*zeile n21* ..... >*zeile n22*	(*append*) Füge die Zeilen *n21* bis *n22* von *datei2* nach Zeile *n11* in *datei1* ein. Die einzufügenden Zeilen aus *datei2* werden immer mit einem vorangestellten `>` gekennzeichnet.
Löschen	*n11,n12***d***n21* <*zeile n11* ..... <*zeile n12*	(*delete*) Lösche die Zeilen *n11* bis *n12* von *datei1*. Die Zeile *n21* aus *datei2* zeigt hierbei an, dass nach diesem Löschen die beiden Dateien bis zu dieser Zeile (nicht eingeschlossen) identisch sind. Die vom Löschvorgang betroffenen Zeile werden mit einem vorangestellten `<` ausgegeben.
Ändern	*n11,n12***c***n21,n22* <*zeile n11* .... <*zeile n12* --- >*zeile n21* ..... >*zeile n22*	(*change*) Ersetze die Zeilen *n11* bis *n12* aus *datei1* durch die Zeilen *n21* bis *n22* aus *datei2*.

-b	Leerzeichen am Ende werden ignoriert und mehrere direkt aufeinanderfolgende Leerzeichen werden zusammengezogen, so dass sich zwei Zeilen nicht unterscheiden, wenn ihr Text bis auf zusätzliche Leerzeichen zwischen den Worten identisch ist, wie z.B. bei "`Hallo Egon`" und "`Hallo  Egon`"
-i	Groß- und Kleinschreibung ignorieren.
-w	Alle Leerzeichen werden ignoriert. Dadurch sind alle Wörter identisch, selbst wenn in ihnen Leerzeichen enthalten sind, wie z.B. bei "`Hallo Egon`" und "`H a l l o  E g o n`". Beide werden zu einem String "`HalloEgon`" zusammengezogen.
-e	Die erforderlichen Änderungen, die an *datei1* vorzunehmen sind, um sie mit *datei2* identisch zu machen, werden in einer dem Editor **ed** verständlichen Form (ed-Skript genannt) ausgegeben.
-f	gibt ein zu **-e** ähnliches Skript aus, aber in umgekehrter Reihenfolge. Dieses Skript ist jedoch nicht für **ed** geeignet.
-D *string*	mischt den Inhalt der beiden Dateien *datei1* und *datei2* zusammen. Dabei werden C-Präprozessor-Anweisungen so eingefügt, dass eine Kompilierung der gemischten Datei ohne die Definition von *string* einer Kompilierung von *datei1* und mit der Definition von *string* einer Kompilierung von *datei2* gleichkommt.

Zum Vergleichen von Directories stehen u.a. die folgenden Optionen zur Verfügung.

-r	gemeinsame Subdirectories werden rekursiv verglichen.
-s	Gleiche Dateien werden auch angezeigt; normalerweise werden diese nicht ausgegeben.
-S *name*	Directory-Vergleich beginnt erst mit der Datei namens *name*.

Zu **diff** existieren u.a. zwei weitere verwandte Kommandos:

- **diff3**: (*3-way differential file comparison*) ermöglicht den Vergleich von 3 Dateien.
- **sdiff**: (*side-by-side difference program*) gibt Inhalte der beiden zu vergleichenden Dateien nebeneinander aus; dabei zeigen die den Zeilen vorangestellten Zeichen an: `<` (ist nur in *datei1* vorhanden), `>` (ist nur in *datei2* vorhanden), `|` (Zeilen sind verschieden) und kein Zeichen deutet an, dass die beiden Zeilen identisch sind.

comm – Vergleichen zweier sortierter Textdateien

comm [**-123**] *datei1 datei2*

comm vergleicht *datei1* und *datei2*; beide Dateien müssen sortiert sein. **comm** gibt eine dreispaltige Liste aus:

Zeilen, die nur in *datei1* vorkommen	Zeilen, die nur in *datei2* vorkommen	Zeilen, die in beiden Dateien (*datei1* und *datei2*) vorkommen

Durch Angabe der Optionen **-1**, **-2** oder **-3** kann die Ausgabe der entsprechenden Spalte unterdrückt werden.´
comm -23 *datei1 datei2*: würde Zeilen ausgeben, die nur in *datei1*, aber nicht in *datei2* vorkommen.
comm -123 *datei1 datei2*: würde überhaupt keine Ausgabe erzeugen.
comm ist auf sortierte Dateien begrenzt; somit wird es häufig beim Vergleich von Dateien mit sortierten Daten verwendet, um z.B. festzustellen, welche Daten in einer Datei noch aufzunehmen bzw. zu entfernen sind.

cmp – Vergleichen von zwei Dateien (auch Nicht-Textdateien)

cmp [*optionen*] *datei1 datei2* [*skip1* [*skip2*]]

cmp vergleicht *datei1* und *datei2* Byte für Byte. Sind Dateien identisch, erfolgt keine Mitteilung am Bildschirm. Sind sie unterschiedlich, wird Bytenr. des ersten Unterschieds ausgegeben. **cmp** kann auch auf binäre Dateien angewendet werden. Mit *skip1* und *skip2* kann festgelegt werden, bei welcher Bytenr. in *datei1* (*skip1*) und in *datei2* (*skip2*) der Vergleich beginnen soll. Für *skip[12]* ist Dezimal- oder Oktalzahl (muss mit `0` beginnen) erlaubt.

-l	Alle Unterschiede der beiden Dateien werden in folgender Form ausgegeben: *Byte-Nummer Byte-Inhalt (oktal) von datei1 Byte-Inhalt (oktal) von datei2*
-s	keine Ausgabe der Unterschiede, sondern es wird nur über den exit-Status dieses Kommandos mitgeteilt, ob Unterschiede vorliegen; dabei bedeutet exit-Status: 0 (identisch). 1 (verschieden) oder 2 (Fehler).

Im allgemeinen verwendet man **cmp**, wenn festzustellen ist, ob zwei Dateien wirklich den gleichen Inhalt haben. **cmp** wird z.B. sehr oft verwendet, um zu prüfen, ob zwei Objektdateien (wie z.B. Programme oder Graphikbilder) den gleichen Inhalt besitzen; wenn ja, kann eine davon gelöscht werden.

join – Mischen von Dateien

join [*optionen*] *datei1 datei2*

join faßt diejenigen Zeilen aus den Dateien *datei1* und *datei2* zusammen, deren Schlüsselfelder identisch sind. Wird für *datei1* ein Querstrich (`-`) angegeben, so wird hierfür die Standardeingabe verwendet. Die beiden Dateien müssen dabei bzgl. des Schlüsselfelds sortiert sein. Als Schlüsselfeld kann dabei jedes Feld innerhalb der Zeilen verwendet werden. Wenn durch Optionen nicht anders festgelegt, wird 1. Feld in beiden Dateien als Schlüsselfeld verwendet. Als Feld-Trennzeichen werden, wenn nicht anders durch die Optionen festgelegt, Leer-, Tab- und Neuezeilezeichen verwendet. Die gemischten Zeilen aus den beiden Dateien werden auf die Standardausgabe ausgegeben.

-j *m*	Das *m*. Feld wird in beiden Dateien als Schlüsselfeld verwendet
-j1 *m* bzw. **-j2** *m*	Das *m*. Feld wird in *datei1* bzw. das *m*. Feld wird in *datei2* als Schlüsselfeld verwendet
-a1	Zeilen aus *datei1* ausgeben, für die keine Zeile mit gleichem Schlüsselfeld-Inhalt in *datei2* existiert.
-a2	Zeilen aus *datei2* ausgeben, für die keine Zeile mit gleichem Schlüsselfeld-Inhalt in *datei1* existiert.
-o *n.m* **...**	Legt die Felder fest, welche auszugeben sind: Aus *n*. Datei das *m*. Feld; wobei für *n* entweder 1 (*datei1*) oder 2 (*datei2*) angegeben werden kann.
-t*z*	Legt *z* als Trennzeichen für die Felder fest; gilt dann sowohl für Eingabe- wie auch für die Ausgabefelder.
-e *string*	Legt fest, dass leere Ausgabefelder durch *string* zu ersetzen sind.

- **join -a1 obst2 obstpreise**: mischt die beiden Dateien `obst2` und `obstpreise`; dabei sind auch die Zeilen aus `obst2` auszugeben, die kein gemeinsames Schlüsselfeld mit `obstpreise` haben.
- **join -t: -j1 1 -j2 2 laender2 sprache**: mischt `laender2` (1. Feld=Schl.) und `sprache` (2. Feld=Schl.)
- **join -t: -j1 1 -j2 2 -e "---" laender2 sprache**: mischt die beiden Dateien `laender2` (1. Feld = Schlüsselfeld) und `sprache` (2. Feld = Schlüsselfeld); leere Felder sind mit `---` auszugeben
- **join -t: -j1 2 -j2 1 -o 1.2 1.1 2.2 sprache laender2**: mischt die beiden Dateien `sprache` (2. Feld = Schlüsselfeld) und `laender2` (1. Feld = Schlüsselfeld); bei der Ausgabe des Mischergebnisses ist zuerst das 2. Feld und dann das 1. Feld von `sprache` auszugeben und dann das 2. Feld von `laender2`.

sort – Sortieren von Dateien

sort [*optionen*] [*datei(en)*]

sort gibt die Zeilen der angegebenen *datei(en)* sortiert am Bildschirm aus. Sind mehrere *datei(en)* angegeben, werden diese als Ganzes sortiert und dabei die Zeilen aus unterschiedlichen Dateien gemischt. Sind keine *datei(en)* angegeben, liest **sort** von der Standardeingabe. Die Ausgabe der sortierten Daten erfolgt auf die Standardausgabe.

-c	(*check*) überprüft, ob die angegebenen *datei(en)* bereits sortiert sind. Bei dieser Option wird nichts ausgegeben, außer die eventuelle Meldung, dass eine angegebene Datei nicht sortiert ist.
-m	(*merge*) gibt die angegebenen *datei(en)*, welche für sich bereits sortiert sein müssen, als ganzes sortiert aus. Dies ist wesentlich schneller, als die Dateien vorher zusammenzumischen und erst dann zu sortieren.
-u	(*unique*) gibt für alle Zeilen mit einem gleichen Sortierschlüssel nur jeweils eine Fundstelle aus.
-o *ausgabedatei*	(*output*) schreibt die sortierte Ausgabe in die Datei *ausgabedatei*; als *ausgabedatei* kann dabei eine der angegebenen *datei(en)* angegeben werden.
-d	(*dictionary*) Lexikographisch sortieren: nur Buchstaben, Ziffern und Leer- bzw. Tabulatorzeichen werden beim Sortieren berücksichtigt; Voreinstellung ist: nach ASCII-Werten sortieren.
-f	Groß- und Kleinschreibung nicht berücksichtigen: alle Buchstaben in Großschreibung vergleichen.
-i	(*ignore*) Nicht druckbare Zeichen ignorieren; Voreinst.: nach ASCII-Werten (auch nicht-druckbare) sortieren.
-M	(*Months*) nach Monaten sortieren; die ersten drei Nicht-Zwischenraumzeichen eines jeden Feldes werden in Großbuchstaben umgewandelt, bevor sie mit den Monatsnamen JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC (legt auch die Sortierreihenfolge fest) verglichen werden. Enthalten Felder keinen gültigen Monatsnamen, so werden sie vor JAN eingeordnet. Diese Option impliziert die Option **-b**.
-n	(*numeric*) Numerisch sortieren. Diese Option impliziert die Option **-b**.
-r	(*reverse*) Absteigend sortieren; Voreinstellung ist: aufsteigend sortieren

Werden diese Optionen vor den Sortierschlüsseln angegeben, so gelten sie global für alle Sortierschlüssel. Sind sie nur zu einzelnen Sortierschlüsseln angegeben, so beziehen sie sich nur auf diese, und schalten eventuell anderslautende globale Optionen für diesen Sortierschlüssel aus. Beim Aufruf von **sort** können so genannte Sortierschlüssel angegeben werden. Sortierschlüssel legen fest, welches Feld in den angegebenen *datei(en)* als Sortierkriterium zu verwenden ist. Dabei ist es möglich, mehrere Sortierschlüssel anzugeben. Wenn die entsprechenden mit dem ersten Sortierschlüssel ausgewählten Felder gleich sind, so werden die über den zweiten Sortierschlüssel festgelegten Felder verglichen; sollten auch die gleich sein, so wird der dritte Sortierschlüssel verwendet usw. Der voreingestellte Sortierschlüssel ist die ganze Zeile. Sortierschlüssel legen die Sortierfelder fest und werden in der folgenden Form angegeben:

+m[.n]	Beginn des Sortierfelds: *n+1*. Zeichen im *m+1*. Feld (*m* Felder und *n* Zeichen im *m+1*. Feld überspringen) Voreinstellung für **.***n* ist **.0**: 1. Zeichen im *m+1*. Feld. Falls Option **-b** angegeben ist, so wird die Zählung für *n* ab dem ersten Nicht-Zwischenraum-Zeichen im *m+1*. Feld begonnen.
-k[.l]	Ende des Sortierfelds: l. Zeichen (einschließlich Trennzeichen) nach Ende des *k*. Feld (Dieses letzte Zeichen gehört nicht mehr zum Sortierfeld). Voreinstellung für **.***l* ist **.0**: Letztes Zeichen im *k*. Feld.

Falls kein Ende für einen Sortierschlüssel angegeben ist, so wird das Zeilenende dafür angenommen.

sort +2 `Das Wetter ist` **heute nicht besonders.**
Überspringe zwei Felder und verwende Rest der Zeile als Sortierschlüssel.

sort +3.2 `Das Wetter ist he`**ute nicht besonders.**
Überspringe drei Felder und zwei Zeichen; Rest der Zeile ist Sortierschlüssel.

sort +2 -3 `Das Wetter` **ist** `heute nicht besonders.`
Überspringe zwei Felder; Sortierschlüssel erstreckt sich von da bis zum Ende des 3. Felds.

Hinter jedem Sortierschlüssel sind die Optionen **dfinr** erlaubt; sie beziehen sich dann nur auf das entspr. Sortierfeld und nicht auf alle Sortierfelder; solche lokale Optionen schalten für dieses Feld eine evtl. anders lautende globale Option aus. Die Trennzeichen für die Felder der Eingabe können durch folg. Optionen geändert werden:

-b	Führende Leerzeichen beim Festlegen des Starts und Endes eines Sortierschlüssels nicht berücksichtigen
-t*z*	Zeichen *z* als Trennzeichen für einzelne Felder; Voreinst. für das Feld-Trennzeichen sind: Leer- und Tabzeichen

Vorsicht ist geboten mit Aufrufen wie **sort datei > datei** Denn hier wird zuerst der Inhalt der Datei `datei` gelöscht, bevor **sort** mit der nun leeren `datei` aufgerufen wird. Entweder man lenkt hier die sortierte Ausgabe in eine andere Datei um, wie z.B. **sort datei > datei2** oder man verwendet die Option **–o**, wie z.B.: **sort datei –o datei**.

2.6 Komprimieren und Archivieren von Dateien

gzip / gunzip / zcat / zless / zmore – (De-)Komprimieren von Dateien

gzip [*optionen*] [*datei(en)*] bzw. **gunzip** *datei(en)*

gzip komprimiert bzw. dekomprimiert die angegebenen Dateien. Komprimierten Dateien wird automatisch die Endung `.gz` angehängt.

-c	beläßt die zu (de)komprimierenden *datei(en)* unverändert und gibt das Ergebnis auf der Standardausgabe (in der Regel der Bildschirm) aus. Von dort kann es mit > in eine Datei umgelenkt werden.
-d	dekomprimiert die angegebenen *datei(en)*, anstatt sie zu komprimieren. Dieser Aufruf entspricht dem Kommando **gunzip**.
-r	(de)komprimiert auch Dateien in allen Subdirectories.
-n	steuert die Geschwindigkeit und Qualität der Kompression. `-1` bewirkt ein schnelles Komprimieren, resultiert aber in einer schlechteren Komprimierung (weniger Platzeinsparung). Dagegen dauert eine Komprimierung mit `-9` zwar länger, führt aber zu kleineren komprimierten Dateien. Die Voreinstellung ist `-6`.

gunzip dekomprimiert die angegebenen Dateien, unabhängig davon, ob sie mit **gzip** oder **compress** komprimiert wurden. Dabei wird automatisch die Kennung `.gz` bzw. `.Z` im Dateinamen entfernt. **gunzip** ist lediglich ein Link auf **gzip**, wobei automatisch die Option **-d** eingeschaltet wird.

- **gzip *.c**: komprimiert alle C-Dateien im Working-Directory. Danach existieren im Working-Directory keine Dateinamen mehr, die mit `.c` enden. Statt dessen findet man im Working-Directory Dateinamen mit der Endung `.c.gz`, welche die komprimierten C-Dateien sind.
- **gzip -d *.c.gz**: dekomprimiert alle zuvor komprimierten C-Dateien (im Working-Directory) wieder.
- **gzip -c laender >laender.gz**: komprimiert die Datei `laender`, beläßt diese Datei aber unverändert und schreibt das komprimierte Resultat in die Datei `laender.gz`.

Um sich den Inhalt von mit **gzip** komprimierten Dateien anzeigen zu lassen, ohne diese vorher explizit zu dekomprimieren, stehen die folgenden Kommandos zur Verfügung:

zcat [*optionen*] *datei(en*) bzw. **zless** *datei(en)* bzw. **zmore** *datei(en)*

Diese Kommandos funktionieren wie **cat**, **less** und **more**. Der einzige Unterschied ist, dass man sich mit ihnen (ohne vorherige Dekomprimierung mit **gunzip**) den Inhalt von komprimierten Dateien (meist haben diese die Endung `.gz`, `.z` oder `.Z`) anzeigen lassen kann. Dies ist vor allem vorteilhaft, wenn sich die komprimierten Dateien auf einer CD-ROM befinden. Daneben existieren u.a. noch folgende Kommandos für komprimierte Dateien:

zgrep	ermöglicht ein Suchen in komprimierten Dateien wie mit dem Kommando **grep** in unkomprimierten Dateien
zdiff	entspricht dem Kommando **diff**, nur dass es auf komprimierte Dateien angewendet werden kann.
zcmp	entspricht dem Kommando **cmp**, nur dass es auf komprimierte Dateien angewendet werden kann.

bzip2 / bunzip2 / bzcat - Neues (De-)Komprimieren von Dateien

bzip2 [*optionen*] *datei(en)*

bzip2 entspricht weitgehend dem Kommando **gzip**, verwendet aber ein besseres Komprimierungsverfahren als **gzip**. Komprimierten Dateien wird automatisch die Endung `.bz2` angehängt. Die Aufrufsyntax von **bzip2** ist:

-c	beläßt die zu (de)komprimierenden *datei(en)* unverändert und gibt das Ergebnis auf der Standardausgabe (in der Regel der Bildschirm) aus. Von dort kann es mit > in eine Datei umgelenkt werden.
-d	dekomprimiert die angegebenen *datei(en)*, anstatt sie zu komprimieren. Dieser Aufruf entspricht dem Kommando **bunzip2**. Komprimierte Dateien können dabei die Endungen `.bz2`, `.bz`, `.tbz` oder `.tbz2` haben.
-n	steuert die Geschwindigkeit und Qualität der Kompression. **-1** bewirkt ein schnelles Komprimieren, resultiert aber in einer schlechteren Komprimierung (weniger Platzeinsparung). Dagegen dauert eine Komprimierung mit **-9** zwar länger, führt aber zu kleineren komprimierten Dateien. Die Voreinstellung ist **-9**.

- **bunzip2** dekomprimiert mit **bzip2** komprimierte Dateien. Dabei werden die Dateiendungen `.bz2`, `.bz`, `.tbz` bzw. `.tbz2` in den Dateinamen entfernt. **bunzip2** ist lediglich ein Link auf **bzip2**, wobei automatisch die Option **-d** eingeschaltet wird. Die Aufrufsyntax von **bunzip2** ist: **bunzip2** [*optionen*] *datei(en)*
- **bzcat** gibt Inhalt von **bzip2**-komprimierten Dateien aus, ohne dass man diese vorher dekomprimieren muss. Dies ist vor allem dann sehr nützlich, wenn sich die komprimierten Dateien auf einer CD-ROM befinden.

pack / unpack / pcat – (De-)Komprimieren von Dateien

- **pack** [*optionen*] [*datei(en)*] - *Komprimieren*
  komprimiert den Inhalt der angegebenen *datei(en)*; dabei werden die ursprünglichen *datei(en)* gelöscht und der komprimierte Inhalt jeder Datei wird in eine Datei mit Namen *datei*.**z** geschrieben.

-f	es wird auch dann komprimiert, wenn daraus keine Platzeinsparung resultiert.

  Zur Komprimierung wird ein *Huffman-Code* verwendet.

- **unpack** *dateiname(n) – Dekomprimieren*
  unpack *dateiname(n)*.**z**
  Der Inhalt der *dateiname(n)*.**z** wird dekomprimiert, wenn es sich dabei um komprimierte Dateien handelt; dabei werden die *dateiname(n)*.**z** mit ihrem komprimierten Inhalt gelöscht und der dekomprimierte Inhalt wird in eine Datei mit Namen *dateiname* geschrieben.

- **pcat** *dateiname(n) – Inhalt von komprimierten Dateien anzeigen*
  pcat *dateiname(n)*.**z**
  Der Inhalt der mit **pack** komprimierten *datei(en)*.**z** wird zum Zwecke der Ausgabe auf dem Bildschirm kurzzeitig dekomprimiert; **pcat** verhält sich bei komprimierten Dateien, welche ja nicht mehr in lesbarer Form vorliegen, wie **cat** bei einfachen Dateien. Eine mit **pack** komprimierte Datei könnte somit auf zwei verschiedene Arten wieder dekomprimiert werden:
  unpack *dateiname*.**z**
  oder
  pcat *dateiname*.**z** >*dateiname*
  rm *dateiname*.**z**

Leistungsfähiger als **pack** sind die Kommandos **gzip** und **bzip2**.

compress / uncompress – (De-)Komprimieren von Dateien

- **compress** [*optionen*] [*datei(en)*]
  komprimiert bzw. dekomprimiert die angegebene Datei. Den komprimierten Dateien wird die Endung `.Z` angehängt.

-d	dekomprimiert die angegebenen *datei(en)*, anstatt sie uzu komprimieren (entspricht dem Aufruf **uncompress**)

- **uncompress** [*datei(en)*]
  dekomprimiert eine mit **compress** komprimierte Datei. Dabei wird die Dateiendung `.Z` automatisch entfernt.
  uncompress ist ein Link auf **compress**, wobei automatisch die Option **–d** eingeschaltet ist.

Leistungsfähiger als **compress** sind die Kommandos **gzip** und **bzip2**.

cpio – Archivieren bzw. Kopieren von Dateien und Directorybäumen

cpio -o[**acBv...**] (*copy out*)

cpio -i[**Bcdmrtuvf...**] [*dateiname(n)*] (*copy in*)

cpio -p[**adlmruv...**] *directory* (*copy pass*)

cpio eignet sich sehr gut dazu, ganze Directorybäume umzukopieren. Auch wird dieses Kommando verwendet, um ganze Directorybäume auf einem externen Speichermedium (wie Diskette oder Magnetband) zu sichern und später eventuell wieder einzukopieren; in diesem Fall wird direkt auf die entsprechende Gerätedatei, wie z.B. die für das Disketten-Laufwerk (`/dev/fd0`) kopiert. **cpio** kann auf drei verschiedene Arten aufgerufen werden:

- **cpio –o**
  liest die Pfadnamen der zu kopierenden Dateien von der Standardeingabe und kopiert deren Inhalt auf die Standardausgabe, wobei die zugehörigen Pfadnamen und Status-Information über die Dateien (Zugriffsrechte, Modifikationsdatum, usw.) mit ausgegeben werden.
  Wird meist auf der rechten Seite einer Pipe und mit Umlenkung angegeben:
  ls ... | cpio -o...>*datei* (wenn bestimmte Dateien eines Directory zu kopieren sind)
  find ... | cpio -o...>*datei* (wenn ein Directorybaum kopiert werden soll)
  Es ist darauf hinzuweisen, dass die Ausgabe dieses Kommandos im allgemeinen nicht lesbar ist, da sie in einem eigenen Format dargestellt wird, das es erlaubt, solche Dateien dann wieder mit **cpio -i** einzukopieren.

- **cpio –i**
  liest ein mit **cpio -o** erzeugtes Archiv von der Standardeingabe und kopiert dessen Inhalt in das Working-Directory. Üblicherweise wird Eingabe-Umlenkung verwendet, um an **cpio -i** eine zuvor angelegte Archivdatei zu übergeben: **cpio -i <archiv-datei**
  Wohin die aus dem Archiv extrahierten Dateien kopiert werden, hängt davon ab, wie die Dateien archiviert wurden. Wenn die Dateinamen relativ zum damaligen Working-Directory (bei **cpio -o**) ins Archiv eingetragen wurden, so werden sie bei **cpio -i** relativ zum nun gültigen Working-Directory kopiert. Wenn z.B. das Archiv mit **ls .. | cpio -o >***datei* angelegt wurde, dann würde **cpio -i** alle Dateien des Archivs ins Parent-Directory zum (neuen) Working-Directory kopieren.
  Normalerweise werden bei **cpio -i** alle Dateien aus einem Archiv kopiert. Sind aber *dateiname(n)* angegeben, so werden nur diese Dateien aus der Standardeingabe extrahiert und in das Working-Directory kopiert. So würde z.B. **cpio -i "*.c"** nur alle C-Programmdateien aus dem Archiv kopieren.
  Bei den *dateiname(n)* sind die Metazeichen `*`, `?`, `[..]` zur Dateinamenexpandierung erlaubt. Die Angabe des Metazeichens `!` (Ausrufezeichen) bedeutet dabei NICHT. So deckt z.B. die Angabe `"!*.txt"` alle Dateien ab, die nicht mit dem Suffix `.txt` enden. Werden Metazeichen verwendet, so muss die Angabe mit Anführungszeichen `"..."` geklammert sein.

- **cpio –p**
  liest (wie **cpio -o**) die Pfadnamen der zu kopierenden Dateien von der Standardeingabe und kopiert deren Inhalt in das entsprechende Directory. So würde z.B. der Aufruf **ls | cpio -p /home/emil** dem Aufruf **cp * /home/emil** entsprechen.

a	(*access*) Zugriffszeit nach Kopieren zurücksetzen auf Zeit vor Kopieren
B	(*Block*) Mit 5120-Byte-Blöcken kopieren
d	Directories anlegen, wenn notwendig
c	Datei-Informationen lesen oder ausgeben
r	interaktives Umbenennen von Dateien möglich
t	(*table*) Inhaltsverzeichnis ausgeben
u	neuere Versionen durch ältere ersetzen
v	(*verbose*) Bearbeitete Dateien melden
l	(*link*) Links erzeugen, wenn möglich
m	(*modification*) Modifikationszeit belassen
f	Alle Dateien außer *datei(en)* kopieren

- **ls -a | cpio -o > /dev/fd0**
  Alle Dateien des Working-Directorys auf Diskette (`/dev/fd0`) kopieren

- **cpio -i "*.c" < /dev/fd0**
  C-Programme von Diskette einkopieren

- **find . –print | cpio -pdv /user2/emil**
  Ganzen Directorybaum des Working-Directorys nach `/user2/emil` kopieren

- **ls a* o* | cpio -o >xx**
  cat xx | cpio -iv "a*"
  kopiert zunächst alle Dateien des Working-Directorys, deren Name mit `a` oder `o` beginnen, nach `xx`. Der zweite Aufruf kopiert aus `xx` alle Dateien, deren Name mit `a` beginnt, in das Working-Directory und gibt dabei alle Namen der kopierten Dateien aus.

- **find . -depth -print | cpio -pdv /tmp**: kopiert den zum Working-Directory gehörigen Directorybaum nach `/tmp`, wobei die Namen der kopierten Dateien ausgegeben werden.

Das **cpio**-Archivformat ist nicht mit dem Format kompatibel, das das Kommando **tar** verwendet.

Um Archivdateien auf externe Speichermedien (wie z.B. Magnetband, Diskette, ..) zu sichern, ist das Kommando **tar** besser geeignet, da dieses blockorientiert ist, so dss jede neue Datei auf einen 1 Kbyte Block untergebracht wird. So würden z.B. drei Dateien, die jeweils nur 15 Bytes belegen, 3 Kbyte auf dem externen Speichermedium belegen. **cpio** dagegen speichert sequentiell (Byte für Byte) und ist damit nicht so platzverschwenderisch, dafür aber langsamer.

Allgemein kann gesagt werden, dass sich heute das Kommando **tar** zum Archivieren durchgesetzt hat.

tar – Archivieren von Dateien und Directorybäumen

tar [*funktion*[*zusatz*]] [*datei(en)*]

Das Kommando **tar** ermöglicht es, Dateien in einem so genannten Archiv zu sichern oder aus einem zuvor erstellten Archiv wieder einzukopieren. Ein Archiv faßt mehrere Dateien zu einer zusammen und enthält zusätzliche Verwaltungsinformation darüber, wo sich die einzelnen Dateien innerhalb des Archivs befinden. Es können sowohl Dateien aus einem Archiv entfernt als auch neue hinzugefügt werden. Auch können Dateien aus einem Archiv extrahiert und in ein Directory kopiert werden. Mit *datei(en)* werden die von **tar** zu bearbeitenden Dateien angegeben. Ist eine der angegebenen *datei(en)* ein Directory, bezieht sich diese Angabe auf ganzen Directorybaum.

- *funktion* legt die auszuführende Aktion fest

r	(*replace*) fügt die angegebenen *datei(en)* am Ende des Archivs an.
x	(*extract*) extrahiert (kopiert) die angegebene *datei(en)* aus Archiv. Falls eine der *datei(en)* ein im Archiv abgelegtes Directory ist, wird ganzee Directorybaum extrahiert. Wenn ein Dateiname im Archiv, aber nicht im Dateisystem existiert, so wird die entsprechende Datei mit den gleichen Zugriffsrechten wie die im Archiv enthaltenen Dateien kreiert, außer dass eventuell gesetzte set-user-id oder set-group-id Bits ausgeschaltet werden. Existiert eine Datei aus dem Archiv bereits auch im Dateisystem, wird die Datei des Dateisystems mit der aus dem Archiv überschrieben, wobei allerdings die Zugriffsrechte unverändert bleiben. Sind keine *datei(en)* angegeben, so wird der vollständige Inhalt des Archivs extrahiert. Wenn das Archiv mehr als eine Version einer Datei enthält, so werden die einzelnen Versionen nacheinander extrahiert, wobei die zuletzt extrahierte Datei alle vorhergehenden überschreibt.
t	(*table*) Alle Namen der im Archiv enthaltenen Dateien auflisten.
u	(*update*) Die *datei(en)* werden nur im Archiv (am Archivende) abgelegt, wenn sie noch nicht dort vorhanden sind oder aber seit letzten Archivierung verändert wurden; ältere Versionen werden im zweiten Fall aus Archiv gelöscht.
c	(*create*) Neues Archiv anlegen und Dateien am Archivanfang und nicht – wie sonst – am Archivende ablegen.

Unter Linux bietet **tar** noch einige weitere *funktionen* an, wie z.B.:

d	vergleicht die Dateien der Archivdatei mit den Dateien des Working-Directorys
--delete	löscht Dateien aus einem Archiv

- *zusatz* läßt Zusatzangaben zu der auszuführenden Aktion zu:

v	(*verbose*) Bei dieser Option wird der Name jeder übertragenen Datei ausgegeben. Wird **v** mit der *funktion* **t** angegeben, wird noch umfangreichere Information (Dateigröße usw.) ausgegeben.
w	(*what*) veranlaßt **tar**, vor jeder Aktion mit der Ausgabe des Dateinamens und der Art der Aktion den Benutzer zu fragen, ob er dies wünscht. Bei der Eingabe von **y** wird die entsprechende Aktion ausgeführt; bei jeder anderen Eingabe wird die Aktion nicht ausgeführt; darf nicht mit der funktion **t** angegeben werden.
f *datei*	(*file*) *datei* wird als Archiv verwendet. Ist dieser *zusatz* nicht angegeben, so wird eine voreingestellte Datei (üblicherweise Gerätedatei eines Magnetbands oder Disketten-Laufwerks, die in `/etc/default/tar` angegeben ist) als Archiv verwendet. Wird für *datei* der Bindestrich – angegeben, so wird abhängig davon, ob **tar** schreibt oder extrahiert entweder die Standardeingabe oder Standardausgabe als Archiv benutzt; dies ermöglicht es, **tar** auf der rechten oder linken Seite einer Pipe anzugeben.
l	(*link*) veranlaßt **tar** zu melden, wenn es nicht alle geforderten Links zu den archivierten Dateien herstellen kann. Dieser *zusatz* macht nur Sinn mit den funktionen **c**, **r** und **u**.
m	(*modify*) veranlaßt **tar** nicht den Modifikations-Zeitstempel einer Datei zu verwenden, sondern als Modifikationszeit für eine Datei den Zeitpunkt des Extrahierens einzutragen.
o	(*ownership*) Für die aus dem Archiv extrahierten Dateien ist die Benutzer- und Gruppenkennung des Aufrufers von **tar** und nicht die aus dem Archiv einzutragen. Dieser *zusatz* ist nur bei der *funktion* **x** erlaubt.
L	Symbolische Links auflösen. Voreinstellung ist, dass symbolische Links nicht aufgelöst werden.

Unter Linux bietet **tar** noch einige weitere *zusätze* an, wie z.B.:

z bzw. **I**	Daten im Archiv (mit **gzip** bzw. mit **bzip2**) komprimieren bzw. beim Extrahieren dekomprimieren.
Z	Daten im Archiv (mit **compress**) komprimieren bzw. beim Extrahieren dekomprimieren.
C *directory*	extrahiert die Dateien in das angegebene *directory* und nicht in das Working-Directory.
N *datum*	archiviert nur Dateien, die neuer als das angegebene *datum* sind.
T *datei*	archiviert bzw. extrahiert nur die in *datei* angegebenen Dateinamen.
W	überprüft nach Schreiben Korrektheit der archivierten Dateien; nicht für komprimierte Archive zugelassen.

Das **tar**-Kommando wird verwendet, um Dateien auf Magnetband oder Diskette zu sichern oder von dort wieder ins Dateisystem einzukopieren. **tar** wird auch häufig verwendet, um ganze Directorybäume in einer Datei zu archivieren.

- **tar r evening.c**
  Die Datei `evening.c` wird an das Ende des voreingestellten Archivs (meist Magnetband oder Diskette) kopiert.
- **tar add*.c**
  Alle Dateien, deren Name mit `add` beginnen und mit `.c` enden, werden an das Ende des voreingestellten Archivs (meist Magnetband oder Diskette) kopiert.
- **cd /home/egon/bin; tar cf – . | (cd /home/gruppe/util; tar xf -)**
  kopiert den gesamten Directorybaum `/home/egon/bin` nach `/home/gruppe/util/bin` um.
- **tar cf /dev/fd0 briefe**
  kopiert das ganze Directory `briefe` auf Diskette. Der Name des Disketten-Laufwerks ist hier `/dev/fd0`.
- **tar xvf /dev/fd0 briefe/hans**
  kopiert die Datei `briefe/hans` von der Diskette `/dev/fd0` mit Meldung ein.
- **tar cf cppquellen.tar c++/src**
  archiviert ohne Meldungen den gesamten Directorybaum `c++/src` in der Datei `cppquellen.tgz`.
- **tar czvf cppquellen.tgz c++/src**
  archiviert mit Ausgabe von Meldungen den gesamten Directorybaum `c++/src` in der komprimierten Datei `cppquellen.tgz`.
- **tar cIf sicher.tar.bz2 .**
  archiviert und komprimiert alle Dateien des Directorybaums zum Working-Directory in der Datei `sicher.tar.bz2`.
- **tar tzf backup.tgz**
  zeigt alle Namen der Dateien an, die sich im Archiv `backup.tgz` befinden. Unter Linux ist **less** meist so konfiguriert, dass man sich auch den Inhalt von Archivdateien mit diesem Kommando anzeigen lassen kann, wie z.B. **less backup.tgz**.
- **tar xzf backup.tgz '*.c'**
  extrahiert nur C-Programmdateien aus dem Archiv `backup.tgz`.

Die typische Dateiendung von **tar**-Archivdateien ist `.tar` oder bei komprimierten Archiven: `.tgz`, `.tar.gz` oder `.tar.bz2`. Zu **tar** existiert auch mit **xtar** eine komfortable graphische Benutzeroberfläche.

2.7 Ändern von Zugriffsrechten, Eigentümer, Gruppe, Zeitmarken

chmod – Zugriffsrechte von Dateien oder Directories ändern

chmod [**-R**] *absolut-modus datei(en)* bzw. **chmod** *symbolischer-modus datei(en)*

Mit **chmod** können Zugriffsrechte von Dateien oder Directories geändert werden. Jedoch kann nur der Superuser oder der Besitzer die Zugriffsrechte für eine Datei bzw. ein Directory ändern. Bei Angabe der Option **–R** durchsucht **chmod** ein Directory rekursiv, d.h. inklusive aller Subdirectories, und ändert die Rechte aller Dateien, die es dort findet. Für *absolut-modus* muss Oktalwert angegeben werden, der festlegt, welche der 12 Bits des Dateimodus für *datei(en)* zu setzen bzw. zu löschen sind. Dabei hat jedes einzelne der 12 Bits folgende Bedeutung:

0400	Leserecht (*read*) für den Eigentümer (*user*)
0200	Schreibrecht (*write*) für den Eigentümer (*user*)
0100	Ausführrecht (*execute*) für den Eigentümer (*user*)
0040	Leserecht (*read*) für die Gruppe (*group*)
0020	Schreibrecht (*write*) für die Gruppe (*group*)
0010	Ausführrecht (*execute*) für die Gruppe (*group*)
0004	Leserecht (*read*) für die Anderen (*others*)
0002	Schreibrecht (*write*) für die Anderen (*others*)
0001	Ausführrecht (*execute*) für die Anderen (*others*)
4000	*set-user-id*: Dieses Bit wird nur für ausführbare Dateien ausgewertet. Ist dieses Bit gesetzt, hat jeder Benutzer, der dieses Programm ausführt, für Dauer der Programmausführung die gleichen Rechte wie der Besitzer dieses Programms
20#0	*set-group-id*: Ist execute-Recht für die Gruppe gesetzt (# ist gleich 7, 5, 3 oder 1), werden dem Aufrufer dieses Programms für die Zeit der Programmausführung die gleichen Rechte gewährt, wie wenn er Mitglied der Gruppe wäre, der diese Datei gehört. Ist execute-Recht für die Gruppe nicht gesetzt (# ist gleich 6, 4, 2 oder 0), wird Datei für alleinigen Lese- und/oder Schreibzugriff zur Verfügung gestellt, d.h. dass diese Datei für Lese- und/oder Schreibzugriffe durch andere Programme gesperrt wird, solange ein Programm auf diese Datei zugreift. Dieses Bit wird bei Directories ignoriert. Soll dieses Bit bei Directories gesetzt oder gelöscht werden, muss der *symbolische-modus* verwendet werden.
1000	*sticky bit*: Nach Ausführung des in dieser Datei enthaltenen Programms wird dieses nicht – wie sonst üblich – aus dem Hauptspeicher entfernt; dieses Bit kann nur vom Superuser eingeschaltet werden. Ist unter Linux dieses Bit für ein Directory gesetzt, so bedeutet dies, dass in diesem Directory jeder Benutzer nur seine eigenen Dateien ändern, löschen oder umbenennen darf. Dieses Sticky-Bit ist z.B. bei dem allen Benutzern zugänglichen Directory `/tmp` gesetzt, so dass dort nicht jeder Benutzer nach Belieben fremde Dateien ändern, löschen oder umbenennen kann.

*symbolischer-modus: **[ugoa]** +|-|= **[rwxslt]**. Dabei bedeuten die einzelnen Zeichen:*

u	für den Eigentümer (*user*)
g	für die Gruppe (*group*)
o	für die anderen Benutzer (*others*)
a	für alle 3 Benutzerklassen (*all*); entspricht der Angabe **ugo**. Keine Angabe entspricht auch der Angabe **a**.
+	Rechte hinzufügen (relativ)
-	Rechte entziehen (relativ)
=	Rechte als neue Zugriffsrechte vergeben (absolut)
r	für Lese-Recht (*read*)
w	für Schreib-Recht (*write*)
x	für Ausführ-Recht (*execute*)
s	für *set-user-id* (in Zusammenhang mit **u**) oder für *set-group-id* (in Zusammenhang mit **g**)
t	für *sticky bit*; nur im Zusammenhang mit **u** wirkungsvoll
l	für exklusiven Lese- und/oder Schreibzugriff

Wenn = verwendet wird, dann muss kein Zugriffsrechte-Muster angegeben sein; fehlendes Zugriffsmuster bedeutet dabei: „*Entfernen aller entsprechenden Zugriffsrechte*“.

Um einer *datei* Zugriffsrechte `s--rwxr-xr--` zu geben, könnte einer der folgenden Aufrufe angegeben werden:

chmod u=rwxs,g=rx,o=r *datei* oder **chmod 4754** *datei*

			user			group			others			
s	-	-	r	w	x	r	-	x	r	-	-	symbolischer Modus
1	0	0	1	1	1	1	0	1	1	0	0	(dual) absoluter Modus
4			7			5			4			(oktal)

Bei der Ausgabe eines Dateinamens mit dem Kommando **ls -l** werden immer nur 9 Bits angezeigt. Wenn das `s`-Bit, `t`-Bit oder `l`-Bit gesetzt ist, so wird in diesem Fall das jeweilige `x`-Bit bei der Ausgabe mit **ls** überschrieben:

`s` (kleines s)	set-user-id-Bit gesetzt und Ausführrecht
`t` (kleines t)	sticky-Bit gesetzt und Ausführrecht
`T` (großes t)	sticky-Bit gesetzt und kein Ausführrecht

Es ist wichtig zu wissen, dass das Recht, eine Datei anzulegen oder zu löschen, ausschließlich von den Zugriffsrechten des Directorys abhängt. Die Kommandos **ln**, **rm** und **mv** geben deswegen zur Absicherung eine Warnung in Form einer Rückfrage aus, wenn man mit ihnen eine Datei überschreiben oder löschen will, die keine Schreibrechte hat. Antwortet man auf diese Rückfrage mit **y** (*yes*), so wird die Datei überschrieben bzw. gelöscht. Bei jeder anderen Eingabe bleibt die Datei unberührt. Ist man sich absolut sicher, dass man die entsprechenden Dateien überschreiben bzw. löschen möchte, und man deshalb die Rückfrage als lästig empfindet, so muss man beim Aufruf dieser Kommandos nur die Option **-f** (*force*) angeben. Dateien, die sich in Directories befinden, die keine Schreibrechte haben, können niemals gelöscht oder überschrieben werden.

- **chmod u+x skript**: Ausführrecht für den Eigentümer der Datei `skript` hinzufügen.
- **chmod 751 add**: An Datei `add` das Zugriffsrechtemuster `rwxr-x–x` vergeben.

chattr – Ändern der Attribute einer Datei (unter Linux)

chattr [*optionen*] [**+/-csSu**] *datei(en)*

chattr ändert Version sowie sechs zusätzl. Dateiattribute, die nur bei ext2-Dateisystemen unterstützt werden. Die Dateiversion gibt an, wie oft Datei bereits geändert wurde. Attribute werden durch + ein- und durch – ausgeschaltet.

+/-a	(*append*) Datei kann zwar verlängert, aber nicht gelöscht oder überschrieben werden.
+/-c	(*compressed*) Datei wird automatisch komprimiert bzw. dekomprimiert.
+/-s	(*secure delete*) Dateiinhalt wird durch Zufallsdaten überschrieben, so dass er nicht mehr rekonstruierbar ist.
+/-S	(*synchronous write*) Jede Änderung an der Datei wird sofort physikalisch durchgeführt und nicht zwischengepuffert.
+/-u	(*undelete*) Datei wird beim Löschen nicht physikalisch überschrieben und könnte eventuell wiederhergestellt werden.
-R	Entsprechende Attribute werden rekursiv in allen Subdirectories geändert.

chown – Eigentümer von Dateien oder Directories ändern

chown [*optionen*] *neuer_eigentümer datei(en)*

Eigentümer einer Datei ist zunächst der Benutzer, der diese angelegt hat. **chown** erlaubt Eigentümer von Dateien oder Directories zu ändern. Für *neuer_eigentümer* ist der Login-Name oder User-ID des neuen Besitzers anzugeben.

-R	Eigentümer von allen Dateien und Subdirectories in einem Directory werden geändert.
-h	Eigentümer eines symbolischen Links und nicht der der Zieldatei wird geändert.

Der Eigentümer einer Datei kann nur dann erfolgreich geändert werden, wenn der Aufrufer von **chown** Superuser oder aber der Besitzer der entsprechenden Datei oder des Directorys ist. Wenn der Eigentümer der Datei (nicht der Superuser) dieses Kommando aufruft, dann werden eventuell gesetzte setuid- und setgid-Bits gelöscht. Unter Linux kann mit **chown** nicht nur der Eigentümer, sondern auch zugleich die Gruppe einer Datei geändert werden. Dazu muss hinter *neuer_eigentümer* noch mit Punkt oder Doppelpunkt abgetrennt die *neue_gruppe* angegeben werden:

chown [*optionen*] *neuer_eigentümer*.*neue_gruppe datei(en)* oder
chown [*optionen*] *neuer_eigentümer*:*neue_gruppe datei(en)*

chgrp – Gruppe von Dateien oder Directories ändern

chgrp [*optionen*] *neuer_gruppe datei(en)*

Die Benutzergemeinde eines Systems ist organisatorisch in verschiedene Gruppen aufgeteilt; die Datei `/etc/passwd` gibt an, welcher Gruppe jeder einzelne Benutzer zugeordnet ist. Jede Gruppe hat eine Gruppennummer (Group-ID) und einen Namen. Die Datei `/etc/group` enthält die Gruppennummer und die Gruppenmitglieder für jede Gruppe. Jede Datei hat nun nicht nur einen Eigentümer, sondern auch eine Gruppenzugehörigkeit. Die Zugriffsmöglichkeiten von Mitgliedern der entsprechenden Gruppe auf eine Datei ist dabei über die *group*-Zugriffsrechte festgelegt. Um für eine Datei die Gruppenzugehörigkeit zu wechseln, steht **chgrp** zur Verfügung. Es verändert die Gruppenzugehörigkeit der angegebenen Dateien oder Directories. Für *neue_gruppe* muss entweder der entspr. Gruppenname oder die Group-ID der neuen Gruppe angegeben werden.

-R	Gruppenzugehörigkeit von allen Dateien und Subdirectories in einem Directory werden geändert.
-h	Gruppenzugehörigkeit eines symbolischen Links und nicht der der Zieldatei wird geändert.

Die Gruppenzugehörigkeit einer Datei kann nur dann erfolgreich geändert werden, wenn der Aufrufer Superuser oder aber der Besitzer der entsprechenden Datei oder des Directorys ist. Wenn der Eigentümer der Datei (nicht der Superuser) dieses Kommando aufruft, dann werden die setuid- und setgid-Bits gelöscht.

umask – Dateikreierungsmaske setzen bzw. ausgeben

umask [*3-stellige-oktalzahl*]

Um die Sicherheit unter Unix etwas zu verbessern, wurde die so genannte *Dateikreierungsmaske* eingeführt: dies ist ein 9-Bit-Wert, der die Rechte festlegt, die nicht beim Anlegen neuer Dateien zu gewähren sind. Das Kommando **umask** setzt die Dateikreierungsmaske mit dem Wert der *3-stellige-oktalzahl*. Wird **umask** ohne Angabe eines Arguments aufgerufen wird, dann gibt es den Wert der momentanen Kreierungsmaske aus. Die Dateikreierungsmaske hat allerdings nur Auswirkungen auf die Zugriffsrechte neu anzulegender Dateien; die Zugriffsrechte bereits bestehender Dateien bleiben vom Verändern der Dateikreierungsmaske unbeeinflußt. Unter Linux gelten z.B. die folgenden Voreinstellungen:

- neue Dateien erhalten die Zugriffsrechte `rw-rw-rw-` (oktal `666`), so dass sie von jedem gelesen und geändert werden können.
- neue Programmdateien, die von einem Compiler erzeugt werden, erhalten die Zugriffsrechte `rwxrwxrwx` (oktal `777`), so dass sie von jedem ausgeführt werden können.

Nachfolgend dazu einige Beispiele:

- **umask 022**: Diese häufig verwendete Dateikreierungsmaske legt fest, dass neue Dateien die Zugriffsrechte 666-022=`644` (`rw-r--r--`) und Programme die Zugriffsrechte 777-022=755 (`rwxr-xr-x`) erhalten.
- **umask 077**: Diese Dateikreierungsmaske wäre für Benutzer denkbar, die mit sehr geheimen Daten umgehen. Sie bewirkt, dass neue Dateien automatisch die Zugriffsrechte 666-077=600 (`rw-------`) und Programme die Zugriffsrechte 777-077=`700` (`rwx------`) erhalten.

Die Dateikreierungsmaske hat keine Auswirkung auf Kommandos wie **cp** oder **mv**, die immer die Zugriffsrechte der Originaldatei mitkopieren. Üblicherweise wird dieses Kommando **umask** in der Datei `~/.profile` aufgerufen, um so automatisch bei jedem Anmeldevorgang die Dateikreierungsmaske entsprechend zu setzen.

touch - Ändern des Zugriffs- und Modifikations-Zeitstempels für Dateien

touch [**-amc**] [*mmtthhmm*[*jj*]] *datei(en)*

Mit **touch** können die im inode eingetragenen Zugriffs- und Modifikations-Zeitstempel für Dateien direkt geändert werden. Eistiert eine der *datei(en)* nicht, wird sie angelegt, allerdings nur, wenn nicht die Option **-c** angegeben ist.

-a	Ändern des Zugriffs-Zeitstempel
-m	Ändern des Modifikations-Zeitstempel
-c	Existiert eine der *datei(en)* nicht, wird sie nicht angelegt; Voreinstellung ist: Anlegen einer nicht existierenden *datei*.

Ist keine Option angegeben, so werden beide Zeitstempel geändert. Die Zeitangabe [*mmtthhmm*[*jj*]] legt die einzutragende Zeit fest: zuerst Monatszahl (*mm*), dann Tag (*tt*), dann Stunde (*hh*) und schließlich Minute (*mm*); Jahresangabe (*jj*) ist auch noch möglich, allerdings nicht gefordert. Fehlt die Zeitangabe, so wird die momentane Uhrzeit und das heutige Datum verwendet.

2.8 Umformen, Extrahieren und Zerteilen von Dateien

tr – Umformen von Dateien

tr [*optionen*] *string1* [*string2*]

tr kopiert den Eingabetext, den es von der Standardeingabe liest, auf die Standardausgabe. Dabei können die gelesenen Zeichen durch andere – auch nicht druckbare – Zeichen ersetzt werden. Wird im Eingabetext ein Zeichen gefunden, das in *string1* vorkommt, so wird es durch das entsprechende Zeichen aus *string2* ersetzt. Innerhalb der *strings* können auch Abkürzungen verwendet werden, um ganze Bereiche von ASCII-Zeichen festzulegen, wie z.B.

[A-Z]	alle Großbuchstaben
[0-9]	alle Ziffern
[a*n]	steht für *n* Wiederholungen von *a*. Fehlt Angabe von *n* oder ist dafür 0 angegeben, wird ein großer Wert angenommen.

Auch kann der ASCII-Wert eines Zeichens innerhalb von *strings* oktal angegeben werden:

\012	für Neuezeilezeichen
[\001-\014]	alle Zeichen mit den oktalen ASCII-Codes von 1 bis 14; dezimal: von 1 bis 12

tr bietet u.a. folgenden Optionen an:

-c	Die Zeichen, die in *string1* vorkommen, werden bezüglich des ASCII-Codes (oktal: 001 bis 377) komplementiert.
-d	Eingabezeichen, die in *string1* vorkommen, werden gelöscht.
-s	Für gleiche hintereinander stehende Ausgabezeichen, die in *string2* vorkommen, wird nur ein Zeichen ausgegeben.

- **tr "[a-z]" "[A-Z]" <laender > laender2**: schreibt den Inhalt der Datei `laender` in die Datei `laender2`, wobei alle Kleinbuchstaben durch Großbuchstaben ersetzt werden.
- **tr -dc "[A-Z]" <laender**: gibt von der Datei `laender` nur die Großbuchstaben aus.
- **cat *.tex | tr –cs "[a-z][A-z]" "[\012]" sort | uniq –c**: gibt zu allen `.tex`-Dateien eine Statistik aus, in der die Worte alphabetisch sortiert sind und zu jedem Wort angegeben ist, wie oft es in den `.tex`-Dateien vorkommt.

uniq – Identische Zeilen nur einmal ausgeben

uniq [*optionen*] [*eingabedatei* [*ausgabedatei*]]

Wenn keine *ausgabedatei* angegeben ist, so erfolgt die Ausgabe auf die Standardausgabe; ist weder eine *ausgabedatei* noch eine *eingabedatei* angegeben, so wird der Eingabetext von der Standardeingabe gelesen und das Ergebnis auf die Standardausgabe geschrieben.

uniq liest den Eingabetext und vergleicht aufeinanderfolgende Zeilen miteinander. Wenn zwei oder mehrere aufeinanderfolgende Zeilen identisch sind, so wird von diesen Zeilen nur eine ausgegeben; alle anderen Zeilen, auf die das nicht zutrifft, werden unverändert ausgegeben. Für *eingabedatei* und *ausgabedatei* sollten zwei verschiedene Dateien angegeben werden.

-u	Nur die Zeilen ausgeben, die nicht mehrfach hintereinander vorkommen.
-d	Nur von den mehrfach hintereinander vorkommenden Zeilen je eine ausgeben.
-c	Zu jeder Zeile angeben, wie oft sie hintereinander vorkommt.
+n	Die ersten *n* Zeichen werden beim Vergleich aufeinanderfolgender Zeilen ignoriert.
-m	Die ersten *m* Felder werden beim Vergleich aufeinanderfolgender Zeilen ignoriert; als Trennzeichen für die Felder werden Leer- und Tabulatorzeichen verwendet.

Um von allen mehrfach vorkommenden Zeilen wirklich nur eine ausgeben zu lassen, ist eventuell eine vorherige Sortierung einer Datei notwendig, da **uniq** nur direkt aufeinanderfolgende Zeilen auf Gleichheit hin überprüft, wie z.B.: **sort namliste | uniq**

expand – Umwandeln von Tabulatoren in Leerzeichen

expand [*optionen*] [*datei(en)*]

expand wandelt in allen angegebenen *datei(en)* die Tabulatorzeichen in eine entsprechende Anzahl von Leerzeichen um. Sind keine *datei(en)* angegeben oder ist ein Minjuszeichen (**–**) angegeben, liest **expand** den umzuwandelnden Text von Standardeingabe. Das Ergebnis der Umwandlung gibt **expand** immer auf der Standardausgabe aus.

-i	(*initial*) nur die am Zeilenanfang stehenden Tabulatoren (vor den ersten wirklichen Zeichen) werden umgewandelt.
-t *n1*[,*n2*,..]	(*tabs*) legt Tabpositionen fest, die mit Leerzeichen zu füllen sind. Voreinst. ist: **-t 8** (alle 8 Spalten eine Tabposition). Wird nur *n1* angegeben, wird alle *n1* Spalten eine Tabposition gesetzt. Ist Liste angegeben (*n1*,*n2*,...), legen Nummern explizit die einzelnen Tabpositionen fest. Statt **-t** *n1*[,*n2*,..] ist auch **-***n1*[,*n2*,..] erlaubt.

cut – Herausschneiden von Spalten oder Felder aus Dateien

cut -c*spalten* [*datei(en)*]
cut -f*felder* [**-d***zeichen*] [**-s**] [*datei(en)*]

Sind keine *datei(en)* angegeben, wird der Eingabetext von der Standardeingabe gelesen. Dasselbe gilt, wenn anstelle von *datei(en)* ein Querstrich (`-`) angegeben ist.

-c*spalten*	Die dabei angegebenen *spalten* legen die herauszuschneidenden Spalten fest. Für *spalten* können dabei mit Komma getrennte ganze Zahlen oder Zahlenbereiche angegeben werden. Z.B. würde **-c-3,7,25-** festlegen, dass die 1. bis 3. Spalte, die 7. Spalte und alle Spalten ab der 25. herauszuschneiden sind.
-f*felder*	Die dabei angegebenen *felder* geben die Nummern der herauszuschneidenden Felder an. Für *felder* können dabei mit Komma getrennte ganze Zahlen oder Zahlenbereiche angegeben werden. Z.B. würde **-f1,3,5-7** festlegen, dass das 1. Feld, das 3. Feld und die Felder 5 bis 7 herauszuschneiden sind. Als Trennzeichen für die einzelnen Felder wird dabei – wenn nicht anders mit der Option **-d***zeichen* angegeben – das Tabulatorzeichen verwendet.
-d*zeichen*	*zeichen* wird als Trennzeichen für die Felder verwendet. Voreinst. ist Tabzeichen als Trennzeichen.
-s	Alle Zeilen, in denen das Trennzeichen nicht vorkommt, werden nicht ausgegeben; normalerweise werden solche Zeilen vollständig ausgegeben.

Um sich z.B. alle Dateien des Working-Directorys auflisten zu lassen, wobei nur die Zugriffsrechte (1. bis 11. Zeichen) und die Dateinamen (ab 56. Zeichen) anzuzeigen sind, könnte man aufrufen: **ls –l | cut –c1-11,56**.

fold – Einfaches Formatieren von Dateien

fold [*optionen*] *datei*

fold bricht Textzeilen der *datei* nach 80 Zeichen um und gibt das Ergbnis auf der Standardausgabe aus:

-w *n*	nach *n* Zeichen statt nach 80 Zeichen umbrechen
-s	versucht, den Umbruch an der Stelle eines Leerzeichens, also zwischen zwei Wörtern durchzuführen. Die Zeilen sind kleiner oder gleich *n* Zeichen lang.

Hat man z.B. die folgende Datei `gedicht.txt`:

```
Ob wir rote, gelbe Kragen, Helme oder Hüte tragen,
Stiefel oder Schuh; oder ob wir Röcke nähen
zu Schuhen Drähte drehen; Das tut, das tut nichts dazu.
```

dann würden die folgenden Aufrufe die nachfolgend gezeigten Ausgaben liefern:

fold -w 30 gedicht.txt

```
Ob wir rote, gelbe Kragen, Hel
me oder Hüte tragen,
Stiefel oder Schuh; oder ob wi
r Röcke nähen
zu Schuhen Drähte drehen; Das
tut, das tut nichts dazu.
```

fold -s -w 30 gedicht.txt

```
Ob wir rote, gelbe Kragen,
Helme oder Hüte tragen,
Stiefel oder Schuh; oder ob
wir Röcke nähen
zu Schuhen Drähte drehen; Das
tut, das tut nichts dazu.
```

fmt – Einfaches Formatieren von Dateien

fmt [*optionen*] [*datei(en)*]

fmt ist ein einfacher Textformatierer, mit dem sich beliebige ASCII-Texte formatieren lassen. **fmt** füllt Zeilen auf oder bricht sie um (Blocksatz). Den zu formatierenden Text liest **fmt** aus den angegebenen *datei(en)* und schreibt die formatierten Zeilen auf die Standardausgabe. Wenn keine *datei(en)* angegeben sind, liest **fmt** den zu formatierenden Text aus der Standardeingabe. Leerzeilen bleiben beim Formatieren ebenso erhalten wie Leerzeichen zwischen den Wörtern. Zeilen, die mit Punkt (`.`) oder mit "`From:`" beginnen, füllt **fmt** nicht auf. Einrückungen im Originaltext werden beibehalten. **fmt** kann auch im **vi** benutzt werden. Um z.B. den Text zwischen der Cursorposition und dem Ende eines Absatzes zu formatieren, muss das vi-Kommando **!}fmt** eingegeben werden.

-s	(*split lines only*) es werden zwar "überlange" Zeilen umbrochen, kürzere Zeilen aber nicht zu einer längeren Zeile zusammengefügt (*Flattersatz*); kann für Programmlistings verwendet werden.
-w *länge*	Zeilen auf *länge* Zeichen auffüllen; Voreinstellung ist **-w 72**.

pr – Formatieren einer Datei zum Drucken

pr [*optionen*] [*datei(en)*]

Ähnlich dem Kommando **cat** gibt das Kommando **pr** den Inhalt von Dateien auf der Standardausgabe aus, aber so aufbereitet, dass sich die Ausgabe für einen Drucker eignet: Für eine Seite werden dabei 66 Zeilen (amerikanisches Format) angenommen und für jede einzelne Seite wird oben ein 5-zeiliger Kopf (zwei Leerzeilen, Datum und Uhrzeit der letzten an dieser Datei vorgenommenen Änderung sowie eine Seitennummer und der Dateiname, und nochmals zwei Leerzeilen) und unten ein 5-zeiliger Fuß (5 Leerzeilen) ausgegeben. Wenn die letzte Seite keine 66 Zeilen umfasst, so wird sie bei der Ausgabe mit Leerzeilen aufgefüllt. Wenn keine *datei(en)* angegeben sind oder Minuszeichen (**–**) für *datei(en)* angegeben ist, so liest **pr** von der Standardeingabe.

+*n*	Ausgabe beginnt mit der Seite *n* (muss ganze Zahl sein); Voreinstellung für *n* ist 1
-*n*	bewirkt, dass die Ausgabe der angegebenen *datei(en)* in *n* Spalten erfolgt; Voreinstellung für *n* ist 1.
-a	bewirkt eine mehrspaltige Ausgabe über eine Seite hinaus; das bedeutet, dass die erste Zeile aus der Eingabe in der ersten , die zweite Zeile in der zweiten Spalte usw. ausgegeben wird; sollte nur in Verbindung mit **-***n* benutzt werden.
-m	bewirkt, dass die angegebenen *datei(en)* nebeneinander ausgegeben werden: jede Datei in einer eigenen Spalte; darf nicht mit **-***n* verwendet werden.
-d	bewirkt eine doppelten Zeilenvorschub für jede Zeile; dadurch entstehende Leerzeilen am Anfang einer Seite werden allerdings wieder entfernt.
-w*n*	setzt für eine mehrspaltige Ausgabe die Zeilenlänge auf *n* Zeichen; Voreinstellung ist 72.
-o*n*	rückt jede Zeile um *n* Zeichen ein; Voreinstellung ist 0.
-l*n*	Bei der in Deutschland üblichen Seitenlänge von 30,5 cm ergeben sich in der Regel 72 Zeilen pro Seite. Dies kann **pr** über die Option **-l72** mitgeteilt werden; Voreinstellung ist 66.
-h *string*	ersetzt den Dateinamen im Kopf durch den angegebenen *string*.
-p	bei der Ausgabe auf dem Bildschirm wird nach jeder Seite angehalten. Mit der Eingabe von (Enter) kann dann die Ausgabe der nächsten Seite erreicht werden.
-f	verwendet ein Zeilenvorschub-Zeichen (*form feed*), um eine neue Seite zu erzeugen. Normalerweise wird mit Leerzeichen aufgefüllt.
-r	für Dateien, die nicht geöffnet werden können, wird keine Fehlermeldung ausgegeben.
-t	unterdrückt die Ausgabe des 5-zeiligen Kopfes und des 5-zeiligen Fußes; auch wird bei der letzten Seite nicht mit Leerzeilen aufgefüllt, um eine Seite zu vervollständigen.
-s*c*	verwendet als Trennzeichen für die einzelnen Spalten das Zeichen *c*; Voreinstellung ist das Tabulatorzeichen.

Um Dateien wirklich auf einen Drucker auszugeben, steht das Kommando **lp** bzw. **lpr** zur Verfügung. Für die Druckaufbereitung wie z.B. Seiten- und Zeilennumerierung kann dagegen **pr** verwendet werden:

pr ... | lp bzw. **pr ... | lpr**

Die Kommandos **join** und **paste** sind eine Alternative zur Option **-a** (Mischen von Dateien bei der Ausgabe). **join** erlaubt es dabei, bestimmte Zeilen aus unterschiedlichen Dateien auszuwählen, während **paste** die Möglichkeit anbietet, Zeilen mit einem Tabulatorzeichen oder einem sonstigen Trennzeichen nebeneinander auszugeben, wobei diese Ausgabe nicht in gleich langen Spalten erfolgt.

split – Zerteilen von Dateien

split [*-n*] [*datei* [*name*]]

Das Kommando **split** liest die angegebene *datei* und zerteilt sie in einzelne Stücke mit je *n* Zeilen. Ist *-n* nicht angegeben, so werden Einzelstücke mit 1000 Zeilen gebildet. Ist *datei* nicht angegeben oder ist für *datei* ein Minuszeichen **–** angegeben, so liest **split** von der Standardeingabe.

name legt dabei ein Präfix fest, aus dem dann die Namen der Dateien gebildet werden, in welche die einzelnen Stücke abgelegt werden; die Namensgebung für diese einzelnen Dateien erfolgt durch Anhängen von `aa`, `ab`, `ac`, ..., `zz` an das Präfix *name*. Ist *name* nicht angegeben, so wird als Präfix `x` verwendet und die Namen für die "Stück-Dateien" wären dann `xaa`, `xab`, `xac` usw. Mit dieser Art der Namensgebung ist es möglich, maximal 676 "Stück-Dateien" zu erzeugen. Die "Stück-Dateien" werden immer im Working-Directory angelegt.

Unter Linux bietet das Kommando **split** noch einige weitere sehr nützliche Optionen:

-b *n*	zerlegt eine Datei in Teildateien mit jeweils *n* Bytes. Wird nach der Zahl *n* einer der folgenden Buchstaben angegeben, so legt *n* nicht Bytes, sondern folgendes fest: **b** (512-Byte-Blöcke), **k** (Kilobytes) oder **m** (Megabytes)
-C *n*	verhält sich wie **-b** *n*, nur dass die entsprechende Datei immer an Zeilengrenzen zerlegt wird, so dass sich in den "Stück-Dateien" immer nur ganze Zeilen befinden, was natürlich dazu führt, dass die betreffenden "Stück-Dateien" meist nicht genau *n* Bytes groß sind.

- **split -b 1430k document.ps disk.**
  zerlegt die Datei *document.ps* in einzelne Dateien zu je 1430 Kbytes und benennt sie `disk.aa`, `disk.ab`, `disk.ac` usw. Diese Dateien könnten dann anschließend auf einzelne Disketten kopiert werden. Um die Einzeldateien wieder zur ursprünglichen Gesamtdatei zusammenzusetzen, müßte nur folgendes aufgerufen werden: **cat disk.* >document.ps**

Das Zerteilen einer Datei kann z.B. dann erforderlich sein, wenn diese für die Bearbeitung mit einem Editor zu groß ist oder wenn sie größer als die Kapazität einer Diskette ist, auf die sie kopiert werden soll. Auch nützlich ist das Zerteilen einer Datei, wenn man eine sehr große Datei, wie z.B. ein PostScript-Dokument mit 40 Megabyte per E-Mail verschicken möchte. In diesem Fall sollte man dieses PostScript-Dokument in einzelne Dateien zerteilen, die man einzelnen verschicken kann. Der Empfänger muss dann nur **cat** mit diesen Einzelstücken aufrufen und die Ausgabe in eine Datei umlenken, die dann das vollständige PostScript-Dokument enthält.

csplit – Kontextabhängiges Zerteilen von Dateien

csplit [*optionen*] *datei schnittstelle(n)*

Das Kommando **csplit** zerteilt eine Datei in mehrere kleinere Dateien.

datei ist der Name der Datei, die zu zerteilen ist. Wenn für *datei* ein Querstrich (`-`) angegeben ist, so liest **csplit** von der Standardeingabe; dies ist nützlich, um **csplit** auf der rechten Seite einer Pipe anzugeben.

schnittstelle(n) sind Argumente, die Punkte festlegen, an denen die angegebene *datei* zu zerteilen ist. Die Schnittstellen können dabei über Zeilennummern oder über einen Kontext festgelegt werden. Die Angabe des Kontexts erfolgt dabei über einen regulären Ausdruck. Die ursprüngliche Datei wird von **csplit** nicht verändert. Im Unterschied zu **split** ist es bei **csplit** möglich, Teile der Originaldatei zu überspringen und somit nicht herauszuschneiden.

-f *präfix*	legt dabei das Präfix für die Namen der neuen kleineren Dateien fest. Ist die Option **–f** *präfix* nicht angegeben, so werden die ausgeschnittenen Teilstücke in Dateien mit den Namen `xx00`, `xx01` usw. abgelegt. Bei den Namen für die "Stück-Dateien" wird also bei dieser Option anstelle von `xx` das hier angegebene *präfix* den Ziffernpaaren `00`, `01` usw. vorangestellt.
-s	**csplit** gibt normalerweise die Zeichenzahl jeder neu erzeugten "Stück-Datei" aus. Mit der Angabe dieser Option kann diese Ausgabe unterdrückt werden.
-k	Normalerweise löscht **csplit** alle seine zuvor kreierten "Stück-Dateien", wenn während seiner Ausführung ein Fehler auftritt, wie z.B., dass eine angegebene Schnittstelle nicht existiert. Mit der Angabe dieser Option kann dies unterbunden werden.

Es gibt mehrere Möglichkeiten, die Schnittstellen für eine Datei festzulegen. Allgemein gilt aber, dass die erste erzeugte "Stück-Datei" alles vom Anfang der Originaldatei bis zum ersten Schnittpunkt enthält. Der Schnittpunkt selbst ist dabei nicht mehr in der "Stück-Datei" enthalten. Die Zeile des ersten Schnittpunktes wird dann die aktuelle Zeile. Die zweite erzeugte "Stück-Datei" enthält dann alles von dieser aktuellen Zeile bis ausschließlich dem nächsten Schnittpunkt usw.

Für die Argumente *schnittstelle(n)* kann folgendes angegeben werden:

/RA/	erzeugt eine "Stück-Datei", in die alles von der aktuellen Zeile bis ausschließlich der nächsten Zeile, die den angegebenen regulären Ausdruck *RA* enthält, kopiert wird. Hinter */RA/* kann auch *-n* oder *+n* (*n* muss eine ganze Zahl sein) angegeben werden; diese Angabe verschiebt dann den Schnittpunkt um *n* Zeilen vor bzw. nach die durch *RA* abgedeckte Zeile.
%RA%	wirkt wie */RA/* mit dem wichtigen Unterschied, dass der dadurch ausgewählte Bereich der Originaldatei nicht in eine "Stück-Datei" kopiert, sondern übersprungen wird.
zeilennr	erzeugt eine "Stück-Date", in die alles von der aktuellen Zeile bis zur Zeile mit der Zeilennummer *zeilennr* kopiert wird.
{zahl}	kann nach einer der drei zuvor angegebenen Formen angegeben werden und wiederholt dann diese *zahl* mal: • wird es nach */RA/* oder *%RA%* angegeben, so wird dieses Argument *zahl* mal angewendet; z.B. würde die Angabe **/PROCEDURE/{7}** bedeuten: Verwende die nächsten 7 Zeilen, in denen **PROCEDURE** vorkommt, als Schnittpunkte • wird es nach *zeilennr* angegeben, so wird die Datei alle *zeilennr* Zeilen (*zahl* mal) zerteilt, z.B. würde die Angabe **100{5}** bedeuten: Verwende die Zeilen mit den Nummern 100, 200, 300, 400 und 500 als Schnittpunkte.

- **csplit brief /Seite 2/ /Seite 3/**
  zerteilt die Datei `brief` in drei Teile:
  1. Der Text vom Anfang bis ausschließlich der ersten Zeile, die den String "`Seite 2`" enthält, wird in die "Stück-Datei" `xx00` kopiert.
  2. Dann wird der Text von dieser Zeile bis ausschließlich der nächsten Zeile, die "`Seite 3`" enthält, in die "Stück-Datei" `xx01` kopiert.
  3. Der Rest wird in die "Stück-Datei" `xx02` kopiert.
- **csplit – f teil lernen.ed 11 25 40**
  zerteilt die Datei `lernen.ed` in vier Teile:
  1. Der Text vom Anfang bis ausschließlich der 11. Zeile wird in die "Stück-Datei" `teil00` kopiert.
  2. Der Text von der 11. Zeile bis einschließlich der 24. Zeile wird in die "Stück-Datei" `teil01` kopiert.
  3. Der Text von der 25. Zeile bis einschließlich der 39. Zeile wird in die "Stück-Datei" `teil02` kopiert.
  4. Der Rest wird in die "Stück-Datei" `teil03` kopiert.
- **csplit -k brief /Seite/+3 {100}**
  zerteilt die Datei `brief`; als Schnittpunkte werden dabei immer die 3 .Zeilen nach jedem Vorkommen des Strings "`Seite`" verwendet. Die Option **-k** bewirkt, dass die "Stück-Dateien" auch dann angelegt werden, wenn in `brief` der String "`Seite`" weniger als 100 mal vorkommt
- **cat *.c | csplit -k –f funk – '/^}/+1' {100}**
  Mit dem **cat**-Kommando werden alle C-Programmdateien des Working-Directorys hintereinander in die Pipe geschrieben. **csplit** würde diesen zusammenhängenden Text dann aus der Pipe lesen und jede C-Funktion (hierbei wird angenommen, dass die C-Funktionen immer mit `}` am Anfang einer Zeile enden) herausschneiden und in die "Stück-Dateien" `funk00`, `funk01` usw. kopieren; dazu ist allerdings anzumerken, dass für die erste Funktion jeder C-Programmdatei nicht nur die Funktion allein, sondern alles vom Dateianfang bis zur schließenden geschweiften Klammer `}`, die diese Funktion beendet, kopiert würde.
- **csplit -k /home/egon/mbox "%^From %" "/^From /" {100}**
  zerteilt die Mailbox `mbox` im Home-Directory von `egon` in einzelne Briefe, die in die Dateien `brief00`, `brief01`,..., `brief99` geschrieben werden. Jeder Brief in dieser Mailbox beginnt immer mit "`From `". Da die erste From-Angabe zwischen Prozentzeichen (`%`) steht, wird keine eigene Stückdatei bis zur ersten "`From `"-Zeile erstellt. Würde diese erste Angabe fehlen, so würde als erste Stückdatei eine leere Datei erzeugt, da die erste "`From `"-Zeile bereits ganz am Anfang der Mailbox steht. Das Erzeugen von leeren Dateien kann man auch mit der Option **–z** unterbinden.

2.9 Drucken von Dateien

lp - Inhalt einer Datei am Drucker ausgeben

lp [*optionen*] [*datei(en)*]

Das Kommando **lp** veranlaßt die Ausgabe der angegebenen *datei(en)* am Drucker. Dazu reicht **lp** den entsprechenden Druckauftrag an den Druckerspooler weiter, welcher alle Druckaufträge entgegen nimmt und die einzelnen Druckaufträge koordiniert.

Sind keine *datei(en)* angegeben, liest **lp** den zu druckenden Text von der Standardeingabe. Es kann auch ein Minuszeichen (–) für eine *datei* angegeben werden, was ebenfalls für die Standardeingabe steht.

Bei jedem Aufruf von **lp** wird diesem Druckauftrag eine eindeutige Kennung (eine Art Auftragsnummer) zugeteilt, die unmittelbar nach der Abgabe des Kommandos **lp** am Bildschirm mitgeteilt (*request id is auftragsnr*) wird. Wenn an späterer Stelle ein solcher Druckauftrag annulliert werden soll, dann kann dies unter Angabe dieser *auftragsnr* mit dem Kommando **cancel** erreicht werden. Wurde die entsprechende *auftragsnr* in der Zwischenzeit vergessen, so kann sie mit dem Kommando **lpstat** wieder erfragt werden.

-n*n*	Es werden *n* Kopien ausgedruckt; normalerweise wird nur eine Kopie ausgegeben.
-c	Es werden temporäre Kopien der angegebenen Dateien erstellt und dann die Kopien am Drucker ausgegeben. Normalerweise wird von der zu druckenden Datei keine Kopie erstellt, sondern die wirkliche Datei gedruckt, was zur Folge hat, dass eventuelle Änderungen, die nach dem Druckauftrag an einer Datei vorgenommen werden, mit ausgedruckt würden. Die Option **-c** ist nützlich, wenn die entsprechende Datei anschließend editiert oder sogar gelöscht wird.
-w	Die Beendigung des abgegebenen Druckauftrags wird an dem Bildschirm gemeldet, an dem der **lp**-Auftrag abgegeben wurde. Hat der Auftraggeber sich zwischenzeitlich vom System abgemeldet, so wird ihm E-Mail geschickt.
-d*drucker*	erlaubt die explizite Angabe eines Druckers oder einer Klasse von Druckern, auf den(en) die angebenen *datei(en)* auszudrucken sind.
-m	Die Beendigung des abgegebenen Druckauftrags wird dem Auftraggeber per E-Mail gemeldet.
-o *optionen*	ermöglicht es, drucker- bzw. druckerklassenspezifische Optionen anzugeben.

Die druckerspezifischen Optionen, die man mit **-o** *optionen* angeben kann, werden vom Systemadministrator definiert. Bei System V.4 sollten mindestens die folgenden Optionen vorhanden sein:

nobanner	unterdrückt den Ausdruck einer Titelseite
length=*n*	legt die Seitenlänge fest. Ohne Suffix hinter *n* wird die Anzahl der Zeilen pro Seite, mit Suffix **i** wird die Seitenlänge in Zoll (*inches*) und mit Suffix **c** in Zentimetern festgelegt.
width=*n*	legt Zeilenlänge fest. Suffix-Angabe wie bei *length* möglich
lpi=*n*	Anzahl der Zeilen pro Zoll (ohne Suffix oder mit **i**) bzw. pro Zentimeter (Suffix **c**)
cpi=*n*	Zeichenbreite pro Zoll (ohne Suffix oder mit **i**) bzw. pro Zentimeter (Suffix **c**)

Wenn man mit **-o** mehrere druckerspezifische Optionen angeben möchte, dann muss man diese in Anführungszeichen angeben, wie z.B. **lp -o "nobanner lpi=8 cpi=12" laender**.

lpstat - Abfragen von Statusinformation zu Druckaufträgen

lpstat [*optionen*] [*druckauftragsnr(n)*]

Das Kommando **lpstat** gibt Statusinformationen zu abgegebenen Druckaufträgen aus. Wird dieses Kommando ohne eine Angabe von *druckauftragsnr(n)* aufgerufen, dann wird der Status aller Druckaufträge angezeigt, welche sich in der Warteschlange befinden, ansonsten wird nur Information über die Druckaufträge mit den angegebenen *druckauftragsnr(n)* ausgegeben.

-t	gibt die gesamte verfügbare Statusinformation aus.

Unter Linux muss statt **lpstat** das Kommando **lpc** verwendet werden.

cancel - Abbrechen von Druckaufträgen, die mit dem lp-Kommando gegeben wurden

cancel [*kennung(en)*] [*druckername(n)*]
cancel -u *login-name* [*druckername(n)*]

cancel storniert abgegebene Druckaufträge. Welche Druckaufträge zu stornieren sind, kann **cancel** entweder über die von **lp** ausgegebenen Auftrags-*kennung(en)* oder über die Angabe der entsprechenden *druckername(n)* mitgeteilt werden. Wird **cancel** mit der Angabe von *druckername(n)* aufgerufen, so wird der gerade am Drucker ausgegebene Auftrag beendet. Mit **-u** *login-name* lassen sich alle Druckaufträge des Benutzers *login-name* abbrechen. Sind noch *druckername(n)* angegeben, so werden nur dort die Druckaufträge storniert, andernfalls werden die Druckaufträge an allen Druckern storniert. Die Kennungen und der Status von abgegebenen Druckaufträgen können mit dem Kommando **lpstat** erfragt werden. Unter Linux muss statt **cancel** das Kommando **lprm** verwendet werden.

lpr - Ausgeben von Dateien am Drucker (unter Linux)

lpr [*optionen*] *datei(en)*

Das Kommando **lpr** veranlaßt dieAusgabe der angegebenen *datei(en)* am Drucker.

-#*n*	die angebenen *datei(en)* *n*-mal am Drucker ausgeben.
-P*druckerid*	Ausgabe erfolgt am Drucker mit der Kennung *druckerid*.
-h	keine Ausgabe einer Kopfzeile.
-m	die Beendigung des Druckauftrags wird per Mail (elektronische Post) gemeldet.
-T *titel*	Bei der Ausgabe wird in der Kopfzeile der *titel* anstelle des Dateinamens ausgegeben.

lpc - Verwalten von Druckern (unter Linux)

lpc [*optionen*] ...

Das Verwalten der Drucker ist unter Linux mit dem Kommando **lpc** möglich. Danach erwartet **lpc** die Eingabe von speziellen **lpc**-Kommandos. Welche **lpc**-Kommandos möglich sind, kann man hierbei mit **h** oder **?** erfragen.

lprm - Löschen von Druckaufträgen (unter Linux)

lprm [*optionen*] [*benutzerkennung(en)*]

Um in der Druckerwarteschlange befindliche Aufträge zu löschen, steht unter Linux das Kommando **lprm** zur Verfügung. Wird **lprm** alleine ohne weitere Angaben aufgerufen, löscht es den momentan aktiven Druckauftrag des Benutzers, der **lprm** aufrief. Für *benutzerkennung* kann entweder ein *loginname* oder eine Auftragsnummer angegeben werden. Ist ein *loginname* angegeben, löscht **lprm** die Druckaufträge, die der Benutzer *loginname* abgesetzt hat, wenn der Aufrufer die entsprechenden Rechte dazu besitzt. Druckaufträge von anderen Benutzern kann nur der Superuser löschen. Um nur einen bestimmten Druckauftrag zu löschen, muss eine Auftragsnummer angegeben werden. Die Auftragsnummern der einzelnen Druckaufträge kann man mit **lpq –l** erfragen.

-P*druckerid*	Löschen von Aufträgen in der Warteschlange des Druckers mit der Kennung *druckerid*.
-	Alle Druckaufträge des aufrufenden Benutzers löschen. Im Falle des Superusers werden alle Druckaufträge in der entsprechenden Warteschlange gelöscht.

lpq - Anzeigen der aktuellen Druckerwarteschlange (unter Linux)

lpq [*optionen*] [*benutzerkennung(en)*]

Um sich unter Linux die Druckerwarteschlange anzeigen zu lassen, steht das Kommando **lpq** zur Verfügung. Wird **lpq** ohne Optionen aufgerufen, gibt es die Druckerwarteschlange des eingestellten Standarddruckers aus.

-P*druckerid*	Warteschlange des Druckers mit der Kennung *druckerid* ausgeben.
-a	Warteschlange zu allen in der Datei `/etc/printcap` angegebenen Druckern ausgeben.
-l	Warteschlange im Langformat (Auftragsnummer, Namen und Größen der Dateien in der Druckerwarteschlange) ausgeben.
-t*n*	Warteschlange automatisch alle *n* Sekunden ausgeben.

2.10 Konvertieren von Dateien

recode – Konvertieren von Zeichensätzen

recode [*optionen*] *von*..*nach* [*datei*]

recode führt eine Konvertierung von Zeichensatz *von* nach Zeichensatz *nach* durch. **recode** kennt eine große Anzahl von Zeichensätzen, die man sich mit **recode -l** anzeigen lassen kann.

- **recode ibmpc..latin1 < dosdatei > linuxdatei**
  konvertiert den Inhalt der Datei `dosdatei`, indem es alle Zeilenenden und deutsche Sonderzeichen in den unter Linux geltenden Zeichensatz *Iso-Latin1* umformt. Das Ergebnis wird in die Datei `linuxdatei` geschrieben.
- **recode latin1..ibmpc < linuxdatei > dosdatei**
  konvertiert den Inhalt der Datei `linuxdatei`, indem es alle Zeilenenden und deutsche Sonderzeichen in den unter MS-DOS geltenden Zeichensatz umformt. Das Ergebnis wird in die Datei `dosdatei` geschrieben.
- **recode latin1/cr-lf..latin1 < windatei > unixdatei**
  konvertiert den Inhalt der Datei `windatei`, indem alle Zeilenenden (*Carriage Return* und *Line Feed*) durch das unter Unix übliche Zeilenende (nur *Line Feed*) ersetzt werden. Der eigentliche Zeichensatz wird unverändert in die Datei `unixdatei` übernommen.

iconv – Konvertieren von Zeichensätzen

iconv -f *von* **-t** *nach* [*datei*]

Mit dem in System V.4 vorhandenen Kommando **iconv** können Zeichensätze konvertiert werden. Mit den Optionen **-f** *von* und **-t** *nach* läßt sich die Konvertierung festlegen. Ist eine *datei* angegeben, so liest **iconv** den Inhalt dieser *datei* und schreibt die konvertierten Daten auf die Standardausgabe. Ist keine *datei* angegeben, so liest **iconv** von der Standardeingabe. Lassen sich bestimmte Sonderzeichen nicht konvertieren, weil sie im gewünschten Ziel-Zeichensatz nicht vorhanden sind, werden sie durch Unterstrich (_) dargestellt.

Für *von* (bei **-f** *von*) und *nach* (bei **-t** *nach*) lassen sich u.a. folgende Angaben machen:

Quell-Zeichensatz	*von*	Ziel-Zeichensatz	*nach*	Quell-Zeichensatz	*von*	Quell-Zeichensatz	*von*
ISO 646	**646**	ISO 8859-1	**8859**	ISO 8859-1	**8859**	ISO 646	**646**
ISO 646de	**646de**	ISO 8859-1	**8859**	ISO 8859-1	**8859**	ISO 646de	**646de**
ISO 646en	**646en**	ISO 8859-1	**8859**	ISO 8859-1	**8859**	ISO 646en	**646en**
ISO 646fr	**646fr**	ISO 8859-1	**8859**	ISO 8859-1	**8859**	ISO 646fr	**646fr**

Die Anhängsel sind dabei immer eine Abkürzung für den Zeichensatz des betreffenden Landes: `de` (Deutschland), `da` (Dänemark), `en` (England), `es` (Spanien), `fr` (Frankreich), `it` (Italien) und `sv` (Schweden). Im deutschsprachigen Raum werden meist die Zeichensätze ISO 626 (US-ASCII, 7 Bit), ISO 646de (deutsche ASCII-Variante, 7 Bit) und ISO 8859-1 (8 Bit, nationalen Zeichen aller europäischen Länder in der oberen Hälfte des 8-Bit-Zeichensatzes) verwendet.

iconv -f 8859 -t 646de datei >zieldatei ISO 8859-1 nach ISO 646de

iconv -f 8859 -t 646 datei >zieldatei ISO 8859-1 nach ISO 646 (US-ASCII)

iconv -f 646de -t 8859 datei >zieldatei ISO 646de nach ISO 8859-1

2.11 Geräte- und FIFO-Dateien

mknod - Anlegen von Gerätedateien bzw. Named Pipes

mknod *name* **b** *major-nr minor-nr* (*blockorientierte Gerätedatei*)
mknod *name* **c** *major-nr minor-nr* (*zeichenorientierte Gerätedatei*)
mknod *name* **p** (*Named Pipe*)

Zum Anlegen einer Gerätedatei steht das Kommando **mknod** zur Verfügung, das allerdings nur ein privilegierter Benutzer wie der Superuser oder der Systemadministrator aufrufen darf.

Bei den ersten beiden Aufrufformen legt die erste Zahl (*major*-nr) den Gerätetyp fest, während die zweite Zahl (*minor-nr*) immer dem Gerätetreiber übergeben wird, der sie nach Belieben interpretieren kann, z.B. zur Unterscheidung von verschiedenen Geräten des gleichen Typs. *major-nr* und *minor-nr* dürfen dabei als Dezimal- oder Oktalzahl (muss mit `0` beginnen) angegeben werden und sind systemspezifisch. Hinsichtlich der Zugriffsrechte gelten bei Gerätedateien die gleichen Regeln wie bei normalen Dateien mit der Ausnahme, dass das Ausführrecht keine Bedeutung hat. Gerätedateien werden üblicherweise im Directory `/dev` angelegt. Neben Gerätedateien enthält das Directory `/dev` auch noch andere Spezialdateien, die besondere Zwecke erfüllen:

/dev/tty	ist immer das aktuelle Terminal, an dem man gerade arbeitet.
/dev/null	ist eine Art Mülleimer. Alle Daten, die nach `/dev/null` kopiert werden, werden einfach weggeworfen. Wenn Programme aus dieser Datei lesen, erhalten sie sofort das Dateiende-Zeichen (EOF).
/dev/zero	ist eine unerschöpfliche Quelle von 0-Bytes, die manchmal verwendet wird, um Dateien bis zu einer bestimmten Größe mit Nullen aufzufüllen.

Die folgende Tabelle zeigt einige Gerätedateien in System V.4:

/dev/cdrom	CD-ROM-Laufwerk	**/dev/dsk/f03ht**	3 ½" Floppy-Disk (1,4 MB)
/dev/console	Systemkonsole	**/dev/lp**	Parallelschnittstelle für Drucker
/dev/cram	RAM-Disk (montierbares Dateisystem)	**/dev/term/00**	Terminal-Schnittstelle 1
/dev/dsk/c0t0d0s0	erste Festplatte	**/dev/term/01**	Terminal-Schnittstelle 2
/dev/dsk/c0t1d0s0	zweite Festplatte		

Die folgende Tabelle zeigt einige Gerätedateien unter Linux:

/dev/fd0 bzw. **/dev/fd1**	erstes bzw, zweites Floppylaufwerk
/dev/hda	erstes IDE-Laufwerk (Festplatten,CD, DVD usw.) **/dev/hda1** bis **/dev/hda15** (Partitionen)
/dev/sda	erste SCSI-Festplatte**/dev/sda1** bis **/dev/sda15** (Partitionen)
/dev/sdb, **/dev/sdc**	zweite bzw. dritte SCSI-Festplatte
/dev/tty, **/dev/console**	aktueller Terminal (Konsole)
/dev/tty1 bis **/dev/tty8**	virtuelle Terminals (Konsolen)
/dev/cdrom	Link auf entsprechendes CD-ROM-Laufwerk
/dev/mouse	Link auf die von Maus verwendete Schnittstelle
/dev/modem	Link auf com-Port, an dem Maus angschlossen
/dev/ttyS0 bis **/dev/ttyS3**	serielle Schnittstellen (COM1 bis COM4)
/dev/lp0 bis **/dev/lp2**	parallele Schnittstellen (LPT1 bis LPT3)

mkfifo - Anlegen von FIFO-Dateien

mkfifo *datei*

mkfifo richtet eine FIFO-Datei (*first-in-first-out*) ein. FIFO-Dateien ermöglichen wie Pipes Datenaustausch zw. zwei Programmen. Der Aufruf **ls –l | less** könnte mittels einer FIFO-Datei auch wie folgt nachgebildet werden:

```
$ mkfifo fifodatei (Enter)
$ ls –l > fifodatei & (Enter)
$ less < fifodatei (Enter)
```

3 Directorykommandos

pwd – Ausgeben des Working-Directorys

Das Kommando **pwd** gibt den Namen des Working-Directorys aus.

cd – Ausgeben des Working-Directorys

cd [*directory*]

Mit dem Kommando **cd** kann das Working-Directory gewechselt werden, indem das neue gewünschte Working-Directory entweder als absoluter oder als relativer Pfadname angegeben wird. Wird **cd** ohne Angabe eines *directory* aufgerufen, so wird zum Home-Directory gewechselt. Es ist einem Benutzer nur dann möglich, in das angegebene *directory* zu wechseln, wenn er für dieses Directory *execute*-Rechte besitzt. In das vorherige Working-Directory kann man mit **cd -** zurückwechseln.

mkdir – Anlegen von Directories

mkdir [*optionen*] *directory-name(n)*

Mit dem Kommando **mkdir** werden die als Argumente angegebenen Directories *directory-name(n)* neu eingerichtet. Um ein neues Directory einrichten zu können, muss man Schreibrechte im Parent-Directory besitzen. Beim Anlegen eines neuen Directory werden immer automatisch die zwei Subdirectories **.** und **..** dort eingerichtet.

-m *absolut-modus*	Setzt die Zugriffsrechte der neu angelegten Directories auf *absolut-modus* (siehe Kommando **chmod**)
-p	legt im *directory-name(n)* erwähnte, aber nicht vorhandene Zwischen-Directories (Parent-Directories) an.

rmdir – Leere Directories löschen

rmdir [*optionen*] *directory-name(n)*

-p	löscht die angegebenen Directories und deren Parent-Directories, wenn diese durch das Löschen leer geworden sind.

rmdir löscht die als Argumente angegebenen Directories aber nur, wenn diese leer sind. Wenn eines der angegebenen Directories noch Dateien oder Subdirectories (**.** und **..** ausgenommen) enthält, dann wird dies gemeldet und das entsprechende Directory wird nicht gelöscht. Möchte man aber ein solches Directory und damit auch die darin enthaltenen Dateien und Subdirectories auf jeden Fall löschen, so ist dies mit folgendem Kommandoaufruf möglich:
rm -r *directory-name(n)*

dircmp – Vergleichen von zwei Directories

dircmp [*optionen*] *dir1 dir2*

Das Kommando **dircmp** vergleicht den Inhalt der beiden Directories *dir1* und *dir2*. Dazu vergleicht es zuerst die Dateinamen der beiden Directories, bevor es die Inhalte von Dateien mit gleichen Namen vergleicht.

dircmp gibt Dateinamen, die nur in einem Directory vorkommen, auf der ersten Ausgabeseite aus. Dateinamen, die in beiden Directories vorkommen, aber unterschiedliche Inhalte haben, werden auf der zweiten Seite angezeigt. Auf der dritten Seite werden gleiche Dateinamen mit gleichem Inhalt ausgegeben.

dircmp gibt immer die Dateinamen aus, die nur in einem Directory vorkommen; die Ausgabe von identischen Dateien kann unterdrückt werden.

-d	gibt ein ed-Skript aus, das aus gleichnamigen Dateien identische Dateien kreiert (siehe auch **diff**).
-s	unterdrückt die Ausgabe der Namen von identischen Dateien
-w*n*	ändert die Länge der Ausgabezeilen von 72 Zeichen auf *n* Zeichen.

4 Benutzer- und Gruppenkommandos

logout / exit – Beenden einer Sitzung

logout bzw. **exit**

logout und **exit** beenden die Shell-Sitzung (in Textkonsole bzw. in Shell-Fenster). Auch mit (Strg)-D möglich.

passwd - Ändern bzw. Vergeben eines Passworts

passwd [*optionen*] [*login-name*]

Das Kommando **passwd** ermöglicht es einem Benutzer, ein Passwort festzulegen oder aber ein bereits vorhandenes Passwort zu ändern. Dabei wird er zunächst nach dem alten Passwort gefragt. Daraufhin muss er das neue Passwort eingegeben, und zwar zweimal. Alle drei Eingaben (altes Passwort und die beiden Eingaben des neuen Passworts) werden nicht am Bildschirm angezeigt. Nur der Superuser (*root*) kann Passwörter anderer Benutzer ändern, indem er **passwd** *login-name* aufruft. Dazu braucht er das alte Passwort nicht zu kennen.

id - Erfragen der eigenen Benutzer-(User-ID) und Gruppenkennung (Group-ID)

id [**-a**]

id gibt die User-ID und Group-ID mit den entsprechenden Login-Namen in Klammern dahinter aus. Falls für ein Programm, das **id** aufruft, das set-user-id- bzw. set-group-id-Bit gesetzt ist, so meldet **id** sowohl die reale wie auch die effektive User-ID bzw. Group-ID, wenn diese verschieden sind. Ist Option **-a** angegeben, gibt **id** alle Gruppen aus, zu denen der aufrufende Benutzer gehört. Bei öfterem Aufruf des Kommandos **su**, das den Wechsel einer Benutzerkennung – ohne Abmelden – am gleichen Terminal zuläßt, kann es vorkommen, dass ein Benutzer den Überblick verliert, unter welcher Kennung er momentan arbeitet. Hier kann er sich mit Kommando **id** weiterhelfen. **id** wird öfters aus Programmen heraus aufgerufen, um mitzuprotokollieren, wer das Programm benutzt.

logname - Erfragen des eigenen Login-Namens

logname

Das Kommando **logname** gibt den Login-Namen des Benutzers aus, der sich am entsprechenden Terminal angemeldet hat. **logname** wird oft wie **id** auch aus Programmen heraus aufgerufen, um mit zu protokollieren, wer dieses Programm benutzt. Während das Kommando **id** immer die aktuelle Benutzerkennung ausgibt, liefert **logname** immer den Login-Namen, unter dem sich der entsprechende Benutzer angemeldet hat, selbst wenn er in der gleichen Unix-Sitzung mit **su** zu einer anderen Benutzerkennung wechselte.

groups – Ausgeben der Gruppenzugehörigkeiten eines Benutzers

groups [*benutzername(n)*]

groups gibt alle Gruppen auf der Standardausgabe aus, zu denen der oder die Benutzer *benutzername(n)* gehören. Jeder Benutzer gehört zu einer voreingestellten Gruppe, die in `/etc/passwd` angegeben ist. Falls ein Benutzer zu mehreren Gruppen gehört, so sind diese weiteren Zugehörigkeiten in der Datei `/etc/group` angegeben.

newgrp - Kurzzeitiges Wechseln der Gruppenzugehörigkeit

newgrp [**-**] [*gruppenname*]

Für *gruppenname* ist der Name (nicht die Group-ID) anzugeben, wie er in der Datei `/etc/group` aufgeführt ist. Jeder Benutzer ist vom Systemverwalter einer bestimmten Gruppe zugeteilt. Die Datei `/etc/group` enthält eine Liste aller Gruppennamen, die Group-ID und die Mitglieder jeder Gruppe. Dabei ist es möglich, dass ein Benutzer mehr als einer Gruppe angehört. Die Datei `/etc/passwd` legt dann fest, welcher Gruppe ein solcher Benutzer beim Anmelden zugeordnet wird. Mit **newgrp** ist es nun einem Benutzer möglich während einer Sitzung in eine andere Gruppe, in der er ebenfalls Mitglied ist, überzuwechseln. Wird **newgrp** ohne die Angabe eines Arguments aufgerufen, so wechselt der entsprechende Benutzer in die Gruppe, der in der Datei /`etc/passwd` zugeordnet ist.

-	alle Systemvariablen behalten die Werte, die ihnen beim Anmelden zugewiesen werden.

who - Ausgeben der momentan angemeldeten Benutzer

who [*optionen*] [*datei*]
who am i
who am I

Das Kommando **who** gibt per Voreinstellung alle Login-Namen, Terminalnamen und Anmeldezeiten zu allen Benutzern aus, die momentan am System angemeldet sind. Über die Angabe von Optionen ist es möglich, sich andere Information ausgeben zu lassen. Normalerweise besorgt sich das Kommando **who** seine Informationen aus der Datei `/var/adm/utmp`; wird dagegen beim Aufruf von **who** eine *datei* angegeben, so liest es seine Informationen aus dieser *datei*.

Bei den Aufrufen **who am i** und **who am I** wird der Login-Name des Aufrufers ausgegeben.

Das Format der **who**-Ausgabe hängt von den angegebenen Optionen ab. Dabei können verschiedene Informationen ausgegeben werden:

name [zustand] leitung zeit [aktivität] [kommentar]

Die einzelnen Felder bedeuten dabei:

name	Login-Name
zustand	zeigt an, ob "fremde" Benutzer auf diesen Terminal (z.B. mit **write**) schreiben dürfen. Dabei bedeutet: `+` (Schreiberlaubnis), `-` (keine Schreiberlaubnis) und `?` (gestörte Verbindung)
leitung	Name (wie in `/dev`) des benutzten Terminals bzw. der benutzten Leitung.
zeit	Anmeldezeit
aktivität	Zeit, die seit der letzten Aktivität am entsprechenden Terminal bzw. Leitung vergangen sind. Dabei bedeutet: `.` (war in der letzten Minute aktiv), `old` (seit über 24 Stunden oder seit letztem Systemstart nicht mehr benutzt) und *std*`:`*min* (sonst).
kommentar	Inhalt des Kommentar-Feldes in der Datei `/etc/inittab`. Ein solcher Kommentar kann z.B. Information geben, wo sich das entsprechende Terminal befindet.

Wenn nicht anders angegeben, so zeigen die nachfolgend angegebenen Optionen die Informations-Felder *name*, *leitung*, *zeit*, *aktivität* und *kommentar* an.

-u	(*used*) zusätzlich das Feld *aktivität* ausgeben.
-T	(*Terminal state*) zusätzlich noch das Feld *zustand* ausgeben. Wenn dies die einzige Option ist, dann werden (bis auf *zustand*) dieselben Felder wie bei der Option **-s** ausgegeben.
-l	(*lines*) nur die Leitungen auflisten, bei denen das System auf ein Anmelden wartet. Dabei wird LOGIN im Feld *name* ausgegeben, und das Feld *zustand* wird nicht angezeigt.
-H	(*Header*) zu jeder Spalte Überschriften ausgeben.
-m	nur Informationen zur aktuellen Konsole ausgeben.
-s	nur die Felder *name*, *leitung* und *zeit* ausgeben.
-q	(*quick*) nur die Namen und die Zahl der momentan angemeldeten Benutzer ausgeben.

Werden keine Optionen angegeben, so entspricht dies der Angabe der Option **-s**.

finger - Abfragen von Informationen zu anderen Benutzern

finger [*optionen*] *benutzername(n)*
finger [**-l**] *benutzername@hostname* ....

Wird **finger** alleine ohne Argumente aufgerufen, so gibt es zu allen momentan angemeldeten Benutzern eine Informationszeile aus, die folgendes enthält:

Login Name TTY Idle When Where

Login ist dabei der Login-Name, *Name* der wirkliche Name, *TTY* der Terminalname (`*` vorangestellt, wenn schreibgeschützt), *Idle* die Zeit in Minuten, seit der Benutzer nichts mehr eingegeben hat, *When* enthält die Zeit der Anmeldung und *Where* zeigt eventuell den Namen des Systems an, von dem aus sich der Benutzer am lokalen System angemeldet hat.

Wenn *benutzername(n)* angegeben sind, wird zu diesen Benutzern eine detailliertere Information ausgegeben. Für *benutzername* kann dabei der Login-Name, der Vor- oder Zuname des entsprechenden Benutzers angegeben werden. Wird **finger** für einen Benutzer auf einem entfernten System aufgerufen, so muss *benutzername@hostname* angegeben werden.

Bei der Ausgabe von detaillierter Information im Langformat wir zu jedem Benutzer zusätzlich zu obiger Information noch folgendes ausgegeben:

- Home Directory und die Login-Shell
- Zeit, die Benutzer schon angemeldet ist, oder wenn er nicht angemeldet ist, wann er das letzte Mal angemeldet war.
- Zeit, wann der Benutzer das letzte Mal Mail empfangen und gelesen hat.
- Inhalt der Datei `.plan` (im Home-Directory des betreffenden Benutzers); `.plan` enthält meist irgendwelche lustigen Sätze, die das sich selbst gesteckte Ziel des Benutzers beschreiben.
- Inhalt der Datei `.project` (im Home-Directory des betreffenden Benutzers); `.project` enthält eine kurze Beschreibung des Projekts, an dem der Benutzer gerade arbeitet.

-l	Langformat-Ausgabe.
-m	*benutzername(n)* nur auf Login-Namen (nicht auf Vor- und Zunamen) anwenden.
-p	Keine Ausgabe von `.plan` bei Langformat-Ausgabe.
-q	(*quick*) nur Login-Name, Terminal, und Login-Zeit ausgeben.
-s	(*short*) Kurzformat-Ausgabe.

last - An- und Abmeldezeiten von Benutzern erfragen

last [**-**[**n**] *zahl*] [**-f** *datei*] [*name(n)*]

In der Datei `/var/adm/wtmpx` oder `/var/log/wtmp` werden alle An- und Abmeldezeiten der einzelnen Benutzer und Terminals gespeichert. **last** liest aus dieser Datei die benötigten Informationen und gibt diese aus.

Für *name* kann entweder ein Benutzer oder ein Terminal (wie z.B. `term/tty08`) angegeben sein. **last** gibt dann die dazugehörige Information aus. Wird z.B. **last egon console** aufgerufen, so werden alle An- und Abmeldezeiten des Benutzers `egon` ebenso ausgegeben wie die Zeiten, in denen an der Konsole gearbeitet wurde.

last gibt immer zuerst die letzte Sitzung, dann die vorletzte Sitzung usw. aus. Die Information zu einer Sitzung umfaßt dabei den Anmeldezeitpunkt, die Dauer der Sitzung und den Namen des Terminals, an dem die Sitzung stattfand. Wird **last** ohne jegliche Argumente aufgerufen, so gibt es alle in der Datei `/var/adm/wtmpx` vermerkten An- und Abmeldezeiten in umgekehrter Reihenfolge aus.

-[**n**]*zahl*	nur *zahl* An- und Abmeldezeiten ausgeben
-f *datei*	An- und Abmeldezeiten werden aus der Datei *datei* und nicht aus `/var/adm/wtmpx` gelesen.

useradd / adduser – Anlegen eines neuen Benutzers

useradd *name*

Das Kommando **useradd** legt einen neuen Benutzer mit dem Namen *name* an. Es kann nur von `root` aufgerufen werden. Auf manchen Linux-Distributionen steht statt **useradd** das Kommando **adduser** zur Verfügung.

-g *gruppe*	legt Hauptgruppe des Benutzers fest.
-G *grup1,grup2,grup2*	legt zusätzliche Gruppen des Benutzers fest.
-m	wenn das Home-Directory `/home/`*name* nicht existiert, wird es angelegt, wobei alle Dateien aus `/etc/skel` dorthin kopiert werden.
-u *n*	legt die User-ID *n* für den Benutzer fest.

userdel – Löschen eines Benutzers

userdel [*optionen*] *name*

Mit dem Kommando **userdel** kann `root` den Benutzer *name* löschen.

-r	löscht auch das gesamte Home-Directory des Benutzers sowie seine Mail-Inbox.

usermod – Ändern von Eigenschaften eines Benutzers

usermod [*optionen*] *name*

Das Kommando **usermod** ermöglicht die Änderung verschiedener Einstellungen (Home-Directory, Gruppenzugehörigkeit, voreingestellte Shell usw.) für den Benutzer *name*. Es kann nur von `root` aufgerufen werden. Die meisten Optionen sind dabei identisch zum Kommando **useradd**.

-L	blockiert diesen Benutzer vorübergehend.
-U	hebt eine mit **–L** vorgenommene Blockierung wieder auf.

groupadd– Anlegen einer neuen Gruppe

groupadd *name*

Mit dem Kommando **groupadd** kann `root` eine neue Gruppe mit dem Namen *name* anlegen.

-g *n*	legt die Group-ID *n* für die Gruppe fest.

groupdel – Löschen einer Gruppe

groupdel *name*

Mit dem Kommando **groupdel** kann `root` die Gruppe *name* löschen.

groupmod – Ändern des Namens und/oder der group-Id einer Gruppe

groupmod [*optionen*] *name*

Mit dem Kommando **groupmod** kann `root` den Namen und die Group-ID der Gruppe *name* ändern.

-g *n*	*n* legt die neue Group-ID fest.
-n *name*	*name* legt den neuen Gruppennamen fest.

5 Bildschirm- und Terminalkommandos

stty – Setzen und Abfragen von Terminaleinstellungen

stty [*optionen*] [*terminal-flag(s)*]

Wird **stty** ohne Argumente aufgerufen, so gibt es eine bestimmte Gruppe von Terminaleinstellungen aus. Werden *terminal-flag(s)* angegeben, so werden die damit spezifizierten Einstellungen für das entsprechende Terminal vorgenommen.

-a	alle Terminal-Einstellungen ausgeben
-g	gibt eine Liste von Terminal-Einstellungen in einer solchen Form aus, dass diese Liste bei einem späteren **stty**-Aufruf wieder verwendet werden kann. Wird eine solche Liste mit Ausgabeumlenkung in einer Datei gesichert, so kann die momentane Terminal-Einstellung später wiederhergestellt werden, wenn diese Datei beim **stty**-Aufruf (unter Verwendung von Kommandosubstitution) angegeben wird.

```
$ stty –a (Enter)
speed 38400 baud; rows 51; columns 133; line = 0;
intr = ^C; quit = ^\; erase = ^?; kill = ^U; eof = ^D;
eol = <undefiniert>;
eol2 = <undefiniert>; start = ^Q; stop = ^S; susp = ^Z; rprnt = ^R;
werase = ^W; lnext = ^V; flush = ^O; min = 1; time = 0;
-parenb -parodd cs8 -hupcl -cstopb cread -clocal –crtscts
-ignbrk -brkint -ignpar -parmrk -inpck -istrip -inlcr –igncr
icrnl ixon –ixoff
-iuclc -ixany –imaxbel
opost -olcuc -ocrnl onlcr -onocr -onlret -ofill -ofdel nl0 cr0
tab0 bs0 vt0 ff0
isig icanon iexten echo echoe echok -echonl -noflsh –xcase
-tostop –echoprt echoctl echoke
$ stty –g (Enter)
500:5:bf:8a3b:3:1c:7f:15:4:0:1:0:11:13:1a:0:12:f:17:16:0:0:2f:0:0:0:0:0:0:0:0:0:0:0
$ stty cols 50 (Enter) [legt z.B. unter Linux fest, dass]
                       [Terminal nur noch über 50 Spalten verfügen soll]
$
```

Um eine Grundeinstellung für ein Terminal wiederherzustellen, muss man unter Linux nur **reset** aufrufen. Alle bei **stty** möglichen Terminalflags kann man sich z.B. unter Linux mit dem folgenden Aufruf einblenden lassen: **stty --help | less**.

Folgender Ausschnitt aus einem Shell-Skript zeigt z.B. wie man die Echo-Funktion für die Eingabe eines Passworts ausschalten kann und dann anschließend die zuvor gesicherten Terminaleinstellungen wiederherstellen kann:

```
.....
echo -n "Passwort: "
stty -g > /tmp/altterm   # Sichern der Terminaleinstellungen
                         # in temporärer Datei
stty -echo               # echo-Funktion ausschalten
read passwort            # Passwort einlesen
stty $(cat /tmp/altterm) # Ursprgl. Terminaleinstellungen wiederherstellen
rm -f /tmp/altterm       # Temporäre Datei wieder löschen
.....
```

setterm - Verändern der Terminaleinstellungen

setterm [*optionen*]

Mit dem Kommando **setterm** können bestimmte Einstellungen des Bildschirms wie etwa Hintergrund- und Vordergrundfarbe verändert werden. Wird **setterm** alleine ohne weitere Angaben aufgerufen, gibt es eine Liste aller möglichen Optionen aus. Um sich z.B. bei jedem Anmelden einen Bildschirm mit blauer Hintergrundfarbe und weißer Schrift einzustellen, empfiehlt es sich, die folgende Zeile in der Datei `~/.profile` einzutragen.

```
setterm -background blue -foreground white –store
```

Weitere wichtige **setterm**-Aufrufe, die entweder interaktiv eingegeben oder aber in der Datei `~/.profile` eingetragen werden können, sind:

- **setterm -blank** *n*: aktiviert nach *n* Minuten den Bildschirmschoner (Bildschirm wird schwarz) im Textmodus..
- **setterm -clear**: löscht den Bildschirm.
- **setterm -inversescreen on** oder **setterm -inversescreen off**: **on** invertiert den Bildschirm (Text in aktueller Hintergrundfarbe und Hintergrund in aktueller Vordergrundfarbe) und **off** stellt den Normalzustand wieder her. Im übrigen ist der Aufruf **setterm -inv** identisch zum Aufruf **setterm -inversescreen on**.

reset - Wiederherstellen eines Zeichensatzes für ein Terminal

reset

reset stellt die Zeichensatzzuordnung wieder her, wenn diese durch die Ausgabe von Sonderzeichen am Bildschirm zerstört wurde. **reset** sollte immer dann aufgerufen werden, wenn am Bildschirm plötzlich nur noch seltsame und unlesbare Zeichen angezeigt werden.

clear - Bildschirm löschen

clear

löscht den ganzen momentanen Bildschirminhalt. In System V.3 mußte dazu noch **tput clear** aufgerufen werden.

mesg – Sperren bzw. Freigeben des Terminals für fremde Benutzer

mesg [**-n**] [**-y**]

Unter Linux muss der Bindestrich vor **y** bzw. **n** weggelassen werden. Das Kommando **mesg** ermöglicht es einem Benutzer, seinen Terminal für das Schreiben durch andere Benutzer (z.B. mit dem Befehl **write**) zu sperren oder aber ein solches "fremdes" Schreiben wieder zu zulassen. Wird **mesg** ohne Angabe einer Option aufgerufen, so meldet es lediglich das momentane Zugriffsrecht dieses Terminals für "fremde" Benutzer.

-n	Schreiben durch "fremde" Benutzer (außer Superuser) am eigenen Terminal verbieten.
-y	Schreiben durch "fremde" Benutzer wieder zulassen.

tty - Erfragen des Terminalnamens

tty [*optionen*]

In Unix werden Geräte wie Dateien behandelt. Zu jedem Gerät existiert eine Gerätedatei. So gibt es z.B. zu jedes angeschlossene Terminal im Directory `/dev` eine Gerätedatei. Das Kommando **tty** gibt nun den Pfadnamen der Gerätedatei des Terminals aus, an dem **tty** aufgerufen wird.

-l	(*list*) Leitungsnummer des Terminals ausgeben; nur möglich für synchrone Leitungen.
-s	(*silent*) unterdrückt die Ausgabe des Pfadnamens der Terminal-Gerätedatei. In diesem Fall wird von **tty** nur ein Exit-Statuswert geliefert. Mögliche Exit-Statuswerte sind: 0 bzw. 1: Standardeingabe ist bzw. ist nicht das Terminal 2: es wurden ungültige Optionen angegeben 3: Ein Schreibfehler ist aufgetreten

tty wird oft aus Programmen heraus aufgerufen, um festzustellen, ob die Standardeingabe momentan auf das Terminal eingestellt ist.

6 Programm- und Prozeßverwaltung

6.1 Auflisten, Beenden und Zeit messen von Prozessen

ps – Ausgeben von Informationen zu aktiven Prozesse

ps [*optionen*]

Werden keine *optionen* angegeben, so gibt **ps** nur Informationen zu den vom jeweiligen Terminal aus gestarteten Prozessen und deren Sohnprozesse aus. Zu jedem entsprechenden Prozeß wird dabei eine Zeile ausgegeben, die folgende Informationen enthält: Prozeßnummer (`PID`), Terminalname (`TTY`), verbrauchte Rechenzeit (`TIME`) und Kommandoname (`COMMAND`).

-a	(*all*) Information über alle Prozesse ausgeben.
-e	(*every*) Ausführliche Information zu allen Prozesse ausgeben.
-f	(*full*) vollständige Informationen zu den entsprechenden Prozessen ausgeben; so wird z.B. nicht nur der Kommandoname, sondern die vollständige Aufrufzeile ausgegeben.
-l	(*long*) viele Informationen zu den einzelnen Prozessen ausgeben.
-p *prozeßnr(n)*	Information zu allen Prozessen mit den Prozeßnummern *prozeßnr(n)* ausgeben. Die einzelnen *prozeßnrn* müssen entweder durch Komma getrennt oder innerhalb von "..." angegeben werden. Innerhalb von "..." sind sie entweder mit Komma oder Leerzeichen zu trennen.
-t *terminal(s)*	Information zu allen Prozessen ausgeben, die den Terminals *terminal(s)* zugeordnet sind. Die einzelnen *terminal(s)* müssen entweder durch Komma getrennt angegeben werden oder innerhalb von "...", wobei sie dann entweder durch Komma oder Leerzeichen zu trennen sind.
-u *benutzer(n)*	Information zu allen Prozessen der Benutzer *benutzer(n)* ausgeben. Für *benutzer(n)* kann dabei entweder die User-ID oder der Login-Name angegeben werden. Die einzelnen Benutzer sind dabei in folgender Form anzugeben: *benutzer1*,*benutzer2*,...,*benutzerx* oder "*benutzer1*,*benutzer2*,...,*benutzerx*" bzw. "*benutzer1 benutzer2 ... benutzerx*".
-x	Information auch zu Dämonprozessen, die keinem Terminal zugeordnet sind, ausgeben.

Nur die beiden Optionen **-f** und **-l** legen fest, wieviel Information zu den einzelnen Prozessen auszugeben ist. Alle anderen Optionen bestimmen, zu welchen Prozessen Informationen auszugeben sind.

pstree - Ausgeben der aktuellen Prozeßhierarchie in Baumform

pstree [*optionen*] [*pid* | *loginname*]

pstree zeigt die aktuelle Prozeßhierarchie in einer Baumform an. Ist keine *pid* und kein *loginname* angegeben, so zeigt **pstree** alle Prozesse an, wobei die Wurzel des Baumes der Urprozeß **init** ist. Ist *pid* oder *loginname* angegeben, dann werden nur die Prozesse des Benutzers *loginname* bzw. die Prozeßhierachie ab *pid* angezeigt.

-a	(*all*) Kommandozeile zu jedem Prozeß ausgeben.
-h	(*highlight*) Aktuellen Prozeß und seinen Elternprozeß, seinen Großelternprozeß usw. durch helle Schrift hervorheben.
-l	(*long*) Überlange Zeilen nicht abschneiden, sondern vollständig ausgeben.
-n	(*numeric sort*) Prozesse, die vom gleichen Elternprozeß abstammen, nicht nach Namen, sondern nach ihrer PID (Prozeß-Kennung) sortieren.
-p	(*pid*) bei jedem Prozeß noch Prozeß-ID ausgeben..

top – Auflisten der Prozesse geordnet nach CPU-Belastung

top [**q**]

Das Kommando **top** zeigt eine Liste von gerade aktiven Prozessen an, wobei diese Liste nach der CPU-Belastung der einzelnen Prozesse (CPU intensivste zuerst) sortiert ist. Diese Prozessliste wird alle fünf Sekunden aktualisiert, bis man das Programm **top** mit Drücken der Taste `Q` verlässt. Ist beim Aufruf der optionale Parameter **q** angegeben, aktualisiert **top** die Liste ständig und beansprucht so die gesamte freie Rechenzeit.
Typische Zustände in der STAT-Spalte sind dabei R (*running*), S (*sleeping*), T (*traced* für gestoppte Prozesse) oder W (*swapped*). Bei **top** kann man auch interaktiv Kommandos eingeben. So ermöglicht z.B. die Taste `K` (*kill*) das Beenden eines Prozesses, indem man die entsprechende Prozeßnummer (PID) eingibt.
Zum textbasierten Kommando **top** existieren auch graphische Oberflächen, wie z.B. das KDE-Programm **ksysguard** oder das Gnome-Programm **gtop**.

kill - Senden von Signalen an Prozesse (über Prozeßnummern)

kill [-*signalnr*] *prozeßnr(n)*

Mit **kill** kann ein Benutzer eigenen Prozessen ein Signal schicken. Dieses geschickte Signal kann von den entspr. Prozessen entweder ignoriert oder aber mit einer Signal-Verarbeitungsroutine behandelt werden. Fängt der entsprechende Prozeß ein so gesendetes Signal nicht ab, so wird er beim Eintreffen dieses Signals beendet.
Über *prozeßnr(n)* werden die Prozeßnummern der zu beendenden Prozesse angegeben. Diese ermittelt man am besten mit Kommando **ps**. Wird 0 für *prozeßnr* angegeben, so bedeutet dies, dass alle Prozesse des entspr. Benutzers zu beenden sind. Mit **kill -l** lassen sich alle vorhandenen Signale auflisten. Beispiele für Signalnummern sind:

1	`SIGHUP`
2	`SIGINT` (intr)
3	`SIGQUIT` (quit)
9	`SIGKILL` (kann niemals abgefangen werden und beendet immer den Prozeß, an den es gesendet wird).
15	`SIGTERM` (voreingestellte Signalnummer: beendet den entspr. Prozeß, wenn dieser dieses Signal nicht explizit abfängt)

Läßt sich ein Prozeß nicht beenden, so sollte man es mit dem Signal 9 versuchen, wie z.B. mit **kill -9** oder mit **kill –s SIGKILL** oder auch mit **kill –KILL**.

kill wird auch verwendet, um aktuell laufende Dämonprozessen dazu zu bringen, ihre Konfigurationsdateien neu zu lesen. Dies ist mit **kill –1** oder **kill –s SIGHUP** bzw. kürzer mit **kill –HUP** möglich.
Unter X Window existiert das Kommando **xkill**, das es erlaubt, ein Programm zu beenden, indem man mit der Maus auf das zugehörige Fenster klickt.

killall – Senden von Signalen an Prozesse (über Prozeßnamen)

killall [-*signalnr*] *prozeßname*

Um Prozesse gewaltsam zu beenden, steht mit **killall** ein weiteres Kommando zur Verfügung, das weitgehend identisch zum vorherigen **kill**-Kommando ist. Anders als bei **kill** muss man hier keine Prozeßnummer angeben, sondern den Namen des Programms, das zu beenden ist, wobei dann alle Prozesse dieses Namens beendet werden. Das zu schickende Signal kann dabei als Nummer *–n* oder mit den Namen ohne vorangestellten `SIG`, wie z.B. **–HUP** oder **–KILL**, angegeben werden. Eine Liste aller verfügbaren Signalnamen läßt sich mit **killall –l** anzeigen.

time - Zeitmessungen für Programme durchführen

time *kommando*

Das Kommando **time** bewirkt, dass das angegebene *kommando* ausgeführt und danach die für die Ausführung benötigte Zeit auf der Standardfehlerausgabe ausgegeben wird. Dabei werden drei Zeiten ausgegeben:

real	vergangene Zeit zwischen Kommandostart und seiner Beendigung (engl. *elapsed time*)
user	gebrauchte CPU-Zeit im Benutzermodus
sys	gebrauchte CPU-Zeit im Systemmodus (z.B. bei der Ausführung von Systemroutinen)

6.2 Hintergrund und periodische Prozesse

& – Prozesse im Hintergrund starten

kdozeile **&**

führt die *kdozeile* im Hintergrund aus, so dass nicht auf das Ende der Ausführung von *kdozeile* gewartet wird, sondern sofort wieder das Promptzeichen ausgegeben wird. Es können also sofort weitere Kommandos eingegeben werden, während dieses Kommando im Hintergrund ausgeführt wird. Folgender Aufruf bewirkt z.B., dass `add.c` im Hintergrund kompiliert und Fehler in die Datei `fehler` geschrieben werden:

```
$ cc -o add add.c 2>fehler &  (Enter)
$
```

jobs – Auflisten angehaltener bzw. im Hintergrund laufender Prozesse

jobs [*optionen*] [*jobs*]

Das Kommando **jobs** gibt eine Liste aller gerade angehaltenen bzw. im Hintergrund ablaufende Prozesse aus, wie z.B.:

```
$ jobs (Enter)
[1]   Stopped    find / -name "*.c"
[2]-  Stopped    vi add.c
[3]+  Stopped    bc
$
```

Dem zuletzt angehaltenen Job wird dabei ein Pluszeichen und dem davor angehaltenen Job wird ein Minuszeichen vorangestellt.

bg – Fortsetzen eines Prozesse im Hintergrund

bg [*prozess*]

bg setzt einen unterbrochenen Prozess im Hintergrund fort. Ist keine Prozessnummer angegeben, wird der zuletzt (meist mit (Strg)–Z) unterbrochene Prozeß im Hintergrund fortgesetzt. Um einen anderen angehaltenen Prozeß im Hintergrund fortzusetzen, muss man dessen Namen oder aber Jobnummer (mit vorangestelltem Prozentzeichen) angeben, die man mit dem Kommando **jobs** erfragen kann. Um z.B. den den ersten Job im Hintergrund fortzusetzen, kann man **bg %1** angeben.

fg – Fortsetzen eines Prozesse im Vordergrund

fg [*prozess*]

fg setzt einen unterbrochenen Prozess im Vordergrund fort. Ist keine Prozessnummer angegeben, wird der zuletzt (meist mit (Strg)–Z) unterbrochene bzw. im Hintergrund gestartete Prozeß im Vordergrund fortgesetzt. Um einen anderen angehaltenen Prozeß im Vordergrund fortzusetzen, muss man dessen Namen oder aber Jobnummer (mit vorangestelltem Prozentzeichen) angeben, die man mit dem Kommando **jobs** erfragen kann. Um z.B. den den ersten Job im Vordergrund fortzusetzen, kann man **fg %1** angeben. Hierfür kann auch nur in Kurzschreibeweise **%1** (ohne **fg**) aufgerufen werden.

cron / crontab – Programme in periodischen Zeitintervallen ausführen lassen

/usr/sbin/cron
crontab [datei]
crontab [*optionen*] [**-u** *login-name*]

cron (`/usr/sbin/cron`) wird durch den Init-V-Prozess beim Systemstart automatisch gestartet. Da der dadurch erzeugte Prozeß niemals beendet wird, nennt man einen solchen Prozeß auch *Dämonprozeß* (*daemon*). Er prüft in bestimmten Zeitabständen (Voreinstellung ist: jede Minute) den Inhalt so genannter crontab-Dateien. crontab-Dateien legen die Zeitpunkte fest, zu denen entsprechende Kommandos automatisch ablaufen sollen. Unter Verwendung des Kommandos **crontab** ist es den Benutzern nun möglich, eigene crontab-Dateien anzulegen.

Die erste Aufrufform von **crontab** (ohne Optionen) kopiert die angegebene crontab-*datei* in ein Directory, das von **cron** gelesen wird. Wurde keine *datei* angegeben, so liest **crontab** den zu kopierenden Text von der Standardeingabe (bis (Strg)-D). Jede Zeile einer crontab-Datei besteht aus einer Zeitvorgabe (5 Felder) und dem Kommando, das dann auszuführen ist, wenn diese Zeitvorgabe zutrifft.

Jeder Benutzer kann nur eine crontab-Datei besitzen und jeder Aufruf von **crontab** überschreibt den vorherigen Inhalt der »crontab«-Datei.

Die Ausgabe der einzelnen an **crontab** übergebenen Kommandos wird, wenn keine Ausgabeumlenkung verwendet wurde, dem entsprechenden Benutzer über Mail zugeschickt.

-l	(*list*) Inhalt der momentanen crontab-Datei ausgeben
-r	(*remove*) Momentane crontab-Datei löschen
-e	(*edit*) ruft **vi** (oder einen anderen Editor, der in Variable `VISUAL` oder `EDITOR` angegeben ist) auf, um eine crontab-Datei zu editieren. Falls keine crontab-Datei existiert, dann wird eine neue Datei zum Editieren eröffnet. Nach dem Beenden des Editors wird die gerade editierte Datei als neue crontab-Datei installiert.
-u *login-name*	(*user*) legt fest, dass sich das **crontab**-Kommando auf den Benutzer *login-name* bezieht. Diese Option darf nur von privilegierten Benutzern wie z.B. dem Superuser verwendet werden.

Jede Zeile einer crontab-Datei muss sechs Felder enthalten, die mit Leer- oder Tabulatorzeichen voneinander getrennt sind. Die ersten fünf Felder sind dabei eine Zeitvorgabe und das sechste Feld ist ein String, der das auszuführende Kommando angibt. Die fünf Zeit-Felder legen dabei in der angegebenen Reihenfolge folgendes fest:

1. Feld	Minute	0-59
2. Feld	Stunde	0-23 (0 ist Mitternacht)
3. Feld	Monatstag	1-31
4. Feld	Monat	1-12
5. Feld	Wochentag	0-7 (0 und 7 ist Sonntag)

Für jedes Feld kann nun angegeben werden:

- eine ganze Zahl aus dem angegebenen Zahlenbereich
- ein Teil-Zahlenbereich (wie z.B. 1-5)
- eine mit Komma getrennte Liste von ganzen Zahlen oder Teil-Zahlenbereichen (wie z.B. 2,4-6,8,10) oder
- ein Stern * (deckt gesamten erlaubten Bereich ab).

0 0 * * *	Jeden Tag um 0.00 Uhr
0 6 * * 1	Jeden Montag um 6.00 Uhr
0 8 1 * *	Jeden Ersten eines Monats um 8.00 Uhr
0,15,30,45 8-17 * * *	Alle 15 Minuten von 8.00 Uhr bis 17.00 Uhr jeden Tag
30 10 15 * 3	Jeden 15.ten eines Monats und jeden Mittwoch um 10.30 Uhr

Die Festlegung eines Tages kann über zwei Felder (Monatstag und Wochentag) erfolgen. Beide Felder werden dabei getrennt interpretiert. So wird im letzten obigen Beispiel z.B. nicht nur an jeden 15. eines Monats, der auch ein Mittwoch ist, das entsprechende Kommando ausgeführt.

Um Tage nur über ein Feld festzulegen, muss für das andere Feld * angegeben werden. So legt z.B. **0 0 * * 2** nur den Dienstag fest.

Für die beiden Zeitfelder sind auch Angaben wie ***/***n* erlaubt, z.B.:

```
*/20 * * * * kdo  # alle 20 Minuten
* */4 * * *  kdo  # alle 4 Stunden
```

Die globalen Einstellungen für **cron** befinden sich in der Datei `/etc/crontab`, während benutzerspezifische **cron**-Einstellungen sich in der Datei `/var/spool/cron/tabs/`*loginname* befinden. Im übrigen enthält die Datei `/etc/crontab` anders als die benutzerspezifischen cron-Dateien ein zusätzliches Feld, das den Benutzer festlegt, für den das entsprechende Programm auszuführen ist.

Bei den meisten Linux-Distributionen enthält die Datei `/etc/crontab` nur einige wenige Einträge, die festlegen, dass die Programme aus den folgenden Directories in den entsprechenden Zeitintervallen auszuführen sind:

`/etc/cron.hourly/`	stündlich
`/etc/cron.daily/`	täglich
`/etc/cron.weekly/`	wöchentlich
`/etc/cron.monthly/`	monatlich

Soll also ein bestimmtes Programm in einem dieser Zeitintervalle ausgeführt werden, muss `root` nur das betreffende Programm in das entsprechende Directory kopieren.

Wird in einem Kommando (6. Feld) das **%**-Zeichen (ohne vorangestellten Backslash \) angegeben, so steht dies für ein Neuezeilezeichen.

Jede Zeile in der crontab-Datei, die mit # beginnt, wird als Kommentar interpretiert und somit ignoriert.

Der Systemverwalter kann die Verwendung des crontab-Kommandos allen Benutzern, nur bestimmten Benutzern oder keinem Benutzer erlauben. Dies kann er über Einträge in eine der beiden folgenden Dateien steuern:

1. `/var/spool/cron/allow` oder `/usr/cron.d/cron.allow`
2. `/var/spool/cron/allow` oder `/usr/cron.d/cron.deny`

Wenn die Datei (1) existiert, so ist nur den dort eingetragenen Benutzern die Ausführung des Kommandos **crontab** erlaubt; im anderen Fall wird der Inhalt von (2) überprüft. Dort sind alle Benutzer einzutragen, denen die Ausführung von **crontab** untersagt ist. Wenn (2) zwar existiert, aber leer ist, so darf jeder Benutzer **crontab** aufrufen. Wenn keine der beiden Dateien existiert, so darf keiner der normalen Benutzer **crontab** aufrufen; dieses Privileg ist dann nur dem Systemverwalter und dem Superuser vorbehalten.

Es ist anzumerken, dass jeder neue **crontab**-Aufruf den Inhalt der alten crontab-Datei überschreibt. Deshalb ist es üblich, sich alle periodisch auszuführenden Aufträge in einer eigenen crontab-Datei zu halten, die bei Änderungen beliebig editiert werden kann und dann beim crontab-Aufruf angegeben wird; so kann man ein ständiges neues Eintippen aller crontab-Aufträge bei neuen **crontab**-Aufrufen umgehen.

Mehr Informationen zu **crontab** kann man sich mit **man 5 crontab** anzeigen lassen.

Unter Linux existieren auch graphische Oberflächen, um die **cron**-Konfigurationsdateien zu editieren, von denen hier einige genannt sind: **gcrontab** (Gnome-Programm), **kcron** (KDE-Programm) und **vcron** (Graphische Tcl/Tk-Oberfläche).

at - Kommandos zu einem bestimmten späteren Zeitpunkt ausführen lassen

at *zeit* [*datum*] [+*increment*] (Enter)
kommando1 (Enter)
kommando2 (Enter)
:
(Strg)-D

zeit und *datum* legen die Zeit und das Datum für den Start der entsprechenden Kommandos fest.
increment legt den Zeitpunkt für den Start der Kommandos relativ (nicht absolut) fest.

at -r *job(s)*
at -l [*job(s)*]
at -d *job*

Für *job(s)* sind die entsprechenden Auftragsnummer(n) (*job number*) anzugeben.

Das Kommando **at** liest die auszuführenden Kommandos über die Standardeingabe ein und bewirkt, dass diese Kommandos zum angegebenen Zeitpunkt ausgeführt werden, selbst wenn der entsprechende Benutzer zu diesem Zeitpunkt nicht am System angemeldet ist.

Die zu einem späteren Zeitpunkt auszuführenden Kommandos müssen nach der Angabe des **at**-Kommandos in den folgenden Zeilen vom Benutzer eingegeben werden. Das Ende der Kommandoeingabe wird dabei **at** mit EOF ((Strg)-D) als einziges Zeichen einer Zeile) mitgeteilt. Natürlich können die entsprechenden Kommandos auch in einer Datei angegeben sein und mit Eingabeumlenkung dem **at**-Kommando übermittelt werden.

Die Ausgabe der einzelnen an **at** übergebenen Kommandos wird, wenn keine Ausgabeumlenkung verwendet wurde, dem entsprechenden Benutzer über Mail zugeschickt.

Die bei **at** angegebenen Kommandos laufen unter den gleichen Bedingungen ab, die vorlagen als sie angegeben wurden (wie z.B. gleiches Working-Directory).

-l	(*list*) Auflisten der mit **at** eingerichteten Jobs
-r	(*remove*) Löschen von Jobs, die mit **at** eingerichtet wurden
-d	(*display*/*delete*) listet den Inhalt des angegebenen *job* auf. Unter Linux löscht es den angegebenen *job*.

zeit legt die Zeit für den Start der Kommandos in Stunden und Minuten fest:

- gibt man für *zeit* nur eine oder zwei Ziffern an, so wird damit eine Stunde festgelegt.
- Sind für *zeit* vier Ziffern angegeben, so spezifizieren die ersten beiden Ziffern die Stunde und die letzten beiden die Minute, z.B. würde die Angabe **1316** der Uhrzeit `13:16` entsprechen.

Unix arbeitet mit einer 24-Stunden-Uhr, es sei denn, es wird explizit die amerikanische Schreibweise: **0617am** (Morgens) oder **0853pm** (Abends) verwendet. Die Stunden und Minuten dürfen auch durch Doppelpunkt getrennt (z.B. **14:53** oder **4:12**) angegeben werden. Falls eine angegebene Zeit für den momentanen Tag schon vorüber ist und es wurde keine *datum*-Angabe gemacht, so wird der Zeitpunkt auf den nächsten Tag gleicher Zeit verschoben.

Bei der *zeit*-Angabe können u.a. auch folgende Wörter verwendet werden:

zulu	für Greenwich Mean Time; ist als Suffix anzugeben, z.B. **2:13zulu** oder **2:13 zulu**
noon	für Mittag
midnight	für Mitternacht
now	für jetzt; ist im Zusammenhang mit *increment* zu verwenden. **at now** liefert die Fehlermeldung *too late*.
next	muss nach *zeit* und vor *datum* angegeben sein.

Für *datum* sind folgende Angaben möglich:

Monat Tag	(z.B. **Feb 23**); Als Monat-Angabe möglich: **Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec**
Monat Tag, Jahr	(z.B. **May 12,1990**)
Wochentag	(z.B. **Wednesday** oder **Wed**); muss in englisch angegeben sein: **Monday**, **Tuesday**, **Wednesday**, **Thursday**, **Friday**, **Saturday**, **Sunday**, wobei auch Abkürzungen erlaubt sind.
Heute oder Morgen	(**today** oder **tomorrow**)

Ist ein angegebenes *datum* (ohne Jahresangabe) bereits verstrichen, so wird es auf das nächste Jahr verschoben.

Für *increment* muss eine Zahl gefolgt von einem der folgenden Wörter angegeben werden: **minutes**, **hours**, **days**, **weeks**, **months** oder **years**; der Singular für diese Wörter (ohne das Plural-S) ist ebenso erlaubt. Um z.B. einen Job in 3 Stunden ablaufen zu lassen, könnte **at now +3 hours** .... angegeben werden. Weitere Beispiele sind: **+3 minutes**; **+10 hours**; **+4 weeks**; **+1 week**; **+2 years**; **+7 days**; **+ 2 months**.

at 1700 May 12 <kdos	Kommandos aus Datei `kdos` am 12.Mai um 17:00 ausführen
at midnight Sun <kdos	Kommandos aus Datei `kdos` nächsten Sonntag um Mitternacht ausführen
at 530am tomorrow <kdos	Kommandos aus Datei `kdos` morgen früh um 5:30 ausführen
at now +3 hours <kdos	Kommandos aus Datei `kdos` in 3 Stunden ausführen
at 17 Fri +1 week <kdos	Kommandos aus Datei `kdos` am Freitag in einer Woche ausführen

Der Systemverwalter kann das Absetzen des **at**-Kommandos allen Benutzern, nur bestimmten Benutzern oder keinem Benutzer erlauben. Dies kann er über Einträge in eine der beiden Dateien steruen:

1. `/var/spool/at/allow` oder `/usr/cron.d/at.allow`
2. `/var/spool/at/allow` oder `/usr/cron.d/at.deny`

Wenn die Datei (1) existiert, so ist nur den dort eingetragenen Benutzern die Ausführung des **at**-Kommandos erlaubt; im anderen Fall überprüft das System den Inhalt von (2), in welchem alle Benutzer einzutragen sind, denen die Ausführung von **at** untersagt ist. Wenn (2) zwar existiert, aber leer ist, so darf jeder Benutzer **at** aufrufen. Wenn keine der beiden Dateien existiert, so darf keiner der normalen Benutzer **at** aufrufen; dieses Privileg ist dann nur dem Systemverwalter und dem Superuser vorbehalten.

batch - Kommandos irgendwann später ausführen lassen

batch
kommando1 (Enter)
kommando2 (Enter)
:
(Strg)-D

Das Kommando **batch** liest die auszuführenden Kommandos über die Standardeingabe ein und bewirkt, dass diese Kommandos zu einem späterem Zeitpunkt ausgeführt werden, wenn das System dafür Zeit hat; wenn also die Systemlast dies zuläßt. Die angegebenen Kommandos werden dann ausgeführt, selbst wenn der entsprechende Benutzer zu diesem Zeitpunkt nicht am System angemeldet ist.

Die später auszuführenden Kommandos müssen vom Benutzer nach der Angabe des **batch**-Kommandos in den folgenden Zeilen eingegeben werden. Das Ende der Kommandoeingabe wird dabei **batch** mit EOF ((Strg)-D) als einziges Zeichen einer Zeile) mitgeteilt. Natürlich können die entsprechenden Kommandos auch in einer Datei angegeben sein und mit Eingabeumlenkung dem **batch**-Kommando übermittelt werden.

Die Ausgabe der einzelnen an **batch** übergebenen Kommandos wird, wenn keine Ausgabeumlenkung verwendet wurde, dem entsprechenden Benutzer über Mail zugeschickt.

Die bei **batch** angegebenen Kommandos laufen unter den gleichen Bedingungen ab, die vorlagen als sie angegeben wurden (wie z.B. gleiches Working-Directory).

nohup - Prozesse bei Sitzungsende weiterlaufen lassen

nohup *kommandoname* [*argumente*]

Soll ein Prozeß, wie z.B. eine umfangreiche Kompilierung, die im Hintergrund läuft, nach der Beendigung einer Unix-Sitzung (durch Schließen des Shell-Fensters bzw. durch Abmelden) weiterlaufen, so ist dies mit dem Kommando **nohup** möglich.

Wenn bei **nohup** die Ausgabe nicht explizit umgelenkt wurde, so wird sowohl die Standardausgabe als auch die Standardfehlerausgabe in die Datei `nohup.out` des Working-Directorys geschrieben. Ist dies wegen der Zugriffsrechte nicht möglich, so werden die Ausgaben in die Datei `nohup.out` des Home-Directorys geschrieben.

sleep – Suspendieren von Prozessen

sleep *zeit*

Das Kommando **sleep** bewirkt, dass *zeit* Sekunden vergehen müssen, bevor der entsprechende Prozeß fortgesetzt wird. Hinter der *zeit* (eine Zahl) kann auch ein Buchstabe (**s**,**m**,**h**,**d**) angegeben werden, um diese *zeit* als Sekunden, Minuten, Stunden oder Tage festzulegen.

Der Benutzer `emil` hat z.B. `egon` versprochen, ihm eine Datei mit Namen `automake.c` in das Directory `/tmp` zu kopieren, damit `egon` sie dort abholen kann. In diesem Fall wäre ein von `egon` getarteter Hintergrundprozeß nützlich, der ständig nachschaut, ob die versprochene Datei bereits in `/tmp` angekommen ist. Da die Shell über eine **while**-Schleife und **if**-Anweisung verfügt, könnte dies wie folgt realisiert werden:

```
$ while true (Enter)
> do (Enter)
>    if [ -f /tmp/automake.c ] (Enter)
>    then (Enter)
>         echo "Datei /tmp/automake.c ist angekommen" (Enter)
>         break (Enter)
>    fi (Enter)
>    sleep 20 (Enter)
> done & (Enter)
821
$
```

true bedeutet in diesem Fall, dass die **while**-Bedingung immer erfüllt ist; es handelt sich hier also um eine "Endlosschleife", die erst mit **break** verlassen wird, wenn die Datei `/tmp/automake.c` existiert. Die Überprüfung, ob diese Datei vorhanden ist, erfolgt mit

if [-f /tmp/automake.c]

Existiert `/tmp/automake.c`, so wird "*Datei /tmp/automake.c ist angekommen*" am Bildschirm ausgegeben und mit **break** die **while**-Schleife verlassen. Da nach der **while**-Schleife keine weiteren Anweisungen angegeben sind, wird danach auch dieser Hintergrundprozeß beendet.

Existiert dagegen diese Datei `/tmp/automake.c` (noch) nicht, so wird als nächstes Kommando **sleep 20** aufgerufen, was bewirkt, dass die Ausführung für 20 Sekunden angehalten wird, bevor sie mit der Überprüfung auf die Existenz der Datei `/tmp/automake.c` (**if [..]**) wieder fortgesetzt wird. Ist diese Datei immer noch nicht vorhanden, so wird mit **sleep 20** die Ausführung wieder für 20 Sekunden angehalten usw.

6.3 Programme (Prozesse) als anderer Benutzer ausführen

su - Ändern der Benutzerkennung, ohne sich abzumelden

su [*optionen*] [*login-name*]

Mit diesem Kommando **su** (*switch user*) kann man sich in der aktuellen Shell bzw. in einem Shell-Fenster, ohne dass man sich abmelden muss, als anderer Benutzer anmelden. Will man sich z.B. mit dem Loginnamen `emil` anmelden, muss man nur **su -l emil** aufrufen: Danach wird nach dem Passwort von `emil` gefragt. Ist die Passworteingabe richtig, arbeitet man ab diesem Zeitpunkt unter dem Login `emil`, so als ob man sich bereits zu Beginn als `emil` angemeldet hätte. Das Beenden dieser Sitzung als `emil` erreicht man mit (Strg)+D bzw. mit der Eingabe von **exit**.

Die Opion **-l** bewirkt im übrigen, dass alle Dateien (`.profile`, `.bashrc` usw.) gelesen werden, die auch bei einem normalen Login gelesen werden. Gibt man keinen Namen an, also ruft nur **su -l** auf, so entspricht dies dem Aufruf **su -l root**.

Arbeitet man unter X Window, so bringt das Anmelden mit **su** als `root` ein kleines Problem mit sich, was nachfolgend gezeigt ist:

```
$ kate /etc/fstab (Enter)
kate: cannot connect to X server
```

Die Fehlermeldung resultiert aus der Tatsache, dass unter X Window voreingestellt ist, dass kein fremder Benutzer (hier `root`) auf dem eigenen Rechner Programme starten darf. Diese Restriktion kann man aufheben, indem man in einem anderen Shell-Fenster (als ursprünglicher Benutzer) folgendes eingibt: **xhost +localhost**. Damit ist das Problem aber noch nicht ganz beseitigt, da beim Start von X Window die Variable `DISPLAY` automatisch auf den eigenen Rechner eingestellt wird. Da man sich nun aber ohne einen neuen Start von X Window als `root` angemeldet hat, ist bei dieser temporären `root`-Sitzung die Variable `DISPLAY` nicht richtig gesetzt. Das kann man nun in der **su**-Sitzung nachholen mit:

```
root# export DISPLAY=localhost:0 (Enter)
root# kate /etc/fstab (Enter)      [funktioniert nun]
....
root# exit (Enter)   [Beenden der temporären root-Sitzung]
$ xhost -localhost (Enter)   [xhost-Rechte wieder entziehen]
```

Jeder **su**-Aufruf (ob erfolgreich oder nicht) wird mitprotokolliert (üblicherweise in `/var/log/messages`).

Möchte man sich die Einstellung von **xhost** und `DISPLAY` ersparen, kann man unter KDE das Programm **kdesu** verwenden, wie z.B. **kdesu kate /etc/fstab**. Nach diesem Aufruf, wird eine Login-Box eingeblendet, in der man nach dem Passwort von `root` gefragt wird.

sudo – Ausführen eines Programms als anderer Benutzer

sudo [*optionen*] *kommando*

Das Kommando **sudo** ermöglicht es, ausgewählten Benutzern bestimmte Programme unter einem anderen Loginnamen ausführen zu lassen. So kann z.B. `root` festlegen, dass einige Benutzer auch administrative Aufgaben durchführen, wie z.B. neue Benutzer anlegen oder neue Drucker einrichten dürfen, ohne dass sie das Passwort von `root` kennen müssen. **sudo** protokolliert alle ausgeführten Kommandos (ob erfolgreich oder nicht) üblicherweise in `/var/log/messages` mit.

Damit ein Benutzer ein bestimmtes Kommando, das er normalerweise nicht ausführen kann, mittels **sudo** aufrufen kann, muss `root` einen entsprechenden Eintrag in der Datei `/etc/sudoers` vornehmen. Dies geschieht normalerweise mit dem Aufruf des Kommandos **visudo**. Nachfolgend ist ein solcher Eintrag gezeigt:

```
hansi  ALL=/usr/sbin/useradd
```

Dieser Eintrag bewirkt, dass der Benutzer `hansi` nun z.B. mit folgendem Aufruf neue Benutzer anlegen kann:

```
hansi$ sudo /usr/sbin/useradd antonia (Enter)
Passwort: *******   [hier Passwort von hansi (nicht von root) eingeben]
```

Möchte man ein Programm mittels **sudo** nicht als `root`, sondern unter einem anderen Loginnamen ausführen, so ist der folgende Aufruf erforderlich:

sudo -u *loginname kommando*

6.4 Priorität von Prozessen ändern

nice - Prozesse mit anderer Priorität ablaufen lassen

nice [*optinen*] *kommando* [*argumente*]

Öfters fallen beim Arbeiten an einem System Aufgaben an, die nicht zeitkritisch sind. In solchen Fällen ist es ratsam, die Priorität des auszuführenden Kommandos (Prozesses) freiwillig herunterzusetzen, um dem System mitzuteilen, dass die Ausführung des gegebenen Auftrags nicht so dringlich ist.

Dazu steht das Kommando **nice** zur Verfügung, mit dem man Kommandos mit niedrigerer bzw. höherer Priorität ausführen lassen kann. Die Priorität legt man dabei nach der Option **-n** fest, indem man eine Zahl zwischen 20 (ganz niedrig) bis -20 (ganz hoch) angibt. Die Voreinstellung ist, dass Prozesse mit der Priorität +10 gestartet werden. Grundsätzlich gilt jedoch, dass nur `root` Programme mit einer höheren Priorität als 0 starten kann. Beim folgenden Aufruf wird das **find**-Kommando mit niedrigerer Priorität als normal gestartet:

```
nice -n 10 find / -atime 7 -print > siebtage.wo 2> /dev/null &
```

renice – Priorität laufender Prozesse ändern

renice *neue-prio pid(s)*

Mit dem Kommando **renice** kann man die Priorität von bereits laufenden Prozessen ändern: Mit dem Kommando **top** und anschließender Eingabe der Taste R kann man ebenso die Priorität eines Prozesses ändern wie mit **ksysguard**, wo man nur auf den entsprechenden Prozeß mit der rechten Maustaste klicken muss und dann im eingeblendeten Popup-Menü den Menüpunkt `Prozess-Priorität ändern` auswählen muss. Auch hier ist zu beachten, dass nur `root` die Priorität eines bereits laufenden Prozesses erhöhen kann.

7 Speicherplatz-Informationen

df - Erfragen des freien Speicherplatzes

df [*optionen*] [*directory* | *gerätedatei* | *dateisystem* ...]

Mit dem Kommando **df** kann man sich den freien Speicherplatz auf allen montierten und auch unmontierten Dateisystemen oder – wenn ein *directory* angegeben ist – von einem bestimmten Dateisystem anzeigen lassen. Ist eine *gerätedatei* angegeben, so gibt **df** Speicherplatz-Information zu diesem Gerät aus.

Ist kein *directory*, keine *gerätedatei* und kein *dateisystem* angegeben, wird der freie Speicherplatz zu allen vorhandenen Dateisystemen ausgegeben.

Bei der Ausgabe von **df** steht in der ersten Spalte der Name des Montierpunktes, zwischen den runden Klammern der Gerätename des Dateisystems und in den folgenden Spalten die Anzahl der freien Blöcke und die Anzahl der noch freien Dateien (inodes).

Die Anzahl der freien Dateien ist nur für das lokale Dateisystem richtig; bei verteilten Dateisystemen steht hier immer der Wert -1. Die Angaben für die Dateisysteme `/dev/fd`, `/stand` und `/proc` sind bedeutungslos, da es sich hierbei nicht um "echte" Dateisysteme handelt.

-F *fstyp*	legt den Dateisystem-Typ auf *fstyp* fest.
-b	nur die Anzahl von freien KBytes ausgeben.
-e	nur die Anzahl der freien Dateien (inodes) ausgeben.
-g	alle verfügbare Information ausgeben; kann nur für montierte Dateisystem verwendet werden. Diese Option darf nicht mit der Option **-o** verwendet werden und schaltet die eventuell gleichzeitig angegebenen Optionen **-b**, **-e**, **-k**, **-n** und **-t** aus.
-i	die gesamte Anzahl von inodes, die Anzahl von freien und belegten inodes ausgeben; zusätzlich wird noch in Prozent ausgegeben, wie viele inodes belegt sind.
-k	der gesamte, der belegte und der freie Speicherplatz wird in Kilobyte sowie der belegte Platz in Prozent ausgegeben.
-l	nur für das lokale Dateisystem Speicherplatz-Information ausgeben; diese Option darf nur für montierte Dateisysteme verwendet und nicht mit der Option **-o** kombiniert werden.
-n	nur Dateisystemtypen ausgeben; darf nicht mit der Option **-o** kombiniert werden.
-t	zusätzlich zum freien Speicherplatz wird noch der gesamte verfügbare Speicherplatz (frei und belegt) ausgegeben.
-o *optionen*	Dateisystem-spezifische *optionen* angeben.
-v	Ausgabe erfolgt im Stil von **dfspace**.

Unter Linux verhält sich das Kommando **df** weitgehend so wie unter anderen Unix-Systemen auch, nur dass es sich in einigen Optionen von diesen unterscheidet. Die wichtigsten Optionen des **df**-Kommandos unter Linux sind:

-a	(**--all**) Ausgeben aller Dateisysteme einschließlich solcher, die 0 Blöcke haben
-h	(**--human-readable**) Ausgeben der Größen in üblicher Sprechweise (wie z.B. 5K, 234M, 2G)
-i	(**--inodes**) Statt des freien Speicherplatzes Informationen über noch freie inodes ausgeben
-k	(**--kilobytes**) Ausgeben der Größen in KBytes (Voreinstellung)
-m	(**--megabytes**) Ausgeben der Größen in MBytes
-T	(**--print-type**) Ausgeben des Dateisystemtyps

dfspace - Erfragen des noch freien Speicherplatzes in allen Dateisystemen

dfspace [**-F** *fstyp*] (unter Linux nicht verfügbar)

dfspace ist ein Shell-Skript, das das **df**-Kommando verwendet. **dfspace** gibt für alle montierten "echten" Dateisystemen (Ausnahme ist z.B. `/proc`) den noch freien Speicherplatz in MBytes und Prozent aus.

-F *fstyp*	nur für *fstyp*-Dateisysteme ausgeben.

du - Erfragen des Speicherplatzes, der von Dateien bzw. Directories belegt wird

du [*optionen*] [*datei(en)*]

Die angegebenen *datei(en)* können dabei Directories oder einfache Dateien sein.

Das Kommando **du** meldet die Anzahl von Speicherblöcken, die von den angegebenen *datei(en)* belegt sind. Handelt es sich bei einer angegebenen *datei* um ein Directory, so wird die Anzahl der Speicherblöcke gemeldet, die vom gesamten Directorybaum belegt wird. Sind keine *datei(en)* angegeben, so nimmt **du** das Working-Directory an; d.h. es meldet die Anzahl der Speicherblöcke, die vom Directorybaum des Working-Directorys belegt werden. Dateien mit zwei oder mehr Links werden dabei nur einmal gezählt und Dateien, die für den Aufrufer von **du** keine Leserechte gewähren, werden nicht mitgezählt.

-a	Es wird für jede einzelne Datei die Blockanzahl ausgegeben; ist die Voreinstellung.
-r	Dateien, die kein Leserecht gewähren, werden gemeldet; Voreinstellung ist, dass dies nicht gemeldet wird.
-s	Es wird nur die Gesamtanzahl der belegten Speicherblöcke für jede der angegebenen *datei(en)* ausgegeben.

Unter Linux verhält sich das Kommando **du** weitgehend so wie unter anderen Unix-Systemen auch, nur dass es sich in einigen Optionen von diesen unterscheidet. Die wichtigsten zusätzlichen Optionen des **du**-Kommandos unter Linux, die oben nicht erwähnt wurden, sind:

-b	(**--bytes**) Ausgeben der Größen in Bytes
-c	(**--total**) Bei Anwendung von **du** auf Dateien (und nicht Directories) wird als abschließender Wert die Endsumme aller zuvor aufgelisteten Größen angezeigt. Mit dieser Option kann leicht festgestellt werden, wieviel Speicherplatz alle Dateien eines bestimmten Typs (wie `*.c`) benötigen.
-h	(**--human-readable**) Ausgeben der Größen in üblicher Sprechweise (wie z.B. 5K, 234M, 2G)
-k	(**--kilobytes**) Ausgeben der Größen in KBytes (Voreinstellung)
-m	(**--megabytes**) Ausgeben der Größen in MBytes
-S	(**--separate-dirs**) Ausgeben des Speicherbedarfs eines Directorys (ohne seine Subdirectories)

du wird oft verwendet, um festzustellen, welche Directories den meisten Speicherplatz benötigen. Dies ist v.a.D. dann notwendig, wenn das Dateisystem schon fast voll ist und der Systemverwalter die einzelnen Benutzer zum Bereinigen ihrer Directories auffordern möchte.

free – Anzeigen von verfügbaren Speicherplatz (RAM und Swap)

free

free zeigt an, wie der verfügbare Speicherplatz (RAM und Swap-Speicher) genutzt wird.

8 Dateisystem-Kommandos

8.1 Montieren, Partitionieren, Formatieren und Kopieren

mount / umount – An- und Abmontieren eines Dateisystems

mount [*optionen*] [*gerät mountpunkt*]
umount [*gerät*]
umount [*mountpunkt*]

Das Dateisystem von Unix ist nicht eine Einheit, sondern setzt sich aus mehreren Teilen zusammen, die sich auf verschiedenen Speichermedien – wie z.B. Festplatten, CD-ROMs, Floppy-Disks oder über ein Netz erreichbaren Systemen – befinden können. Die Möglichkeit, Dateisysteme in den vorhandenen Directorybaum einzuhängen, ermöglicht bei Bedarf einen Ausbau der vorhandenen Speicherkapazität.

Um sich alle montierte Dateisysteme anzeigen zu lassen, muss man **mount** (bzw. **/etc/mount**) ohne jegliche Argumente aufrufen.

```
$ mount (Enter)
....
/dev/sda5 on / type ext2 (rw)
proc on /proc type proc (rw)
/dev/sda1 on /C type vfat (rw,noexec,nosuid,nodev,user=herold)
/dev/sda7 on /usr type ext2 (rw)
/dev/sda8 on /home type ext2 (rw)
....
```

Um ein Dateisystem wieder abzumontieren, muss das Kommando **umount** mit dem Directorynamen, an dem das jeweilige Dateisystem anmontiert ist, bzw. mit dem Device-Namen des Dateisystems aufgerufen werden. Beim Herunterfahren eines Systems werden die einzelnen anmontierten Dateisysteme automatisch durch **umount**-Aufrufe abmontiert. Die wichtigsten Optionen, die **mount** für jeden Dateisystemtyp unterstützt, sind:

-a	Montieren aller in `/etc/fstab` aufgeführten Dateisysteme, wenn für diese nicht Option `noauto` angegeben wurde. Wird mit **-t** *fstype* zuätzl. ein Dateisystemtyp angegeben, werden nur Geräte dieses Typs montiert.
-r	Dateisystem wird als *read-only* anmontiert, was kein Schreiben auf diesem Dateisystem zuläßt.
-w	Dateisystem wird als *read-write* anmontiert, so dass Schreiben und Lesen auf diesem Dateisystem möglich ist.
-t *fstype*	Angabe des Typs des zu montierenden Dateisystems. Dies ist nur notwendig, falls der Kernel anhand des Superblocks nicht den Typ des jeweiligen Dateisystems ermitteln kann.
-o *optionen*	Kommaseparierte Liste von Optionen. Nachfolgend sind die wichtigsten Optionen kurz aufgezählt:
	async Ein- und Ausgaben auf dem Dateisystem erfolgen asynchron, z.B. werden Daten zwischengespeichert ("*gecached*") und zyklisch geschrieben.
	atime Die Zugriffszeit einer Datei wird bei jedem Zugriff aktualisiert (im Inode).
	auto Automatisches Montieren (nur als Option in der Datei `/etc/fstab` sinnvoll).
	defaults Voreingestellte Optionen verwenden (**rw**, **suid**, **dev**, **exec**, **auto**, **nouser**, **async**).
	dev Devices des zu montierenden Dateisystems werden auch als solche interpretiert.
	exec Ausführen von Programmen erlaubt.
	remount Zum erneuten Montieren eines schon anmontierten Dateisystems.
	ro (*read-only*) Nur Leseberechtigung.
	rw (*read-write*) Lese- und Schreibberechtigung.
	suid setuid-Bits behalten ihre Wirkung
	sync Alle Ein- und Ausgaben erfolgen synchron (ohne Zwischenpufferung).
	user Dateisystem darf von normalen Benutzern anmontiert werden (nur in `/etc/fstab` sinnvoll).

Eigenschaften wie das Dateisystem anzumontieren ist, werden durch **-o** eingeleitet. Bei vorangestelltem **no** (**noauto**, **nodev** usw.) werden die Eigenschaften negiert (es gibt aber kein **noremount**, **noro** und **norw**).

VFS-Dateisysteme: Das VFS von Unix verwaltet u.a. die folgenden Dateisysteme:

s5	ist das traditionelle Dateisystem von System V.3, bei dem die Namen von Dateien nur 14 Zeichen lang sein dürfen.
sysv	wird von XENIX, SCO- und Coherent-Systemen eingesetzt
ufs	Implementierung des *Fast Filesystem* von BSD Unix. Namen dürfen bis zu 255 Zeichen lang sein.
rfs	Implementierung des *Remote File Sharing* (RFS) von AT&T.
nfs	Implementierung des *Network File Systems* (NFS) von SunOS.
proc	Dateisystem in System V.4, über das auf Datenstrukturen von Prozessen zugegriffen werden kann.
bfs	enthält alle für den Systemboot notwendigen Dateien, Kernel und Bootloader.
fdfs	erlaubt Zugriffe auf Dateikanäle eines Prozesses.
fifofs	bietet eine Schnittstelle zu Named Pipes.
specfs	ist eine Schnittstelle zu den Gerätedateien.

Während das **s5**-, das **sysv**-, das **ufs**- und das **rfs**-Dateisystem "echte" Dateisysteme sind, stehen auf den anderen Dateisystemen nicht unbedingt alle zur Dateibearbeitung notwendigen Operationen zur Verfügung.

Linux-Dateisysteme: Die wichtigsten von Linux unterstützten Dateisysteme sind u.a.:

ext2	(*extended filesystem, Version2*) dies ist heute das Standard-Dateisystem unter Linux.
ext3	Nachfolger des **ext2**-Dateisystems. Bei **ext3** handelt es sich jedoch um ein *Journaling Filesystem*.
reiserfs	ist ebenfalls wie **ext3** ein *Journaling Filesystem*, das von *Hans Reiser* entwickelt wurde.
xfs	ist ein Workstation-System der Firma SGI, bei dem es sich auch um ein *Journaling Filesystem* handelt.
jfs	ist ein *Journaling Filesystem* der Firma IBM.
sysv	ermöglicht den Zugriff auf SCO-, XENIX- und Coherent-Partitionen.
ufs	ermöglicht den Lesezugriff auf Partitionen von SunOS, FreeBSD, NetBSD und NextStep.
msdos	ermöglicht Zugriff auf MS-DOS-Disketten/-Festplatten. Dabei ist nicht nur Lesen, sondern auch Schreiben möglich.
umsdos	ermöglicht Zugriff auf MS-DOS-Disketten/-Festplatten, wobei jedoch auch lange Dateinamen mit Unix-Zugriffsrechten und Links verwendet werden können, um Linux auch in DOS-Partition installieren zu können.
vfat	ermöglicht den Zugriff auf Dateisysteme von WindowsXX.
ntfs	ermöglicht nun auch den Zugriff auf das Windows-NT-Dateisystem.
hpfs	ermöglicht den Lesezugriff auf Partitionen von OS/2.
hfs	ermöglicht den Zugriff auf dieses MacIntosh-Dateisystem.
iso9660	hat sich als Norm für die Dateiverwaltung auf CD-ROMs durchgesetzt.
udf	(*universal disk format*) ist der Nachfolger zu **iso9660**.
nfs	(*Network File System*) Unter Unix das übliche Netzwerk-Dateisystem.
ncpfs	(*Network Core Protocol*) Netzwerk-Dateisystem von Novell.
smb	(*Server Message Buffer*) Netzwerk-Dateisystem von Microsoft.
proc	kein echtes Dateisystem; für Abbildung von Verwaltungsinfos des Kernels bzw. der Prozeßverwaltung benutzt.
ramdisk	Montieren eines Bereichs des Hauptspeichers als Dateisystem
shmfs	(*shared memory*) ermöglicht effizienten Datenaustausch zwischen zwei Programmen.
devfs	neues Device-Dateisystem, das alte Konzept mit Tausenden von Gerätedateien im Directory `/dev` ersetzen soll.
loopback	kein echtes Dateisystem, sondern ermöglicht ein ganzes Dateisystem in einer Datei abzulegen und diese Datei mit **mount** als Dateisystem anzumontieren.
auto	ist kein Dateisystem, ist aber als Schlüsselwort bei **mount** bzw. in `/etc/fstab` erlaubt, um Linux anzuweisen, das entsprechende Dateisystem automatisch zu erkennen.
autofs, **autofs4**	keine echten Dateisysteme, sondern Programme, die *Automount-Funktion* übernehmen, also automatisch bei Bedarf Dateisysteme anmontieren und auch wieder abmontieren, wenn diese längere Zeit nicht benutzt werden.

Die Datei /etc/fstab: In `/etc/fstab` werden Dateisysteme spezifiziert, die beim Systemstart zu montieren sind, und alle Dateisysteme, die z.B. auch von einem normalen Nutzer an- bzw. abmontiert werden dürfen. Jedes Dateisystem wird durch eine eigene Zeile beschrieben. In jedem Fall müssen die beiden folgenden Zeilen in `/etc/fstab` vorhanden sein, die natürlich abhängig von der Konfiguration auch anders aussehen können:

```
# in /etc/fstab muss Montieren von root- und proc-Dateisystem vorhanden sein
/dev/hda6    /       reiserfs     defaults 1 2 # root-Dateisystem
proc         /proc   proc         defaults 0 0 # proc-Dateisystem
```

Alle weiteren Zeilen sind optional. Jede Zeile besteht aus sechs Spalten, die folgende Bedeutung besitzen:

Device Mountpoint Type Options Dump Check

- *Device* (1. Spalte): gibt Gerätenamen des Datenträgers an. Sollen Directories von fremden Rechnern über `nfs` anmontiert werden, muss hier folgendes angegeben werden: `rechnername:`*/directory_auf_fremden_rechner*
- *Mountpoint* (2.Spalte): gibt Directory an, an das der entspr. Datenträger im Dateisystem anmontiert werden soll. Directory muss existieren, muss aber nicht leer sein, allerdings kann bei erfolgreichen Montierung auf darin enthaltenen Dateien nicht mehr zugegriffen werden, sondern nur noch auf Dateien des montierten Datenträgers. Erst mit dem Abmontieren des montierten Datenträgers werden die zuvor enthaltenen Dateien wieder sichtbar.
- *Type* (3. Spalte): Typ des zu montierenden Dateisystems an: `ext2`, `ext3`, `reiserfs`, `msdos`, `nfs`, `proc`, `vfat`, `iso9660`, usw. `auto` steht hier für automat. Erkennung bei Disketten- und CDROM-Laufwerken.
- *Options* (4.Spalte): gibt Optionen für den Zugriff auf den Datenträger an. Mehrere Optionen können durch Kommas (keine Leerzeichen) getrennt werden. Die wichtigsten Optionen sind:

defaults	Voreinstellungen (**rw**, **suid**, **auto**, **nouser**...)
(**no**)**auto**	**auto** bewirkt beim Systemstart das automatische Anmontieren dieses Datenträgers. **noauto** verhindert dies, hat aber den Vorteil, dass man zum späteren Anmontieren dieses Datenträgers nur **mount** *name* aufrufen muss. Für *name* ist dabei entweder der Gerätename (aus 1. Spalte) oder der Mountpunkt (aus 2. Spalte) anzugeben. Die fehlenden Optionen werden dabei aus entspr. Zeile in `/etc/fstab` (4. Spalte) gelesen.
(**no**)**dev**	Voreinstellung ist, dass zeichen- bzw. blockorientierte Gerätedateien auf dem montierten Dateisystem entsprechend behandelt werden. **nodev** unterbindet dies.
(**no**)**exec**	legt fest, ob auf dem Dateisystem befindliche Programme ausgeführt werden dürfen oder nicht. Voreinstellung ist hier meist **exec** (Programmausführung zulassen).
(**no**)**suid**	bei **suid** behalten die setuid-Bits auf dem montierten Dateisystem ihre Wirkung und bei **nosuid** nicht.
(**no**)**sync**	**sync** legt fest, dass Schreibzugriffe nicht zwischenzupuffern, sondern sofort physikalisch zu schreiben sind. Dies bedeutet zwar einen langsameren, aber sichereren Zugriff.
(**no**)**user**	bei **user** darf jeder Benutzer diesen Datenträger mit **mount** montieren bzw. mit **umount** abmontieren. Bei der Voreinst. **nouser** ist dies nur `root` erlaubt. Die Option **user** ist für oft zu wechselnde Medien wie CD-ROMs oder Disketten sinnvoll. Die Option **user** impliziert die Optionen **noexec**, **nosuid** und **nodev**.
ro,**rw**	Auf Dateisystem nur Lesen (**ro**) bzw. Lesen und Schreiben (**rw**) erlauben.

- *Dump* (5. Spalte): gibt an, ob Dateisystem vom Kommando **dump** zu sichern ist. Dieser Eintrag wird derzeit nur bei `ext2` bewertet. Üblicherweise wird hier für Systempartitionen 1 und für alle anderen 0 angegeben.
- *Check* (6. Spalte): gibt an, ob zu montierendes Dateisystem auf Konsistenz zu prüfen ist. Für Root-Directory sollte 1 und für alle anderen veränderlichen Dateisysteme 2 angegeben werden. Bei Dateisystemen, die keiner Prüfung bedürfen, wie CD-ROM, Windows-Partitionen, Disketten, `proc`, `swap` usw., ist hier 0 anzugeben.

Fehlt die 5. oder 6. Spalte, wird 0 hierfür angenommen.

Sollen nachträglich alle in `/etc/fstab` angegebenen Datenträger anmontiert werden, z.B. weil zwischenzeitlich einer abmontiert wurde oder aber beim Hochfahren keine Diskette im Laufwerk war, so muss nur **mount –a** aufgerufen werden. In diesem Fall werden alle in `/etc/fstab` angegebenen Datenträger anmontiert, soweit sie nicht schon montiert sind. Ein mögliches Aussehen der Datei `/etc/fstab` ist nachfolgend noch gezeigt:

```
# /etc/fstab
/dev/hda6    /              reiserfs  defaults                 1 2
/dev/hda1    /boot          ext2      defaults                 1 2
/dev/cdrom   /media/cdrom   auto      ro,noauto,user,exec      0 0
devpts       /dev/pts       devpts    defaults                 0 0
/dev/fd0     /media/floppy  auto      noauto,user,sync         0 0
proc         /proc          proc      defaults                 0 0
usbdevfs     /proc/bus/usb  usbdevfs  noauto                   0 0
/dev/hda2    /windows/C     ntfs      ro,noauto,user,umask=022 0 0
/dev/hda5    /windows/D     vfat      noauto,user              0 0
/dev/hdd4    /media/zip     auto      noauto,user              0 0
/dev/hda7    swap           swap      pri=42                   0 0
```

Die Datei /etc/mtab: Um sich alle momentan montierten Dateisysteme auflisten zu lassen, muss man nur **mount** ohne weitere Angaben aufrufen. **mount** liest dann die Datei `/etc/mtab`, in der alle aktuell montierten Dateisysteme angegeben sind. Zusätzlich befindet sich diese Information auch noch in der Datei `/proc/mounts`.

Anmontieren von Windows-Partitionen: Linux kann DOS- und Windows(3.1/9x/ME)-Partitionen des Dateisystemtyps `vfat` lesen und schreiben. Bei Windows(NT/2000/XP)-Partitionen des Dateisystemtyps NTFS ist in jedem Fall ein fehlerfreies Lesen möglich. Ein Schreiben ist zwar auch möglich, aber diese Funktion ist noch nicht ausgereift, weshalb man sie nicht verwenden sollte, um größeren Schaden zu vermeiden. Nachfolgend zwei typische Einträge in `/etc/fstab` zum Montieren von Windows-Partitionen:

```
# /etc/fstab
...
/dev/hda5   /windows/D   vfat  user              0 0
/dev/hda2   /windows/C   ntfs  ro,user,umask=022 0 0
```

Zum Anmontieren von Windows- und DOS-Dateisystemen stehen eine Vielzahl von Optionen zur Verfügung, von denen hier nur die drei folgenden kurz vorgestellt werden: **uid=*n*, gid=*n*, umask=*n***

Da Windows-Dateisysteme nicht das Unix-Zugriffskonzept (Eigentümer, Gruppe usw.) unterstützen, werden solche Dateisystem so montiert, dass alle Dateien `root` gehören. Bei `vfat`-Partitionen können die Dateien von allen Benutzern gelesen, aber nur von `root` geändert werden. Bei `ntfs`-Partitionen können die Dateien nur von `root` gelesen werden. Durch die Angabe einer User-ID bzw. Group-ID bei **uid=** bzw. **gid=** kann man festlegen, dass die Dateien des anmontierten Windows-Dateisystems nicht `root`, sondern eben dem angegebenen Benutzer und der angegebenen Gruppe gehören. Mit **umask=** kann man einen Oktalwert festlegen, der ein invertiertes Zugriffsrechtemuster für alle Dateien des anmontierten Windows-Dateisystems festlegt. So legt z.B. **umask=0** fest, dass jeder Benutzer alle Dateien lesen, schreiben und ausführen kann (`rwxrwxrwx`). Eine Angabe wie **umask=022** legt für alle Dateien das Zugriffsrechtemuster `rwxr-xr-x` fest, so dass alle Benutzern die Dateien lesen und ausführen können, aber nur der Eigentümer sie verändern darf.

Anmontieren von CD-ROMs und DVDs: CDROM- und DVD-Laufwerke werden weitgehend wie andere Laufwerke (Disketten, Festplatten usw.) anmontiert, nur dass als Dateisysmtyp die Kennung **iso9660** oder **udf** anzugeben ist. Typische Aufrufe, um eine CD bzw. eine DVD anzumontieren, sind:

- **mount -t iso9660 -o ro /dev/hdc /cdrom**
  montiert das erste Laufwerk am zweiten IDE-Controller an das Directory `/cdrom`. Die Angabe **-o ro** legt fest, dass dieses Dateisystem *readonly* (nur lesbar) ist.
- **mount -t auto -o ro /dev/dvd /dvd**
  montiert das DVD-Laufwerk an das Directory `/dvd`.

Neben diesem direkten Montieren sind auch wieder entsprechende Einträge in `/etc/fstab` möglich, wie z.B.:

```
# /etc/fstab
...
/dev/cdrom    /cdrom    auto    ro,noauto,user,exec 0 0
/dev/dvd      /dvd      auto    ro,noauto,user,exec 0 0
```

auto (in der dritten Spalte) legt hier fest, dass das System automatisch erkennt, ob die CD bzw. DVD das **udf**-Format (*universal disk format*) oder aber das **iso9660**-Format verwendet. **noauto** (in der vierten Spalte) legt hier fest, dass beim Systemstart nicht versucht wird, CDs bzw. DVDs anzumontieren, was auch sinnvoll ist, da sich oft zu diesem Zeitpunkt keine CD bzw. DVD im entsprechenden Laufwerk befindet. **user** legt fest, dass jeder Benutzer das CD/DVD-Laufwerk anmontieren darf und **exec** erlaubt die Ausführung von Programmen, die sich auf der CD bzw. DVD befinden.

Bei solchen Einträgen in `/etc/fstab` reichen dann Aufrufe wie **mount /cdrom** oder **mount /dvd** zum Anmontieren aus. Wichtig ist in jedem Fall, dass man vor dem Entnehmen der CD-ROM aus dem Laufwerk diese immer zuerst wieder mit **umount** abmontiert, wobei kein Zugriff darauf mehr vorliegen darf.

Anmontieren von Netzwerk-Dateisystemen: Die beiden folgenden Einträge in `/etc/fstab` sollen verdeutlichen, wie man Unix-/Windows-Netzwerk-Directories (NFS, Samba) anmontieren kann:

```
# /etc/fstab
...
hilbert.mathe.de:/nfs  /nfs      nfs    noauto,user,exec 0 0
//gauss/c              /windows  smbfs  noauto,user,exec 0 0
```

Anmontieren von Diskettenlaufwerken: Die Diskettenlaufwerke werden über `/dev/fd0` (erstes Diskettenlaufwerk) und `/dev/fd1` (zweites Diskettenlaufwerk) angesprochen. Befindet sich auf einer Diskette (im ersten Laufwerk) ein `ext2`-Dateisystem, so kann dieses z.B. an das Directory `/A` wie folgt montiert werden:

`root#` **mkdir /A** (Enter) [*Nur, wenn Directory noch nicht existiert*]
`root#` **mount -t ext2 /dev/fd0 /A** (Enter) [*Anmontieren der Floppy als ext2-Dateisyst. an Directory /A*]

Statt `/A` wird häufig auch das Directory `/floppy` verwendet. Nun kann auf die Diskette ganz einfach zugegriffen werden, da sie nun im Root-Directorybaum integriert ist. Befindet sich auf der Diskette ein DOS-Dateisystem, so müßte folgender Aufruf angegeben werden:

`root#` **mount -t msdos /dev/fd0 /A** (Enter) [*Anmontieren der Floppy als msdos-Dateisyst. an Directory /A*]

Nun können auf `/A` mit gewissen Einschränkungen (wie z.B. begrenzte Dateinamenlänge: 8 für Name plus 3 für Dateinamenerweiterung, keine Links usw.) Linux-Dateien kopiert werden, die dann unter DOS wieder gelesen werden können. Mit den **mtools**-Kommandos steht eine andere Gruppe von Kommandos zur Verfügung, die eigens zum leichten Umgang mit MS-DOS-Partitionen unter Linux konzipiert wurden. Soll das System selbst ermitteln, welches Dateisystem sich auf der Diskette befindet, muss man nur folgendes aufrufen:

`root#` **mount -t auto /dev/fd0 /A** (Enter) [*Anmontieren der Floppy mit automat. Dateisystem-Erkennung*]

Hat man z.B. den folgenden Eintrag in `/etc/fstab`:

```
# /etc/fstab
...
/dev/fd0  /floppy   auto    noauto,user,sync 0 0
```

muss man zum Anmontieren des Diskettenlaufwerks nur **mount /floppy** aufrufen. Wichtig ist in jedem Fall, dass man vor dem Entnehmen der Diskette aus dem Laufwerk diese immer zuerst mit **umount** abmontiert, wobei kein Zugriff darauf mehr vorliegen darf.

Der Automounter: Der Kernel-Automounter montiert automat. bei einem Zugriff auf ein Dateisystem dieses im Hintergrund und montiert es auch automatisch wieder ab, wenn in einer festgelegten Zeitspanne nicht mehr darauf zugegriffen wird. Die beiden Voraussetzungen zur Verwendung des Automounters sind das installierte Paket `autofs` und die Unterstützung durch den Kernel. Danach sind folgende Schritte notwendig:

- Anlegen einer Datei `/etc/auto.master` durch `root`, wie z.B.:
  ```
  # /etc/auto.master
  # Unterhalb von /mnt sollen lokales CDROM und Floppy gemountet werden;
  # beide sollen nach 60 Sekunden automatisch abmontiert werden
  /mnt   /etc/auto.local   --timeout 60
  ```
- Anlegen der Konfigurationsdateien zu in `/etc/auto.master` erwähnten Mountpkten mit folg. Einträgen:
  Mountpunkt Mount-Optionen [*Rechner*]**:***Device*
  Mountpunkt bezeichnet das nicht existierende Directory, das unterhalb dem in `/etc/auto.master` genannten Mountpunkt zur Aufnahme des Dateisystems dient. Die *Mount-Optionen* entsprechen denen des Kommandos **mount** und *Device* ist bei lokalen Dateisystemen Name der Gerätedatei (Angabe eines Rechnernamens entfällt) und bei Dateisystemen, die von anderem Rechner importiert werden, der dortige Mountpunkt.
  ```
  # /etc/auto.local
  cdrom   -fstype=iso9660,ro,user  :/dev/cdrom
  floppy  -fstype=auto,rw,user     :/dev/fd0
  ```
  Hier lautet also der vollständige Mountpunkt für das CDROM-Laufwerk `/mnt/cdrom` und für das Disketten-Laufwerk `/mnt/floppy`.
- Die vom Automounter zu verwaltenden Dateisysteme müssen diesem nun bekannt gegeben werden. Hierzu sollte ein Skript **autofs** im Directory der Init-Skripte existieren. Dieses Skript wertet die oben beschriebenen Konfigurationsdateien aus, extrahiert die Parameter und ruft Kommando **automount** mit diesen auf, wie z.B.:
  `root#` **/sbin/init.d/autofs start** (Enter) oder `root#` **/etc/init.d/autofs start** (Enter)
  Aktueller Zustand des Automounters kann auch mittels des **autofs**-Skripts (**/.../autofs status**) erfragt werden.

Soll der Automounter nach jedem Systemstart zur Verfügung stehen, sind die entsprechenden Links in den jeweiligen Runleveln zu erzeugen. Ein erster Blick in das Directory `/mnt` zeigt, dass noch nichts montiert ist:

```
$ ls /mnt  (Enter)
$
```

Das Montieren des entsprechenden Dateisystems geschieht erst mit dem ersten Zugriff auf dieses:

```
$ ls /mnt/cdrom  (Enter)
.   allprog.txt    lcgi_hilf.txt   Linuxbgi   teil_i     teil_iii ......
$
```

Wartet man nun eine Minute, so ist die CDROM wieder automatisch abmontiert.

fdisk – Partitionieren von Festplatten

fdisk [*optionen*] [*gerätedatei*]

fdisk kann nur von `root` aufgerufen werden, um Festplatten zu partitionieren. Ist keine *gerätedatei* angegeben, bezieht sich **fdisk** auf die erste IDE-Platte (`/dev/hda`). Mit **fdisk –l** kann man sich Liste aller Partitionen auf allen IDE- und SCSI-Platten ausgeben lassen. **fdisk** sollte nicht auf Festplatten angewendet werden, die anmontiert sind. Nachfolgend einige wichtige Tastenkürzel zur interaktiven Bedienung von **fdisk**, die immer mit (Enter) abzuschließen sind:

M	(*menu*) Kurze Übersicht der zur Verfügung stehenden Tastenkürzel anzeigen
P	(*print*) Liste der Partitionen zu der aktuell ausgewählten Festplatte anzeigen
N	(*new*) Neue Partition anlegen
D	(*delete*) Partition löschen
W	(*write*) Partitionstabelle ändern
V	(*verify*) Partitionstabelle überprüfen
L	(*list*) Partitions-ID-Nummer anzeigen
T	Partitionstyp ändern (z.B. für Swap-Partitionen)
Q	(*quit*) **fdisk** beenden

cfdisk – Komfortables Partitionieren von Festplatten

cfdisk [*optionen*] [*gerätedatei*]

cfdisk ist ein komfortableres Kommando zum Partitionieren von Festplatten. Die Bedienung erfolgt über ein Menü und mit den Cursortasten. Nach dem Start zeigt **cfdisk** alle gefundenen Partitionen mit ihrer Größe an. Mit (Pfeil o) und (Pfeil u) kann dabei eine Partition und mit (Pfeil li) und (Pfeil re) kann ein Kommando ausgewählt werden. Daneben stehen noch die folgenden Tastenkürzel zur Verfügung:

H	Online-Hilfe anzeigen
N	Neue Partition anlegen
D	Ausgewählte Partition löschen
T	Partitionstyp ändern (z.B. für Swap-Partitionen)
(Shift)-W	Partitionstabelle ändern
Q	**cfdisk** beenden, ohne die Partitionstabelle zu ändern

parted – Anlegen, Verkleinern, Vergrößern und Verschieben von Partitionen

parted [*optionen*] [*gerätedatei*]

Das Kommando **parted** ist mächtiger als die beiden Kommandos **fdisk** und **cfdisk**, da es nicht nur das Anlegen von Partitionen ermöglicht, sondern auch z.B. das Verkleinern, Vergrößern oder Verschieben von Partitionen.

fdformat – Formatieren von Disketten

fdformat [**-n**] *gerätedatei*

fdformat führt eine Low-Level-Formatierung einer Diskette durch. Der Diskettentyp, die Kapazität und das Laufwerk werden durch die entsprechende *gerätedatei* ausgewählt.

Auf der Diskette wird kein Dateisystem eingerichtet. Dazu stehen die Kommandos **mkfs** für Linux/Unix-Dateisysteme bzw. **mformat** für MS-DOS-Dateisysteme zur Verfügung. Die roh formatierte Diskette kann aber auch mit den Kommandos **tar** oder **dd** direkt beschrieben werden. Die Angabe der Option **-n** verhindert eine anschließende Überprüfung (Verifizierung) der Diskette auf Fehler. Verwendet man **fdformat**, ohne vorher auf die Diskette zugegriffen zu haben, wird eventuell der Fehler "*no such device*" gemeldet. In diesem Fall muss zunächst einmal mit **mdir a:** bzw. **mdir b:** auf die Diskette zugegriffen werden.

dd - Konvertieren und Kopieren von Datei(systemen) und Partitionen

dd [*option*=*wert*] ....

Das Kommando **dd** kopiert die angegebene Eingabedatei (**if**=*datei*) auf die angegebene Ausgabedatei (**of**=*datei*), was auch eine Gerätedatei sein kann. Beim Kopieren führt **dd** abhängig von den angegebenen Optionen entsprechende Konvertierungen (wie z.B. von ASCII nach EBCDIC) durch. Die Voreinstellung von **dd** ist, dass es von der Standardeingabe liest und auf die Standardausgabe ausgibt.

if=*datei*	Verwendet *datei* als Eingabedatei; kann auch eine Gerätedatei sein.
of=*datei*	Verwendet *datei* als Ausgabedatei; kann auch eine Gerätedatei sein.
ibs=*n*	liest die Eingabe in Blöcken von *n* Bytes.
obs=*n*	schreibt die Ausgabe in Blöcken von *n* Bytes.
bs=*n*	setzt sowohl die Eingabe wie die Ausgabe auf Blöcke von *n* Bytes; schaltet die Option **ibs**= und **obs**= aus.
cbs=*n*	legt die Größe des Konvertierungspuffers auf *n* Bytes fest.
skip=*n*	überspringt *n* Eingabeblöcke, bevor es mit dem Kopieren beginnt.
seek=*n*	überspringt *n* Blöcke in der Ausgabedatei, bevor es mit dem Kopieren beginnt.
count=*n*	kopiert nur *n* Eingabeblöcke.
conv=ascii	konvertiert EBCDIC -> ASCII
conv=ebcdic	konvertiert ASCII -> EBCDIC
conv=ibm	konvertiert ASCII -> EBCDIC (verwendet dabei eine andere Abbildung)
conv=lcase	konvertiert Großschreibung in Kleinschreibung
conv=ucase	konvertiert Kleinschreibung in Großschreibung
conv=swap	vertauscht die Bytes jedes Byte-Paares; manche Maschinen speichern 2-Byte-Wörter mit dem höheren Byte zuerst und andere mit dem niedrigeren Byte zuerst.
conv=noerror	Bearbeitung wird beim Auftreten von Fehlern nicht beendet.

Ein Komma kann verwendet werden, wenn mehr als eine Konvertierung erwünscht ist, z.B. **conv=ascii,ucase**.

Das Kommando **dd** ermöglicht das Übertragen von Daten zwischen verschiedenen Speichermedien (Festplatte, Diskette usw.). Es kann nicht nur einzelne Dateien kopieren, sondern auch direkt auf Gerätedateien zugreifen. Mit **dd** können z.B. ganze Festplatten bzw. Festplattenpartitionen kopiert werden, wobei auf dem Ziel-Speichermedium kein Dateisystem eingerichtet sein muss.

- **dd if=/dev/hdb of=/boot/bootsektor.bak bs=512 count=1**
  überträgt den Bootsektor der zweiten IDE-Platte in die Backup-Datei `bootsektor.bak`.
- **dd if=/vmlinuz of=/dev/fd0**
  kopiert den Linux-Kernel direkt in die ersten Sektoren der Diskette, die dann als Bootdiskette verwendet werden kann.

8.2 Einrichten und Prüfen von Dateisystemen

mkfs – Einrichten von Dateisystemen

mkfs [*optionen*] *gerätedatei* [*blöcke*]

Mit dem Kommando **mkfs** kann auf einer zuvor formatierten Diskette oder auf einer partionierten Festplatte ein Dateisystem eingerichtet werden. Das Kommando **mkfs** kann nur vom Superuser (`root`) ausgeführt werden. Je nach dem angegebenen Dateisystemtyp ruft **mkfs** eines der Kommandos **mkfs.minix**, **mke2fs**, **mkxfsi**, **mkreiserfs**, **mkfs.xiafs** usw. auf. Für *gerätedatei* muss entweder der Pfad der entsprechenden Gerätedatei (wie z. B. `/dev/fd0`, `/dev/hda1`, `/dev/sdb2` usw.) oder der Montierpunkt des entsprechenden Dateisystems (wie z.B. `/A`, `/usr`, `/home` usw.) angegeben werden. Mit *blöcke* kann die Anzahl der Blöcke angegeben werden, die im entsprechenden Dateisystem zu benutzen sind. Die wichtigste Option bei **mkfs** ist:

-t *fstyp*	gibt den Typ des Dateisystems an. Für *fstyp* kann dabei u.a. folgendes angegeben werden: **minix**, **ext2** (Voreinstellung unter Linux), **xiafs** (Alternative zu ext2), **msdos** (MS-DOS-Dateisystem; entspricht einem **mformat**-Aufruf) usw. angegeben werden. Ist die Option **-t** nicht angegeben, versucht **mkfs**, den Dateisystemtyp aus der Datei `/etc/fstab` zu ermitteln. Die Option **-t** muss immer als erste Option angegeben werden.

Die weiteren Optionen sind abhängig vom angegebenen *fstyp*.

mke2fs - Anlegen eines ext2- bzw. ext3-Dateisystems

mke2fs [*optionen*] *gerätedatei* [*blöcke*]

-b *n*	legt die Blockgröße für das Dateisystem fest (Voreinstellung ist 1024 Bytes). *n* muss dabei eine Zweierpotenz größer oder gleich 1024 sein (1024, 2048, 4096).
-c	führt vor dem Einrichten des Dateisystems einen Test auf dem Datenträger durch, um defekte Blöcke aufzufinden.
-N *n*	legt die maximal mögliche Anzahl von Dateien (Inodes) für dieses Dateisystem fest.
-m *n*	legt fest, wie viel Prozent des Datenträgers für Daten des Superusers (`root`) reserviert werden sollen (Voreinst.: 5%). Dies ermöglicht es dem Superuser auch dann noch zu arbeiten, wenn Dateisystem bereits für andere Benutzer voll ist.

Da das **ext3**-Dateisystem weitgehend kompatibel zu **ext2** ist, sind die Kommandos zur Administration dieses Dateisystems identisch zu denen des **ext2**-Dateisystems. Zum Einrichten eines **ext3**-Dateisystem steht deshalb auch das Kommando **mke2fs** zur Verfügung, wobei man beim Aufruf von **mke2fs** allerdings die Option **-j** angeben muss, wie z.B.: `root#` **mke2fs -j /dev/hda5**

Um **ext2**-Dateisystem in **ext3**-Dateisystem umzuwandeln, muss man nur **tune2fs** mit Option **-j** aufrufen, wie z.B.:
`root#` **tune2fs -j /dev/hda4** Dadurch wird zum **ext2**-Dateisystem eine Jorunaling-Datei hinzugefügt.

ext3 kennt folg. *Journaling-Modi*, von denen man einen beim Anmontieren des Dateisystems auswählen kann:

data=journal	Alle Änderungen werden zweimal gespeichert: zuerst im Journal und dann in entspr. Datei. Dies ermöglicht bei Rechnerabsturz auch das Wiederherstellen von Dateien, bei denen die Änderungen bereits im Journal, aber noch nicht in die Datei eingetragen wurden. Jedoch ist **ext3** in diesem Modus auch deutlich langsamer.
data=ordered	Dies ist die Voreinstellung. Im Journal werden nur so genannte Metadaten gespeichert. Metadaten sind nur Informationen über betroffene Dateien, aber nicht die Inhalte der Dateien. Im Journal werden Dateien erst dann als *committed* markiert, wenn sie vollständig auf Festplatte geschrieben wurden. Hier kann Dateisystem nach Rechnerabsturz sehr schnell wieder in einen konsistenten Zustand gebracht werden, da das Journal alle nicht ordentlich gespeicherten Dateien enthält. Allerdings sind die nicht gespeicherten Daten verloren.
data=writeback	Dieser Modus unterscheidet sich vom `ordered`-Modus dahingehend, dass das Journal nicht unbedingt den aktuellen Zustand des Dateisystems widerspiegelt, da Einträge im Journal bereits als *commited* markiert werden, bevor die entsprechenden Speicheroperationen wirklich abgeschlossen sind. Dadurch ist dieser Modus der schnellste, wobei jedoch nach einem Systemabsturz Dateien noch alte Daten enthalten können. Der Zustand des Dateisystems entspricht in solchen Fällen einen mit **fsck** reparierten **ext2**-Dateisystem, wobei **ext3** hierbei jedoch bedeutend schneller ist.

Der folgende Aufruf bewirkt, dass ein **ext3**-Dateisystem im `journal`-Modus anmontiert wird:

`root#` **mount -t ext3 -o data=journal /dev/hda3 /data**

Das Journaling-Verfahren kann auch in `/etc/fstab` festgelegt werden:

```
/dev/hda3     /data     ext3     data=journal  1 0
```

Für die regelmäßige Aktualisierung der Journaling-Datei ist der Dämonprozeß **kjournald** zuständig, der automatisch beim Anmontieren eines **ext3**-Dateisystems gestartet wird.

mkswap / swapon / swapoff - Einrichten von Swap-Partitionen und -Dateien

mkswap *gerätedatei* [*Einrichten einer Swap-Partition*]
mkswap *datei* [*Einrichten einer Swap-Datei*]
swapon *(geräte-)datei* [*Aktivieren einer Swap-Partition bzw. -Datei*]
swapoff *(geräte-)datei* [*Deaktivieren einer Swap-Partition bzw. -Datei*]

Um Speicherengpässe evtl. etwas abzumildern, kann man Teil der Festplatte als "*Verlängerung des RAM*" (einen so genannten Swap) einrichten. Das System kümmert sich dann darum, dass die Daten des jeweils benötigten Programms im physischen RAM liegen und Speicherseiten, die im Augenblick keine Verwendung finden, im Bereich des "*virtuellen RAM (des Swaps)*" auf Festplatte gehalten werden. Swap-Partitionen werden üblicherweise während der Installation eingerichtet. Dies kann man dann an entspr. Zeilen in `/etc/fstab` erkennen, wie z.B.:

```
# /etc/fstab
...
/dev/hda7     swap     swap     pri=42 0 0
```

Die Option `pri` legt die Priorität für diese Swap-Partition (bei mehreren) fest.

Informationen zu Swap-Partitionen kann man sich mit dem Kommando **free** anzeigen lassen:

Einrichten einer Swap-Partition:

Um eine Swap-Partition einzurichten, sind die folgenden Schritte unter Linux notwendig:

1. Zuerst muss die entspr. Partition als Swap-Bereich eingerichtet werden. Hierzu verwendet man **mkswap**, dem als Argument die Partition übergeben wird. Das Kommando verwendet den gesamten verfügbaren Speicher, maximal jedoch 128 MB. Um größere Partitionen vollständig zu nutzen, muss Option **-v1** verwendet werden. Durch Angabe der Anzahl von 1k-Blöcken als letztes Argument kann die Größe weiter beschränkt werden.
   `root#` **mkswap /dev/hdb1** (Enter) [*hier hat /dev/hdb1 100 MByte*]
   `Swapbereich Version 0 mit Größe 104857600 Bytes wird angelegt`
   `root#`
2. Nun existiert zwar die Swap-Partition, aber diese ist dem System noch nicht bekannt. Dem System muss die Existenz erst mit dem Kommando **swapon** mitgeteilt werden:
   `root#` **swapon /dev/hdb1**
   Um diesen Befehl nicht immer von Hand ausführen zu müssen, kann man eine Zeile wie die folgende in `/etc/fstab` eintragen:
   ```
   # /etc/fstab
   ...
   /dev/hdb1     swap     swap     sw 0 0
   ```
3. Ein aktivierter Swap-Bereich kann mit **swapoff** aus dem laufenden System entfernt werden:
   `root#` **swapoff /dev/hdb1**

Einrichten einer Swap-Datei

Eine Swap-Datei sollte allerdings nur als Notlösung in Betracht gezogen werden, wenn z.B. keine Partition für den Swap mehr zur Verfügung steht. Ein Zugriff auf eine Datei erfordert immer den Umweg über das Dateisystem und ist somit langsamer als ein Zugriff auf eine Swap-Partition. Zunächst muss man eine Datei der gewünschten Größe anlegen, wie z.B.:

`root#` **dd if=/dev/zero of=/dev/meine_swap_datei bs=1024 count=102400** (Enter)
`102400+0 Records ein`
`102400+0 Records aus`

Hier wurde mit **dd** eine 100 MByte große Datei `meine_swap_datei` im Directory `/dev` (üblicherweise das Directory für Swap-Dateien) angelegt und mit Nullen beschrieben. Nun sind noch folg. Aufrufe erforderlich:

`root#` **mkswap /dev/meine_swap_datei 102400** (Enter)
`Swapbereich Version 1 mit der Größe 104853504 Bytes wird angelegt`
`root#` **sync** (Enter) [*alle Puffer leeren und auf Festpl. speichern*]
`root#` **swapon -v /dev/meine_swap_datei** (Enter)
`swapon für /dev/meine_swap_datei`

Swap-Dateien können wie swap-Partitionen in `/etc/fstab` aufgenommen werden:

```
# /etc/fstab
...
/dev/meine_swap_datei     none     swap     sw 0 0
```

mkreiserfs - Anlegen eines ReiserFS-Dateisystems

mkreiserfs [*optionen*] *gerätedatei* [*blöcke*]

Um ein **reiserfs**-Dateisystem anzulegen, steht das Kommando **mkreiserfs** zur Verfügung, wie z.B.:

```
root# mkreiserfs /dev/hdb3
```

Das Journaling-Verfahren des reiserfs-Dateisystems entspricht dem **ext3**-Modus `data=writeback`. Der folgende Aufruf bewirkt, dass ein reiserfs-Dateisystem anmontiert wird:

```
root# mount -t reiserfs /dev/hdb3 /data
```

Natürlich kann man auch wieder einen entsprechenden Eintrag in `/etc/fstab` vornehmen, wie z.B.:

```
# /etc/fstab
...
/dev/hdb3     /data     reiserfs    defaults  1 2
```

Es ist zu bechten, dass sowohl der Kernel als auch die Daten des Bootmanagers nicht auf einer mit **mkreiserfs** formatierten Partition liegen, da der Bootmanager dieses nicht lesen kann.

badblocks - Physikalisches Prüfen eines Speichermediums

badblocks [*optionen*] *gerätedatei* [*startblock*]

Um die physische Unversehrtheit eines Speichermediums zu überprüfen, steht das Kommando **badblocks** zur Verfügung, das nach defekten Blöcken auf einer Diskette oder Festplatte sucht. Das Kommando **badblocks** kann nur vom Superuser (`root`) ausgeführt werden.

Für *gerätedatei* muss der Pfad der entsprechenden Gerätedatei (wie z. B. `/dev/fd0`, `/dev/hda1`, `/dev/sdb2` usw.) angegeben werden. Optional kann mit *startblock* der erste zu testende Block festgelegt werden. Nachfolgend einige wichtige Optionen zum Kommando **badblocks**:

-b *blockgröße*	legt die Anzahl der Bytes je Block fest. Dies ist nur notwendig, wenn beim Einrichten des Dateisystems eine von der Voreinstellung abweichende Größe gewählt wurde.
-o *datei*	eine Liste der beschädigten Blöcke wird in die Datei *datei* geschrieben, die von anderen Programmen zum Einrichten eines Dateisystems verwendet werden kann, damit diese die beschädigten Blöcke nicht verwenden.
-s	zeigt das Fortschreiten der Überprüfung an.

Als Beispiel soll eine 1.44 MByte-Diskette überprüft werden. Das Ergebnis des Tests wird in einer Datei `defekt` gespeichert:

```
root# badblocks -s -o defekt /dev/fd0 1440
```

fsck - Prüfen und Reparieren von Dateisystemen

fsck [*optionen*] [*gerätedatei*]

Je nach Dateisystemtyp ruft **fsck** eines der Kommandos **fsck.ext2** oder **fsck.minix** auf. Ein ReiserFS-Dateisystem kann mit dem Kommando **reiserfsck** überprüft werden. Die beiden Kommandos **fsck** und **reiserfsck** können nur vom Superuser (`root`) ausgeführt werden.

fsck sollte immer auf nicht montierte oder auf als *read-only* montierte Dateisysteme angewendet werden, da das Programm mitunter das Dateisystem modifiziert, ohne dass das System dies realisiert. Ein Systemabsturz könnte dann die Folge sein. Um defekte Dateisysteme frühzeitig zu erkennen und wieder in einen konsistenten Zustand zu bringen, findet üblicherweise beim Systemstart automatisch der Aufruf **fsck -a -A** statt. Die Option **-A** veranlasst das Kommando, alle in der Datei `/etc/fstab` als "*zu testen*" markierten Dateisysteme zu überprüfen. Die Option **-a** bewirkt eine automatische Reparatur, sofern dies möglich ist.

8.3 Weitere Dateisystem-Kommandos

dumpe2fs – Informationen zu einem ext2-/ext3-Dateisystem

dumpe2fs *gerätedatei*

dumpe2fs gibt viele interne Informationen über den Zustand eines ext2- bzw. ext3-Dateisystems aus.

tune2fs – Ändern der Systemparameter eines ext2-/ext3-Dateisystems

tune2fs [*optionen*] *gerätedatei*

Mit dem Kommando **tune2fs** können nachträglich verschiedene Systemparameter eines ext2- bzw. ext3-Dateisystems geändert werden.

-j	wandelt ein ext2-Dateisystem in ein ext3-Dateisystem um.
-c *n*	legt fest, nach wie vielen mount-Operationen die Partition beim Booten mit **fsck** zu überprüfen ist.
-i *n*	legt fest, nach wie vielen Tagen die Partition beim Booten mit **fsck** zu überprüfen ist.
-m *n*	legt fest, wieviel Prozent des Datenträgers für Daten des Superusers (`root`) reserviert werden sollen.

tune2fs darf nur auf nicht montierte Partitionen angewendet werden.

sync – Schreiben von gepufferten Daten auf Festplatten

sync

Der Aufruf dieses Kommandos bewirkt, dass alle gepufferten Daten wirklich auf den entsprechenden Datenträger geschrieben werden. Dies ist nützlich, wenn man das System nicht mehr ordnungsgemäß mit Kommandos wie **shutdown**, **reboot**, **halt** oder aber mit der Tastenkombination (Strg)-(Alt)-(Entf) herunterfahren kann. In diesem Fall sollte man dieses Kommando **sync** aufrufen, bevor man den Rechner ausschaltet.

9 Benutzerkommunikation

wall - Nachrichten an alle Benutzer schicken

/etc/wall bzw. nur **wall**

wall liest den zu übermittelnden Text von der Standardeingabe (bis zur Eingabe von EOF) und schreibt dann den gelesenen Text auf alle Terminals der momentan angemeldeten Benutzer. Hiermit informiert üblicherweise der Superuser alle angmeldeten Benutzer über bevorstehende Änderungen des Systemzustands (wie z.B. Abschalten des Systems oder Veränderung des Laufzeitverhaltens für Benutzerprogramme wegen notwendiger Tests).
wall kann allerdings nicht nur der Superuser, sondern jeder Benutzer aufrufen; jedoch ist nur der Superuser in der Lage, die Zugriffsrechte anderer Benutzer-Terminals zu durchbrechen, wenn diese mit dem Kommando **mesg n** für das Schreiben durch "fremde" Benutzer gesperrt wurden.

write - Nachrichten an andere Benutzer senden

write *login-name* [*terminal-name*]

Mit dem Kommando **write** ist es möglich, dass ein Benutzer Mitteilungen an die Terminals anderer Benutzer sendet. **write** kann dabei benutzt werden, um entweder Informationen einkanalig zu verschicken oder aber zwei Benutzern einen Dialog miteinander führen zu lassen. Nach dem **write**-Aufruf wird die Meldung
Message from absender on rechner-name (tty..) [datum] ...
auf dem Terminal von *login-name* ausgegeben, falls dorthin eine Verbindung hergestellt werden konnte. Dem Sender wird erfolgreicher Verbindungsaufbau mit zweimaligen Klingeln der Terminalglocke angezeigt. Nun kann der Sender Text eingeben; jede mit (Enter) abgeschlossene Zeile wird am Empfängerterminal ausgegeben. Ende der Nachricht zeigt der Sender mit Eingabe von (Strg)-D (EOF) an; dies wird am Empfänger-Teminal mit *<EOT>* oder (*end of message*) angezeigt und danach wird Verbindung abgebrochen. Der zu übermittelnde Text kann auch in Datei geschrieben werden und mit Eingabeumlenkung beim Aufruf gesendet werden.
Ist ein Dialog zwischen zwei Benutzern erwünscht, so müßte der Empfänger der Nachricht
Message from absender-login-name on rechner-name (tty..) [datum] ...
seinerseits den Befehl **write** *absender-login-name* abgeben, um eine "Schreibleitung" zum Absender aufzubauen.
Wenn **write** an einen Benutzer abgegeben wird, der an mehr als an ein Terminal arbeitet, muss zusätzl. zum *login-name* noch der *terminal-name* (z.B. `ttyic`) angegeben werden, zu dem eine "Schreibleitung" herzustellen ist. Ist in diesem Fall kein *terminal-name* angegeben, wird eine "Schreibleitung" zum Terminal-Namen aufgebaut, der vom entspr. Benutzer zuletzt benutzt wurde und der Schreiben erlaubt. Der Sender erhält dann folgende Meldung:
login-name is logged on more than one place
You are connected to "terminal-name"; Other locations are: terminal-name1.....
Wird bei der Eingabe der zu übermittelnden Nachricht als erstes Zeichen ein **!** in einer Zeile angegeben, so wird bei manchen **write**-Versionen der Rest der Zeile als ein Unix-Kommando interpretiert, das ausgeführt wird.
Ein Benutzer kann sein Terminal mit Kommando **mesg** für einen **write**-Befehl eines anderen Benutzers sperren.

talk - Interaktive Kommunikation mit anderen Benutzern

talk *login-name* [*terminal-name*]

talk ist ein komfortableres Kommando als **write**. Die Aufrufsyntax von **talk** ist identisch zu **write**. Anders als bei **write** können mit **talk** auch Benutzer auf anderen Systemen in einem lokalen Netz erreicht werden. Dazu muss zusätzlich zum Benutzernamen noch der Systemname angegeben werden, wobei die beiden Namen mit @ voneinander zu trennen sind, wie z.B. **micha@rosenrot** für den Benutzer **micha** auf dem System **rosenrot**. **talk** meldet sich beim Gesprächspartner eventuell mehrmals mit
Message from Talk_Daemon@rosenrot at 15:52 .....
talk: connection requested by egon@wiesengruen.
talk: respond with: talk egon@wiesengruen
und erwartet die Annahme des Gesprächs. Während des Verbindungsaufbaus erscheinen auf dem Terminal des Senders nacheinander die beiden folgenden Meldungen. *[No connection yet][Waiting for your party to respond]*
Meldet sich der Gesprächspartner nicht, erscheint immer wieder die Meldung *[Ringing your party again]*
bis eine Verbindung aufgebaut werden konnte oder der Sender mit (Strg)-C **talk** abbricht.
Bei einem erfolgreichen Verbindungsaufbau teilt **talk** den Bildschirm in zwei Hälften, in denen unabhängig voneinander jeweils die Eingaben und die Antworten des Gesprächspartners angezeigt werden. **talk** wird auf beiden Seiten beendet, wenn einer der beiden Gesprächspartner das Kommando mit (Strg)-C abbricht.

notify - Sofortige Meldung neu angekommener Mail

notify [**-y**] [**-n**]

Gib man die Option **-y** an, so wird jede neu angekommene Mail nicht nur sofort dem betreffenden Benutzer gemeldet, wenn er noch angemeldet ist und sein Terminal nicht mit **mesg -n** gesperrt wurde, sondern es werden zusätzlich noch die ersten Zeilen dieser Mail am Bildschirm ausgegeben. Mit **notify -n** kann die sofortige Benachrichtigung wieder abgeschaltet werden.

Ruft man nur **notify** ohne Angabe von **-n** oder **-y** auf, so meldet es, ob momentan sofortige Benachrichtigung ein- oder ausgeschaltet ist.

vacation - Anrufbeantworter einrichten

vacation [option(en)]

Seit System V.4 wird (für **rmail** als MTA) das Kommando **vacation** angeboten, mit dem sich eine Art Anrufbeantworter für elektronische Mail einrichten läßt. Neu eintreffende Mail wird dazu über den *Forwarding-Mechanismus* an ein Programm weitergeleitet, das den Eingang der Mail bestätigt und die Mail in der Mailbox des Benutzers ablegt. Zusätzlich schreibt das Programm den Namen des Absenders in die Datei `$HOME/.maillog`, so dass der gleiche Absender die automatische Antwort auf seine Mail nur einmal erhält.

-l *logfile*	Datei *logfile* und nicht `$HOME/.maillog` zur Protokollierung der Namen verwenden.
-M *antwortdatei*	Der Standardtext des Anrufbeantworters ist in der Datei `/usr/share/lib/mail/std_vac_msg` enthalten. Mit dieser Option kann ein Name einer anderen Datei (*antwortdatei*) festgelegt werden, in dem ein eigener Text steht.
-m *datei*	Normalerweise wird die ankommende Mail in der Mailbox des betreffenden Benutzers abgelegt. Mit dieser Option kann eine andere *datei* festgelegt werden, in der ankommende Mail abzulegen ist.
-f *forward-id*	ankommende Mail nicht nur in der Mailbox des Benutzers speichern, sondern auch an den Benutzer mit der Kennung *forward-id* weiterleiten.
-i *forward-id*	ankommende Mail nicht in der Mailbox des Benutzers speichern, sondern sofort an den Benutzer mit der Kennung *forward-id* weiterleiten.
-d	Tagesdatum am Ende der *datei* schreiben, die bei **-m** angegeben ist.

Bei Verwendung von **sendmail** als MTA kann eine automatische Beantwortung auch mit dem Programm **vacation** eingerichtet werden. Dazu erstellt man im Home-Directory eine Datei `.forward` z.B. mit folgenden Inhalt:

```
\egon,"|/usr/ucb/vacation egon"
```

Danach muss man noch mit dem Aufruf **/usr/ucb/vacation -I** eine Protokolldatei initialisieren, in der **vacation** die Namen von bereits benachrichtigten Benutzern festhält. Der Antworttext kann in der Datei `$HOME/.vacation.msg` angegeben werden. Die automatische Beantwortung läßt sich durch das Entfernen der Datei `.forward` (im Home-Directory) wieder aufheben.

mail / mailx – Klassische Mail-Programme von Unix

mail [*optionen*] [*login-name(n)*] oder **mailx** [*optionen*] [*login-name(n)*]

- Ankommende Briefe werden in *Mailbox*-Datei `/var/mail/`*login* oder `/usr/mail/`*login* oder `/usr/spool/mail/`*login* oder `/var/spool/mail/`*login* abgelegt; *login* ist Loginname des Empfängers.
- Wird **mail**/**mailx** ohne Argumente aufgerufen, ermöglicht es Lesen der Briefe, die sich in *Mailbox* befinden.
- Jeder gelesene Brief wird automatisch aus der *Mailbox* entfernt und in einer so genannten *Sekundär-mailbox* (Datei `mbox`, die sich normalerweise im Home-Directory befindet) abgelegt.
- **mail**/**mailx** kennt zwei Arbeitszustände:
  - *Eingabemodus*: Hier kann Brieftext eingegeben werden. mail/mailx-Kommandos (mit Tilde) auch mögl.
  - *Kommandomodus:* Hier können mail/mailx-Kommandos eingegeben werden.
- Ist *login-name(n)* angegeben, heißt dies "*Senden eines Briefs*". Für *login-namen* sind Loginnamen der Empfänger (mit Leer- bzw. Tabulatorzeichen getrennt) anzugeben.
- Sind keine *login-name(n)* angegeben, so bedeutet dies "*Lesen von angekommenen Briefen*".

-f [*datei*]	**mail**/**mailx** liest Briefe aus Mailbox *datei* und nicht aus voreingest. Mailbox.
-s"*Thema*"	Das hier angegebene Thema des Briefes entspricht der deutschen *Betrifft:*-Angabe und wird dem Empfänger als *Subject:* Thema des Briefes vor dem eigentlichen Brieftext übermittelt.

Senden von Briefen

Nach einem Aufruf wie **mailx** *login-name(n)* oder **mail** *login-name(n)* liest **mailx** bzw. **mail** den zu übermittelnden Brieftext von der Standardeingabe. Nach Eingabe der *Subject*-Zeile (eventuell leer) befindet sich **mailx**/**mail** im Eingabemodus und der Brieftext kann eingegeben werden. Die Eingabe des Brieftextes kann mit `~.` als einzige Zeichen einer Zeile beendet werden. Danach wird der Brieftext abgeschickt.
Im Eingabemodus können auch **mailx**/**mail**-Kommandos aufgerufen werden. Jedes dieser Kommandos besteht aus einem Buchstaben. Diesem ist ein ~ voranzustellen (gilt nur im Eingabemodus); zudem muss dieses Zeichenpaar am Anfang einer Zeile angegeben sein. Nachfolgend einige mail/mailx-Kommandos für den Eingabemodus:

~!*kdo*	angegebene Unix-Kommando *kdo* wird ausgeführt und danach wird wieder in Eingabemodus zurückgekehrt.
~.	bewirkt das Verlassen des Eingabemodus und führt zum Abschicken des eingegebenen Briefes.
~?	gibt eine Zusammenfassung der möglichen ~ -Kommandos aus.
~b *login-name(n)*	ermöglicht es, neue Adressaten zur `Bcc:`-Liste hinzuzufügen.
~c *login-name(n)*	ermöglicht es, neue Adressaten zur `Cc:`-Liste hinzuzufügen.
~e	ruft den Editor auf, der in der Variablen `EDITOR` angegeben ist; meist der **vi**.
~v	ruft den Editor auf, der in der mail/mailx-Variablen `VISUAL` angegeben ist; meist der **vi**.
~f [*liste*]	kopiert die mit *liste* ausgewählten Briefe in den Brieftext.
~h	zeigt nacheinander alle Komponenten eines Briefkopfes mit bisherigem Inhalt an: `To:`, `Subject:`, `Cc:`, `Bcc:`. Bei jeder Komponente kann nun deren Inhalt geändert oder neuer Text hinzugefügt werden.
~m [*liste*]	kopiert die mit *liste* ausgewählten Briefe in Brieftext mit vorangestellten Tabzeichen vor jeder Zeile.
~M [*liste*]	wie **~m**, nur dass alle Kopfzeilen der betroffenen Briefe mit einkopiert werden.
~p	gibt den bisher eingegebenen Brieftext nochmals von Beginn an am Bildschirm aus.
~q	sofortiger Abbruch von **mail** bzw. **mailx**. Eingegebener Brieftext wird in `~/dead.letter` gesichert
~r *dateiname*	kopiert die Datei *dateiname* in den Brieftext ein.
~< **!***unix-kdo*	führt das Unix-Kommando *unix-kdo* aus und kopiert dessen Ausgabe in den Brieftext.
~s *string*	besetzt das `Subject:`-Feld mit den angegebenen *string*.
~t *login-name(n)*	ermöglicht es, neue Adressaten zur To:-Liste hinzuzufügen.
~w *dateiname*	sichert den Brieftext in der Datei *dateiname*.
~x	bewirkt sofortigen Abbruch von **mail**/**mailx**. Eingegebener Brieftext wird weder verschickt noch gesichert.

Lesen von angekommenen Briefen

Zum Umgang mit angekommenen Briefen bietet **mail** bzw **mailx** eine Vielzahl von Kommandos an. Die allgemeine Syntax der Kommandos im Kommandomodus ist: [*kommando*] [*liste*] (Enter) Wird kein *kommando* angegeben, so wird hierfür das Kommando `next` angenommen. Bei den Kommandonamen ist der hier fette Teil die kürzest mögliche Form der Angabe; allerdings kann dabei auch jede mögliche Zwischenform angegeben werden. So ist z.B. für das Kommando `next` die kürzest mögliche Form **n**; dieses Kommando kann also mit **n**, **ne**, **nex** oder **next** aufgerufen werden. Die *liste* legt dabei fest, für welche Briefe das angegebene Kommando auszuführen ist. Wenn ein Kommando eine *liste* zuläßt und es wird keine angegeben, so wird das Kommando für den aktuellen Brief ausgeführt. Bei der Ausgabe der Briefköpfe kennzeichnet **mail**/**mailx** den aktuellen Brief immer mit >.

Eine *liste* ist eine Liste von einzelnen mit Leerzeichen getrennten Angaben; jede einzelne Angabe spezifiziert dabei bestimmte Briefe. Als Angabe ist dabei möglich:

n bzw. **.**	Brief mit der Nummer *n* bzw. aktueller Brief
^	erster nicht als "gelöscht" markierter Brief
$ bzw. *****	letzter Brief bzw. alle Briefe
n-m	Briefe mit den Nummern *n* bis *m*
login-name	alle Briefe des Benutzers *login-name*
/text	alle Briefe, bei denen *text* in der Subject:-Zeile vorkommt
:*c*	alle Briefe vom Typ *c*, wobei für *c* folgendes angegeben werden darf: **d** als "gelöscht" markierte Briefe (*deleted*) **n** alle neuen Briefe (*new*) **o** alle alten Briefe (*old*) **r** alle bereits gelesenen Briefe (*read*) **s** alle gesicherten Briefe (*saved*) **u** alle noch nicht gelesenen Briefe (*unread*)

Hier werden bereits die Kommandos `delete` (als "gelöscht" markieren) und `save` (auf Datei sichern) verwendet:

d 2-5	Briefe mit den Nummern 2, 3, 4 und 5 als "gelöscht" markieren
s :u ungelesen	noch nicht gelesene Briefe in Datei `ungelesen` sichern
p toni	alle Briefe von `toni` ausgeben
2	Brief mit der Nummer 2 ausgeben
d :r	alle bereits gelesenen Briefe als "gelöscht" markieren

Wenn neue Post ankommt, so wird dies dem entspr. Benutzer mit *you have mail* oder *you have new mail* mitgeteilt. Dieser Hinweis erfolgt entweder sofort, wenn der Benutzer gerade angemeldet ist, oder aber beim nächsten Anmelden. Um neu angekommene Post zu lesen, muss Benutzer **mail** bzw. **mailx** ohne Angabe von *login-name(n)* aufrufen. Danach wird eine Liste von Kopfzeilen zu den in der Mailbox vorhandenen Briefen ausgegeben, wie z.B.

```
mailx version 4.0  Type ? for help.
"/var/spool/mail/egon":  3 messages  1 new  2 unread
 U  1 miller         Fri Jul  10 09:10   9/247  Jubilaeum von Schorsch
 U  2 marketing      Fri Jul  10 09:14  11/437  Marketing-Termine
>N  3 molly          Fri Jul  10 09:18   7/137
?
```

Erste Zeile zeigt Versionsnummer des mail-Programms und gibt Hinweis, dass mit **?** eine Kurzbeschreibung von mail-Kommandos angefordert werden kann. Zweite Zeile zeigt Pfadnamen der Datei, die als Mailbox verwendet wird; zusätzlich wird in dieser Zeile angezeigt, wie viele Briefe in dieser Mailbox vorhanden sind, wie viele davon neu und wie viele davon ungelesen sind. Die restlichen Zeilen geben zu den in der Mailbox vorhandenen Briefen die Überschriften an. Links von den Nummern kann dabei Statusinformation angegeben sein, wie z.B.:

`N`	(*new*) ist seit dem letzten mail-Aufruf neu eingetroffen
`R`	(*read*) ist neu eingetroffen und bereits gelesen
`U`	(*unread*) ist schon älter, aber noch nicht gelesen
`O`	(*old*) ist schon älter und gelesen
`S`	(*saved*) wurde in einer Datei gesichert
`M`	(*mbox*) wird bei Verlassen in `mbox` aufgehoben

Das Zeichen > steht dabei immer vor dem aktuellen Brief. Desweiteren wird zu jedem einzelnen Brief der Login-Name des Absenders, das Datum und die Uhrzeit der Zustellung, die Anzahl der Zeilen und Zeichen des Briefes. und die *Subject:*-Zeile angegeben. Am Ende dieser Liste erscheint dann das Promptzeichen `?` oder `&,` um dem Benutzer anzuzeigen, dass er nun mailx/mail-Kommandos eingeben kann.

Nachfolgend sind einige mail/mailx-Kommandos aufgezählt, die hier eingegeben werden können. Dabei gilt allgemein, dass das jeweilige Kommando sich auf die mit der *liste* ausgewählten Mails bezieht. Ist keine *liste* angegeben, bezieht sich das jeweilige Kommando auf den aktuellen Brief.

!*unix-kdo*	führt das angegebene Unix-Kommando *unix-kdo* aus.
?	gibt eine Zusammenfassung der mail-Kommandos am Bildschirm aus.
delete [*liste*]	bewirkt, dass die mit *liste* ausgewählten Briefe als "gelöscht" markiert werden. Die als "gelöscht" markierten Briefe werden erst beim Verlassen bzw. beim Wechseln in eine andere Mailbox entfernt.
edit [*liste*]	bewirkt das Editieren der mit *liste* ausgewählten Briefe. Als Editor wird dabei der in der mail-Variablen EDITOR angegebene Editor verwendet; Voreinstellung ist **vi**.
exit	bewirkt, dass mail/mailx unmittelbar verlassen wird und keine Briefe in mbox gesichert werden.
from [*liste*]	bewirkt, dass zu den mit *liste* ausgewählten Briefen die Kopfzeilen ausgegeben werden.
help	gibt eine Zusammenfassung der mail/mailx-Kommandos am Bildschirm aus.
hold [*liste*]	bewirkt, dass die mit *liste* ausgewählten Briefe in Mailbox verbleiben, obwohl sie bereits gelesen wurden.
list	gibt alle verfügbaren mail/mailx-Kommandos ohne sonstige Erklärungen am Bildschirm aus.
mail *login*	ermöglicht das Schreiben eines Briefes an die Benutzer *login*.
New [*liste*]	Die mit *liste* ausgewählten Briefe bzw. den aktuellen Brief als "ungelesen" markieren.
next [*angabe*]	bewirkt, dass zum nächsten Brief gesprungen wird, auf den die *angabe* passt; dies ist nützlich, wenn für *angabe* entweder *login-name* oder */text* angegeben wird.
print [*liste*]	bewirkt die Ausgabe der mit *liste* ausgewählten Briefe.
quit	bewirkt, dass vor dem Verlassen von mail/mailx alle gelesenen Briefe in mbox gesichert werden und nur die ungelesenen Briefe in der Mailbox verbleiben. Briefe, die explizit in einer Datei gesichert oder als "gelöscht" markiert wurden, werden in keiner dieser beiden Dateien aufgehoben.
reply [*brief*]	bewirkt, dass nicht nur dem Absender des mit *brief* ausgewählten Briefes, sondern auch allen anderen Adressaten eine Antwort geschickt wird. Nach Abgabe dieses Kommandos werden die To:- und Subject:-Zeilen eingeblendet, bevor in Eingabemodus umgeschaltet wird. Nach Verlassen des Eingabemodus mit ~. wird der gerade geschriebene Antwortbrief an alle Adressaten geschickt, die in der zuvor eingeblendeten To:-Zeile erwähnt wurden. Ist kein *brief* angegeben, so wird auf den aktuellen Brief geantwortet.
save [*liste*] *datei*	sichert die mit *liste* ausgewählten Briefe in Datei *datei*. Wird weder *datei* noch *liste* angegeben, so wird der aktuelle Brief in der Datei ~/mbox gesichert. Die so gesicherten Briefe werden als "gesichert" markiert, d.h. dass sie beim Verlassen normalerweise aus der Mailbox entfernt und nicht in mbox gesichert werden.
shell	Durchschalten auf die Unix-Kommandoebene (auch mit **!** möglich); Rückkehr nach mail/mailx ist mit der Eingabe von **exit** oder (Strg)-D möglich.
top [*liste*]	bewirkt die Ausgabe der fünf ersten Zeilen der mit *liste* ausgewählten Briefe.
undelete [*list*]	bewirkt, dass bei den mit *list* ausgewählten Briefen die Markierung "gelöscht" wieder aufgehoben wird.
unread [*liste*]	Die mit *liste* ausgewählten Briefe bzw. den aktuellen Brief als "ungelesen" markieren; dasselbe wie **New**.
visual [*liste*]	bewirkt das Editieren der mit *liste* ausgewählten Briefe. Als Editor wird dabei der in der mail-Variablen VISUAL angegebene Editor verwendet; Voreinstellung ist **vi**.
z[**+**]	Vorwärtsblättern in der Kopfzeilen-Liste (Inhaltsverzeichnis); + ist optional
z-	Zurückblättern in der Kopfzeilen-Liste (Inhaltsverzeichnis)

mail/mailx-Variablen

Mit dem **set**-Kommando ist es möglich, mail/mailx-Variablen zu setzen:

set	Ausgabe aller definierten Variablen und deren Werte
set *var1* [[*var2*] ...]	setzt Variablen *var1*, *var2*, .. **no** vor Variablennamen schaltet entspr. Variable aus.
unset *var1* [[*var2*] ...]	entspricht der Angabe **set no***var1* [[**no***var2*] ...]
set *var*=*wert*	weist *var* den *wert* zu. *wert* kann dabei ein String (mit '.. zu klammern) oder eine Zahl sein.

Die mail/mailx-Variablen können dabei entweder während einer **mail**/**mailx**-Sitzung oder aber in der Datei ~/.mailrc gesetzt werden. Auf eine Vorstellung der einzelnen mail-Variablen wird hier verzichtet. Interessierte Leser seien auf die man-Seite von **mail** bzw. **mailx** verwiesen.

10 Netzkommandos

10.1 Allgemeine Netz-Kommandos

ping – Testen von Netzwerk-Verbindungen

ping [*optionen*] *host*

Mit dem Kommando **ping** kann man feststellen, ob ein System momentan erreichbar ist. Für *host* muss der Name des zu testenden Rechners bzw. dessen IP-Adresse angegeben werden. **ping** sendet ständig Datenpakete an das entfernte System und mißt die Zeit, bis diese wieder zurückgeschickt werden (*round-trip*). **ping** kann man mit (Strg)-C beenden. Am Ende gibt **ping** eine Statistik darüber aus, wie viele Pakete erfolgreich übertragen werden konnten und wie viele nicht.

uname – Erfragen des eigenen Rechnernamens

uname [*optionen*]

uname gibt den Namen des lokalen Unix-Systems auf der Standardausgabe aus.

-s	(*system name*) Name des lokalen Systems; ist die Voreinstellung, wenn **uname** ohne Angabe von Optionen aufgerufen wird.
-n	(*node name*) Knotenname des lokalen Systems im Netzwerk.
-r	(*release*) Freigabe-Nummer des lokalen Systems
-v	(*version*) Versions-Nummer des lokalen Systems
-m	(*machine*) Hardware des lokalen Systems
-p	(*processor type*) Prozessortyp des lokalen Systems
-a	(*all*) alle obigen Informationen

hostname – Anzeigen bzw. Ändern des Netzwerknamens des Rechners

hostname [*optionen*] *name*

Das Kommando **hostname** zeigt den Netzwerknamen des Rechners an bzw. ändert ihn auf *name*.

-a	vollständigen Hostnamen (einschließlich Domainnamen) anzeigen

10.2 FTP, Secure-Shell und Telnet

ftp – Übertragen von Dateien von/zu einem anderen Rechner

ftp [*optionen*] [*host*]

Mit dem Kommando **ftp** kann man sich auf einen anderen vernetzten Rechner begeben, dort im Directorybaum herumwandern und Dateien zwischen den beiden Systemen hin und her kopieren. Das Programm **ftp** kommuniziert mit dem Server über das FTP-Protokoll, das von vielen Betriebssystemen unterstützt wird, so dass Datenübertragungen über **ftp** auch mit Nicht-Unix-Systemen möglich sind.

Wenn für *host* Rechnername oder Internet-Adresse angegeben ist, versucht **ftp** eine Verbindung zu diesem System aufzubauen. Anschließend meldet sich dann **ftp** im Kommandomodus mit Prompt `ftp>` und erwartet Eingabe eines ftp-Kommandos. Wird kein *host* angegeben, meldet sich **ftp** ohne Verbindungsaufbau sofort im Kommandomodus.

Nach erfolgreichem Verbindungsaufbau meldet sich noch der FTP-Server, bevor die Aufforderung zur Anmeldung erscheint. Dabei wird der Login-Name des Aufrufers als Voreinst. angeboten. Nur wenn Benutzer diesem System unter anderem Login-Namen bekannt ist, muss er diesen hier eingeben, ansonsten reicht Bestätigung mit (Enter).

-g	(*globbing*) Sonderzeichen für Dateinamen-Expandierung bei den ftp-Kommandos **mget**, **mput** und **mdelete** ausschalten (siehe auch ftp-Kommando **glob**).
-i	Interaktive Abfrage nach zu kopierenden Dateien bei den ftp-Kommandos **mget** und **mput** ausschalten.
-t	(*tracing*) Kontrollausgaben für übertragene Datenpakete einschalten.
-v	(*verbose*) Ausgabe von Informationsmeldungen des FTP-Servers auf dem entfernten System einschalten; Voreinstellung bei interaktiven ftp-Prozessen.

ftp kennt u.a. die folgenden Kommandos:

! [*kdo*]	Das Shell-Kommando *kdo* auf der lokalen Maschine ausführen. Ist kein *kdo* angegeben, so wird eine interaktive Shell am lokalen System aufgerufen.
append *lokaldatei* [*ferndatei*]	Inhalt von *lokaldatei* an einer Datei am entfernten System anhängen. Falls *frndatei* nicht angegeben ist, wird auf dem entfernten System der Name *lokaldatei* benutzt.
ascii	Übertragungsart von Daten auf ASCII-Format einstellen; dies ist die Voreinstellung.
binary	Übertragungsart von Daten auf binäres Format einstellen.
bye oder **quit**	ftp beenden; auch mit (Strg)-D möglich.
case	Ein- bzw Ausschalten, dass bei Übertragung mit (**m**)**get** oder (**m**)**put** Dateinamen automatisch von Klein- in Großschreibung umgewandelt werden. Per Voreinst. ist diese Umwandlung ausgeschaltet.
cd *dir*	Wechseln in das Directory *dir* auf dem entfernten System.
cdup	Wechseln in das Parent-Directory auf dem entfernten System; entspricht **cd ..** auf Unix-Systemen.
close, **disconnect**	ftp-Verbindung abbauen und in den ftp-Kommandomodus zurückschalten.
cr	Bei der Übertragung von ASCII-Dateien werden Neuezeilezeichen durch CR/NL (*Carriage-Return/Linefeed*) dargestellt. Wenn **cr** eingeschaltet ist (Voreinst.), werden alle CR (*Carriage-Return*) entfernt, um so die unter Unix verwendende Darstellung zu haben. Wenn nun eine Übertragung einer ASCII-Datei zu/von Nicht-Unix-Systemen stattfindet, so mag diese Datei NL-Zeichen enthalten, die keine Zeilenbegrenzung darstellen. In diesem Fall ist **cr** aufzurufen.
delete *ferndatei*	Datei *ferndatei* auf dem entfernten System löschen.
debug	Debugging-Modus ein- bzw. wieder ausschalten. Wenn der Debugging-Modus eingeschaltet ist, so zeigt ftp jedes Kommando mit vorangestelltem "`-->`" an, das es zum entfernten System schickt.
dir [*ferndir*] [*lokaldatei*]	Inhalt des Directorys *ferndir* auf entferntem System auflisten. Ist kein *ferndir* angegeben, wird der Inhalt des Working-Directory aufgelistet. Ist *lokaldatei* angegeben, so erfolgt die Ausgabe in diese Datei auf dem lokalen System. Fehlt die Angabe von *lokaldatei*, erfolgt die Ausgabe am Terminal.
get *ferndatei* [*lokaldatei*]	*ferndatei* vom entfernten System auf lokales System kopieren. Sind keine *lokaldatei* angegeben, wird auf dem lokalen System der gleiche Name wie auf dem entfernten System verwendet.
glob	Bedeutung der Sonderzeichen für Dateinamen-Expandierung bei den Kommandos **mdelete**, **mget** und **mput** aus- bzw. einschalten; per Voreinstellung ist Dateinamen-Expandierung eingeschaltet.

hash	Automatische #-Anzeige bei Dateiübertragung ein- bzw. wieder ausschalten. Ist #-Anzeige eingeschaltet, wird für jeden übertragenen Datenblock (8192 Bytes) das Nummernzeichen # ausgegeben. Dies ist nützlich, wenn man die Übertragung großer Dateien mitverfolgen möchte.
help [*ftp-kommando*] od. **?** [*ftp-kommando*]	Help-Information zu *ftp-kommando* ausgeben. Ist *ftp-kommando* nicht angegeben, so wird eine Kurzübersicht zu allen ftp-Kommandos ausgegeben.
lcd [*dir*]	Wechseln in Directory *dir* auf lokalem System. Fehlt *dir*, wird in das Home-Directory gewechselt.
ls [*ferndir*] [*lokaldatei*]	Inhalt des Directorys *ferndir* auf entferntem System in Kurzform listen. Ist kein *ferndir* angegeben, wird Inhalt des Working-Directorys aufgelistet. Ist *lokaldatei* angegeben, erfolgt Ausgabe in diese Datei auf dem lokalen System. Fehlt die Angabe von *lokaldatei*, erfolgt die Ausgabe am Terminal.
mdelete *ferndatei(en)*	Die Dateien mit den Namen *ferndatei(en)* auf dem entfernten System löschen.
mdir *ferndateien* [*lokaldatei*]	Wie **dir**, nur dass mehrere *ferndateien* angegeben werden können.
mget *ferndateien*	Dateien mit Namen *ferndateien* vom entfernten System in Work.-Dir. auf lokalem System kopieren.
mkdir *ferndir*	Anlegen eines Directorys *ferndir* auf dem entfernten System.
mls *ferndir*... [*lokaldatei*]	Wie **ls**, nur dass mehrere *ferndirectories* angegeben werden können.
mput *lokaldateien*	*lokaldateien* vom lokalen System in das Working-Directory auf dem entfernten System kopieren.
open *host* [*port*]	Verbindung zum Rechner *host* aufbauen. Ist *port* angegeben, versucht ftp Verbindung über das Port.
prompt	Interaktive Abfrage von **mget** und **mput**, ob eine Datei zu kopieren ist, aus- bzw. wieder einschalten; per Voreinstellung ist diese interaktive Abfrage eingeschaltet.
put *lokaldatei* [*ferndatei*] **send** *lokaldatei* [*ferndatei*]	*lokaldatei* vom lokalen System auf entfernte System kopieren. Wenn keine *ferndatei* angegeben ist, so wird auf dem entfernten System der gleiche Name wie auf dem lokalen System verwendet.
pwd	Momentanes Working-Directory auf dem entfernten System ausgeben.
rename *alt neu*	Datei *alt* auf dem entfernten System in *neu* umbenennen.
rmdir *ferndir*	Das Directory *ferndir* auf dem entfernten System löschen.
status	Momentane Einstellung des FTP-Servers ausgeben.
type [*typ*]	Für *typ* darf **ascii** (ASCII-Format) oder **binary** bzw. **image** (binäres Format) angegeben werden. Voreinst. ist **ascii**. Wird kein *typ* angegeben, wird der momentan gesetzte *typ* ausgegeben.
verbose	Um mehr Informationen vom FTP-Server zu erhalten, wie z.B. Informationen über die Übertragungsgeschwindigkeit nach einem Dateitransfer, muss man das ftp-Kommando **verbose** aufrufen. Ein erneuter Aufruf schaltet diese zusätzlichen Informationen des FTP-Servers wieder aus.

Falls Argumente, die nicht optional sind, bei einem ftp-Kommando weggelassen werden, erfragt **ftp** diese interaktiv.

Argumente zu den ftp-Kommandos, die Leerzeichen beinhalten, müssen mit Anführungszeichen geklammert werden, wie z.B. **ls -CF "\| lpr -n5 -c"**

Soll der Inhalt einer zu kopierenden Datei von der Standardeingabe gelesen oder auf die Standardausgabe geschrieben werden, so muss anstelle eines Dateinamens ein Minuszeichen (**-**) angegeben werden. So bewirkt z.B. der Aufruf **get marketing -** die Ausgabe der Datei `marketing` auf dem Bildschirm.

Ist das erste Zeichen eines Namens ein Pipe-Zeichen (Senkrechtstrich **|**), so muss der Rest ein Unix-Kommando sein, an dessen Standardeingabe die Standardausgabe des ftp-Komandos über die Pipe weitergeleitet wird. So könnte man z.B. mit **dir . |less** bzw. **dir . |more** die Dateien des Working-Directorys auf dem entfernten System seitenweise im Langformat auflisten.

Mit (Strg)-C kann man eine Datenübertragung abbrechen.

Im Internet stellen viele Rechner freie Software-Archive zur Verfügung. Jeder Benutzer kann per anonymous FTP auf diese Archive zugreifen. Der Systemadministrator richtet dabei auf Archiv-Rechner ein Login **anonymous** ein. Es ist Konvention, dass jeder Gastbenutzer dann als Passwort seine Mail-Adresse eingibt.

Nach einer Anmeldung befindet man sich dann im Home-Directory von **anonymous**, in dem üblicherweise weitere Subdirectories enthalten sind, die Kommandos, Protokolldateien und die frei verfügbare Software (meist im Subdirectory `pub`) enthalten. In diese Subdirectories kann man dann mit dem ftp-Kommando **cd** wechseln.

Meist ist im `pub`-Directory ein Inhaltsverzeichnis der verfügbaren Software enthalten, das komprimiert ist. Um ein solches Inhaltsverzeichnis auf den lokalen Rechner zu kopieren, müßte man zunächst das ftp-Kommando **binary** aufrufen, dann mit **get** die entsprechende Datei auf den lokalen Rechner kopieren, wo diese Datei dann noch dekomprimiert werden muss.

tftp – Einfaches Übertragen von Dateien von/zu einem anderen Rechner

tftp [*host*]

Anders als **ftp** baut **tftp** keine feste Verbindung zu einem entfernten System auf, sondern stellt immer wieder neu bei Kopier-Anforderungen eine Verbindung zum betreffenden System her. Zu welchem System dabei eine Verbindung herzustellen ist, kann entweder beim Aufruf durch die Angabe von *host* oder danach mit **connect** festgelegt werden. Nach dem Aufruf meldet sich **tftp** im Kommandomodus mit dem Prompt `tftp>` und erwartet die Eingabe eines tftp-Kommandos. **tftp** kennt u.a. die folgenden Kommandos:

- **connect** *host* [*port*]: Für zukünftige Dateiübertragungen Rechner mit Namen *host* (und evtl. *port*) festlegen.
- **mode** *modus*: Übertragungsmodus auf *modus* (**ascii** oder **binary**) einstellen. Voreinstellung ist: **ascii**.
- **put** *lokale-datei*
  put *lokale-datei entfernte-datei*
  put *lokale-datei(en) entferntes-directory*
  lokale-datei bzw. *lokale-datei(en)* vom lokalen System auf das entfernte System kopieren. Für *entfernte-datei* oder *entferntes-directory* kann entweder nur ein einfacher Name angegeben werden, wenn der Zielrechner *host* bereits zuvor festgelegt wurde, oder aber *host:dateiname*, um zugleich den *host* und den Dateinamen festzulegen. Im letzteren Fall wird auch zugleich *host* als Zielrechner für zukünftige Dateitransfers festgelegt.
- **get** *entfernte-datei*
  get *entfernte-datei lokale-datei*
  get *entfernte-datei1 entfernte-datei2 entfernte-datei3 ...*
  entfernte-datei bzw. *entfernte-datei(en)* vom entfernten System auf das lokale System kopieren. Für *entfernte-datei* kann entweder nur ein einfacher Name angegeben werden, wenn der Zielrechner *host* bereits zuvor festgelegt wurde, oder aber *host:dateiname*, um zugleich den *host* und den Dateinamen festzulegen. Im letzteren Fall wird auch zugleich *host* als Zielrechner für zukünftige Dateitransfers festgelegt.
- **quit**: tftp beenden; auch mit (Strg)-D möglich.
- **ascii**: Übertragungsart von Daten auf ASCII-Format einstellen (entspricht **mode ascii**); dies ist die Voreinstellung.
- **binary**: Übertragungsart von Daten auf Binär-Format einstellen (entspricht **mode binary**).
- **verbose**: Um mehr Informationen über Transfers zu erhalten, wie z.B. Informationen über die Übertragungsgeschwindigkeit nach einem Dateitransfer, muss man das tftp-Kommando **verbose** aufrufen. Ein erneuter Aufruf schaltet diese zusätzlichen Informationen wieder aus.
- **status**: Information über den momentanen Status ausgeben.
- **?** [*tftp-kommando(s)*]: Help-Information zu *tftp-kommando(s)* ausgeben. Ist kein *tftp-kommando* angegeben, so wird eine Kurzübersicht zu allen tftp-Kommandos ausgegeben.

ssh – Starten einer sicheren Shell auf anderem Rechner

ssh -l *loginname rechnername*

Wenn man mit **ssh** zum ersten Mal eine Verbindung zu einem anderen Rechner herstellt, erscheint oft eine Warnung. Diese Warnung erscheint, da **ssh** hier nachfragt, ob es dem anderen Rechner mit seiner IP-Adresse vertrauen darf. Beantwortet man diese Frage mit `yes`, speichert **ssh** den Namen und den *RSA-Fingerprint* (Code zur eindeutigen Identifizierung des anderen Rechners) in der Datei `~/.ssh/known_hosts`, so dass diese Nachfrage beim nächsten Anmelden nicht mehr erscheint.

Danach fragt **ssh** nach dem Passwort des Benutzers *loginname*. Nach der Eingabe des richtigen Passworts beginnt die Sitzung am anderen Rechner, was durch die Ausgabe des Promptzeichens `$` angezeigt wird.

rlogin und **telnet** verwenden ein sehr unsicheres Protokoll, bei dem u.a. das Passwort unverschlüsselt übertragen wird. Um schlimmstes zu vermeiden, ist es deshalb u.a. nicht möglich, sich mittels **rlogin** oder **telnet** als `root` auf einem anderen Rechner anzumelden.

Die Secure-Shell **ssh** dagegen ist ein Login-Programm, das eine vollständig verschlüsselte Sitzung zwischen Rechnern unterstützt. Für jede Verbindung wird hierzu zwischen den Partnern ein neuer Sitzungsschlüssel ausgehandelt, so dass einem potentiellen Angreifer kaum die Zeit bleibt, einen solchen Schlüssel schnell genug zu knacken. Als Verschlüsselungsalgorithmus kann man zwischen BlowFish, IDEA (*International Data Encryption Standard*), RSA (*Rivest-Shamir-Adelman-Algorithmus*) und TripleDES (*Data Encryption Standard*) wählen. Alle vier Algorithmen gelten als absolut sicher, weshalb **ssh** wesentlich sicherer ist als **rlogin** bzw. **telnet**.

telnet – Anmelden auf einem anderen Rechner

telnet [*host* [*port*]]

Mit dem Kommando **telnet** (*teletype network*) kann man sich ähnlich wie mit **rlogin** auf einem anderen System anmelden. **telnet** wird üblicherweise für Verbindungen zu Nicht-Unix-Systemen benutzt.

host muss Name des betreffenden Rechners oder seine Internet-Adresse aus `/etc/hosts` sein. Die *port*-Angabe ermöglicht Verbindungsaufbau zu einem bestimmten Dienst, dessen Name oder Nummer hierbei anzugeben ist.

Nach einem erfolgreichen Verbindungsaufbau befindet sich **telnet** im Eingabemodus, was bedeutet, dass alle eingegebenen Zeichen an das entfernte System übertragen und von diesem verarbeitet werden. Ausgaben von Kommandos auf dem entfernten System erscheinen dagegen auf dem lokalen Bildschirm.

Wird **telnet** ohne Argument aufgerufen, befindet man sich im telnet-Kommandomodus, was durch Ausgabe des Prompts `telnet>` angezeigt wird. Man kann mit (Strg)-] vom Eingabe- in Kommandomodus wechseln. Um sich eine Kurzübersicht der möglichen telnet-Kommandos anzeigen zu lassen, muss man nur **?** eingeben. Mit **open** kann man eine Verbindung zum entfernten System aufbauen. Nach erfolgreicher Anmeldung kann man auf entfernten System arbeiten. Mit der Abmeldung von diesem System wird normalerweise auch Verbindung abgebrochen. Sollte dies nicht der Fall sein, kann man die Verbindung mit den telnet-Kommandos **close** oder **quit** abbrechen. Dadurch werden alle telnet-Verbindungen abgebrochen, wenn man sich über mehrere Systeme hinweg mit einem bestimmten Rechner verbunden hatte. Im Kommandomodus können u.a. die folgenden telnet-Kommandos eingegeben werden:

open [*host* [*port*]]	eine Verbindung zu einem entfernten System aufbauen.
close bzw. **quit**	alle telnet-Verbindungen abbrechen. Eine Abmeldung mit (Strg)-D ist auch möglich.
z	**telnet** suspendieren.
mode *typ*	Nach Verbindungsaufbau befindet sich **telnet** im Eingabemodus. Es arbeitet dabei entw. mit Einzelzeichen- (*character-at-a-time*) oder mit Zeilen-Übertragung (*line-by-line-modus*), je nachdem, welchen Modus das entfernte System beim Verbindungsaufbau angefordert hat. Mit **mode** *typ* läßt sich dieser Modus ändern. Für *typ* kann **character** oder **line** angegeben werden. Bei Einzelzeichen-Übertragung wird jede gedrückte Taste sofort an entfernte System gesendet. Bei Zeilen-Übertragung werden nur ganze Zeilen, nachdem (Enter) eingegeben wurde, gesendet. Hierbei ist immer die Echo-Funktion am Terminal eingeschaltet, was bedeutet, dass jedes eingegebene Zeichen am Bildschirm angezeigt wird. Echo-Funktion kann mit (Strg)-E ausgeschaltet werden, um z.B. Passwort-Eingaben zu tätigen.
status	gibt den momentan eingestellten Übertragungsmodus aus.
display [*argument(e)*]	gibt alle aktuellen Einstellungen oder die Einstellungen zu *argument(e)* (siehe **toggle**) aus.
? [*telnet-kommando*]	gibt Kurzbeschreibung zu *telnet-kommando* aus. Ist kein *telnet-kommando* angegeben, so wird eine Kurzübersicht zu allen telnet-Kommandos ausgegeben.
send *name(n)*	sendet telnet-Sequenzen an das entfernte System, so als seien sie durch Steuerzeichen veranlaßt wurden. Mögliche Angaben für *name(n)* sind z.B.:

brk	(*break*) Unterbrechungssignal (*quit*)	**ec**	(*erase character*) letzt. Zeichen löschen
el	(*erase line*) ganze Zeile wieder löschen	**ip**	(*interrupt process*) Prozeß abbrechen
escape	Escape-Zeichen; voreingestellt auf (Strg)-]	**ao**	(*abort output*) Ausgabe beenden
synch	(*synch operation*) Eingabe beenden	**ayt**	(*are you there*) Verbindung prüfen
ga	(*go ahead*) Fortsetzung	**nop**	(*no operation*) Leeres Kommando
?	Kurzbeschreibung zu **send** ausgeben		

set *name wert*	Steuerzeichen ändern, wie z.B. **set erase ^x**. Mögl. Angaben für *name* sind:

echo	((Strg)-E) Echo-Funktion	**escape**	((Strg)-]) in Kdomodus umschalten
interrupt	Interrupt; telnet-Sequenz **ip** schicken	**quit**	Interrupt; telnet-Seq. **brk** schicken
flushoutput	telnet-Sequenz **ao** schicken	**erase**	letzt. Zeichen löschen; **ec** schicken
kill	ganze Zeile löschen; **el** schicken	**eof**	Dateiende-Zeichen

toggle *name(n)*	telnet-Einstellungen ändern, was abhängig vom momentanen Zustand entweder ein- oder ausschalten bedeutet. Mögliche Angaben für *name(n)* kann man sich mit **toggle ?** anzeigen lassen.

10.3 r-Kommandos von Berkeley

rlogin – Anmelden auf einem anderen Netzwerkrechner

rlogin [*optionen*] *host*

rlogin ermöglicht es, sich auf einem entfernten Rechner im Netz anzumelden. Als Login-Name wird normalerweise der Login-Name des aufrufenden Benutzers verwendet, außer es wurde die Option **-l** *login-name* angegeben.

-l *login-name*	Unter *login-name* am entfernten System anmelden.

Als *host* sind dabei die in `/etc/hosts` aufgezählten Rechner-Namen erlaubt.

Der Systemadministrator legt fest, auf welchen Rechnern im Netz er eine Login-Kennung für die einzelnen Benutzer einrichtet. Hat ein Benutzer auf einem anderem Rechner im Netz eine Kennung, so kann er sich dort anmelden (eventuell mit eigenem Passwort). Jeder lokale Rechner im Netz kann eine Datei `/etc/hosts.equiv` enthalten, die eine Liste von Rechnernamen enthält. Die Benutzer auf diesen darin erwähnten Rechnern gelten als so genannte "trusted" Benutzer (Vertrauenspersonen), die sich mit ihrem Login-Namen an diesem lokalen Rechner anmelden können, ohne dass von Ihnen ein Passwort verlangt wird. Wenn in `/etc/hosts.equiv` nur eine einzige Zeile mit einem Pluszeichen (+) beginnt, gelten alle Benutzer der im Netz vorhandenen Rechner als Vertrauenspersonen, denen ohne Passwort Zugang zum lokalen System gestattet ist. Daneben kann aber auch noch jeder einzelne Benutzer festlegen, wer sich von entfernten Rechnern am lokalen Rechner unter seiner Login-Kennung anmelden darf. Dazu muss der entsprechende Benutzer in seinem Home-Directory eine Datei `.rhosts` anlegen, in der er alle seine Vertrauenspersonen aufzählt. Eine Zeile in `.rhosts` muss zwei oder mehr mit Leerzeichen getrennte Einträge enthalten: den *hostname* und *login-name(n)*. Ein Eintrag in `.rhosts` bedeutet dann, dass sich Benutzer *login-name* vom entfernten Rechner *hostname* am lokalen Rechner ohne Eingabe eines Passwortes anmelden darf. Wenn z.B. `egon` auf seinem System `ahorn` folgende Datei `.rhosts` in seinem Home-Directory hat:

```
birke.klug.de egon
eiche.klug.de eg eh
kiefer.klug.de mueller
```

dann wird der Zugang unter der Kennung `egon` für das System `birke` gestattet. Anmeldungen vom Rechner `eiche` aus müssen unter `eg` oder `eh`, und Anmeldungen vom Rechner `kiefer` aus unter `mueller` erfolgen.
Die Datei `.rhosts` wird nur ausgewertet, wenn nicht bereits in `/etc/hosts.equiv` ein passender Eintrag für den Rechner gefunden wurde. Der Eigentümer von `.rhosts` muss der Benutzer selbst oder der Systemadministrator sein, sonst wird der Inhalt von `.rhosts` ignoriert. Ist einem Benutzer kein Zugang als "Vertrauensperson" gestattet, so wird von ihm ein Passwort verlangt, wie dies auch bei einer gewöhnlichen Benutzeranmeldung an einem System der Fall ist. Die Sitzung mit **rlogin** wird beendet, wenn man sich auf dem entfernten System mit **exit** oder (Strg)-D abmeldet. Falls eine solche normale Abmeldung aus irgendwelchen Gründen nicht mehr möglich sein sollte, kann man die Sitzung auch durch die Eingabe der Zeichenfolge **~.** beenden. Soll nur die letzte Verbindung abgebrochen werden, muss man **~~.** eingeben.

rwho – Auflisten aller momentan aktiver Benutzer im Netz

rwho [**-a**]

rwho liefert Informationen zu allen Benutzer, die in einem Netz angemeldet sind. Benutzer, die seit mehr als einer Stunde inaktiv waren, werden in der Ausgabe von **rwho** nicht angezeigt, außer es wurde die Option **-a** angegeben.

ruptime – Auflisten aller Systeme im Netz

ruptime [*optionen*]

Mit dem Kommando **ruptime** kann man feststellen, welche Systeme seit wann im Netz angeschlossen sind.

-a	auch die Benutzer mitzählen, die seit mehr als einer Stunde keine Eingaben mehr getätigt haben.
-l	nach *load average* sortieren.
-r	umgekehrt sortieren.
-t	nach Laufzeit (*uptime*) sortieren.
-u	nach Anzahl der Benutzer sortieren.

rcp – Kopieren von Dateien im Netz

rcp [**-px**] *datei1 datei2*

rcp [**-prx**] *datei(en) directory*

Mit dem Kommando **rcp** (*remote copy*) kann man Dateien oder ganze Directorybäume über ein Netz kopieren. Ansonsten verhält sich **rcp** wie das Kommando **cp**.

-p	Für jede kopierte Datei werden die gleichen Zugriffsrechte, die gleiche Zugriffs- und Modifikationszeit eingetragen, die für die Originaldatei gelten.
-r	Sollen ganze Directorybäume kopiert werden, so muss diese Option angegeben werden.
-x	schaltet die DES-Verschlüsselung für alle Daten ein, die mit **rcp** übertragen werden. Dies erhöht die Sicherheit, so dass kein Unbefugter diese Daten lesen kann, verringert aber die Geschwindigkeit.

- **rcp add.c weide:/home/egon/add.c**: kopiert Datei `add.c` vom lokalen Rechner in Directory `/home/egon` auf dem Rechner `weide`. Man braucht nicht unbedingt den absoluten Pfadnamen der Zieldatei anzugeben. Fehlt diese Angabe, wird die Datei ins Home-Directory des Aufrufers auf dem jeweiligen System kopiert. Falls auch noch der gleiche Dateiname bei der Kopie zu verwenden ist, kann auch noch der Zieldateiname weggelassen werden. Für obigen Aufruf hätte `egon` z.B. auch folgenden Aufruf angeben können. **rcp add.c weide:**
- **rcp eiche:add.c kiefer:add2.c**: kopiert `add.c` auf System `eiche` in Datei `add2.c` auf System `kiefer`.

Normalerweise benutzt **rcp** auf dem entfernten System die Benutzerkennung, unter der es am lokalen System aufgerufen wurde. Ist aber bei den Argumenten vor dem Rechnernamen mit `@` abgetrennt ein Login-Name angegeben, so findet der Kopiervorgang unter der Kennung dieses Login-Namens statt. Voraussetzung dafür ist, dass die lokale Kennung in der Datei `.rhosts` des entsprechenden Benutzers enthalten ist.

- **rcp egon@tanne:add.c addiere.c**: kopiert `add.c` vom Home-Directory des Benutzers `egon` auf dem System `tanne` in das Working-Directory auf dem lokalen System unter dem Namen `addiere.c`.
- **rcp egon@*.c buche:src**: kopiert alle C-Programme des Working-Directorys in das Directory `/home/egon/src` auf den Rechner `buche` unter der Kennung `egon`.
- **rcp -r uebung1/src kiefer:sources**: kopiert den ganzen Directorybaum `uebung/src` in das Directory `sources` (unter dem Home-Directory des Aufrufers auf Rechner `kiefer`). Dabei wird das angegebene Zieldirectory `sources` automatisch erstellt, wenn es noch nicht existiert. Falls `sources` bereits existiert, wird das zu kopierende Directory `src` dort als Subdirectory angelegt.

rsh – Starten einer Shell auf anderen Netzwerkrechner

rsh [*optionen*] *hostname* [*kommando*]

-l *login-name*	anstelle des eigenen lokalen Login-Namens wird *login-name* auf dem entfernten System verwendet.
-n	Standardeingabe in `/dev/null` umlenken; z.B. nützlich, wenn **rsh** im Hintergrund gestartet wird.
-x	schaltet die DES-Verschlüsselung für alle Daten ein, die mit **rsh** übertragen werden. Dies erhöht die Sicherheit, so dass kein Unbefugter diese Daten lesen kann, verringert aber die Geschwindigkeit.

Mit **rsh** kann man einen Kommandoaufruf auf einem entfernten System ausführen lassen, ohne das man sich dort mit **rlogin** anmelden muss. Die Ausführung von **rsh** muss dabei jedoch auf dem entfernten Rechner entweder über `/etc/hosts.equiv` oder `.hosts` (im Home-Directory) gestattet sein. Bei Angabe eines *kommando* wird dieses auf dem System *hostname* ausgeführt. Wenn kein *kommando* angegeben ist, verhält sich **rsh** wie **rlogin**. Das Kommando **rsh** wird ausgeführt, als ob man sich gerade am System angemeldet hätte und dort dann die entspr. Kommandozeile aufgerufen hätte. Das Working-Directory ist das Home-Directory. Die Standard-E/A-Kanäle von **rsh** bleiben bei dem auszuführenden Kommando erhalten. Das heißt, dass weiter vom aufrufenden Terminal gelesen und auf dieses geschrieben wird. Jedoch wird anders als bei **rlogin** der Inhalt der `TERM`-Variablen nicht übertragen, was bedeutet, dass bildschirmorientierte Kommandos wie **less** oder **vi** nicht aufgerufen werden können. Metazeichen der Shell müssen ausgeschaltet werden, wenn sie nicht durch das lokale System, sondern erst durch das entfernte System ausgewertet werden sollen. Dies ist auf zwei verschiedene Arten möglich, entweder man schließt das ganze *kommando* in Anführungszeichen ein oder man stellt den entspr. Metazeichen einen Backslash `\` voran. Um z.B. auf dem System `kiefer` den Inhalt einer Datei `namen` sortiert und numeriert in eine Datei `namen.sort` zu schreiben, könnte man einen der beiden folgenden Aufrufe am lokalen System abgeben:

rsh kiefer "sort namen | nl –ba >namen.sort"
rsh kiefer sort namen \| nl –ba \>namen.sort

10.4 Netz-Kommandos von UUCP

uuname – Erfragen der im Netz erreichbaren Rechnernamen

uuname [*optionen*]

Die Namen aller Systeme, die mit dem lokalen System über ein Netzwerk gekoppelt sind, können mit dem Kommando **uuname** erfragt werden. Dabei gibt **uuname** die Namen aller Unix-Systeme aus, die über das UUCP-Packet angeschlossen sind.

uuencode – Kodieren von Binärdateien für Übertragung im Netz

uuencode [*quelldatei*] *zieldatei*

Um auf älteren Systemen über ein externes Netz binäre Dateien verschicken zu können, stehen die beiden Kommandos **uuencode** und **uudecode** zur Verfügung. **uuencode** konvertiert den binären Inhalt der Datei *quelldatei* in ASCII-Zeichen und schreibt diese auf die Standardausgabe. Falls keine *quelldatei* angegeben ist, liest **uuencode** die umzuwandelnden Zeichen von der Standardeingabe. *zieldatei* muss immer angegeben sein. Dieses Argument und die Zugriffsrechte der Datei werden in einer Kopfzeile vor dem generierten ASCII-Text auf die Standardausgabe ausgegeben. Lenkt man die Ausgabe von **uuencode** in eine Datei um, wie z.B.

uuencode add add >add.enc

dann kann man diese Datei (`add.enc`) problemlos verschicken. Der Empfänger muss dann nur noch **uudecode** aufrufen. Durch die Konvertierung mit **uuencode** wird eine Datei um ca. 35% größer. Falls man große Textdateien zu übertragen hat, kann es eventuell angebracht sein, diese zunächst zu komprimieren und die so erhaltene Binärdatei dann mit **uuencode** für die Übertragung wieder in ASCII-Format umzuwandeln.

uudecode - Dekodieren von uuencode-kodierten Dateien

uudecode [**-o** *ausgabedatei*] [*ascii-datei*]

uudecode ist für die Rückumwandlung von ASCII-Dateien, die mit **uuencode** erzeugt wurden, in Binärdateien zuständig. Bei dieser Rückumwandlung entnimmt **uudecode** der Kopfzeile die Zugriffsrechte und den Namen für die zu erzeugende Ausgabedatei. Falls *ascii-datei* nicht angegeben ist, liest **uudecode** von der Standardeingabe.

uucp - Kopieren von Dateien von einem Unix-System auf ein anderes

uucp [*optionen*] *quell-datei(en) ziel-datei*

uucp kopiert die angegebenen *quell-datei(en)* in die Datei bzw. das Directory *ziel-datei*; falls *ziel-datei* ein Directory ist, so werden für die dort angelegten neuen Dateien die Namen der *quell-datei(en)* verwendet als Name wird – wie bei **cp** – die letzte Komponente des Pfadnamens der *quell-datei(en)* in das Directory eingetragen.

- **uucp *.c frankfurt2!~egon**: Alle C-Programmdateien in das Home-Directory von `egon` auf dem entfernten Rechner `frankfurt2` kopieren.

uuto – Kopieren von Dateien zwischen vernetzten Unix-Systemen

uuto [*optionen*] *quell-datei(en) ziel*

uuto kopiert die *quell-datei(en)* an das angegebene *ziel*.

- **uuto -m * hh3!emil**: Alle Dateien des Work.-Dir. (einschl. aller Directorybäume dort) auf Rechner mit Knotennamen `hh3` kopieren; dort werden sie in das Directory `/var/spool/uucppublic/receive/emil/` kopiert; dem Sender wird am Ende des Kopiervorgangs E-Mail geschickt (**-m**).

uupick – Abholen von mit uuto kopierten Dateien

uupick [*optionen*]

-s *knotenname*	im Directory `/var/spool/uucppublic/receive/`*login-name*/ nur nach Dateien suchen, die vom System mit dem Namen *knotenname* gesendet wurden

uustat – Statusinformationen zu uuto-/uupick-Aufträgen

uustat [*optionen*]

Mit dem Kommando **uustat** ist es möglich, den Status von abgegebenen **uucp**- oder **uuto**-Aufträgen zu erfragen oder sogar solche Aufträge abzubrechen. Wird nur **uustat** (ohne Optionen) aufgerufen, so gibt es Informationen über alle unerledigten Aufträge des aufrufenden Benutzers aus.

uux – Ausführen eines Kommandos auf anderen Unix-Rechner

uux [*optionen*] "*kommandozeile*"

Das Kommando **uux** ermöglicht es, Unix-Kommandos auf einem Fremdsystem ausführen zu lassen. **uux** ist in der Lage, Dateien auf verschiedenen Systemen zu lesen, das entsprechende Kommando auf dem gewählten Fremdsystem auszuführen und die Standardausgabe dieses Kommandos wiederum in eine Datei auf einem Fremdsystem zu schreiben. Falls die geforderte Kommandoausführung dabei nicht realisiert werden kann, so meldet dies **uux** mit **mail**. Für *kommandozeile* kann dabei eine übliche Unix-Kommandozeile angegeben werden, außer dass dem entsprechendem Kommando oder den angegebenen Dateien *knotenname***!** vorangestellt werden darf; wird vor einem Kommando oder einem Dateinamen kein *knotenname* angegeben, so bezieht sich diese Angabe auf das lokale System.

- **uux "munich3!diff hamburg1!~petersen/hexd.c !hdump.c >!vergl":** überträgt `hdump.c` (im Working-Directory des lokalen Systems) auf Rechner `munich3`; zusätzl. wird noch `/home/petersen/hexd.c` vom Rechner `hamburg1` auf den Rechner `munich3` übertragen. Anschließend wird das Kommando **diff** auf dem Rechner `munich3` gestartet, um diese beiden so eben dorthin kopierten Dateien zu vergleichen. Die Ausgabe dieses Kommandos wird dann in die Datei `vergl` auf dem lokalen System geschrieben.
- **uux "!diff munich3!~alfons/hexd.c hamburg1!/user3/petersen/hdump.c >!vergleich"**: vergleicht die Dateien `hexd.c` (im Home-Directory von `alfons` auf Knoten `munich3`) und `hdump.c` (im Directory `/user3/petersen` auf Knoten `hamburg1`) und schreibt Vergleichsergebnis in Datei `vergleich` (Home-Directory des Aufrufers). Das Vergleichskommando **diff** wird dabei auf dem lokalen System ausgeführt.
- **cat namliste | sort | nl | uux -p frankfurt3!lpr**
  gibt den Inhalt der Datei `namliste` sortiert und mit vorangestellten Zeilennummern auf einem Drucker des Knotens `frankfurt3` aus.

uulog – Ausgeben von UUCP-Logdateien

uulog [*optionen*]

Wenn **uucp** Dateien zwischen Unix-Systemen überträgt, schreibt es alle anfallenden Aktionen in Log-Datei (`/var/spool/uucp/.Log/uucico`); gilt auch für Kommando **uux** (`/var/spool/uucp/.Log/uuxqt`). Mit dem Kommado **uulog** kann nun der Inhalt solcher Log-Dateien angezeigt werden.

uuglist – Ausgeben der verfügbaren UUCP-Grades (Prioritäten)

uuglist [*optionen*]

Bei System V.4 wurden sogenannte *Grades* eingeführt, mit denen Kopierjobs eine Priorität zugeteilt werden kann. Ein *Grade* ist dabei entweder eine symbolische Bezeichnung (vom Systemadministrator festgelegt) oder ein Buchstabe (`A` bis `Z` und `a` bis `z`). Dabei steht der Buchstabe `A` für die höchste und `z` für die niedrigste Priorität. Die Voreinstellung ist, dass alle Kopieraufträge die Priorität `Z` haben. Elektronische Mail, die mit UUCP übertragen wird, hat meist eine höhere Priorität (z.B. `D`). Mit dem Kommando **uuglist** kann man sich die am System verfügbaren Grades anzeigen lassen.

-u	die hier ausgegebenen Grades darf der Benutzer bei **uucp** und **uux** mit Option **-g** *grade* angeben.

```
$ uuglist –u  (Enter)
high
low
medium
```

Sollten keine symbolischen Bezeichnungen (wie `low`, `medium` und `high`) existieren, dann muss ein Buchstabe verwendet werden. In diesem Fall erscheint beim **uuglist**-Aufruf ein entsprechender Hinweis.

cu – Koppeln eines lokalen Rechners mit anderem Netzrechner

cu [*optionen*] *telefonnummer* | *knotenname*

Das Kommando **cu** stellt eine Verbindung zwischen einem lokalen Rechner und einem "entfernten" Rechner her; es ermöglicht somit, dass ein Benutzer gleichzeitig an beiden Rechnersystemen angemeldet ist. Das bedeutet, dass er zwischen beiden Rechnersystemen hin- und herschalten kann und so auf beiden Rechnern Kommandos ausführen oder aber Dateien zwischen diesen beiden austauschen kann:

~.	Verbindung zum Fremdsystem abbrechen.
~!	Vorübergehend nur auf dem lokalen System arbeiten; in die Verbindung zum Fremdsystem kann mit (Strg)-D wieder zurückgekehrt werden.
~!*unix_kdo*	*unix_kdo* auf dem lokalen System ausführen.
~$*unix_kdo*	*unix_kdo* auf dem lokalen System ausführen, dessen Ausgabe aber an das andere System schicken.
~%cd	**cd**-Kommando auf dem lokalen System ausführen.
~%take *q* [*z*]	kopiert die Datei *q* auf dem Fremdsystem in die Datei *z* auf dem lokalen System. Fehlt die Angabe von *z*, so wird für *z* der Pfadname von *q* genommen.
~%put *q* [*z*]	kopiert die Datei *q* des lokalen Systems in die Datei *z* auf dem Fremdsystem. Fehlt die Angabe von *q*, so wird für *z* der Pfadname von *q* genommen.

ct – Remote-Callback eines Terminals

ct [*optionen*] *telnr*....

Mit dem Kommando **ct** (*call terminal*) kann man sich zurückrufen lassen, z.B. um ein Terminal mit dem Rechner in der Firma zu verbinden, die die Telefonkosten für die Verbindung übernehmen soll. In der Telefonnummer *telnr* sind neben den Ziffern noch folgende Angaben erlaubt:

=	bedeutet "Warten auf einen zweiten Wählton"
-	bedeutet "4 Sekunden warten, bevor weiter zu wählen ist"

Wenn mehrere Telefonnummern angegeben sind, probiert **ct** diese der Reihe nach durch, bis eine Verbindung zustandekommt.

11 PostScript-Programme

gs – Konvertieren von PostScript- und PDF-Dateien

gs (GhostScript) kann PostScript- und PDF-Dateien in eine Vielzahl von Druckformaten umwandeln. Das Programm verwendet dabei die im Directory `/usr/share/ghostscript/fonts/` liegenden Dateien, um PostScript-Fonts in eine Bitmap-Darstellung zu konvertieren. Nahezu alle Drucksysteme unter Linux verwenden GhostScript, um die Postscript-Daten in das Format des jeweiligen Druckers zu konvertieren.

Man kann **gs** allerdings auch explizit aufrufen. Z.B. konvertiert der folgende Aufruf die Datei `skriptum.ps` in das Format *HP-Laserjet 4* und schreibt das Ergebnis in die Datei `skriptum.hp`:

```
$ gs -sDEVICE=ljet4 -sOutputFile=skriptum.hp \(Enter)
>    -sPAPERSIZE=a4 -dNOPAUSE skriptum.ps quit.ps (Enter)
```

gs arbeitet interaktiv und wartet nach der Konvertierung der letzten Seite auf eine Benutzereingabe. Hier muss man dann (Strg)+D eingeben. Möchte man, dass sich **gs** nach der letzten Seite automatisch beendet, muss man nur `quit.ps` als letzte Datei beim Aufruf angeben.

Die Option **-dNOPAUSE** verhindert, dass **gs** nach jeder Konvertierung einer Seite auf eine Bestätigung durch den Benutzer (durch Eingabe von (Enter)) wartet. Eine Liste aller möglichen Angaben bei **-sDEVICE=** kann man sich mit **gs -h** anzeigen lassen. Mit der Zusatzoption **-r** kann man bei einem **gs**-Aufruf die gewünschte Auflösung in DPI (*dots per inch*) angeben, wie z.B. **-r300**.

Zu **gs** stehen auch graphische Oberflächen zur Verfügung, wie z.B. **ghostview**: (Ursprüngliche graphische Bedienoberfläche zu **gs**.), **gv** (Komfortablere graphische Bedienoberfläche zu **gs**), **kghostview** (KDE-Variante zu **ghostview**) und **ggv** (Gnome-Variante zu **ghostview**).

Zum Anzeigen von PDF-Dateien stehen die beiden Programme **acroread** und **xpdf** zur Verfügung.

a2ps - Umwandeln von Textdatei in Postscript

a2ps [*optionen*] *textdate i* **-o** *psdatei*

Das Programm **a2ps** (*Any to PostScript*) kann viele Dateitypen (ASCII, HTML usw.) in PostScript umwandeln, wobei es aber meist zur Konvertierung von Textdateien (insbesondere C-Programme, Java-Programme, Shell-Skripts usw.) herangezogen wird, um diese dann entsprechend formatiert an einem PostScript-Drucker auszugeben.

Die Voreinstellung ist, dass **a2ps** den Text im Querformat mit zwei Spalten ausgibt, wobei die beiden Spalten automatisch mit Rahmen und Überschriften versehen sind. Bei Programmen führt **a2ps** eine entspr. Syntaxhervorhebung (Schlüsselwörter fett, Kommentare kursiv usw.) durch. So würde der folgende Aufruf das C-Programm `list.c` in PostScript umwandeln und Ergebnis in die Datei `list.ps` schreiben: **a2ps list.c -o list.ps**.

dvips - Umwandeln von DVI-Dateien in Postscript

dvips [*optionen*] *dvidatei*

Das Kommando **dvips** wandelt eine DVI-Datei (durch **latex**-Aufruf erzeugt) in eine Postscript-Datei um.

-A bzw. **-B**	nur ungerade bzw. nur gerade Seiten umwandeln.
-l *n* bzw. **-p** *n*	beendet bzw. beginnt die Umwandlung nach bzw. ab der *n*-ten Seite.
-pp *liste*	wandelt nur in *liste* angegebenen Seiten um. In *liste* sind auch Bereiche erlaubt, wie z.B. **-pp 2,5-11,14-18,22**.
-o *datei*	schreibt das Ergebnis der Umwandlung in die Datei *datei*.

enscript - Umwandeln von Textdateien in PostScript

enscript [*optionen*] *textdatei* **-p** *psdatei*

Das Programm **enscript** leistet Ähnliches wie das Programm **a2ps**. Der folgende Aufruf würde z.B. das C-Programm `bruchrec.c` in PostScript umwandeln und das Ergebnis in die Datei `bruchrec.ps` schreiben: **enscript --columns=3 -E -r -j bruchrec.c -p bruchrec.ps**.

--columns=3	dreispaltige Ausgabe.	**-r**	Ausgabe im Querformat.
-E	Syntaxhervorhebung (*Syntax-Highlighting*).	**-j**	Rahmen um Spalten zeichnen.

psutils - Programmpaket zur Manipulation von Postscript-Dateien

Nachfolgend werden die wichtigsten Programme des **psutils**-Pakets kurz vorgestellt:

psnup - Anordnen mehrerer Seiten (verkleinert) auf einem Blatt

Dieses ist eines der nützlichsten Programme des **psutils**-Pakets. Mit **psnup** kann man mehrere Seiten auf einem Blatt ausgeben. So sind z.B. oft viele Unterrichts-Skripten oder -Folien mit sehr großen Schriften gesetzt, so dass man den Text auch noch gut lesen kann, wenn man zwei oder vier Seiten auf ein Blatt drucken lässt. **psnup** hat die folgende Aufrufsyntax: **psnup** [*optionen*] *input.ps output.ps*
Nachfolgend sind einige mögliche Aufrufe zu **psnup** gezeigt:

- **psnup -4 -d folien.ps folienklein.ps**: erzeugt aus Datei `folien.ps` eine Datei `folienklein.ps`, in der sich vier Seiten auf einem Blatt befinden. Option **-d** bewirkt, dass um jede Seite ein Rahmen gezeichnet wird.
- **psnup -32 -b0.1cm -d buch.ps buchvorschau.ps**: erzeugt aus `buch.ps` eine Datei `buchvorschau.ps`, in der sich 32 Seiten auf einem Blatt befinden. Die Option **-b0.1cm** bewirkt, dass die einzelnen Seiten mit einem Rand von 0,1 cm voneinander getrennt werden.

psresize - Verändern der Papiergröße

Hiermit kann man das Papierformat einer PostScript-Datei ändern. Dieses bietet sich z.B. an, um Texte von amerikanischen Autoren auf einem A4-Drucker ausgeben zu können:

- **psresize -Pletter -pa4 input.ps output.ps**
  Dieser Aufruf setzt das *letter*-Format in das in Deutschland verwendete *A4*-Format um. Die Option **-pA4** kann hier eigentlich entfallen, da **psresize** standardmäßig das A4 Format benutzt. Das Programm kennt z.B. die Papierformate: *a3*, *a4*, *a5*, *b5*, *letter*, *legal*, *tabloid*, *statement*, *executive*, *folio*, *quarto* und *10x14*.

psselect - Extrahieren einzelner Seiten

Dieses Programm ermöglicht es, aus einer PostScript-Datei bestimmte Seiten zu extrahieren. Die gewünschten Seiten können mit folgenden Optionen ausgewählt werden:

-e bzw. **-o**	alle Seiten mit einer geraden bzw. mit einer ungeraden Seitennummer
-p*pages*	*pages* ist eine durch Kommas getrennte Liste der gewüschten Seitenbereiche.

Um ersten drei Seiten, die Seiten 7 bis 11, die Seiten 13 bis 20 und ab Seite 50 alle restlichen Seiten aus `input.ps` in `output.ps` zu kopieren, müsste man folg. aufrufen: **psselect -p-3,7-11,13-20,50- input.ps output.ps**.

pstops - Umsortieren der Seiten

Dieses Kommando erlaubt das Umsortieren der Seiten in einer PostScript-Datei:
pstops [*optionen*] *neueanordnung input.ps output.ps*

psbook - Anordnen der Seiten für den Druck eines Buches

Dieses Programm erlaubt es, die Seiten einer PostScript-Datei so anordnen zu lassen, dass ganze Bögen (etwa mit je 16 Seiten) für das Drucken eines Buches entstehen: **psbook** [*optionen*] *input.ps output.ps*

epsffit - Anpassen der Druckbildgröße einer PostScript-Datei

Dieses Programm verändert die Größe des Druckbilds einer EPS-Datei, so dass es in einen vorgegebenen Rahmen (*bounding box*) passt: **epsffit** [*optionen*] *lux luy rox roy input.ps output.ps*
Die linke untere Ecke wird dabei mit (*lux*,*luy*) und die rechte obere Ecke mit (*rox*,*roy*) in PostScript-Einheiten (Punkten) festgelegt.

extractres, *includeres* - Extrahieren und Einfügen benötigter Ressourcen

Das Kommando **extractres** analysiert eine PostScript-Datei und liefert alle benötigten Ressourcen (Fonts, Dateien usw.). Die extrahierten Ressourcen werden im gleichen Directory unter entsprechenden Namen abgelegt. Das Kommando **includeres** fügt die mit **extractres** ermittelten Ressourcen in die PostScript-Datei ein. Der folgende Kommandoaufruf bewirkt, dass alle Ressourcen einer PostScript-Datei in den Dokument-Prolog aufgenommen werden: **extractres input.ps | includeres >output.ps**

Weitere psutils-Programme

Folgende Programme des **psutils**-Pakets erlauben die Anpassung von PostScript-Dateien, die mit anderen Werkzeugen erstellt wurden, an die **psutils**-Konventionen:

fixdlsrps	DviLaser/PS-Dokumente
fixfmps	Framemaker-Dokumente
fixmacps	MacIntosh-Dateien
fixnt	PostScript-Dokumente unter Windows NT (3.5, 4.0)
fixscribeps	Scribe-Dokumente
fixtpps	Troff/Tpscript-Dokumente
fixwfwps	MS-Word-Dokumente
fixwpps	Word-Perfect-Dokumente
fixwwps	MS-Write-Dokumente

mpage - Umwandeln von Text- bzw. PostScript-Dateien in PostScript

mpage [*optionen*] *eingabedatei(en)* > *psdatei*

Das Kommando **mpage** hat zwei Vorteile: Erstens kann es bis zu acht Seiten auf einem Blatt drucken und zweitens akzeptiert **mpage** als Eingabe nicht nur Text-, sondern auch Postscript-Dateien. Somit kann eine existierende Postscript-Datei neu formatiert werden, wie z.B. zwei Seiten auf ein Blatt.

-1, -2, -4, -8	eine, zwei, vier oder acht Seiten auf einem Blatt; bei zwei oder acht Seiten wird das Querformat verwendet. Voreinstellung sind vier Seiten auf einem Blatt.
-bA4	Ausgabe im Din-A4-Format; Voreinstellung ist US-Letter.
-C ISO-Latin.1	wird für den Ausdruck von deutschen Sonderzeichen benötigt.
-f	lange Zeilen auf mehrere Zeilen verteilen; Voreinstellung ist Abschneiden.
-j *n-m*	nur die Blätter *n* bis *m* drucken; Voreinstellung ist: alle Blätter.
-l	im Querformat drucken; Voreinstellung ist Hochformat.

Möchte man sich z.B. die typischen deutschen Optionen automatisch einschalten lassen, muss man in der Datei `~/.profile` nur folgendes eintragen: `export MPAGE='-A -C ISO-Latin.1'`.

dvilj - Umwandeln von DVI-Dateien in Laserjet-Format

dvilj [*optionen*] *dvidatei*

dvilj wandelt DVI-Datei (durch **latex**-Aufruf erzeugt) in Datei mit Laserjet-Format um. Als Ergebnis liefert dieser Aufruf eine gleichnamige Datei mit der Endung `.lj`, die für den Ausdruck an einem 300-DPI-HP-Laserjet-Drucker geeignet ist. Weitere verwandte Kommandos sind u.a. **dvilj2p**, **dvilj4**, **dvilj4l** und **dvilj6**, die die entsprechenden Modelle (einschließlich 600 DPI und eingebauter TrueType- und Intellifont-Zeichensätze) unterstützen.

html2ps - Umwandeln von HTML-Dateien in PostScript

html2ps [*optionen*] *htmldatei* > *psdatei*

-D	bewirkt, dass DSC-konforme Kommentare in die PostScript-Datei eingefügt werden, was für die Weiterverarbeitung der PostScript-Datei (wie z.B. zum Einfügen in andere Dokumente) sehr wichtig ist.

pdf2ps, ps2pdf - Umwandeln von PDF-Dateien in PostScript bzw. umgekehrt

- **pdf2ps datei.pdf datei.ps**: konvertiert die PDF-Datei `datei.pdf` in das PostScript-Format (`datei.ps`).
- **ps2pdf datei.ps datei.pdf**: konvertiert die PostScript-Datei `datei.ps` in das PDF-Format (`datei.pdf`).

Beide Programme verwenden das GhostScript-Programm **gs** zum Konvertieren.

12 Systemkommandos

shutdown – Herunterfahren des Systems

shutdown [*optionen*] *zeitpunkt* [*nachricht*]

Das Kommando **shutdown** fährt das System zu dem angegebenen *zeitpunkt* herunter. Für *zeitpunkt* muss dabei entweder eine Uhrzeit (*hh*:*mm*), die Anzahl der Minuten von jetzt an (+*m*) oder **now** für sofort angegeben werden. **shutdown** informiert alle Benutzer (mit einer eigenen Nachricht bzw. mit der angegebenen *nachricht*) darüber, dass das System bald heruntergefahren wird, und lässt keine neuen Anmeldungen mehr zu.

-c	bricht einen gestarteten shutdown-Vorgang ab, wenn dies noch möglich ist.
-f, -r	starten nach dem Herunterfahren das System neu, wobei **–f** schneller ist.
-n	fährt das System sehr schnell herunter, indem der Init-V-Prozess umgangen wird.
-h	hält das System nach dem Herunterfahren an, so dass keine Eingaben mehr möglich sind und der Rechner nur noch ausgeschaltet werden kann bzw. sich auch automatisch selbst ausschaltet.
-t *sek*	legt fest, wie viele Sekunden zwischen der Warnnachricht an die Benutzer und dem Herunterfahren zu warten ist. Voreinstellung ist 20 Sekunden.

Unter Linux lässt sich das System auch meist mit (Strg)-(Alt)-(Entf) herunterfahren.

reboot – Beenden aller Prozesse und Neustarten des Systems

reboot [*optionen*]

Das Kommando **reboot** beendet alle laufenden Prozesse und startet das System dann neu. Befindet sich das System im Runlevel 1 bis 5, wird dazu **shutdown** aufgerufen.

halt – Beenden aller laufenden Prozesse

halt [*optionen*]

Das Kommando **halt** beendet alle laufenden Prozesse. Das System wird damit angehalten und reagiert nicht mehr auf Eingaben.

dmesg – Anzeigen der Boot-Meldungen des Kernels (unter Linux)

dmesg

Die Kernelmeldungen während des Bootvorgangs kann man sich später mit dem Kommando **dmesg** anzeigen lassen. Dies ist z.B. hilfreich, wenn man nachträglich wissen möchte, welche Hardware erkannt wurde.

13 Online-Hilfe

man - Traditionelle Online-Hilfe für Unix

man [*optionen*] [*bereich*] *thema*

man sucht die als *thema* angegebene Manual-Datei in allen ihm bekannten man-Directories (kann über die Variable `MANPATH` in `~/.profile` festgelegt werden). Mit optionaler Angabe von *bereich* kann Suche auf bestimmten Bereich eingeschränkt werden. Das **man** von Linux beispielsweise kennt die Themenbereiche **1** bis **9** und **n**:

1	Benutzerkommandos	**6**	Spiele
2	Systemaufrufe	**7**	Diverses
3	Funktionen der Programmiersprache C	**8**	Kommandos zur Systemadministration
4	Dateiformate, Gerätedateien	**9**	Kernel-Funktionen
5	Konfigurationsdateien	**n**	Neue Kommandos

In vielen Unix-Dokumentationen und -Büchern wird zum Kommando- oder Funktionsnamen oft noch die entsprechende Bereichsnummer – wie etwa `ls(1)` oder `fopen(3)` – angegeben. Um nun nicht alle man-Directories durchsuchen zu lassen, kann man Suche auf einen Bereich eingrenzen, wie z.B. **man 1 ls** oder **man 3 fopen** . Das Kommando **man** gibt die Informationen seitenweise am Bildschirm aus. Mit (Enter) kann man zeilenweise und mit dem (Leertaste) seitenweise vorwärts blättern. Mit der Eingabe von q wird **man** beendet. Die Beschreibung eines Kommandos im gedruckten Handbuch oder im Online-Manual nennt man *Manpage*.

-a	zeigt der Reihe nach alle gleichnamigen man-Seiten an. Nachdem man den ersten man-Text gelesen hat, muss man die Taste q drücken und man gelangt zum nächsten man-Text, der das angegebene *thema* betrifft. Ohne diese Option wird gewöhnlich nur die erste von mehreren gleichnamigen Dateien aus unterschiedlichen Themengebieten angezeigt. Auf manchen Systemen ist **-a** die Voreinstellung.
-k *schlüsselwort*	zeigt eine Liste aller man-Seiten an, in denen das *schlüsselwort* vorkommt; entspr: **apropos** *schlüsselwort*
-f *schlüsselwort*	zeigt die Bedeutung des *schlüsselworts* in Form eines einzeiligen Textes an; entspr: **whatis** *schlüsselwort*

Welche Tasten zum interaktiven Blättern und Suchen in einem angezeigten man-Text zur Verfügung stehen, hängt vom eingestellten Programm zur seitenweise Anzeige des man-Textes ab. Unter Linux ist das meist **less**, und wichtige Tasten zum interaktiven Bedienen von **man** sind dort:

q	man-Text verlassen
(Leertaste), b und auch (Bild unten), (Bild oben)	eine Seite weiter- bzw. zurückblättern
(Pfeil o), (Pfeil u)	Text nach oben/unten verschieben
(Pos1), (Ende) und auch g, G	an Anfang/Ende des man-Textes springen
/*text* (Enter), ?*text* (Enter)	nach unten bzw. nach oben nach dem String *text* suchen
n, N	Letzte Suche in gleiche/umgekehrte Richtung wiederholen
h	Hilfstext zu weiteren Tastenkürzeln anzeigen

Die Manpages sind wie folgt gegliedert:

NAME	Name und Kurzbeschreibung des Kommandos	EXIT CODES	Rückgabewerte des Kommandos
SYNOPSIS	Syntaxbeschreibung des Kommandos	SEE ALSO	Hinweise auf verwandte Kommandos
DESCRIPTION	ausführliche Beschreibung des Kommandos	DIAGNOSTICS	Fehlermeldungen des Kommandos
OPTIONS	Bedeutung der Optionen und Argumente	WARNINGS	Einschränkungen oder andere Hinweise
FILES	Dateien, die das Kommando benutzt	AUTHOR	Autor des Programms
EXAMPLES	Anwendungsbeispiele zum Kommando	BUGS	bekannte Fehler
NOTES	allgemeine Hinweise		

Will man sich die Manualpage zu **man** selbst ausgeben lassen, muss man nur **man man** aufrufen. Es existieren auch grafische Oberflächen zum **man**-Kommando, wie z.B. **xman** und das Tcl/Tk-Programm **tkman**.

Falls Optionen **-k** und **-f** bzw. **whatis** und **apropos** unter Linux nicht funktionieren, fehlt wahrscheinlich die Datenbank mit Inhaltsangaben zu man-Texten. Dann ist zuvor noch **makewhatis** vom Superuser aufzurufen.

apropos - Suchen von Schlüsselwörtern in den man-Seiten

apropos *schlüsselwort*

Das Kommando **apropos** listet die Namen aller man-Seiten auf, in denen das angegebene *schlüsselwort* vorkommt. **apropos** *schlüsselwort* entspricht dem Aufruf **man -k** *schlüsselwort.*

whatis - Kurzbeschreibung zu einem Kommando bzw. Schlüsselwort

whatis [*optionen*] *schlüsselwort*

Das Kommando **whatis** zeigt die Bedeutung des *schlüsselworts* in Form eines einzeiligen Textes an. **whatis** *schlüsselwort* entspricht dem Aufruf **man -f** *schlüsselwort* aufrufen.

info - Online-Manual von GNU

info [*begriff*]

Das Kommando **info** ist das bevorzugte Help-System für die umfangreiche bei Linux mitgelieferte GNU-Software. Die wichtigsten Tasten zur interaktiven Bedienung von **info** sind:

(Leertaste)	Nach unten blättern
(Backspace)	Nach oben blättern
b, e	An Anfang/Ende des info-Textes springen
(Tab)	Cursor zum nächsten Querverweis
(Enter)	Zum info-Text wechseln, auf den der aktuelle Querverweis zeigt
h	Ausführliche Bedienungsanleitung zu **info** anzeigen
?	Kommandoübersicht anzeigen
(Strg)+x,0	Hilfsfenster wieder schließen

14 Sonstige Kommandos

alias - Vergeben von Kurznamen an Kommandos

alias [*kurzname* [=*kommando*]]

alias definiert einen Kurznamen (*Alias*) für einen Kommandoaufruf, wie z.B.

```
alias m='less '
alias ll='ls -CF '
```

Wird **alias** ohne jegliche weitere Angaben aufgerufen, gibt es alle momentan definierten Kurznamen mit den zugehörigen Kommandozeilen aus. Wird **alias** nur mit einem Kurznamen aufgerufen, so gibt es den zu diesem Kurznamen definierten Kommandoaufruf aus.

Üblicherweise definiert man mit **alias** alle benötigten Kurzformen in der Datei `~/.alias`, um nützliche Kurzformen nicht immer wieder neu definieren zu müssen. Diese Datei wird automatisch bei jedem Anmelden gelesen.

Um ein Alias zu löschen, muss man nur **unalias** *kurzname* aufrufen.

banner - Zeichenketten (Strings) in Spruchband-Form ausgeben

banner *string(s)*

Das Kommando **banner** gibt die angegebenen *string(s)* in Spruchband-Form auf die Standardausgabe aus. Jeder angegebene einzelne *string* wird dabei in einer (Groß-)Zeile ausgegeben. Die maximale Anzahl von Zeichen, die in eine solche (Groß-)Zeile passen, hängt vom Bildschirm ab; für einen 80 spaltigen Bildschirm ist dieses Maximum 10 Zeichen. Um mehrere Wörter in einer (Groß-)Zeile ausgeben zu lassen, sind diese mit " .. " zu klammern. Unter Linux ist das Kommando **banner** nicht in allen Distributionen enthalten. Einige **banner**-Versionen geben die *string(s)* von oben nach unten aus.

basename / dirname – Basis- bzw. Directoryname eines Pfads

basename *string* [*suffix*]
dirname *string*

Jede Datei hat einen so genannten Basisnamen und einen Pfadnamen. Der Basisname ist der Name, der im entsprechenden Directory zu dieser Datei eingetragen ist. So ist z.B. beim Pfadnamen `/home/egon/uebung1/laender` der Basisname `laender`. Der Pfadname gibt dabei den Pfad vom Root-Directory zu dieser Datei an.

- **basename** entfernt nun aus dem angegebenen *string* (Pfadname) alles von Beginn bis einschließlich dem letzten `/` und gibt dann den Rest, also den Basisnamen aus; so würde z.B. der Aufruf **basename /home/egon/uebung1/laender** die Ausgabe `laender` liefern. Ist ein *suffix* angegeben, so wird auch dieses – falls es am Ende des Basisnamens vorhanden ist – noch vor der Ausgabe vom Basisnamen entfernt; so würde z.B. der Aufruf **basename /home/egon/uebung1/add1.c .c** die Ausgabe `add` liefern.
- Das Kommando **dirname** ist das Gegenstück zum Kommando **basename**: Es gibt zu dem angegebenen *string* (Pfadname) nur den Directory-Pfad ohne den Basisnamen aus. So liefert z.B. der Aufruf **dirname /home/egon/uebung1/laender** die Ausgabe `/home/egon/uebung1` und der Aufruf **dirname add1.c** würde den Punkt (**.**) als Ausgabe liefern.

bc - Taschenrechner

bc [*optionen*] [*datei(en)*]

Sind *datei(en)* angegeben, liest **bc** zunächst deren Inhalt, bevor er von der Standardeingabe liest, ansonsten liest **bc** sofort von der Standardeingabe. **bc** beendet sich, wenn es EOF ((Strg)-D) oder **quit** liest.

-c	(*compile only*) Ausgabe wird nicht an **dc** weitergeleitet, sondern auf die Standardausgabe geschrieben. **dc** ist ein rudimentärer Taschenrechner ist, der Eingaben in Postfix-Schreibweise entgegennimmt.
-l	(*library*) Aus einer mathematischen Bibliothek werden Funktionen geladen, die dann aufgerufen werden können (siehe Bibliotheksfunktionen unten).

- *Variablen:* **bc** bietet 26 Variablen (Kleinbuchstaben a, b,.., z) an. Der Wert eines Ausdrucks wird mit Zuweisung (=) in den Variablen gespeichert. Bei einem Ausdruck ohne `=` wird das Ergebnis ausgegeben. Postfix-/Präfix `++` und `--` sind erlaubt.
- *Ausdrücke:* In einem arithmetischen Ausdruck sind ganze und reelle Zahlen sowie folgende Operatoren erlaubt: `( ) ^ * % / + -`. Eine Division von ganzen Zahlen liefert eine ganze Zahl als Ergebnis. Beim Potenzoperator `^` muss die Potenz ganzzahlig sein.
- *Zusammengesetzte Zuweisungsoperatoren*:

`x =+ y` (für `x = x + y`)	`x =- y` (für: `x = x - y`)	`x =* y` (für `x = x * y`)
`x =/ y` (für `x = x / y`)	`x =% y` (für `x = x % y`)	`x =^ y` (für `x = x ^ y`)

- *builtin-Funktionen für arithmetische Ausdrücke:*

length(*A*)	signifikante Ziffernzahl von Ausdruck *A*.
scale(*A*)	Nachkommastellenzahl von Ausdruck *A*.
sqrt(*A*)	Quadratwurzel aus Ausdruck *A*.

- *Skalierungsfaktor:* legt fest, wie viele Nachkommastellen beim Berechnen zu berücksichtigen sind. Über **scale** kann ein Skalierungsfaktor (0..99) festgelegt werden. Voreinstellung: **scale=0**.
- *Zahlensysteme*: **bc** arbeitet normal im Zehnersystem (Basis=10). Um andere Zahlensysteme einzuschalten, sind die Variablen **ibase** (Eingabe) und **obase** (Ausgabe) entsprechend zu setzen.
- *Funktionen*: Als Funktionsname ist ein Kleinbuchstabe anzugeben. Neben 26 Variablen sind noch 26 Funktionen möglich:

```
define funktionsname([parameterliste]) {
   [auto  var1, var2, ..]
   anweisungen
   ::::
}
```

  Aufruf erfolgt mit *funktionsname(p1,..)*. Eine Funktion wird am Ende der Funktion oder bei **return** beendet: **return** (Rückgabe: 0) bzw. **return**(*a*) (Rückgabe: Wert des Ausdrucks *a*). Funktionslokale Variablen werden mit **auto** vereinbart, die automatisch auf 0 gesetzt werden. Rekursive Funktionsaufrufe sind möglich.

- *Eindimensionale Arrays:* Arrays werden automatisch beim Zugriff auf ein Arrayelement angelegt. Zugriff auf ein Arrayelement erfolgt durch Arrayname (Kleinbuchstabe), gefolgt von **[***ausdruck***]**. Ein Array kann als Parameter bei der Funktionsdefition angegeben oder mit **auto** deklariert werden: *array***[]**.
- *Verzweigungen und Schleifen:*
  if, while, for, break
  if (*bedingung*) *anweisung* oder
  if (*bedingung*) { *anweisung1*; *anweisung2*; ... }

  while (*bedingung*) *anweisung* oder
  while (*bedingung*) { *anweisung1*; *anweisung2*; ... }

  for (*ausdr1*;*bedingung*;*ausd2*) *anweisung* oder
  for (*ausdr1*;*bedingung*;*ausdr2*) { *anw1*; *anw2*; ... }

  break bewirkt das Verlassen von **for**-/**while**-Schleifen.

- *Bedingungen:* a<b, a>b, a<=b, a>=b, a==b, a!=b.
- *Mathematische Bibliotheksfunktionen:* Mit **bc -l** werden aus einer mathematischen Bibliothek (wie z.B. /usr/lib/lib.b) bestimmte Funktionen geladen: Durch die Option **-l** wird **scale** auf **20** gesetzt.

s(*x*), c(*x*), a(*x*)	Sinus, Cosinus, Arcustangens von *x*
l(*x*), e(*x*)	nat. Logarithmus von *x* bzw. $\mathbf{e}^x$
j(*n*,*x*)	Bessel-Funktion

- *Kommentare und Textausgaben:* Kommentare müssen mit /* .. */ geklammert sein und auszugebender Text ist in ".." anzugeben.

```
$ bc (Enter)
34+19 (Enter)
53
1234*81728 (Enter)
100852352
2^3^4 (Enter)   [entspricht: 2^(3^4) = 2^81; Auswertung erfolgt von rechts nach links]
241785163922925834941235 2
2^(3^4) (Enter)
241785163922925834941235 2
(2^3)^4 (Enter) [entspricht: 8^4]
4096
8+4*5 (Enter)
28
(8+4)*5 (Enter)
60
8*5/4 (Enter)   [entspricht: (8*5)/4; Auswertung erfolgt von links nach rechts]
10
b=4 (Enter)
b (Enter)
4
b=b+17 (Enter)
b (Enter)
21
sqrt(15) (Enter)
3
a=sqrt(1243) (Enter)
a (Enter)
35
sqrt(sqrt(15)+a) (Enter)
6
scale (Enter)
0
1/3 (Enter)
0
scale=scale+10 (Enter)
1/3 (Enter)
.3333333333
++scale (Enter)
11
1/3 (Enter)
.33333333333
scale=2 (Enter)
2/3 (Enter)
.66
(Strg)-D
$
```

cal - Kalender zu einem Monat oder einem Jahr ausgeben lassen

cal [[*monat*] *jahr*]

Das Kommando **cal** gibt einen Kalender zu einem bestimmten Monat oder Jahr auf die Standardausgabe aus. Es sind folgende Aufrufe von **cal** möglich:

- ohne jede Argumente: Kalender für den laufenden Monat wird ausgegeben
- mit einem Argument: Kalender für das angegebene jahr (1-9999 möglich) wird ausgegeben
- mit zwei Argumenten: Kalender für den angegebenen *monat* (1. Argument; 1-12 möglich) des angegebenen *jahres* (2. Argument; 1-9999 möglich) wird ausgegeben.

Bitte beachten Sie folgende Hinweise:

- **cal 9 1752** gibt den Kalender für September des Jahres 1752 aus. Das Besondere an dieser Ausgabe ist, dass in diesem Monat 11 Tage fehlen, um den bis dahin benutzten Kalender zu korrigieren.
- **cal 03** gibt den Kalender für das Jahr 3 und nicht für das Jahr 2003 aus. Dies ist ein häufiger Fehler, den Benutzer begehen, wenn sie sich Kalender aus diesem Jahrhundert ausgeben lassen möchten.

Auf manchen Unix-Systemen verhält sich **cal** geringfügig anders:

- ohne jede Argumente: Es wird zunächst Datum und Uhrzeit ausgegeben, bevor dann drei Monate (letztes, dieses und nächstes Monat) ausgegeben werden.
- statt einer Monatszahl kann auch ein Monatsname oder dessen eindeutigen Anfangsbuchstaben angegeben werden, wie z.B. **cal may**, **cal ja** (für Januar), **cal jun 1956** oder **cal april**. Nicht erlaubt, da nicht eindeutig, wäre z.B. **cal ju** (jun oder jul?) oder **cal ma** (mar oder may?).

calendar - Automatisches Erinnern an Termine

calendar [-]

Das Kommando **calendar** sucht im Working-Directory nach einer Datei mit dem Namen `calendar`. Findet es diese Datei, so gibt es aus ihr alle Zeilen aus, in denen das heutige oder morgige Datum vorkommt. An einem Freitag oder an einem Wochenende werden nicht nur die Zeilen ausgegeben, die das heutige und morgige Datum enthalten, sondern auch alle Zeilen, die ein Datum für dieses Wochenende und den darauffolgenden Montag enthalten. **calendar** erkennt jedes Datum, das im amerikanischen Format: *Monat Tag* (wie z.B. `Jan 25` oder `January 25` oder `1/25`) angegeben ist. Hat die Datei `~/calendar` z.B. folgenden Inhalt:

```
Fruehlingsanfang am Mar 21
December 1 : Mit dem Weihnachtseinkauf anfangen
7/19 : Hochzeitstag
Besprechung am Jul 18 (14.30 Uhr, Raum 412)
```

und man ruft nun am 18. Juli das Kommando **calendar**, so würde es folgende Ausgabe liefern:

7/19 : Hochzeitstag
Besprechung am Jul 18 (14.30 Uhr, Raum 412)

Wenn **calendar** – aufgerufen wird, durchsucht das **calendar**-Programm die login-Directories aller Benutzer und stellt gegebenenfalls eine Erinnerung daran den betreffenden Benutzern per Mail zu. Diese Aufrufform wird nur vom Systemadministrator verwendet, wenn er das System so konfiguriert, dass **calendar** regelmäßig zentral gestartet wird. Wenn ein System nicht Tag und Nach durchläuft und so **calendar** nicht automatisch aufgerufen wird, sollte man **calendar** in `~/.profile` aufrufen.

cc - Kompilieren und Linken von C-Programmen (C Compiler)

cc [*optionen*] *datei(en)*

Das Kommando **cc** kompiliert die angegebenen *datei(en)*, welche C-Programmdateien sein müssen, deren Namen mit `.c` enden. Für jede kompilierte C-Programmdatei wird eine Objektdatei erstellt, deren Name aus dem Namen der entsprechenden C-Programmdatei hergeleitet wird, indem statt der Endung `.c` die Endung `.o` genommen wird. Die `.o`-Datei wird normalerweise gelöscht, wenn mit einem **cc**-Aufruf ein einzelnes C-Programm kompiliert und dann sofort auch gelinkt werden soll.

-c	Die angegebenen *datei(en)* werden nur kompiliert; d.h. das Linken wird durch diese Option ausgeschaltet. In diesem Fall werden die von **cc** erzeugten Objektdateien nicht gelöscht.
-o *progname*	Normalerweise erzeugt **cc** eine Programmdatei mit den Namen `a.out`. Wird ein anderer Name für ein Programm, das mit **cc** erzeugt wird, gewünscht, so ist dies mit dieser Option möglich.

crypt - Verschlüsseln und Entschlüsseln von Texten

crypt [*passwort*]

Das Kommando **crypt** liest den zu ver-/entschlüsselnden Text von der Standardeingabe und gibt den ent-/verschlüsselten Text wieder auf die Standardausgabe aus. Ist *passwort* beim Aufruf nicht angegeben, so verlangt **crypt** interaktiv vom Benutzer die Eingabe eines Passworts. Das *passwort* dient als Schlüssel beim Ver- und Entschlüsseln. Aus Sicherheitsgründen wird seit Unix System V.3 dieses Kommando außerhalb der USA nicht mehr zur Verfügung gestellt.

- **crypt geheim <obst >obst.cr**
  Der zu verschlüsselnde Text wird aus der Datei `obst` gelesen. Zum Verschlüsseln wird das Passwort `geheim` verwendet. Der verschlüsselte Text wird in die Datei `obst.cr` geschrieben. Die Datei `obst` könnte nun mit **rm obst** gelöscht werden und somit würde ihr ursprünglicher Inhalt nur noch in verschlüsselter Form (in Datei `obst.cr`) vorliegen. Entschlüsseln kann man dann wieder mit **crypt geheim <obst.cr >obst**.

ctags - Automatische Generierung einer tags-Datei für vi bzw. ex

ctags [*optionen*] *datei(en)*

Das Kommando **ctags** erzeugt aus den vorgegebenen *datei(en)*, die z.B. C-, Pascal-, Fortran-, lex- oder yacc-Programmdateien sein können, eine *tags*-Datei, die beim Arbeiten mit den Editoren **vi** oder **ex** zum Positionieren in bestimmten Dateien verwendet werden kann. Eine *tags*-Datei gibt pro Zeile folgendes (mit Tabulatorzeichen getrennt) an:

name datei adresse

name ist dabei der Name einer Funktion oder eines sonstigen Datenobjekts. *datei* ist der Name der Datei, in dem sich das betreffende Objekt *name* befindet. *adresse* legt die Position fest, an dem sich das betreffende Objekt *name* in Datei *datei* befindet. Für *adresse* wird entweder ein vi-Suchkommando (*/reg.ausdruck*) oder eine Zeilennummer angegeben.

Normalerweise schreibt **ctags** die *tags*-Information in eine Datei mit dem Namen `tags` im Working-Directory. Ist ein anderer Name erwünscht, so muss die Option **-f** benutzt werden.

Für die Funktion `main()` in C-Programmen gilt die Besonderheit, dass vor dem Dateinamen in der *tags*-Datei der Buchstabe `M` angegeben und das Suffix `.c` am Ende entfernt wird. In diesem Fall werden auch alle führenden Pfadangaben beim Dateinamen entfernt. So ist es möglich, dass man sich nur eine *tags*-Datei für mehrere Programme in einem Directory hält.

-a	(*append*) Ausgabe am Ende einer existierenden *tags*-Datei anhängen.
-f *tagsdatei*	(*file*) Normalerweise schreibt **ctags** die *tags*-Information in eine Datei mit dem Namen `tags` im Working-Directory. Ist ein anderer Name erwünscht, so ist dies mit **-f** *tagsdatei* möglich.

- **ctags *.c**
  Zu allen C-Programmen des Working-Directorys eine *tags*-Datei `tags` (im Working-Directory) erstellen.

echo - Ausgeben von Text

echo [*optionen*] [*argument(e)*]

Das Kommando **echo** gibt die angegebenen *argument(e)* auf die Standardausgabe aus. Jedes der angegebenen *argument(e)* wird dabei bei der Ausgabe mit einem Leerzeichen vom nächsten getrennt.

-n	kein Zeilenvorschub nach Ausgabe der argument(e)

- **echo Guten Morgen Hans**
  würde ausgeben: `Guten Morgen Hans`

- **echo 'Guten Morgen Hans'**
  würde ausgeben: `Guten Morgen Hans`

date - Erfragen (bzw. Setzen) des Datums und der Uhrzeit

date [**-s**] *mmttHHMM*[*cc*]*jj* (nur für Superuser erlaubt)
date [*optionen*] [*+format*]

Die erste Aufrufform ist dem Superuser vorbehalten und ermöglicht das Setzen des Datums und der Uhrzeit. Die zweite Aufrufform gibt das Datum und die Uhrzeit aus. Mit *format* kann die gewünschte Form der Ausgabe festgelegt werden.

-u	Datum als GMT (*Greenwich Mean Time*) und nicht als lokale Zeit anzeigen bzw. setzen

Wird **date** ohne Angabe von Argumenten aufgerufen, so wird das heutige Datum und die momentane Uhrzeit ausgegeben, wie z.B. *Thu Aug 2 12:03:34 GMT 1990*.

mm	2-ziffrige Monatsnummer (01-12)
tt	2-ziffrige Tagesnummer (01-31)
HH	2-ziffrige Stundenangabe (00-23)
MM	2-ziffrige Minutenangabe (00-59)
jj	2-ziffrige Jahresangabe (für dieses Jahrhundert)
ccjj	4-ziffrige Jahresangabe, wie z.B. 2003

Anstelle der vollen Angabe *mmttHHMM*[*cc*]*jj* sind auch folgende kurzen Angaben erlaubt: *mmtt* oder *HHMM*; in diesem Fall wird für die weggelassenen Teile das momentane Datum bzw. die momentane Zeit eingesetzt.

Das *format* ist üblicherweise ein String, der mit '.. ' geklammert ist. Die Klammerung bewirkt dabei, dass die ganze *format*-Angabe als ein String aufgefaßt wird. Alle Zeichen, außer die, denen ein **%** vorangestellt ist, werden unverändert ausgegeben; so steht z.B. **%j** für die Tagesnummer des laufenden Jahres und die Angabe **date '+Heute ist der %j.Tag des Jahres'** würde z. B. folgende Ausgabe liefern: *Heute ist der 214.Tag des Jahres*.

Folgende speziellen *format*-Angaben sind u.a. erlaubt:

%a bzw. **%A**	Abgekürzter (Sun-Sat) bzw. voller (Sunday-Saturday) Name des Tages
%b bzw. **%B**	Abgekürzter (Jan-Dec) bzw. voller (January-December) Monatsname
%c	Länderspezifisches Datums- und Zeitformat
%d bzw. **%e**	Tag des Monats (01-31) bzw. (1-31);
%D	Datum im Format **%m/%d/%y**
%H bzw. **%I**	Stunde (00-23) bzw. (01-12)
%j bzw. **%m**	Tag (001-366) bzw. Monat (01-12) im Jahr
%M bzw. **%S**	Minute (00-59) bzw. Sekunden (00-61)
%n bzw. **%t**	Neuezeile- bzw. Tabulatorzeichen
%p	Ausgabe von **AM** bzw. **PM**
%r	Zeit im Format **%I:%M:%S %p**
%R bzw. **%T**	Zeit im Format **%H:%M** bzw. **%H:%M:%S**
%U	Nummer der Woche im Jahr (00-53); Sonntag gilt als erster Wochentag
%w	Tag der Woche (0-6); 0 ist Sonntag
%W	Nummer der Woche im Jahr (00-53); Montag gilt als erster Wochentag
%x bzw. **%X**	Länderspezifisches Datums- bzw. Zeit-Format
%y bzw. **%Y**	Jahr im Jahrhundert (00-99) bzw. volle Jahresangabe (4 Ziffern)
%Z	Name der Zeitzone
%%	Prozentzeichen

Die Kommandozeile **date '+Datum: %d.%m.%y%n Zeit: %H:%M.%S'** könnte z.B. folgende Ausgabe liefern:

```
Datum: 03.08.93
 Zeit: 12:31.56
```

factor - Durchführen einer Primfaktorzelegung

factor [*n*]

Das Kommando **factor** zerlegt ganze Zahlen in ihre Primfaktoren. Wird beim Aufruf von **factor** eine ganze Zahl *n* angegeben, so wird die Primfaktorzerlegung zu dieser Zahl ausgegeben.

Wird **factor** ohne das Argument *n* aufgerufen, so wartet es auf die Eingabe einer ganzen Zahl. Nach der Eingabe einer ganzen Zahl, gibt es dann die Primfaktorzerlegung zu dieser Zahl aus und wartet auf die Eingabe der nächsten ganzen Zahl, zu der eine Primfaktorzerlegung durchzuführen ist. **factor** kann hierbei beendet werden, wenn entweder 0 oder ein anderes Zeichen als eine Ziffer eingegeben wird. Unter Linux exsitiert das Programm **primes**, mit dem man sich Primzahlen ausgeben lassen kann.

primes – Ausgeben von Primzahlen (unter Linux)

primes [*startwert* [*endwert*]]

Mit dem Kommando **primes** kann man sich die Primzahlen aus einem Zahlenbereich ausgeben lassen. Wird kein *startwert* und kein *endwert* beim Aufruf angegeben, liest **primes** den *startwert* von der Standardeingabe. Fehlt die Angabe von *endwert*, wird hierfür meist die größte darstellbare ganze Zahl angenommen.

tee - Sichern der Daten, die durch eine Pipe geleitet werden (T-Stück)

tee [*optionen*] [*datei(en)*]

Das Kommando **tee** leitet die Standardeingabe an die Standardausgabe weiter, wobei es eine Kopie der weitergeleiteten Daten in den eventuell angegebenen *datei(en)* sichert. Das heißt also: Wenn *datei(en)* angegeben sind, so wird das Zwischenergebnis aus der Pipe in diese *datei(en)* kopiert. Sind keine *datei(en)* angegeben, so hat **tee** keine Auswirkung.

Bei dem Aufruf **ls | wc –w** würde nur die Anzahl der Dateien ausgegeben, aber nicht die Dateinamen selbst, da die Ausgabe von **ls** vom **wc**-Kommando "geschluckt" wird. Dagegen würde z.B. der Aufruf

ls | tee dliste | wc -w

zwar auch die Ausgabe des **ls**-Kommandos an das Kommando **wc** weiterleiten, allerdings würde er zusätzlich noch die Ausgabe des **ls**-Kommandos in die Datei `dliste` schreiben: Der folgende Aufruf

ls –l | tee dateiliste

würde die Dateien des Working-Directorys im Langformat nicht nur am Bildschirm auflisten, sondern gleichzeitig auch noch in die Datei `dateiliste` schreiben.

-i	Interrupts (wie z.B. Unterbrechungs-Tasten) ignorieren
-a	Zwischenergebnisse an den alten Inhalt der *datei(en)* anhängen; die Voreinstellung ist: alten Inhalt überschreiben.

line - Lesen einer Zeile von der Standardeingabe

line

line liest von der Standardeingabe nur eine Zeile und gibt diese auf die Standardausgabe aus. Anwendungen für **line** werden im Buch "*Linux--Unix-Shells*" vorgestellt. Auf manchen Unix-Systemen, wie z.B. auch auf Linux, wird das Kommando **line** nicht angeboten. Auf diesen Systemen empfiehlt es sich, ein Alias in der Datei `.profile` oder `.alias` (beide im Home-Directory) der folgenden Form zu definieren: **alias line='head -1'**.

mtools - Zugriff auf MS-DOS-Disketten und -Festplatten (unter Linux)

mtools ist ein Programm, das einen einfachen und leichten Zugriff auf Disketten im MS-DOS-Format ermöglicht, ohne das man diese mit **mount** anmontieren muss. Die **mtools**-Kommandos (**mdir**, **mtype**, **mcd** usw.) sind durch Links auf das zentrale Programm **mtools** realisiert. **mtools** selbst kann nicht ausgeführt werden, sondern nur über diese Links. Alle **mtools**-Kommandos haben einige gemeinsame Merkmale:

1. Laufwerksangaben erfolgen wie unter DOS mit `A:`, `B:`, `C:` usw. Ist kein Laufwerk angegeben, greifen die Kommandos immer automatisch auf `A:` bzw. auf das mit **mcd** eingestellte Working-Directory zu.
2. In Pfadangaben kann zur Abtrennung von Directories sowohl `/` als auch `\` verwendet werden.
3. Das Metazeichen `*` funktioniert wie unter Unix und Linux üblich. Zum Zugriff auf alle Dateien ist deshalb nur `*` und nicht wie unter DOS `*.*` anzugeben.
4. Dateinamen sind auf die DOS-Konventionen (8+3 Zeichen) limitiert.

Zur Konfiguration von **mtools** steht Datei `/etc/mtools` bzw. `/etc/mtools.conf` zur Verfügung. Normalerweise gibt es keine Probleme, so dass keine Veränderungen an der Konfigurationsdatei notwendig sind.
Hier nun ein kurzer Überblick zu den wichtigsten Programmen des **mtools**-Programmpakets:

- **mattrib** [+|-**ahrs**] *datei(en)* (*DOS-Kommando attrib*): liest bzw. verändert die Attribute von DOS-Dateien.
- **mbadblocks**: markiert unbrauchbare Blöcke in FAT
- **mcd** *directory* (*DOS-Kommando cd*)
  wechselt in das angegebene Directory auf einer DOS-Diskette oder -Festplatte. Dabei kann auch ein Laufwerksbuchstabe angegeben werden, wie z.B. **mcd c:\programme**
- **mcopy** [*optionen*] *quelldatei zieldatei*
  mcopy [*optionen*] *datei(en) zieldirectory* (*DOS-Kommando copy*)
  kopiert Dateien von oder auf DOS-Disketten bzw. -Festplatten. Die Optionen unterscheiden sich etwas von den Optionen des DOS-**copy**. Die beiden wichtigsten Optionen sind:

-n	keine Ausgabe einer Warnung vor dem Überschreiben einer Datei
-t	transformiert die unter DOS übliche Zeichenkombination *Carriage-Return*, *Line-Feed* für Zeilenende in ein einfaches *Line-Feed*-Zeichen und auch umgekehrt. Diese Option sollte nur für Textdateien verwendet werden.

- **mdel** *datei(en)* (*DOS-Kommando del*)
  löscht die angegebenen *datei(en)* auf der entsprechenden DOS-Partition.
- **mdeltree** *directory(s)* (*DOS-Kommando deltree*)
  löscht die angegebenen *directory(s)* auf der entsprechenden DOS-Partition.
- **mdir** [*optionen*] [*datei(en)*|*directory(s)*] (*DOS-Kommando dir*)
  zeigt den Inhalt der angegebenen *directory(s)* oder listet die angegebenen *datei(en)* der entsprechenden DOS-Partition auf. Mit der Option **-w** werden die Dateinamen in mehreren Spalten nebeneinander aufgelistet.
- **mformat** *laufwerk* (*DOS-Kommando format*)
  richtet ein MS-DOS-Dateisystem auf einer mit **fdformat** bereits vorformatierten Diskette ein.
- **mlabel** *laufwerk* (*DOS-Kommando label*)
  zeigt den Namen (*volume label*) des entsprechenden MS-DOS-Dateisystems an. Anschließend kann man einen neuen Namen festlegen.
- **mmd** *directory* (*DOS-Kommando md*)
  legt ein neues *directory* auf dem entsprechenden MS-DOS-Dateisystem an.
- **mrd** *directory* (*DOS-Kommando rd*)
  löscht das angegebene *directory* auf dem entsprechenden MS-DOS-Dateisystem.
- **mread** *dos_datei linux_datei*
  kopiert die angegebene *dos_datei* in das Linux-Dateisystem. **mread** ist eine eingeschränkte Version des **mcopy**-Kommandos, das Kopieren in beide Richtungen erlaubt.
- **mren** *alt_name neu_name* (*DOS-Kommando ren*)
  benennt die Datei *alt_name* im entsprechenden MS-DOS-Dateisystem in *neu_name* um.
- **mtype** *datei* (*DOS-Kommando type*)
  gibt den Inhalt der *datei* des entsprechenden MS-DOS-Dateisystems am Bildschirm aus. **mtype** entspricht in etwa dem Unix-Kommando **cat**.
- **mwrite** *linux_datei dos_datei*
  kopiert die angegebene *linux_datei* aus dem Linux-Dateisystem in ein DOS-Dateisystem. **mwrite** ist wie **mread** eine eingeschränkte Version des **mcopy**-Kommandos, das Kopieren in beide Richtungen erlaubt.

15 Editoren

emacs/xemacs

emacs [*optionen*] [*datei(en)*] bzw. **xemacs** [*optionen*] [*datei(en)*]

Allgemeines

Anders als der **vi** befindet sich der Emacs standardgemäß im Eingabemodus und Kommandos müssen bei gedrückter (Strg)-, (Alt)- und/oder (Shift)-Taste eingegeben werden: `C-` = gedrückte (Strg)-Taste, `A-` = gedrückte (Alt)-Taste und `S-` = gedrückte (Shift)-Taste. Grundsätzlich können Emacs-Kommandos auf zwei verschiedene Arten eingegeben werden:

1. Verwendung von Tastenkürzeln, wie z.B. `C-D` ((Strg)-D) zum Löschen des Zeichens an Cursorposition.
2. Eingabe des gesamten Kommandonamens, eingeleitet mit `A-X` ((Alt)-X), wie z.B. `A-X` **delete-char** (Enter) zum Löschen des Zeichens an Cursorposition. Während der Eingabe des Kommandonamens kann (Tab)-Taste gedrückt werden, um diesen, soweit er eindeutig ist, automatisch zu vervollständigen. Bestehen mehrere Möglichkeiten zum Vervollständigen, werden diese bei erneutem Drücken der (Tab)-Taste alle angezeigt.

Bei manchen Emacs-kompatiblen Editoren wird die (Alt)-Taste nicht unterstützt. Dort muss z.B. für `A-X` die Tastenkombination ESC,X eingegeben werden. Beim Start vom Emacs wird die Konfigurationsdatei `~/.emacs` gelesen, in der sich persönliche Einstellungen für den Emacs befinden können.

Laden und Speichern

`C-X,C-S`	(*save*) Datei speichern
`C-X,C-W` *name* (Enter)	(*write*) Datei unter *name* speichern
`C-X,S`	alle offenen Dateien (mit Rückfrage) speichern
`C-X,S,!`	alle offenen Dateien (ohne Rückfrage) speichern
`C-X,C-F` *dateiname* (Enter)	(*find*) Datei laden
`C-X,I`	(*insert*) Datei in vorhandenen Text einfügen
`C-X,C-C`	Emacs verlassen
`C-Z`	Emacs kurzzeitig verlassen; Rückkehr mit **%emacs**(Enter)

Emacs erstellt eine Sicherheitskopie *dateiname*~. Zudem wird in regelmäßigen Zeitintervallen in der Datei *#dateiname#* gesichert, auf die wie folgt zugegriffen werden kann: `A-X` **recover-file** (Enter).

Markieren, Löschen, Einfügen und Undo

`C-`(Leertaste)	an Cursorposition Markierung setzen
(Backspace)	Zeichen vor Cursor löschen
(Entf) oder `C-D`	Zeichen an Cursorposition löschen
`A-D`	nächstes bzw. bis Ende des aktuellen Wort löschen
`A-`(Backspace)	Vorheriges Wort bzw. bis Anfang des aktuellen Worts löschen
`C-K`	Von Cursor bis Zeilenende löschen
`A-0,C-K`	Von Cursor bis Zeilenanfang löschen
`A-Z,`*zeichen*	Text von Cursorposition bis zum nächsten Vorkommen von *zeichen* löschen
`A-M`	Nächsten Absatz löschen
`C-Y`	Zuletzt gelöschten Text an Cursorposition einfügen
`C-X,U`	(*undo*) mehrstufiges Undo
`C-W`	Text zwischen Cursor (inklusive) und Markierungspunkt löschen
`C-X,C-X`	Cursorposition und Markierungspunkt vertauschen

Cursor bewegen und Blättern

(Pfeil re) oder C-F	(*forward*) ein Zeichen nach rechts
(Pfeil li) oder C-B	(*backward*) ein Zeichen nach links
(Pfeil o) oder C-P	(*previous*) ein Zeichen nach oben
(Pfeil u) oder C-N	(*next*) ein Zeichen nach unten
(Bild oben) oder C-V	Eine Seite nach oben blättern
(Bild unten) oder A-V	Eine Seite nach unten blättern
A-F bzw. A-B	ein Wort nach rechts (*forward*) bzw. nach links (*backward*)
C-A bzw. C-E	an den Anfang bzw. an das Ende der Zeile
A-A bzw. A-E	an den Anfang bzw. an das Ende des Absatzes
A-< bzw. A-S->	an den Anfang bzw. an das Ende der Datei
C-L	so blättern, dass Cursor in der Bildmitte
A-X **goto-line**(Enter) *n* (Enter)	in die Zeile *n* springen
A-X **what-line** (Enter)	aktuelle Zeilennummer anzeigen, in der sich Cursor befindet
A-X **line-number-mode**(Enter)	Zeilennumerierung ein/ausschalten; aktuelle Zeilennummer wird in der Mitte der vorletzten Zeile angezeigt.

Kommandos können im Emacs auch durch A-*n* (*n* ist beliebige Zahl) mehrfach durch einen Aufruf wiederholt werden. Die Ziffern für die Zahl *n* müssen bei gedrückter (Alt)-Taste vom alphanumerischen Tastaturteil (nicht vom Zehnerblock im rechten Teil der Tastatur) eingegeben werden, wie z.B. A-10,A-F (Cursor 10 Wörter vorwärts) oder A-12,(Pfeil o) (Cursor 12 Zeilen nach oben).

Cursorposition speichern

C-X,R,(Leertaste),*x* (Enter)	speichert aktuelle Cursorposition in benannten Puffer *x* (a, b, ..., z, 0, .., 9).
C-X,R,J,*x* (Enter)	springt an Cursorposition, die im benannten Puffer *x* (a, b, ..., z, 0,..., 9) gespeichert ist.

Benannte Puffer speichern und einfügen

C-X,R,S *x* (Enter)	(*register save*) Text zwischen Cursorposition und Markierungspunkt in benannten Puffer *x* (a, b, c, ..., x, y, z, 0, 1, ..., 8, 9) speichern.
C-X,R,I *x* (Enter)	(*register insert*) Inh. des benannten Puffer *x* (a, b, c, ..., x, y, z, 0, 1, ..., 8, 9) an Cursorpos. einfügen.

Einfaches Suchen (ohne reguläre Ausdrücke)

C-S *suchtext* bzw. C-R *suchtext*	Inkrementelles Suchen vorwärts bzw. rückwärts
C-S,A-N	(*next*) Vorwärtsblättern in allen bisher verwendeten Suchstrings
C-S,A-P	(*previous*) Rückwärtsblättern in allen bisher verwendeten Suchstrings
C-G	Abbrechen der Suche
C-S,C-S	Letzte Suche (vorwärts) wiederholen
C-R,C-R	Letzte Suche (rückwärts) wiederholen

Suchen (mit regulären Ausdrücke)

C-A-S *suchtext*	Inkrementelles Suchen nach Mustern vorwärts
C-A-R *suchtext*	Inkrementelles Suchen nach Mustern rückwärts

Reguläre Ausdrücke

^ bzw. $	Anfang bzw. Ende einer Zeile
\< bzw. \>	Anfang bzw. Ende eines Wortes
.	Ein beliebiges Zeichen (außer Zeilenende-Zeichen)
.*	Kein, ein oder mehrere beliebige Zeichen
.+	Ein oder mehrere beliebige Zeichen
.?	Kein oder ein beliebiges Zeichen
[...]	Eines der in [...] angegebenen Zeichen
[^...]	Ein Zeichen, das nicht in [^...] angegeben ist
\(bzw. \)	Beginn bzw. Ende einer Gruppe
\n	An dieser Stelle den Text, der für die *n*. te Gruppe gefunden wurde, einsetzen
\&	Hier gesamten gefundenen Text einsetzen
\x	Sonderzeichen *x* ausschalten

Beim Suchen nach Mustern wird zwischen Groß- und Kleinschreibung unterschieden.

Ersetzen (mit und ohne reguläre Ausdrücke)

ESC,%	Suchen und Ersetzen ohne Muster
A-X **query-replace-r** (Enter)	Suchen und Ersetzen mit Muster
C-S--	(Minuszeichen) fehlerhafte Ersetzungen schrittweise wieder rückgängig machen

Nachdem ein Suchtext gefunden wurde, kann man mit folgenden Tastenkombinationen steuern, was mit diesem Text zu tun ist und wie das weitere Suchen und Ersetzen fortgesetzt werden soll:

Interaktive Eingaben zum Ersetzen

(Leertaste) oder Y	Ersetzen und Suche fortsetzen
(Backspace) oder N	Nicht ersetzen und Suche fortsetzen
,	Ersetzen und Suche erst einmal anhalten, um Ersetzung zu kontrollieren. Ist die Ersetzung richtig, dann kann die Suche mit (Leertaste) fortgesetzt werden
ESC	Nicht Ersetzen und Suche abbrechen
!	Alle weiteren Ersetzungen ohne Rückfrage durchführen
C-R	Suche kurzzeitig unterbrechen, um an aktueller Cursorposition eine manuelle Korrektur vorzunehmen
C-A-R	Eine mit dem vorherigem Kommando unterbrochene Suche wieder aufnehmen

Umschalten Einfüge-/Überschreibmodus

A-X **overwrite-mode** (Enter)	(auch mit Taste (Einfg) möglich)

Vertauschen von Zeichen, Wörter und Zeilen

C-T	Vorheriges Zeichen mit Zeichen an Cursorposition vertauschen
A-T	Zwei Wörter vertauschen: Steht der Cursor am Wortanfang, wird dieses Wort mit vorherigem Wort vertauscht. Steht Cursor nicht am Wortanfang, wird dieses Wort mit nachfolgenden Wort vertauscht.
C-X,C-T	Aktuelle Zeile mit vorheriger Zeile vertauschen. Wiederholte Anwendung dieses Kommandos läßt Zeile oberhalb vom Cursor immer weiter nach unten rutschen.

Groß- in Kleinschreibung

A-C	(*capitalize*) Buchstabe an Cursorposition in Großbuchstaben und restliche Buchstaben des Wortes in Kleinbuchstaben umwandeln
A-L	(*lower*) alle Buchstaben des aktuellen Worts ab Cursor in Kleinbuchstaben umwandeln.
A-U	(*upper*) alle Buchstaben des aktuellen Worts ab Cursor in Großbuchstaben umwandeln.
ESC,-,A-C	(*capitalize*) ersten Buchstaben groß, Rest klein. Steht der Cursor am Anfang eines Worts, wird das vorherige Wort entsprechend umgeformt.
ESC,-,A-L	(*lower*) alle Buchstaben des aktuellen Worts klein. Steht der Cursor am Anfang eines Worts, wird das vorherige Wort in Kleinbuchstaben umgeformt.
ESC,-,A-U	(*upper*) alle Buchstaben des aktuellen Worts groß. Steht der Cursor am Anfang eines Worts, wird das vorherige Wort in Großbuchstaben umgeformt.
C-X,C-L	Bereich zwischen Cursor und Markierungspunkt in Kleinbuchstaben umwandeln.
C-X,C-U	Bereich zwischen Cursor und Markierungspunkt in Großbuchstaben umwandeln.
ESC,-	kann vielen Emacs-Kommandos vorangestellt werden und verändert meist Richtung (rückwärts statt vorwärts).

Puffer-Kommandos

C-X,B (Enter)	zurück zum zuvor verwendeten Puffer
C-X,B *name* (Enter)	Puffer *name* aktivieren und im aktuellen Fenster darstellen
C-X,C-B	im aktuellen Fenster Liste aller Puffer anzeigen; mit C-X,1 kann Anzeige wieder gelöscht werden
C-X,K *name* (Enter)	Puffer *name* löschen; enthält Puffer nicht gesicherten Text, wird nachgefragt, ob er zu löschen ist.

Fenster-Kommandos

C-X,O	(Oh) zum nächsten Fenster springen
C-X,0	(Null) aktuelles Fenster löschen
C-X,1	außer aktuelles Fenster alle Fenster löschen
C-X,2	aktuelles Fenster in zwei horizontale Fenster teilen
C-X,3	aktuelles Fenster in zwei vertikale Fenster teilen
C-X,^	aktuelles Fenster vertikal vergrößern
ESC,-,C-X,^	akuelles Fenster vertikal verkleinern
C-X,} bzw. C-X,{	aktuelles Fenster horizontal vergrößern bzw. verkleinern
C-X,< bzw. C-X,>	Inhalt des aktuellen Fensters nach links bzw. nach rechts schieben

Das Löschen eines Fenstern bewirkt nicht das Löschen des zugehörigen Puffers. Dieser wird unsichtbar, bleibt aber im Speicher und kann später wieder angezeigt werden

Aufzeichnen und Ausführen von Makros

C-X,(bzw. C-X,)	Starten bzw. Beenden einer Makro-Aufzeichnung
C-X,E	Ausführen des zuletzt aufgezeichneten Makros
A-X **name-last-kbd-macro** (Enter) *name* (Enter)	An ein aufgezeichnetes Makro einen Namen vergeben
A-X **insert-kbd-macro** (Enter) *name* (Enter)	Makro *name* in aktueller Datei (z.B. .emacs) einfügen
A-X name (Enter)	Ausführen des Makros *name*

Abkürzungs-Kommandos

ESC,/	Vervollständigen des bisher eingegebenen Wortanfangs durch Wort aus vorherigem oder folgendem Text, oder aus einer anderen offenen Datei
C-X,A,I,G *text* (Enter)	definiert zum zuvor eingegebenen Kürzel einen vollständigen *text*
C-X,A,E	ersetzt gerade eingegebene Abk. durch zugehörigen vollständigen Text
A-X **abbrev-mode** (Enter)	aktiviert Abkürzungsmodus
A-X **edit-abbrevs** (Enter)	Anzeigen und Editieren der aktuellen Abkürzungstabelle
C-X,C-S	Abkürzungstabelle speichern
C-X,B	Von Abkürzungstabelle zurück zum Text
A-X **write-abbrev-file** (Enter) *name* (Enter)	Abkürzungstabelle in Datei *name* speichern
A-X **read-abbrev-file** (Enter) *name* (Enter)	Abkürzungstabelle aus Datei *name* lesen

Online-Hilfe

Allgemeine Online-Hilfe

F1,F1	Übersicht zu den Hilfe-Kommandos
F1,A *text* (Enter)	(*apropos*) Übersicht über alle Emacs-Kommandos anzeigen, die *text* enthalten
F1,B	(*bindings*) Übersicht über alle Tastenkürzel
F1,C *tastenkürzel*	(*command*) Kurzbeschreibung zum Emacs-Kommando anzeigen, dem *tastenkürzel* zugeordnet ist
F1,F *kdo* (Enter)	(*function*) Kurzbeschreibung zum Kommando *kdo* anzeigen
F1,S-F	(*frequently asked questions*) Häufige Fragen mit entsprechenden Anwtorten anzeigen
F1,I	(*info*) startet **info**-Programm, um Texte mit Querverweisen anzuzeigen
F1,N	(*new*) Neuheiten in der aktuellen Version
F1,T	(*tutorial*) Einführung in die Emacs-Bedienung
F1,C-C	(*copyright*) Copyright-Informationen anzeigen
F1,C-F *kdo* (Enter)	**info**-Programm starten und Informationen zu Kommando *kdo* anzeigen

Online-Hilfe zu den Modi

F1,A *mode* (Enter)	Übersicht zu allen verfügbaren Modi
F1,M	Informationen zum gerade aktiven Modus

Online-Hilfe zu Emacs-Lisp-Programmierung

F1,F *funktion* (Enter)	Kurzbeschreibung zur Lisp-Funktion *funktion* anzeigen
F1,F (Tab)	Liste aller Lisp-Funktionen zeigen. Um Liste zu betrachten, muss C-G eingegeben werden. Zum Blättern kann mit C-X,B ***Com** (Tab) (Enter) in Funktionsliste gewechselt werden.
F1,V *varname* (Enter)	Kurzbeschreibung zur Lisp-Variable *varname* anzeigen
F1,V (Tab)	Liste aller vorhandenen Lisp-Variablen zeigen. Um die Liste zu betrachten, muss dieses Kommando mit C-G abgebrochen werden. Dann kann zum Blättern in der Variablenliste mit C-X,B ***Com** (Tab) (Enter) in den Variablenliste-Puffer gewechselt werden.
F1,C *tastenkombi* (Enter)	Kurzbeschreibung des Kommandos, das der Tastenkombination *tastenkombi* zugeordnet ist

Auf älteren Emacs-Versionen muss statt F1 die Tastenkombination C-H gedrückt werden. Die entsprechende Hilfs-Information wird in einem eigenen Fenster angezeigt, von dem mit C-X,B (Enter) oder C-X,0 (Null) zurück in das Textfenster gewechselt werden kann.

Ein-/Ausrücken von Text am Zeilenanfang

C-X,(Tab)	Text zwischen Markierungspunkt und Cursorposition um ein Zeichen einrücken
ESC,-,C-X,(Tab)	Text zwischen Markierungspunkt und Cursorposition um ein Zeichen ausrücken
A-*n*,C-X,Tab	Text zwischen Markierungspunkt und Cursorposition um *n* Zeichen einrücken
ESC,-,A-*n*,C-X,(Tab)	Text zwischen Markierungspunkt und Cursorposition um *n* Zeichen ausrücken

Rechteck-Kommandos

C-X,R,O	(*rectangle open*) im rechteckigen Bereich Leer- bzw. Tabzeichen einfügen
C-X,R,K	(*rectangle kill*) rechteckigen Bereich löschen
C-X,R,R	(*rectangle register*) rechteckigen Bereich in internen Puffer speichern
C-X,R,Y	(*rectangle yank*) zuvor gelöschten bzw. im internen Puffer gespeicherten rechteckigen Bereich an Cursorposition einfügen
A-X **clear-rectangle** (Enter)	rechteckigen Bereich mit Leerzeichen überschreiben
A-X **string-rectangle** (Enter) *text* (Enter)	*text* vor 1. Zeichen jeder Zeile des markierten Rechteck-Bereichs einfügen

Tabulatoren in Leerzeichen und umgekehrt

A-X **untabify** (Enter)	Im markierten Bereich Tabulatoren in Leerzeichenfolgen umwandeln
A-X **tabify** (Enter)	Im markierten Bereich Leerzeichenfolgen in Tabulatoren umwandeln
A-X **set-variable** (Enter) **tab-width** (Enter) *n* (Enter)	Einstellen der Tabulatorweite auf *n* Zeichen; Voreinstellung: *n*=8.

Aufrufen von Unix-Kommandos im Emacs

ESC,! *kommando* (Enter)

Aktivieren wichtiger Hauptmodi

A-X **fundamental-mode** (Enter)	Standard-Modus
A-X **c-mode** (Enter) bzw. A-X **c++-mode** (Enter)	C-Modus bzw. C++-Modus
A-X **html-mode** (Enter)	HTML-Modus
A-X **shell** (Enter)	Shell-Modus (für Eingabe von Unix-Kommandos)
A-X **text-mode** (Enter)	Einrück-Modus
A-X **sh-mode** (Enter)	Modus für das Editieren von Shell-Skripts
A-X **latex-mode** (Enter)	LaTeX-Modus

Aktivieren der wichtigsten Nebenmodi

A-X **auto-fill-mode** (Enter)	Fließtext-Modus (Automatischer Zeilenumbruch)
A-X **iso-accents-mode** (Enter)	Modus für die Eingabe fremdsprachiger Sonderzeichen
A-X **font-lock-mode** (Enter)	Modus, in dem Syntaxkonstrukte farbig hervorgehoben werden
A-X **abbrev-mode** (Enter)	Abkürzungs-Modus (Abkürzungen werden automatisch ersetzt)

Emacs versucht aus Dateinamen-Endung (`.c`, `.tex` usw.) und Inhalt der ersten Zeilen zu erkennen, um welchen Texttyp es sich handelt, um dann den entspr. Modus einzuschalten. Um Emacs zu helfen, kann man in der ersten Zeile der entsprechenden Datei einen Kommentar angeben, der folgendes enthält (*name*=gewünschte Modus): `*-*` *name* `*-*`

Wichtige Tasten im C-Modus

A-A bzw. A-E	An den Anfang bzw. an das Ende der aktuellen Klammerebene springen

Kompilierung von C-Programmen

A-X **compile** (Enter)	Kompilierung starten
C-X,`	zur nächsten Fehlermeldung
ESC,-,C-X,`	zur vorherigen Fehlermeldung
C-U,C-X,`	zur ersten Fehlermeldung

Automatischer Zeilenumbruch (Fließtext-Modus)

A-Q	Zeilenumbrüche für aktuellen Absatz durchführen
C-X,F	maximale Zeilenlänge (für automatischen Zeilenbruch) an aktueller Cursorposition festlegen

Einrücken im Fließtext-Modus

C-X,.	Einrückspalte an aktueller Cursorposition festlegen; Zeichen vor Cursor legen einzufügenden String fest und Zeilenrest sollte leer sein.
A-M	an Beginn einer eingerückten Zeile springen (ähnlich zu C-A)
A-X **fill-individual-paragraphs** (Enter)	Bereich zwischen Markierungspunkt und Cursor formatieren, wobei Einrückungen erhalten bleiben

Wichtige Tasten zum Text-Modus

A-Q	manuellen Umbruch durchführen
A-S	aktuelle Zeile zentrieren
A-S-S	aktuellen Absatz zentrieren
A-X **center-line** (Enter)	Zeilen zentrieren
A-X **center-paragraph** (Enter)	Absätze zentrieren

Wichtige Tasten im LaTeX-Modus

C-C,C-E	fügt \end{name} zum letzten offenen \begin{name} ein
C-C,C-O *nam1*(Enter) *nam2*(Enter)	fügt folgendes ein: \begin[*nam1*]{*nam2*} \end{*nam2*}
C-C,}	zu abschließender } der aktuellen Klammerebene springen
C-J	beendet Absatz, fügt Leerzeile ein und führt Syntaxprüfung für Absatz durch
A-X **validate-text-buffer**(Enter)	aktuellen Text auf LaTeX-Syntaxfehler prüfen

Maustasten beim Emacs unter X Window

Linke Maustaste	Cursorpositionierung
Bewegen Maus bei gedrückter linker Maustaste	Markierung eines Bereichs. Entsprechender Bereich wird in internen Puffer kopiert und kann an anderer Stelle mit `C-Y` oder dem Drücken der mittleren Maustaste im Text eingefügt werden.
mittlere Maustaste (im Text)	zuvor mit gedrückter linker Maustaste markierten Bereich beim Mauscursor einfügen.
mittlere Maustaste (in inverser Infozeile)	vergrößert das so ausgewählte Textfenster auf seine maximale Größe
rechte Maustaste	Markierungsende des zuvor markierten Bereichs versetzen
Doppelklick auf rechte Maustaste	Markierten Textbereich löschen; mit mittlerer Maustaste wieder einfügbar
Linke Maustaste bei gedrückter (Shift)-Taste	Popup-Menü zur Auswahl des Zeichensatzes
Linke Maustaste bei gedrückter (Strg)-Taste	Popup-Menü zur Auswahl eines Puffers
Mittlere Maustaste bei gedrückter (Strg)-Taste	Popup-Menü zum Ändern von Schriftart bzw. Farbe
Rechte Maustaste bei gedrückter (Strg)-Taste	Popup-Menü mit Kommandos zum aktuellen Modus

Wichtige Optionen für emacs unter X

-fg *farbe*	Vordergrundfarbe (Textfarbe; Voreinstellung ist `black`)
-bg *farbe*	Hintergrundfarbe (Textfarbe; Voreinstellung ist `white`)
-cr *farbe*	Farbe des Textcursors (Textfarbe; Voreinstellung ist `black`)
-nw	Emacs im Shell-Fenster starten. Üblicherweise wird X-Variante von Emacs in eigenem Textfenster gestartet
-fn *zeichensatz*	angegebenen *zeichensatz* verwenden

Alle obigen Optionen können auch in der Konfigurationsdatei `~/.Xdefaults` eingestellt werden, wie z.B.

```
emacs.cursorColor: white
emacs.pointerColor: red
emacs.shell*background: blue
emacs*menubar.background: yellow
emacs*menubar.foreground: red
emacs*background: blue
emacs*buttonForeground: white
emacs*attributeBackground: white
emacs*attributeForeground: black
emacs*modeline.attributeBackground: green
emacs*modeline.attributeForeground: black
emacs*region.attributeBackground: yellow
```

Sind in `~/.Xdefaults` nicht existierende Farben oder Zeichensätze angegeben, kann Emacs nicht gestartet werden.

Interaktive Emacs-Konfiguration

Viele Konfigurationsmöglichkeiten ergeben sich mit der Eingabe des folgenden Kommandos:

^X **customize-browse** (Enter)

Dieses Kommando bewirkt, dass ein Menü eingeblendet wird, das mit der mittleren Maustaste bedient werden kann. Klickt man eines der Icons mit der mittleren Maustaste an, wird im unteren Fenster die entsprechende Option mit einer kurzen Beschreibung angezeigt, wobei man diese Option nun dort auch ändern kann. Änderungen werden dabei erst übernommen, wenn man den `Set`-Button anklickt. Um Änderungen für spätere **emacs**-Sitzungen zu speichern, muss der `Save`-Button (für jede einzelne Änderung) angeklickt werden. Die Einstellungen werden in der Konfigurationsdatei `~/.emacs` gespeichert.

Emacs-ähnliche Editoren

Unter Linux existiert eine Vielzahl von Editoren, wie z.B. die beiden Editoren **jove** und **jed**, die abgemagerte Emacs-Versionen sind und deshalb gerade für den Anfänger leichter zu bedienen sind. Auf eine kleine Besonderheit des **jed** sei hier hingewiesen: Die (Entf)-Taste kann im **jed** erst verwendet werden, wenn zuvor ein Bereich markiert wurde. Das Setzen eines Markierungspunktes erfolgt dabei wie im Emacs mit (Strg)-(Leertaste). Ein Bereich erstreckt sich dann vom Markierungspunkt bis zur aktuellen Cursorposition. Mit der Taste (Entf) kann dann ein solcher Bereich gelöscht und mit der Taste (Einfg) an einer anderen Stelle wieder eingefügt werden.

vi

vi [*optionen*] [*datei(en)*]

-t *marke*	(*tag*) Editieren der in `tags` mit *marke* festgelegten Datei
-r*datei*	(*recover*) Editieren der angegebenen *datei* nach Editor- bzw. Systemzusammenbruch
-L	alle geretteten Dateien nach Editor- bzw. Systemzusammenbruch melden
-w*n*	(*window*) vi-Fenster soll *n* Zeilen hoch sein
-R	(*Readonly*) *datei(en)* nur zum Lesen öffnen
-c *kdo*	ex-*kdo* vor Editieren ausführen

Drei Arbeitszustände des vi

- **direkter Kommandomodus**
  Nach dem **vi**-Aufruf befindet sich **vi** immer in diesem Modus. Umschalten in den Eingabemodus ist möglich mit:

i, I, a, A	insert bzw. append
c, C, o, O	change bzw. open
s, S	substitute
R	Replace

- **Eingabemodus**: Im Eingabemodus ist Texteingabe möglich. Mit ESC kann die Texteingabe beendet und zurück in den direkten Kommandomodus gewechselt werden.
- **Zeilen-Kommandomodus**: Umschalten vom direkten Kommandomodus in diesen Modus ist möglich mit:

:*exkdo* (Enter)	*exkdo* (**ed**-ähnlich) ausführen
/*RA* (Enter)	Vorwärtssuche nach *RA*
?*RA* (Enter)	Rückwärtssuche nach *RA*
:!*kdo* (Enter)	Unix-Kommando *kdo* ohne Verlassen des **vi** ausführen

Wichtige vi-Tasten

ESC	- Eingabemodus beenden - im Zeilen-Modus: *exkdos* abbrechen
(Enter)	- im Zeilen-Modus: *exkdos* ausführen - im Eingabemodus: neue Zeile - im Kommandomodus: in nächste Zeile
(Strg)-C	Aktion abbrechen.

Sondertasten in Eingabemodus

(Strg)-I	Tabulatorzeichen einfügen
(Strg)-T	auf nächste Tabulator-Marke positionieren
(Strg)-V	Nächstes Zeichen nicht als Kommando interpretieren
(Strg)-W	Letztes Wort löschen
(Strg)-H, (Backspace)	Letztes Zeichen löschen

Interne vi-Puffer

Neben dem allgemeinen Arbeitspuffer gibt es noch 36 weitere Puffer:

- 26 benannte Puffer (`a, b, c, .. , z`); Großbuchstabe; bedeutet Anhängen am jeweiligen Puffer
- Neun Zifferpuffer (`1, 2, ..., 9`), in denen die neun zuletzt gelöschten Texte liegen (in 1 zuletzt gelöschter, in 2 der davor gelöschte Text usw.).

Diese Puffer können mit "*x* (*x* = Kleinbuchstabe bzw. Ziffer) angesprochen werden.

Sichern und Beenden

ZZ oder **:wq** (Enter)	Sichern und vi beenden
:q (Enter)	vi beenden, wenn Puffer nicht geändert
:q! (Enter)	vi ohne Puffer-Sicherung beenden
:w (Enter)	Puffer sichern (vi nicht verlassen)
:w *datei* (Enter)	Puffer in *datei* sichern (vi nicht verlassen)
:w! *datei* (Enter)	Puffer in (eventuell existierende) *datei* sichern (vi nicht verlassen)
:*n*,*m***w** *datei* (Enter)	Zeilen *n* bis *m* in *datei* sichern (vi nicht verlassen)

Cursor-Positionierungen

Zeichen-Positionierung

l, (Pfeil re), (Leertaste)	ein Zeichen nach rechts
h, (Pfeil li), (Strg)-H	ein Zeichen nach links
j, (Pfeil u), (Strg)-N	nach unten (selbe Spalte bzw. Zeilenende)
k, (Pfeil o), (Strg)-P	nach oben (selbe Spalte bzw. Zeilenende)
+, (Enter), (Strg)-M	nach unten (auf erstes sichtbare Zeichen)
-	nach oben (auf erstes sichtbare Zeichen)
^	Zeilenanfang (auf erstes sichtbare Zeichen)
0	Zeilenanfang (auf erstes Zeichen)
$	Zeilenende
n|	zur *n*. ten Spalte

Wort-Positionierung

w bzw. *n***w**	zum nächsten bzw. zum *n*. ten nächsten Wort
b bzw. **e**	Ein Wort zurück bzw. zum Wortende

Zeilen-Positionierung

H	(*Home*) auf erste Bildschirmzeile springen
M	(*Middle*) auf mittlere Bildschirmzeile springen
L	(*Last*) auf letzte Bildschirmzeile springen
G	letzte Zeile
*n***G**	zur *n*. ten Zeile springen (`1G` auf erste Zeile der Datei positionieren)

Weitere Positionierungsmöglichkeiten

(Bild unten), (Strg)-f	Eine Seite vorblättern
(Bild oben), (Strg)-b	Eine Seite zurückblättern
z (Enter)	aktuelle Zeile zur obersten Bildschirmzeile machen
z-	akuelle Zeile zur untersten Bildschirmzeile machen
z.	akuelle Zeile zur mittleren Bildschirmzeile machen
``	zur vorherigen Position zurück
%	auf korrespondierende Klammer

Eingeben/Ändern mit Umschalten in Eingabemodus

a bzw. **A**	(*append*) Rechts vom Cursor bzw. am Zeilenende einfügen
i bzw. **I**	(*insert*) Links vom Cursor bzw. am Zeilenanfang einfügen
o bzw. **O**	(*open*) In neuer Zeile danach bzw. davor einfügen
s bzw. **S**	(*substitute*) Cursorzeichen bzw. ganze Cursorzeile ersetzen
R	(*Replace*) Überschreiben einschalten
cw bzw. *n***cw**	(*change*) nächstes Wort bzw. nächsten *n* Worte ersetzen
cc bzw. **C**	(*change*) ganze Zeile bzw. Rest der Zeile ersetzen

Ändern ohne Umschalten in Eingabemodus

r*z* bzw. *n***r***z*	(*replace*) Zeichen an Cursorposition bzw. nächsten *n* Zeichen durch *z* ersetzen
~ bzw. *n*~	Cursorzeichen bzw. nächsten *n* Zeichen von Klein- in Großbuchstaben umwandeln bzw. umgekehrt
J bzw. *n***J**	(*Join*) Zwei bzw. *n* Zeilen zusammenfügen.
.	Letztes Änderungskommando wiederholen

Löschen, Kopieren und Verschieben

dw bzw. *n***dw**	(*delete*) nächstes Wort bzw. nächsten *n* Worte löschen
dd bzw. *n***dd**	aktuelle Zeile bzw. nächsten *n* Zeilen löschen
d/was bzw. **dG**	ab Cursor bis zum nächsten String "`was`" bzw. bis Dateiende löschen
D	Zeilenrest löschen (entspricht **d$**)
x bzw. *n***x**	Cursorzeichen bzw. nächsten *n* Zeichen löschen
X	Zeichen vor Cursor löschen
yy bzw. *n***yy**	aktuelle Zeile bzw. nächsten *n* Zeilen in den allgemeinen Puffer kopieren
y$	ab Cursor Zeilenrest in den allgemeinen Puffe kopieren
>%	Text bis zur korrespondierenden Klammer einrücken
"*x***yw**	kopiert nächstes Wort in den Puffer *x*
"ayy	kopiert aktuelle Zeile in den Puffer `a`
"*xn***yy**	kopiert die nächsten *n* Zeilen in den Puffer *x*
"add	löscht aktuelle Zeile und kopiert sie in den Puffer `a`
"*xn***dw**	*n* Worte löschen und in Puffer *x* kopieren
p bzw. **P**	(*put*) allgemeinen Puffer hinter bzw. vor Cursor kopieren
xp	zwei Zeichen vertauschen
"*x***p** bzw. **"***x***P**	Puffer *x* hinter bzw. vor Cursor kopieren
"*n***p**	*n*. te letzte Löschung hinter Cursor kopieren (für *n* ist Ziffer 1 bis 9 anzugeben)
:r *datei*	*datei* hinter aktuelle Zeile kopieren

Änderungen rückgängig

u bzw. **U**	(*undo*) macht die letzte Änderung bzw. die Änderung in aktueller Zeile rückgängig
:e!	alle Änderungen seit letztem Sichern wegwerfen
:q!	vi ohne Sichern verlassen

Markieren und Positionieren

m*x*	Cursorposition mit *x* (Kleinbuchstabe) markieren.
`*x*	Cursor auf die mit *x* markierte Stelle positionieren.
'*x*	Cursor auf erstes Zeichen der Zeile mit Marke *x* positionieren.

Editieren mehrerer Dateien

:e *datei* (Enter)	(*edit*) *datei* in Arbeitspuffer kopieren; alten Puffer überschreiben, wenn bereits gesichert.
:e# (Enter)	zurück zur vorherigen Datei (wie (Strg)-^)
:n (Enter)	(*next*) Nächste Datei (aus vi-Aufruf) editieren
:n! (Enter)	wie :**n** (Enter), jedoch Puffer "blind" überschreiben
:f *datei* (Enter)	Dateinamen in *datei* ändern. Ist *datei* nicht angegeben, wird gerade gemerkter Dateiname und Nummer der aktuellen Zeile angezeigt (wie (Strg)-G).

Suchen

Suchen in aktueller Zeile

f*x*	Auf Zeichen *x* in aktueller Zeile vorwärts positionieren
F*x*	zurück auf Zeichen *x* in aktueller Zeile positionieren
t*x*	vor Zeichen *x* in aktueller Zeile positionieren
T*x*	zurück hinter Zeichen *x* in aktueller Zeile
;	letztes **f**-,**F**-, **t**- oder **T**-Kommando wiederholen
,	letztes **f**-,**F**-, **t**- oder **T**-Kommando wiederholen (mit umgekehrter Suchrichtung)

Suchen im Arbeitspuffer

/*RA* (Enter)	Vorwärtssuche nach regulären Ausdruck *RA*
/*RA*/*n* (Enter)	sucht vorwärts *n*. ten regulären Ausdruck *RA*
?*RA* (Enter)	Rückwärtssuche nach regulären Ausdruck *RA*
?*RA***?***n* (Enter)	sucht rückwärts *n*. ten regulären Ausdruck *RA*
/*RA*/+*n* (Enter)	Cursor auf *n*. te Zeile nach gefundenen regulären Ausdruck *RA*
/*RA*/-*n* (Enter)	Cursor auf *n*. te Zeile vor gefundenen regulären Ausdruck *RA*
n	letzten Suchvorgang (/*RA* oder **?***RA*) wiederholen
N	letzten Suchvorgang umgekehrt wiederholen

RA (regulärer Ausdruck): siehe Ende dieses Buches.

Suchen und gleichzeitiges Ersetzen

:s/*alt*/*neu*/ (Enter)	in aktueller Zeile erstes *alt* durch *neu* ersetzen
:s/*alt*/*neu*/**g** (Enter)	in aktueller Zeile alle *alt* durch *neu* ersetzen
:*n*,*m***s**/*alt*/*neu*/**g** (Enter)	in der *n*. ten bis zur *m*. ten Zeile alle *alt* durch *neu* ersetzen.
:1,$s/*alt*/*neu*/**g** (Enter)	im gesamten Arbeitspuffer alle *alt* durch *neu* ersetzen.
:1,$s/*alt*/*neu*/**gc** (Enter)	im gesamten Arbeitspuffer alle *alt* durch *neu* ersetzen, jedoch mit Rückfrage: Soll ersetzt werden, muss **y**(Enter) eingegeben werden, sonst findet keine Ersetzung statt.

Neuaufbauen einer Bildschirmseite

(Strg)-**L**, (Strg)-**R**	Bildschirmseite neu aufbauen
z*n***.**	Bildschirmgröße auf *n* Zeilen festlegen

Makros

Bei Definition von Makros sind Steuerzeichen durch vorangestelltes (Strg)-V auszuschalten.

@x	Makro aus Puffer *x* aufrufen
:ab *abk text* (Enter)	Textmakro *abk* für *text* definieren
:ab (Enter)	alle definierten Textmakros anzeigen
:map *x kdos* (Enter)	Kommando-Makro *x* für *kdos* definieren; *x* muss ein Zeichen oder #*n* (*n*=0,..,9) sein; :**map #1 :set number** (Strg)-V (Enter) (Enter) legt z.B. fest, dass bei Drücken der F1-Taste die Zeilennumerierung einzuschalten ist
:map (Enter)	alle definierten Kommando-Makros anzeigen

vi konfigurieren

Konfigurierung mit **set** in `~/.exrc` oder mit **:set** während vi-Sitzung möglich:

:set *option* (Enter)	Einschalten von *option*
:set no*option* (Enter)	Ausschalten einer *option*
:set *option*=**wert** (Enter)	Zuweisen von *wert* an *option*
:set (Enter)	Alle geänderten Optionen anzeigen
:set all (Enter)	Belegung aller Optionen anzeigen
:set *option***?** (Enter)	Belegung von *option* anzeigen

:set showmatch (Enter)	korrspondierende Klammer zeigen
:set showmode (Enter)	Modus immer anzeigen
:set number (Enter)	Zeilennumerierg einschalten

ed

ed [*optionen*] [*datei*]

-s	keine Meldung über gelesene bzw. geschriebene Zeichenzahl
-p*prompt*	*prompt* als Promptzeichen im Kommandomodus verwenden (Voreinstellung: kein Prompt)

Zwei Arbeitszustände des ed

- **Kommandomodus:** Verlassen mit **a**, **i**, **c**
- **Eingabemodus**: Verlassen mit **.** (Enter) (*Punkt, Return*)

Aufbau eines ed-Kommandos

[*adresse1*[,*adresse2*]] [*editier-kommando* [*parameter*]]

keine Adresse angegeben: adressiert aktuelle Zeile

eine Adresse angegeben: adressiert die Zeile, die Adresse besitzt

beide Adressen angegeben: adressiert den Bereich (*von*,*bis*) von Zeilen

Mögliche Angaben für *adresse1* und *adresse2* sind:

.	adressiert die aktuelle Zeile
$	adressiert die letzte Zeile
n	adressiert die *n*. te Zeile
'*x*	adressiert die Zeile mit Marke *x* (*x* muss dabei ein Kleinbuchstabe sein)
/*RA*/	adressiert erste Zeile (von aktueller Zeile zum Dateiende hin), die einen String beinhaltet, der durch den regulären Ausdruck *RA* abgedeckt ist. Wird bis Dateiende keine solche Zeile gefunden, so wird vom Dateianfang bis einschließlich aktueller Zeile nach solcher Zeile gesucht.
?*RA***?**	wie /*RA*/, nur in umgekehrter Richtung

ed-Kommandos

In der folgenden Liste der **ed**-Kommandos werden die *default*-Adressen in Klammern davor angegeben. Daraus ist zugleich auch erkennbar, wie viele Adressen jeweils maximal erlaubt sind.
Verwendete Abkürzungen sind: *ra* (Regulärer Ausdruck), *kdos* (ed-Kommandos) und *ers* (Ersetzungstext)

(.)	**a**	(*append*) Text anfügen; bis zur Eingabe von **.** (Punkt)
(.,.)	**c**	(*change*) Zeilen durch neue Zeilen ersetzen; bis zur Eingabe von **.** (Punkt)
(.,.)	**d**	(*delete*) Zeilen löschen
	e *datei*	(*edit*) Puffer mit Inhalt von *datei* laden
	E *datei*	(*Edit*) wie **e** ohne Warnung über Änderungen
	f *dateiname*	(*file*) *dateiname* merken; fehlt *dateiname*, wird gerade gemerkter Dateinamen ausgegeben
(1,$)	**g**/*ra*/*kdos*	*kdos* für alle Zeilen mit *ra* ausführen (*global*); mehrere *kdos* mit \(Enter) trennen
(1,$)	**G**/*ra*/	(*Global*) interaktive Version zum **g**-Kommando
	h	(*help*) zur letzten ?-Warnung Erklärung ausgeben
	H	(*Help*) statt ? richtige Fehlermeldung ausgeben
(.)	**i**	(*insert*) Text vor Zeile einfügen; bis zur Eingabe von **.** (Punkt)
(.)	**k***x*	(*mark*) Zeile mit Kleinbuchstaben *x* markieren
(.,.+1)	**j**	(*join*) Zeilen aneinanderhängen
(.,.)	**l**	(*list*) Zeilen ausgeben; alle Zeichen sichtbar machen und überlange Zeilen zerteilen
(.,.)	**m***adr*	(*move*) Zeilen hinter Zeile *adr* verlagern
(.,.)	**n**	(*number*) Zeilen mit Zeilennummern ausgeben
(.,.)	**p**	(*print*) Zeilen ausgeben
	P	(*Prompt*) ed-Promptzeichen ein-/ausschalten
	q	(*quit*) ed verlassen
	Q	(*Quit*) wie **q** ohne Warnung über Änderungen
($)	**r** *datei*	(*read*) Inhalt der Datei *datei* hinter adressierte Zeile kopieren
(.,.)	**s**/*ra*/*ers*/	(*substitute*) Von *ra* abgedeckten Text durch *ers* ersetzen
(.,.)	**t***adr*	(*transfer*) Zeilen hinter Zeile *adr* kopieren
	u	(*undo*) letzte Änderung rückgängig machen
(1,$)	**v**/*ra*/*kdos*	(*veto*) wie **g**-Kommando, aber nicht für Zeilen mit *ra*
(1,$)	**V**/*ra*/	(*Veto*) interaktive Version zum **v**-Kommando
(1,$)	**w** *datei*	(*write*) Zeilen in Datei *datei* schreiben
(1,$)	**W** *datei*	(*Write*) Zeilen an Datei *datei* anhängen
($)	**=**	Zeilennummer ausgeben
	!*unix-kdo*	*unix-kdo* ausführen
(.+1)	(Enter)	Zeile `.+1` ausgeben

red - Eingeschränkte Version des Editors ed (restricted editor)

red [**-**] [**-p***string*] [*datei(en)*]
Mit **red** können nur Dateien im Working-Directory editiert werden.
Es ist nicht der Aufruf **!***unix-kdo* erlaubt, um Unix-Kommandos aus dem Editor heraus aufzurufen.
Sonst arbeitet **red** genauso wie **ed**.
red ist für bestimmte Benutzer gedacht, denen nicht der vollständige Zugang zum System erlaubt sein soll.

16 Übersicht über die regulären Ausdrücke

deckt ab / bewirkt	Dateinamen-Expandierung	awk, egrep	sed, ed, grep,pg, csplit	ex, vi, emacs	lex
ein beliebiges Zeichen	?	.	.	.	.
beliebige Zeichenkette	*	.*	.*	.*	.*
keine, eine oder mehrmalige Wiederholung	--	*	*	*	*
eine oder mehrmalige Wiederholung	--	+	\{1,\}	+	+
keine oder eine Wiederholung	--	?	\{0,1\}	?	?
n-malige Wiederholung	--	--	\{n\}	--	\{n,n\}
n- bis m-malige Wiederholung	--	--	\{n,m\}	--	\{n,m\}
mindestens n-malige Wiederholung	--	--	\{n,\}	--	--
Klasse von Zeichen	[...]	[...]	[...]	[...]	[...]
Komplement-Klasse von Zeichen	[!...]	[^...]	[^...]	[^...]	[^...]
Zeilenanfang	--	^RA	^RA	^RA	^RA
Zeilenende	--	RA$	RA$	RA$	RA$
Wortanfang	wort*	--	--	\<RA	--
Wortende	*wort	--	--	RA\>	--
a oder b (Alternation)	--	RA_1|RA_2	--	--	RA_1|RA_2
\runde Klammern	--	--	\(RA\)	\(RA\)	--
n-ter Teilausdruck	--	--	\n	\n	--

RA, RA_1, RA_2 stehen für reguläre Ausdrücke.

Zu **awk** und **egrep** sei noch folgendes angemerkt:

- *runde Klammern*: **(***r***)** deckt gleichen String wie *r* ab; mit Klammern kann man vorgegebene Prioritäten aufheben
- *Die Priorität der Operatoren (in aufsteigender Folge)*:
  Konkatenation, ***** **+** **?** (untereinander gleiche Priorität), **( )** (untereinander gleiche Priorität)
- Die Operatoren *****, **+** und **?** beziehen sich immer auf das vorhergehende Zeichen; sollen sie sich auf einen längeren Ausdruck beziehen, so ist dieser mit **(** .. **)** zu klammern. Um runde Klammern in einem Text abzudecken, ist deren Sonderbedeutung mit \ auszuschalten: **\(** bzw. **\)**. Die Alternation kann bei **egrep** auch durch ein Neuezeilezeichen (*Carriage-Return*) angegeben werden.

17 Index