Zusammenfassung UNIX/LINUX-Einführungskurs -Aufbaukurs und -Shell-Programmierung

Version 1.22 — 19.2.2016

Hinweise auf inhaltliche Fehler, Schreibfehler und unklare Formulierungen sowie Ergänzungen, Kommentare, Wünsche und Fragen können Sie gerne an den Autor richten:

OSTC Open Source Training and Consulting GmbH
Thomas Birnthaler
E-Mail: tb@ostc.de
Web: www.ostc.de

Inhaltsverzeichnis

1 Schreibweise in dieser Zusammenfassung

Folgende groß geschriebenen, englischen Namen werden in den Befehlen als Platzhalter verwendet. Sie sind jeweils durch passende echte Namen zu ersetzen (ebenso wie die anderen `GROSS` geschriebenen Namen in Befehlen).

`ARCH`	Archiv-Datei [**archive**]
`CMD`	Kommando-Name [**command**]
`DEST`	Ziel-Datei/Verzeichnis [**destination**]
`DEV`	Gerät [**device**]
`DIR`	Verzeichnis [**directory**]
`EXPR`	Ausdruck (numerischer oder logischer) [**expression**]
`FILE`	Datei-Name
`GID`	Gruppen-Nummer [**group identity**]
`GROUP`	Gruppen-Name [**group**]
`HOST`	Rechner-Name
`ID`	Nummer [**identity**]
`JOBNR`	Hintergrundprozeß/Druckauftrag-Nummer
`MOVE`	Bewegungs-Kommando im Vi
`NN`	Zahl
`NUM`	Zahl [**number**]
`OPT`	Options-Name
`PATH`	Pfad zu einer Datei
`PATTERN`	Muster
`PID`	Prozeß-ID
`REGEX`	Regulärer Ausdruck [**regular expression**]
`SCRIPT`	Skript-Datei (Awk, Sed, Shell, ...)
`SRC`	Quell-Datei/Verzeichnis [**source**]
`TEXT`	Beliebiger Text (*meist in Anführungzeichen zu setzen!*)
`TYPE`	(Daten)Typ
`UID`	Benutzer-Nummer [**user identity**]
`USER`	Benutzer-Name
`VALUE`	Beliebiger Wert
`VAR`	Variablen-Name

Optionale Angaben werden in eckige Klammern `[...]` eingeschlossen.

`<...>`	Taste
`Strg-<...>`	Steuerung/Control-Taste + Taste gemeinsam drücken [**control**]
`CR RETURN`	Return-Taste [**carriage return**]
`ESC`	Escape-Taste
`NL / LF`	Return-Taste [**newline, linefeed**]
`TAB`	Tabulator-Taste
`DEL`	Entfernen-Taste [**delete**]

2 Grundlagen

2.1 Datei-Inhalt ansehen

`more FILE`	Datei `FILE` seitenweise anzeigen [**mehr**]
`less FILE`	Datei `FILE` seitenweise anzeigen (*verbesserte Version!*) [**weniger**]

`<SPACE>`	Um eine Seite weiterblättern (Leertaste)
`b`	Zurückblättern (*nur* `less`*!*) [**back**]
`<RETURN>`	Um eine Zeile weiterblättern
`/TEXT <RETURN>`	`TEXT` suchen
`n`	Letzte Suche wiederholen [**next**]
`q`	`more`/`less` verlassen [**quit**]
`h`	Hilfe anzeigen [**help**]

2.2 Hilfestellung abrufen

`man [[-s] NN] CMD`	Beschreibung zu `CMD` aus `man`-Abschnitt `NN` [**section**]
`man -f CMD`	Einzeilige `man`-Beschreibungen zu `CMD` ausgeben [**find**]
`man -k TEXT`	Einzeilige `man`-Beschreibungen mit `TEXT` darin ausgeben [**key**]
`man -t CMD | lpr`	`CMD`-Beschreibung mit `troff` setzen und ausdrucken [**troff**]
`whatis CMD`	Analog `man -f CMD`
`apropos TEXT`	Analog `man -k TEXT`
`CMD --help`	Usage-Meldung zu `CMD` (GNU-Programme)
`CMD -?`	Usage-Meldung zu `CMD` (einige Programme)
`CMD -h`	Usage-Meldung zu `CMD` (einige Programme) [**help**]
`CMD -.`	Usage-Meldung zu `CMD` (einige Programme)
`help`	Alle eingebauten Shell-Kommandos ausgeben (`bash`)
`help CMD`	Usage-Meldung zu eingebautem Shell-`CMD` ausgeben
`hilfe`	Browsergestütztes SuSE-Hilfesystem aufrufen
`info CMD`	GNU-Beschreibung zu `CMD` anzeigen (interaktiv, Hypertext)

2.2.1 man-Abschnitte (Sections)

1	Ausführbare Programme oder Shellbefehle
2	(C-)Systemaufrufe (Kernelfunktionen)
3	(C-)Bibliotheksaufrufe (Funktionen in System-Bibliotheken `/usr/lib`)
4	Gerätedateien (gewöhnlich in `/dev`)
5	Formate und Bedeutung von Konfigurations-/Logdateien (z.B. `/etc/passwd`)
6	Spiele
7	Makropakete und Konventionen (z.B. `man(7)`, `groff(7)`)
8	Systemadministrationsbefehle (in der Regel nur für `root`)
9	Kernelroutinen (kein Standard)
l	Lokale Erweiterungen

2.2.2 man-Seitenaufbau

NAME	Name
SYNOPSIS	Syntax-Zusammenfassung
DESCRIPTION	Beschreibung
OPTIONS	Optionen
OVERVIEW	Übersicht
DEFAULTS	Normaleinstellungen
ENVIRONMENT	Umgebung (Variablen)
FILES	(Konfigurations)Dateien
EXAMPLES	Beispiele
NOTES	Bemerkungen
SEE ALSO	Siehe auch (*Verweise auf verwandte Kommandos!*)
REFERENCES	Verweise auf verwandte Kommandos
BUGS	Fehler
AUTHOR	Autor
HISTORY	Entwicklungs-Geschichte

2.3 Wichtige Steuertasten

<RETURN>	Eingegebenes Kommando ausführen
<BACKSPACE>	Letztes eingegebenes Zeichen entfernen
<Strg-C>	Laufendes Kommando abbrechen [**break**]
<Strg-D>	Abmelden/Dateiende anzeigen [**end of file**]
<Strg-Z>	Aktuellen Prozeß stoppen + in Hintergrund stellen
<Strg-S>	Terminal-Scrolling anhalten [**stop**]
<Strg-Q>	Terminal-Scrolling laufen lassen [**quit**]
<Strg-U>	Ganze Eingabezeile löschen [**undo**]
<Strg-W>	Letztes Wort auf der Eingabezeile löschen [**word**]
<Strg-H>	Backspace
<Strg-I>	Tabulator [**indent**]
<Strg-V>	Nächstes Steuer-Zeichen schützen (Escape) [**verbose**]
<Strg-G>	Klingel
<Strl-L>	Bildschirm neu aufbauen [**load**]

2.4 Wichtige LINUX-spezifische Steuertasten

`Cursor-Auf/Ab`	In alten Kommandos blättern
`Cursor-Links/Rechts`	In aktuellem Kommando bewegen
`<DEL>/<BACKSPACE>`	In aktuellem Kommando löschen
`Shift-Cursor-Auf/Ab`	Zeilenweise in Kommandoausgabe blättern
`Shift-Bild-Auf/Ab`	Seitenweise in Kommandoausgabe blättern
`Alt-F1...F6`	Auf Textterminal 1..6 umschalten
`Strg-Alt-F1...F6`	Auf Textterminal 1..6 umschalten (in Grafikoberfläche)
`Alt-Cursor-Links/Rechts`	Auf vorheriges/nächstes Textterminal umschalten
`Strg-Alt-F7`	Auf Grafikoberfläche umschalten
`Strg-Alt-BACKSPACE`	Grafikoberfläche beenden
`Strg-Alt-+/-`	Bildschirmauflösung verändern (+/− *auf Zahlentastatur!*)

2.5 Wichtige KDE-spezifische Steuertasten

`Alt-TAB`	Zum nächsten Fenster springen
`Shift-Alt-TAB`	Zum vorherigen Fenster springen
`Strg-F1..F8`	Zur Arbeitsfläche Nummer 1..8 springen
`Strg-TAB`	Zur nächsten Arbeitsfläche springen
`Shift-Strg-TAB`	Zur vorherigen Arbeitsfläche springen
`Alt-F1`	Startmenü des aktuellen Fensters öffnen
`Alt-F2`	Fenster für Befehl ausführen öffnen
`Alt-F3`	Systemmenü des aktuellen Fensters öffnen
`Alt-F4`	Aktuelles Fenster schließen
`Strg-Alt-ESC`	Fenster abschießen (*mit Totenkopf-Maus anklicken, ESC=Abbruch!*)
`Strg-ESC`	Systemüberwachungs-Programm starten (Prozeßliste)
`Strg-F4`	Alle Fenster minimieren

2.6 Maus-Bedienung unter X Window

`Linke Taste`	Text markieren durch Halten+Ziehen
`Linke Taste 2x`	Doppelklick markiert Wort, Halten+Ziehen markiert wortweise
`Mittlere Taste`	Markierten Text an Cursorposition einfügen (auch Linke+Rechte Taste)
`Rechte Taste`	Kontextmenü öffnen

2.7 Die wichtigsten UNIX-Kommandos

`man CMD`	Beschreibung zu `CMD` aus `man`-Abschnitt `NN` [**section**]
`cd`	Verzeichnis wechseln [**change directory**]
`ls`	Verzeichnis-Inhalt anzeigen [**list**]
`ls -l`	Verzeichnis-Inhalt ausführlich anzeigen [**long**]
`*`	Steht für alle Dateien in einem Verzeichnis
`exit`	Abmelden bzw. Terminal-Fenster schließen
`clear`	Bildschirminhalt des Terminals löschen
`date [+FORMAT]`	Datum + Uhrzeit ausgeben (gemäß `FORMAT`, z.B. `+%T`)
`echo "TEXT"`	`TEXT` auf dem Terminal ausgeben
`echo $VAR`	Inhalt der Variablen `VAR` anzeigen
`more FILE`	Datei `FILE` seitenweise anzeigdn [**mehr**]
`CMD | more`	Ausgabe von `CMD` seitenweise anzeigen
`grep "TEXT" FILE`	Nur Zeilen mit `TEXT` ausgeben [**global regular expr print**]
`CMD | grep "TEXT"`	Ausgabe von `CMD` seitenweise anzeigen
`wc -l FILE`	Zeilen in Datei `FILE` zählen [**word count/lines**]
`CMD | wc -l`	Zeilen in Ausgabe von `CMD` zählen
`lp(r) FILE`	Datei `FILE` ausdrucken [**line print**]
`CMD | lp(r)`	Ausgabe von `CMD` ausdrucken

3 Benutzer- und Gruppenverwaltung

3.1 Kommandos zur Benutzerverwaltung

`who`	Alle angemeldeten Benutzer anzeigen
`whoami`	Eigene Kennung anzeigen
`who am i`	Eigene Kennung anzeigen (ausführlicher)
`finger USER`	Weitere Informationen über Benutzer beschaffen
`su`	In Kennung `root` wechseln [**substitute user**]
`su -`	In Kennung + Umgebung von `root` wechseln
`su USER`	In Kennung `USER` wecheln (alte Umgebung)
`su - USER`	In Kennung + Umgebung von `USER` wechseln
`exit`	Rücksprung zur vorherigen Kennung bzw.
`exit`	Abmelden/Fenster schließen
`<Strg-D>`	Abmelden/Fenster schließen
`passwd`	Paßwort ändern [**password**]
`yppasswd`	NIS-Paßwort ändern [**yellow pages**]
`chsh`	Login-Shell ändern (LINUX) [**change shell**]
`chfn`	Benutzer-Beschreibung ändern (LINUX) [**change full name**]
`ypcat passwd`	NIS-Benutzerdatenbank ausgeben [**yellow pages catalog**]
`ypcat group`	NIS-Gruppendatenbank ausgeben
`ypcat hosts`	NIS-Rechnerdatenbank ausgeben

3.2 Kommandos zur Gruppenverwaltung

`groups`	Eigene Gruppenkennungen ausgeben
`id [-a]`	Benutzerkennung + alle Gruppenk. mit IDs anzeigen [**identity, all**]
`newgrp GROUP`	In Gruppe `GROUP` wechseln (*Paßwort notwendig!*) [**new group**]
	(legt Standardgruppe für Datei/Verzeichnis-Neuanlage fest)

4 Dateisystem

4.1 Auskunftsfunktionen

`cat FILE1 FILE2...`	Dateien aneinanderhängen + auf `stdout` ausgeben [**concatenate**]
`cd PATH`	In Verzeichnis `PATH` wechseln [**change directory**]
`cd`	In eigenes Heimat-Verz. wechseln (`$HOME`, z.B. `/home/USER`)
`cd /`	In Root-Verzeichnis wechseln
`cd ..`	In Eltern-Verzeichnis des aktuellen Verz. wechseln
`cd -`	In vorheriges akt. Verz. wechseln (hin- und her, `bash` und `ksh`)
`cd ~`	In eigenes Heimat-Verz. wechseln
`cd ~USER`	In Heimat-Verz. von `USER` wechseln
`df`	Größe + Belegungsgrad aller Partitionen anzeigen [**disk free**]
`df -k DIR`	Größe + Belegungsgrad der `DIR`-Partition anzeigen [**kilobyte**]
`du`	Platzbedarf Dateibaum ab aktuellem Verz. anzeigen [**disk used**]
`du DIR`	Platzbedarf Dateibaum ab `DIR` (inkl. Unterverz.) anzeigen
`du -s DIR`	Platzbedarf Dateibaum ab `DIR` (nur Summe) anzeigen [**sum**]
`du -k  du -h`	In Kilobyte bzw. für Menschen lesbar [**kilobyte, human readable**]
`file FILE`	Typ einer Datei anhand Datei-Inhalt ermitteln (`/etc/magic`)
`less FILE`	Datei seitenweise anzeigen (auch `more`, `pg`, `page`) [**mehr**]
`ls`	Dateinamen des akt. Verz. alphabetisch sortiert auflisten [**list**]
`ls DIR`	Dateinamen des Verzeichnisses `DIR` auflisten
`ls -l`	Dateinamen + Dateiattribute ... [**long**]
`ls -d`	Verzeichnis selbst, nicht seinen Inhalt ... [**directory**]
`ls -l -d`	Beide Optionen gleichzeitig (auch `ls -ld`)
`ls -a`	Auch versteckte Dateien (`. .. .xxx`) anzeigen [**all**]
`ls -i`	Inode-Nummer der Dateien mit ausgeben [**inode**]
`ls -r`	Sortierrichtung absteigend (Std: aufsteigend) [**reverse**]
`ls -t`	Dateien nach Änderungsdatum sortieren [**time**]
`ls -F`	Dateityp an Namen anhängen (`/`=Verz, `@`=Link, `*`=Exec, ...)
`ls -R`	Inhalt von Unterverzeichnissen mit auflisten [**recursive**]
`ll`	Häufig eine Abkürzung (Alias) für `ls -l` [**list long**]
`more FILE`	Datei seitenweise anzeigen (auch `less`, `pg`, `page`) [**mehr**]
`pwd`	Aktuelles Verzeichnis ausgeben [**print working directory**]

4.2 Dateitypen (Kommando ls -l)

-	Normale Datei
d	Verzeichnis [**directory**]
l	Symbolischer Link [**link**]
c	Zeichenorientiertes Gerät [**character device**]
b	Blockorientiertes Gerät [**block device**]
p	Named Pipe [**pipe**]
s	Socket [**socket**]

4.3 Verändernde Kommandos

`cp SRC DEST`	Datei `SRC` in Datei `DEST` kopieren (*überschreiben!*) [**copy**]
`cp SRC... DIR`	Dateien `SRC...` in Verzeichnis `DIR` kopieren
`cp -f SRC DEST`	Überschreiben von Dateien erzwingen [**force**]
`cp -i SRC DEST`	Rückfrage falls Dateien überschrieben werden [**interactive**]
`cp -r SRC DEST`	Dateibaum von `SRC` nach `DEST` kopieren [**recursive**]
`ln SRC DEST`	(Harten) Link namens `DEST` auf Datei `SRC` erzeugen [**link**]
`ln -s SRC DEST`	Symbol. Link namens `DEST` auf Datei `SRC` erzeugen [**symbolic**]
`mkdir DIR`	Verzeichnis anlegen [**make directory**]
`mkdir -p PATH`	Kompletten Verzeichnispfad anlegen [**path**]
`mv SRC DEST`	Datei `SRC` in `DEST` umbenennen [**move**]
`mv SRC... DIR`	Dateien `SRC...` in Verzeichnis `DIR` verschieben [**move**]
`rm FILE`	Datei löschen [**remove**]
`rm -f FILE`	Löschen von Dateien erzwingen [**force**]
`rm -i FILE`	Rückfrage pro zu löschende Datei [**interactive**]
`rm -r DIR`	Dateibaum komplett löschen [**recursive**]
`rmdir DIR`	Verzeichnis entfernen (*muss leer sein!*) [**remove directory**]
`rkdir -p PATH`	Kompletten Verzeichnispfad entfernen (*muss leer sein!*) [**path**]
`touch FILE`	Änd.Datum aktual. bzw. neue (leere) Datei anlegen [**berühren**]

4.4 Besitzverhältnis und Zugriffsrechte

4.4.1 Kommandos

`chgrp GROUP FILE`	`FILE` der Gruppe `GROUP` zuordnen (*Mitglied!*) [**change group**]
`chown USER FILE`	`FILE` dem Benutzer `USER` zuordnen (*nur* `root`*!*) [**change owner**]
`chown USER.GROUP`	Benutzer + Gruppe gleichzeitig ändern (*nur* `root`*!*)
`chmod MODE FILE`	Dateirechte ändern [**change mode**]
	`MODE` = `[ugoa][+-=][rwxst]` oder `(0)000-(7)777` (s.u.)
`chmod u+rwx FILE`	Besitzer erhält alle Rechte an `FILE` [**user**]
`chmod g-w FILE`	Besitzergruppe wird Schreibrecht an `FILE` entzogen [**group**]
`chmod o-rwx FILE`	Andere haben keine Rechte mehr an `FILE` [**others**]
`chmod a+r FILE`	Alle haben Leserecht für Datei `FILE` [**all**]
`chmod 644 FILE`	Rechte `rw-r--r--` für Datei `FILE` setzen
`chmod -R ...`	Rechte für ganzen Verzeichnisbaum ändern [**recursive**]
`umask [NNN]`	Standardrechte `NNN` für Datei/Verz-Neuanlage anzeigen/setzen, definiert die zu entfernenden(!) Rechte [**usage mask**]
`umask`	Gibt meist `022` aus (d.h. Schreibrecht für `g+o` entfernen)
`umask 027`	Schreibrecht für Gruppe + alle Rechte für Others entfernen

4.4.2 Zugriffsrechte für Dateien

`r`	[**read**]	Datei-Inhalt lesbar
`w`	[**write**]	Datei-Inhalt veränderbar
`x`	[**execute**]	Datei als Programm ausführbar
`s`	[**set user id**]	Programmausführung unter der Besitzer-Kennung
`s`	[**set group id**]	Programmausführung unter der Besitzergruppen-Kennung
`t`	[**sticky**]	Programm nach dem Start im Speicher/Swap halten (*veraltet!*)

4.4.3 Zugriffsrechte für Verzeichnisse

`r`	read	Verzeichnis lesbar (`ls`)
`w`	write	Verzeichnis veränderbar (Dateien anlegen/löschen/umbenennen verschieben, z.B. `cp`, `ln`, `mkdir`, `mv`, `rm`, `rmdir`, `touch`)
`x`	execute	Wechsel in Verzeichnis erlaubt (`cd`)
`s`	set user id	(keine Funktion)
`s`	set group id	Neue Dateien/Verz. automatisch der gleichen Gruppe zuordnen, neue Verzeichnisse haben wieder set group id Recht gesetzt
`t`	sticky	Nur Verzeichnis-Eigentümer (oder `root`) darf Dateien löschen

4.4.4 Dateirechte in oktaler + symbolischer Form

0	`---`	
1	`--x`	1
2	`-w-`	2
3	`-wx`	2 + 1
4	`r--`	4
5	`r-x`	4 + 1
6	`rw-`	4 + 2
7	`rwx`	4 + 2 + 1

000	`a-rwx`	Keine Rechte
700	`u+rwx,go-rwx`	Nur Benutzer-Rechte
070	`g+rwx,uo-rwx`	Nur Gruppen-Rechte
007	`o+rwx,ug-rwx`	Nur Rechte für alle anderen
777	`a+rwx`	Rechte für alle
1000	`u+s`	User-Set-ID-Recht
2000	`g+s`	Group-Set-ID-Recht
4000	`o+t`	Sticky Recht

4.4.5 LINUX-Sonderrechte

`lsattr FILE`	Sonderrechte auflisten [**list attributes**]
`chattr [+/-]OPT FILE`	Sonderrechte setzen/löschen [**change attributes**]

a	Nur Anfügen an Datei erlaubt [**append**]
A	Zugriffszeit nicht aufzeichnen (*Zeitersparnis!*) [**access date**]
c	Datei komprimieren (*in* `ext2` *nicht implementiert!*) [**compress**]
d	Datei nicht sichern [**dump**]
i	Datei nicht veränderbar (Löschen/Umbenennen/Schreiben/Link) [**immutable**]
s	Datei sicher löschen (mit 0-Bytes überschreiben) [**secure**]
S	Änderungen synchron schreiben (ohne Pufferung) [**synchron**]
u	Datei wiederherstellbar nach Löschen (*in* `ext2` *nicht implementiert!*) [**undelete**]

4.5 Dateien und Verzeichnisse

4.5.1 Standard-Verzeichnisnamen

/	Root-Verzeichnis
.	Aktuelles Verzeichnis
..	Eltern-Verzeichnis
/home	Standort der Heimat-Verzeichnisse aller Benutzer (auch `/usr/home` oder `/export/home`)
~	Eigenes Heimat-Verzeichnis (`$HOME`, nur `tcsh`, `bash`, `ksh`)
~USER	Heimat-Verzeichnis von Benutzer `USER` (nur `tcsh`, `bash`, `ksh`)

4.5.2 UNIX Standard-Verzeichnisse

`/bin` *	Grundlegende Befehle, zum Booten nötig [**binary**]
`/dev` *	Gerätedateien [**device**]
`/etc` *	(Fast alle) Konfigurationsdateien [**et cetera**]
`/home`	Heimat-Verzeichnisse aller Benutzer
`/lib` *	Grundlegende Systembibliotheken [**library**]
`/lost+found`	Für die von `fsck` [**file system check**] geretteten Dateien
`/opt`	Zusätzliche (kommerzielle) Software [**optional**]
`/sbin` *	Befehle für Systemadministrator, zum Booten nötig [**system binary**]
`/tmp`	Temporäre Dateien aller Benutzer [**temporary**] (für jeden schreibbar, z.B. Editor-Zwischendateien)
`/usr`	Zweite UNIX-Dateihierarchie [**unix system resources**]
`/usr/X(11R6)`	X Window (Version 11, Release 6)
`/usr/bin`	Weitere Befehle
`/usr/doc`	Dokumentationen (veraltet)
`/usr/etc`	Konfigurationsdateien
`/usr/local`	Lokale Installationen (z.B. `bin`, `lib`, `man`)
`/usr/man`	Manual-Seiten (veraltet)
`/usr/sbin`	Weitere Systembefehle (nicht für Single-User-Mode nötig)
`/var`	Für vom System angelegte Zwischendateien [**variable**]

Mit * gekennzeichnete Verzeichnisse werden zum Booten des Systems benötigt. Sie müssen daher auf der Root-Partition liegen (d.h. können nicht auf extra Partitionen liegen).

4.5.3 LINUX-spezifische Standard-Verzeichnisse

`/boot`	Kern + Bootdateien (`lilo`)
`/cdrom`	Mount-Point für CDROM-Laufwerk (auch `/media/cdrom`)
`/dvd`	Mount-Point für `DVD`-Laufwerk (auch `/media/dvd`)
`/floppy`	Mount-Point für Floppy-Laufwerk (auch `/media/floppy`)
`/mnt`	Mount-Point für temporäre Dateisysteme [**mount**]
`/opt/kde[23]`	Programme und Dateien für KDE-Oberfläche (Version 1/2/3)
`/opt/gnome`	Programme und Dateien für GNOME-Oberfläche
`/proc`	Pseudo-Verz. für Prozeß/Systeminformationen [**process**]
`/root`	Heimat-Verzeichnis der `root` (evtl. auch `/`)
`/usr/src`	Quellen der Systemsoftware (LINUX-Kern)
`/usr/share`	Dokumentationen (neu)
`/usr/share/doc`	Dokumentationen (neu)
`/usr/share/man`	Manual-Seiten (neu)
`/usr/share/doc/howto`	Howto-Seiten (neu)

4.5.4 Die wichtigsten Gerätedateien von LINUX

`/dev/cdrom`	Link auf verwendetes CDROM Laufwerk
`/dev/console`	UNIX-Konsole
`/dev/fd0 fd1`	1./2. Diskettenlaufwerk
`/dev/hda`	Erste IDE-Festplatte
`/dev/hda1 ... hda63`	Partitionen der ersten IDE-Festplatte
`/dev/hdb hdc ...`	Zweite, dritte, ... IDE-Festplatte
`/dev/mouse`	Link auf von der Maus verwendete Schnittstelle
`/dev/modem`	Link auf angeschlossenes Modem (meist am COM-Port)
`/dev/null`	Verschluckt Eingaben, liefert EOF (*Papierkorb, Schwarzes Loch!*)
`/dev/psaux`	PS/2-Maus
`/dev/pts/NN`	Virtuelle Konsole `NN` (in grafischer Oberfläche)
`/dev/scd0`	SCSI-CDROM (*für* `CD`*-Brennen notwendig!*)
`/dev/sda`	Erste SCSI-Festplatte
`/dev/sda1 ... sda15`	Partitionen der ersten SCSI-Festplatte
`/dev/sdb sdc ...`	Zweite, dritte, ... SCSI-Festplatte
`/dev/lp0 lp1 lp2`	1-3. parallele Schnittstelle (LPT1 - LPT3)
`/dev/ttyNN`	Virtuelle Textkonsole `NN`
`/dev/ttyS0 ... ttyS3`	1-4. serielle Schnittstelle (COM1 - COM4)
`/dev/zero`	Liefert beliebig viele Nullbytes

4.5.5 Wichtige zentrale Konfigurationsdateien

`/etc/fstab`	Beim Booten autom. zu mont. Dateisysteme [**file system table**]
`/etc/group`	Gruppenkonten (bis auf Paßwort)
`/etc/gshadow`	Gruppenpaßworte (verschlüsselt)
`/etc/inittab`	Bootvorgang steuern (Runlevel + Startup-Skripte) [**initialization table**]
`/etc/magic`	Datenbank mit Dateityp-Signaturen für `file`-Kommando
`/etc/passwd`	Benutzerkonten (bis auf Paßwort)
`/etc/profile`	Login-Skript der Shells `sh`, `bash`, `ksh`
`/etc/printcap`	Konfiguration der verfügbaren Drucker (`lpd`)
`/etc/shadow`	Benutzerpaßworte (verschlüsselt)
`/etc/shells`	Erlaubte Login-Shells (`chsh`)
`/etc/XF86Config`	X-Window Konfiguration

4.5.6 Wichtige zentrale Netzwerk-Konfigurationsdateien

`/etc/exports`	Exportierte NFS-Laufwerke (zum Einhängen übers Netz)
`/etc/hostname`	Rechnername (evtl. `HOSTNAME`)
`/etc/hosts`	Zuordnung Rechnername ↔ IP-Adresse (Nameserver-Ersatz)
`/etc/hosts.lpd`	Erlaubte Hosts für Netzwerkdrucker-Zugriff [**line print demon**]
`/etc/hosts.allow`	Zugriffsteuerung `tcpd` erlaubt diesen Rechner den Zugang
`/etc/hosts.deny`	Zugriffsteuerung `tcpd` verbietet diesen Rechner den Zugang
`/etc/inetd.conf`	Von `inetd` zu startende Netzwerkdienste
`/etc/named.conf`	`DNS`-Konfiguration
`/etc/networks`	Zuordnung Netzname ↔ IP-Adresse (Nameserver-Ersatz)
`/etc/nsswitch.conf`	Definiert die Reihenfolge der Namensauflösung [**name service switch**]
`/etc/protocols`	Verfügbare Netzwerkprotokolle
`/etc/resolv.conf`	`DNS`-Suchreihenfolge + Nameserver festlegen
`/etc/route.conf`	Routing-Tabelle
`/etc/services`	Zuordnung Dienstname ↔ Portnummer

4.5.7 Wichtige lokale Konfigurationsdateien/Verzeichnisse

`.alias`	Alias-Definitionen (*bei SuSE von* `.bashrc` *gelesen!*)
`.bashrc`	Beim Start einer interaktiven Bash ausgeführt
`.cshrc`	Beim Start einer interaktiven `csh` ausgeführt
`.emacs`	Emacs-Konfiguration
`.exrc`	Vi/Ex-Konfiguration
`.forward`	Mail-Umlenkung
`.login`	Beim Start einer Login-`csh` ausgeführt
`.logout`	Beim Verlassen einer Login-`csh` ausgeführt
`.netscape`	Netscape-Einstellungen
`.profile`	Beim Start einer Login-Shell vom Typ `sh`, `bash`, `ksh` ausgeführt
`.rhosts`	Erlaubte Hosts für remote-Kommandos ohne Paßwort
`.vimrc`	Vim-Konfiguration
`.xinitrc`	X Window Start-Skript (`startx` → `xinit`)
`.xsession`	X Window Start-Skript (`xdm` oder `kdm`)

5 Prozeßverwaltung

5.1 Kommandos zur Prozeßverwaltung

`ps`	Eigene Prozesse auflisten (inkl. `ps` selbst) [**process status**]
`ps -A`	Alle Prozesse [**all**]
`ps -e`	Alle Prozesse [**every**]
`ps -f`	Prozesse + alle Prozeßattribute [**full**]
`ps -l`	Prozesse + viele Prozessattribute [**long**]
`ps -t TERM`	Prozesse zu Terminal `TERM` anzeigen [**terminal**]
`ps -u USER`	Prozesse zu User `USER` anzeigen [**user**]
`ps a`	Prozesse aller Anwender (mit zugeordnetem Terminal) [**all**]
`ps f`	Prozessabhängigkeiten anzeigen [**forest**]
`ps u`	Username mit anzeigen [**user**]
`ps w`	Aufruf des Prozesses ausführlich anzeigen (*mehrfach!*) [**wide**]
`ps x`	Prozesse ohne zugeordnetes Terminal (Dämonen) [**extended**]
`kill [-SIGNAL] PID`	Prozess (durch Signal `SIGNAL`) beeinflussen [**töten**]
`kill PID`	Prozess abbrechen (Standardsignal `SIGTERM` = 15, ignorierbar)
`kill -9 PID`	Prozeß unbedingt abbrechen (`SIGKILL` = 9, nicht ignorierbar)
`kill -1 PID`	Prozeß neu initialisieren (`SIGHUP` = 1)
`kill -l`	Liste aller Signal-Namen + Nummern ausgeben [**list**]
`<Strg-C>`	Aktuellen Prozeß abbrechen (`SIGINT` = 2) [**break**]
`(k)top`	Prozessliste + Prozessattribute ständig anzeigen (LINUX)
`ksysguard`	Analog (LINUX)
`p(s)tree`	Prozessabhängigkeiten (Baum) anzeigen (LINUX) [**process tree**]

Die `ps`-Optionen der UNIX-Hauptlinien BSD und System V unterscheiden sich in ihrer Bedeutung (leider). Daher wird unter LINUX eine Option mit führendem `-` als BSD-Option und ohne führendes `-` als System V-Option interpretiert.

5.2 Kommandos zur Hintergrund-Prozeß-Verwaltung

`CMD ... &`	Kommando im Hintergrund starten
`<Strg-Z>`	Aktuellen Prozeß stoppen + in Hintergrund stellen (`SIGTSTP` = 20)
`bg [JOBNR]`	Prozeß mit `JOBNR` im Hintergrund starten [**background**]
`fg [JOBNR]`	Prozeß mit `JOBNR` in den Vordergrund holen [**foreground**]
`jobs`	Hintergrundprozesse auflisten
`kill %JOBNR`	Hintergrundprozeß `JOBNR` beenden (`%` *notwendig!*)
`nohup CMD...`	`CMD` wird beim Abmelden nicht abgebrochen [**no hangup**] (Ein/Ausgabe von/auf Datei erhalten/machen, Std: `nohup.out`)
`stop JOBNR`	Prozeß mit `JOBNR` im Hintergrund anhalten (`kill -STOP`, `ksh`)

5.3 Die wichtigsten Signale

SIGHUP	1	Konfigurationsdatei lesen [**hangup**]
SIGINT	2	Abbruch durch Eingabe von <Strg-C> [**interrupt**]
SIGQUIT	3	Prozeßende erreicht
SIGKILL	9	Bedingungsloser Prozeßabbruch [**töten**]
SIGTERM	15	Prozeß beenden, wird evtl. ignoriert [**terminate**]
SIGCONT	18	Ausgabe weiterlaufen lassen durch Eingabe von <Strg-Q> [**continue**]
SIGSTOP	19	Ausgabe anhalten durch Eingabe von <Strg-S>
SIGTSTP	24	In Hintergrund stellen durch Eingabe von <Strg-Z>

6 Shell

Kommentar wird in der Shell durch # eingeleitet und gilt bis zum Zeilenende.

6.1 Shell-Typen

sh	Bourne-Shell	Ur-Shell, Skript-orientiert
csh	C-Shell	Angelehnt an C, Interaktions-orientiert
tcsh	Tenex-C-Shell	Erweiterung + Verbesserung der C-Shell (BSD)
ksh	Korn-Shell	Standard unter UNIX System V
bash	Bo(u)rn(e)-Again-Shell	Standard unter LINUX/GNU
ash	Anfang des Alphabets	Sehr einfache Shell, klein
zsh	Ende des Alphabets	Ultimative Shell, extremer Funktionsumfang

Shells sind sowohl zur interaktiven Benutzung auf der Kommandozeile als auch zur Ausführung von Shell-Skripten (Batch-Dateien) gedacht. Hier eine Bewertung ihrer Eignung für den interaktiven (I) bzw. den Batch-Gebrauch (B):

Name	SRC	Int	Batch
sh		−−	+
csh		+	−
tcsh	csh	++	+−
ksh	sh	+	++
bash	sh	++	++
ash	sh	+−	+−
zsh	sh	++	++

6.2 Shell-Konfigurationsdateien

Es muss unterschieden werden zwischen Login-Shells (die beim Anmelden gestartet werden) und (interaktive) Sub-Shells, die zum Ausführen eines Shell-Skriptes oder Kommandos gestartet werden.

6.2.1 Konfigurationsdateien der Bourne-Shell

1	/etc/profile	Zentrales Login-Skript
2	~/.profile	Benutzerabhängiges Login-Skript
		Interaktive Bourne-Shells lesen keine Konf.dateien!
		Bourne-Shells führen beim Logout kein Skript aus!

6.2.2 Konfigurationsdateien der Bash

1	/etc/profile	Zentrales Login-Skript
2a	~/.bash_profile	Benutzerabhängiges Login-Skript oder
2b	~/.bash_login	Benutzerabhängiges Login-Skript oder
2c	~/.profile	Benutzerabhängiges Login-Skript
	~/.bashrc	Beim Start einer interaktiven Bash ausgeführt
	~/.alias	Alias-Definitionen (nur bei SuSE?)
	~/.bash_logout	Beim Verlassen einer Bash-Login-Shell ausgeführt

6.2.3 Konfigurationsdateien der C-Shell

1	/etc/cshrc	Beim Start einer Login-csh ausgeführt
2	~/.cshrc	Beim Start einer Login-csh ausgeführt
3	~/.login	Benutzerabhängiges Login-Skript
	~/.cshrc	Beim Start einer interaktiven csh ausgeführt
	~/.logout	Beim Verlassen einer Login-csh ausgeführt

6.2.4 Konfigurationsdateien der TC-Shell

1	/etc/csh.cshrc	Zentrales Login-Skript der tcsh
2	/etc/csh.login	Zentrales Login-Skript der tcsh
3a	~/.tcshrc	Beim Start einer Login-tcsh ausgeführt oder
3b	~/.cshrc	Beim Start einer Login-tcsh ausgeführt
4	~/.history	Benutzerabhängiges Login-Skript der tcsh
5	~/.login	Benutzerabhängiges Login-Skript der tcsh
6	~/.cshdirs	Benutzerabhängiges Login-Skript der tcsh
1	/etc/csh.cshrc	Zentrales Skript beim Start einer interaktiven tcsh
2a	~/.tcshrc	Beim Start einer interaktiven tcsh ausgeführt oder
2b	~/.cshrc	Beim Start einer interaktiven tcsh ausgeführt
1	/etc/csh.logout	Beim Verlassen einer Login-tcsh ausgeführt
2	~/.logout	Beim Verlassen einer Login-tcsh ausgeführt

6.2.5 Konfigurationsdateien der Korn-Shell

1	`/etc/profile`	Zentrales Login-Skript
2	`~/.profile`	Benutzerabhängiges Login-Skript
	`$ENV`	Bestimmt die für eine interaktive `ksh` auszuführende
	`~/.kshrc`	Datei (Inhalt `~/.kshrc`) [**environment**]

6.3 Shell-Operationen

1	History-Ersetzung (Teile alter Kommandos wiederverwenden)
2	Zerlegung in Token (Whitespace trennt die einzelnen Worte)
3	Quotierung berücksichtigen (`@0@'...'` und `\`)
4	Aliase ersetzen
5	I/O-Umlenkungen einrichten (`< > >> 2> 2>> >& 1>&2 |`)
6	Variablen durch ihren Wert ersetzen (`$VAR`)
7	Kommando-Substitution `` `...` `` `$(...)` durchführen (durch Ergebnis ersetzen)
8	Dateinamen expandieren (Metazeichen `* ? [] ~ {}` ersetzen)
9	Kommando suchen (Pfadsuche anhand Variabler `PATH`)

6.4 Pfadsuche (bei Aufruf von CMD)

1	Built-in/Keyword (z.B. `cd`, `echo`, `exit`, ...)
2	Alias (z.B. `alias ll='ls -lF'`)
3	Funktion (z.B. `ll() { /bin/ls -lF $*; }`)
4	Von links nach rechts in allen im Suchpfad `PATH` angegebenen Verzeichnissen (durch `:` getrennt) nach einer Datei namens `CMD` suchen; diese muss ausführbar (+ lesbar bei Shell-Skript) sein
5	Nicht gefunden → Meldung: `error: command CMD not found`

Standardmäßig wird nicht nach Kommandos im aktuellen Verzeichnis gesucht, außer im Suchpfad steht `.` (Punkt). Die Suche kann simuliert werden durch:

`type CMD`	Kommando-Typ ermitteln (Builtin, Alias, Funktion, Programm)
`type -a CMD`	Alle passenden Programme (nicht nur das 1.) ausgeben [**all**]
`type -p CMD`	Nur Suchpfad zu passendem Programm ausgeben [**path**]
`which CMD`	Entspricht `type -p` bei der `(t)csh`
`whence CMD`	Entspricht `type -p` bei der `ksh`

6.5 Shell- und Umgebungs-Variablen

6.5.1 Kommandos für Shell-Variablen

`VAR="TEXT"`	Erzeugt eine Shell-Variable und weist ihr den Wert `TEXT` zu (*keine Leerzeichen um = erlaubt!*)
`set`	Alle Shell-Variablen auflisten
`$VAR`	Zugriff auf den Wert (Inhalt) einer Shell-Variablen
`${VAR}xxx`	Alternative Variante falls direkt dahinter Text `xxx` steht
`VAR=`	Löschen einer Shell-Variablen (leer)
`unset VAR`	Löschen einer Shell-Variablen (undefiniert != leer)

6.5.2 Kommandos für Umgebungs-Variablen

`export`	Alle nicht exportierten Variablen auflisten
`export VAR`	Shell-Variable zu einer Umgebungs-Variable machen
`export VAR="TEXT"`	Analog + gleichzeitige Wertzuweisung (`bash`,`ksh`)
`env`	Alle Umgebungs-Variablen anzeigen

6.5.3 Variablen-Kommandos unter der (T)C-Shell

`set VAR = "TEXT"`	Erzeugt eine Shell-Variable (*Leerzeichen um = nötig!*)
`set VAR = (aaa bbb)`	Array-Variable mit Werteliste füllen
`set`	Alle Shell-Variablen auflisten
`unset VAR`	Löschen einer Shell-Variablen (undefiniert != leer)
`setenv VAR "TEXT"`	Erzeugt eine Umgebungs-Variable (*kein =!*)
`setenv VAR (aaa bbb)`	Array-Variable mit Werteliste füllen
`printenv`	Alle Umgebungs-Variablen anzeigen
`unsetenv VAR`	Löschen einer Umgebungs-Variablen

6.5.4 Einige Standard-Variablen

CDPATH	Suchpfad für cd [**change directory path**]
HOME	Standardverzeichnis für cd (Heimat-Verzeichnis)
EDITOR	Definiert den Kommandozeilen-Editor (vi/emacs)
IFS	Whitespace-Zeichen für read [**internal field separator**]
LANG	Sprache für Fehlermeldungen einstellen [**language**]
LOGNAME	Aktueller Loginname (auf manchen Systemen nicht vorhanden)
MANPATH	Suchpfad für man, whatis, apropos
OLDPWD	Vorheriges aktuelles Verzeichnis [**print working directory**]
PAGER	Programm zum seitenweisen Blättern (more)
PATH	Suchpfad für Kommandoaufruf
PS1	Shell-Prompt ($)
PS2	Fortsetzungs-Prompt (>)
PS3	Prompt der select-Menüabfrage
PS4	Debug-Prompt (+)
PWD	Aktuelles Verzeichnis [**print working directory**]
RANDOM	Zufallszahl (0..32767)
SHELL	Name der Login-Shell
TERM	Terminaltyp (Editoren, more, curses-Bibliothek)
TZ	Zeitzone [**time zone**]
USER	Aktueller Loginname (auf manchen Systemen nicht vorhanden)

6.5.5 Eingabeprompt-Definition (bash)

Die Variable PS1 legt das Aussehen des Eingabeprompts fest. Sie wird entweder für alle Benutzer gemeinsam in /etc/profile oder für einzelne Benutzer in ~/.profile initialisiert. Folgende Sonderzeichen in PS1 setzen spezielle (variable) Komponenten im Prompt ein:

\d	Datum in der Form Sun Dec 24 [**date**]
\H	Kompletter Rechner-Name (FQDN) [**host**]
\h	Rechner-Name bis zum ersten Punkt (.) [**host**]
\n	Zeilenvorschub [**newline**]
\r	Zeilenrücklauf [**carriage return**]
\s	Shell-Name [**shell**]
\t	Zeit in der Form hh:mm:ss [**time**]
\u	Benutzer-Name [**user**]
\w	Arbeitsverzeichnis [**working directory**]
\W	Arbeitsverzeichnis (nur Teil nach dem letzten /) [**working directory**]
\$	$-Zeichen für normalen Benutzer, #-Zeichen für root
\#	Nummer des aktuellen Kommandos (in der aktuellen Sitzung)
\!	History-Nummer des aktuellen Kommandos (über Sitzungen hinweg)
\[	Beginnt Terminalsteuersequenz (zählt nicht zur Kommandolänge mit)
\]	Beendet Terminalsteuersequenz (zählt nicht zur Kommandolänge mit)

6.5.6 Eingabeprompt-Definition (tcsh)

Die Variable `prompt` legt das Aussehen des Eingabeprompts fest. Sie wird entweder für alle Benutzer gemeinsam in `/etc/tcshrc.rc` oder für einzelne Benutzer in `~/.login` initialisiert. Folgende Sonderzeichen in `prompt` setzen spezielle (variable) Komponenten im Prompt ein:

`%d/%D`	Datum-Tag [**day**]
`%m/%M`	Datum-Monat [**month**]
`%y/%Y`	Datum-Jahr [**year**]
`%t/%T`	Uhrzeit [**time**]
`%M`	Kompletter Rechner-Name (FQDN) [**machine**]
`%m`	Rechner-Name bis zum ersten Punkt (`.`) [**machine**]
`%n`	Benutzer-Name [**name**]
`%c`	Arbeitsverzeichnis [**current directory**]
`%C2`	Arbeitsverzeichnis (nur 2 Teile) [**current directory**]
`%#`	`$`-Zeichen für normalen Benutzer, `#`-Zeichen für root
`%h`	Nummer des aktuellen Kommandos [**history**]
`%b/%B`	Fettschrift an/aus [**bold**]
`%s/%S`	Hervorhebung an/aus [**standout**]

6.6 Kommando-Wiederholung (bash + ksh)

`HISTFILE`	Dateiname zur Speicherung der Kmdos. (Std: `~/.bash_history`)
`HISTSIZE`	Anzahl zu merkender Kommandos (Std: 50/500)
`history`	Die letzten `HISTSIZE` Kommandos auflisten
`history NN`	Die letzten `NN` Kommandos auflisten
`!NR`	Kommando mit der Nummer `NR` wiederholen (`bash`)
`!TEXT`	Kommando mit Textanfang `TEXT` wiederholen (`bash`)
`r NR`	Kommando mit der Nummer `NR` wiederholen (`ksh`) [**repeat**]
`r TEXT`	Kommando mit Textanfang `TEXT` wiederholen (`ksh`)
`^OLD^NEW`	Letztes Kommando wiederh., zuvor `OLD` durch `NEW` ersetzen
`!!`	Letztes Kommando wiederholen (auch in der Form `vi `!!``)
`!-2`	Vorletztes Kommando wiederholen, usw.

6.7 Dateinamen-Vervollständigung (bash + ksh)

In der `bash` sind verwendbar:

`NNN <TAB>`	Kommando-Namen `NNN` mit passend. Kdo. vervollständigen (*1. Wort!*)
`NNN <TAB><TAB>`	Liste der Alternativen bei mehrdeutigem Namen `NNN` anzeigen
`CMD NNN <TAB>`	Datei-Namen vervollständigen (*nicht 1. Wort!*)
`$NNN <TAB>`	Variablen-Namen vervollständigen (`bash`)
`~NNN <TAB>`	Benutzer-Namen vervollständigen (`bash`)
`@NNN <TAB>`	Rechner-Namen vervollständigen (`bash`)

In der `ksh` sind verwendbar:

`<ESC> *`	Aktuelles Wort zu allen passenden Dateinamen expandieren
`<ESC> \`	Aktuelles Wort durch eindeutigen passenden Dateinamen ersetzen
`<ESC> /TEXT`	Kommando mit `TEXT` darin suchen und ausführen

Funktioniert für Kommandos (1. Wort) und Dateinamen (ab 2. Wort).

6.8 Aliase

`alias ll='/bin/ls -lF'`	Alias `ll` definieren (*Rekursion vermeiden!*)
`ll *.c *.h`	Wird zu `/bin/ls -lF *.c *.h` expandiert
`alias`	Alle Aliase anzeigen
`alias ll`	Alias `ll` anzeigen
`unalias ll`	Alias `ll` löschen

6.8.1 Nützliche Aliase

`c`	`clear`	Bildschirm löschen
`cx`	`chmod u+x`	Ausführungsrecht setzen
`h`	`history`	Die letzten Befehle anzeigen
`hg`	`history | grep`	In den letzten Befehlen suchen
`ld`	`ls -l | grep '^d'`	Verzeichnisse des aktuellen Verzeichnisses ausgeben
`lt`	`ls -ltr`	Dateien nach Änderungsdatum sortieren (neueste zuletzt)
`lss`	`ls -l | sort +4 -5`	Dateien nach Größe sortieren (größte zuletzt)
`m`	`less`	Dateien seitenweise anzeigen
`mfl`	`mount /floppy`	Floppy einhängen
`umfl`	`umount /floppy`	Floppy aushängen
`va`	`vi $HOME/.alias`	Eigene Aliase editieren
`sa`	`. $HOME/.alias`	Eigene Aliase aktivieren

6.9 Funktionen

`ll() {`
`  /bin/ls -lF`
`}`
	Funktion `ll` definieren (*Rekursion vermeiden!*)

`ll() { /bin/ls -lF; }`	Analog in einer Zeile (`;` *+ Leerzeichen nach* `}` *sind nötig!*)
`typeset -f`	Alle Funktionen anzeigen
`typeset -f ll`	Funktion `ll` anzeigen
`unset -f ll`	Funktion `ll` löschen

6.10 Ein/Ausgabe-Umlenkung

`CMD < FILE`	`stdin` von Datei `FILE` lesen
`CMD > FILE`	`stdout` auf Datei `FILE` schreiben (*vorher gelöscht!*)
`CMD >> FILE`	`stdout` an Datei `FILE` anhängen
`CMD 2> FILE`	`stderr` auf Datei `FILE` schreiben (*vorher gelöscht!*)
`CMD 2>> FILE`	`stderr` an Datei `FILE` anhängen
`CMD1 | CMD2`	`stdout` von `CMD1` mit `stdin` von `CMD2` verbinden [**pipe**]

6.11 Spezielle Ein-/Ausgabeumlenkung

`CMD 1>&2`	`stdout`(1) zu `stderr`(2) hinzufügen (kombinieren)
`CMD 2>&1`	`stderr`(2) zu `stdout`(1) hinzufügen (kombinieren)
`CMD &> FILE`	`stdout` und `stderr` auf Datei `FILE` schreiben (*vorher gelöscht!*) entspricht `> FILE 2>&1` oder `2> FILE 1>&2`
`CMD >& FILE`	`stdout` zu `stderr` hinzufügen (`csh`)
`CMD >| FILE`	`stdout` auf Datei `FILE` schreiben (*erzwingen bei* `noclobber`*!*)
`CMD1 |& CMD2`	`stdout` und `stderr` an `CMD2` übergeben

6.12 Here-Dokument

`CMD <<TEXT`	`stdin` folgt, bis `TEXT` in einer Zeile für sich steht (Ersetzung von `$VAR`, `` `CMD` ``, `$(CMD)` und `\` wird durchgeführt)
`CMD <<-TEXT`	Analog, aber Tabulatoren am Zeilenanfang ignorieren
`CMD <<"TEXT"`	Analog, aber keine Ersetzung von `$VAR`, `` `CMD` ``, `$(CMD)` und `\`
`CMD <<'TEXT'`	Analog, aber keine Ersetzung von `$VAR`, `` `CMD` ``, `$(CMD)` und `\`
`CMD <<\TEXT`	Analog, aber keine Ersetzung von `$VAR`, `` `CMD` ``, `$(CMD)` und `\`

6.13 Dateinamen-Expansion

`/`	Verzeichnis-Trenner (*nicht in Dateinamen verwendbar!*)
`?`	Genau ein beliebiges Zeichen
`*`	0 oder mehr beliebige Zeichen (*Wh. von Z. ist nicht möglich!*)
`\X`	Das Metazeichen `X` selbst (der Backslash `\` quotiert)
`[abc] [a-z]`	Genau ein Zeichen aus der angegebenen Menge
`[!abc] [!a-z]`	Genau ein Zeichen nicht(!) aus der angegebenen Menge (`sh`)
`[^abc] [^a-z]`	Genau ein Zeichen nicht(!) aus der angegebenen Menge (`csh`)
`~`	Heimat-Verzeichnis des aktuellen Benutzers (`tcsh`, `bash`)
`~USER`	Heimat-Verzeichnis des Benutzers `USER` (`tcsh`, `bash`)
`{abc,def,...}`	Liste der angegebenen Zeichenketten (`csh`, `ksh`, `bash`)

6.14 Kommando-Substitution

`` `CMD` ``	`CMD` ausführen und durch Ergebnis ersetzen (alte Form, in jeder Shell)
`$(CMD)`	Analog (neue Form, nur in `bash`/`ksh`, ähnelt Variablen-Ersetzung)

6.15 Shell-Quotierung

`'...'`	Sämtliche Sonderzeichen abschalten [**single quote**/**tick**]
`"..."`	Alle Sonderzeichen bis auf `$`, `` `...` ``, `$(...)` und `\` abschalten [**double quote**]
`\X`	Genau ein (das folgende) Sonderzeichen `X` abschalten [**backslash**]

6.16 Kommando-Listen

`(CMD; ...)`	Kmdos in neuer Shell ausführen (gemeinsam Umlenken, Hintergrund)
`{CMD; ...; }`	Kmdos in akt. Shell ausführen (gemeinsam Umlenken, Hintergrund)
`CMD1 ; CMD2`	`CMD1` ausführen, danach `CMD2` ausführen (*wartet auf Ende von* `CMD1`*!*)
`CMD1 & CMD2`	`CMD1` starten, danach `CMD2` starten (*wartet nicht auf Ende von* `CMD1`*!*)
`CMD1 && CMD2`	`CMD1` ausführen, bei Erfolg `CMD2` ausführen (`CMD1` hat Exit-Status 0)
`CMD1 || CMD2`	`CMD1` ausführen, bei Mißerfolg `CMD2` ausführen (`CMD1` hat Exit-Status 1)

7 Sonstige Kommandos

`clear`	Löscht den Bildschirminhalt eines Terminals
`echo TEXT1 ...`	Gibt `TEXT1 ...` auf Terminal aus (durch je 1 Leerzeichen getrennt)
`echo $SHELL`	Name der Login-Shell ausgeben
`banner "TEXT"`	`TEXT` groß geschrieben ausgeben
`date [+FORMAT]`	Gibt Datum + Uhrzeit aus (gemäß `FORMAT`, s.u.)
`cal [MON] YEAR`	Jahres/Monatskalender ausgeben (*Sep 1752 ist ungewöhnlich!*)
`sleep [NN]`	1 (`NN`) Sekunden warten
`fmt -w 70 FILE`	Absätze (Leerzeilen trennen) auf 70 Z. Breite umbrechen [**format**]
`pr FILE`	Datei `FILE` seitenweise für Ausdruck aufteilen [**print**]
`spell FILE`	Datei `FILE` korrekturlesen [**buchstabieren**]
`troff FILE`	UNIX-Textformatierung von `FILE` [**text raster output file format**]
`write USER@HOST`	Nachricht (`stdin`) auf Bildschirm von `USER` schreiben
`wall`	Nachricht (`stdin`) auf Bildschirm aller User schreiben [**write all**]
`talk USER@HOST`	Mit `USER` telefonieren (zweigeteilter Bildschirm)

7.1 Wichtige Format-Angaben zu date

`%H`	Stunde [**hour**]
`%M`	Minute [**minute**]
`%S`	Sekunde [**second**]
`%T`	Uhrzeit `hh:mm:ss` [**time**]
`%d`	Tag [**day**]
`%M`	Monat [**month**]
`%y/%Y`	Jahr zweistellig/vierstellig [**year**]
`%D`	Datum `dd/mm/yy` [**date**]
`%a`	Wochentagname (`Sun`, ..., `Sat`)
`%w`	Wochentagnummer (`0`=Sonntag, ..., `6`=Samstag) [**weekday**]
`%b`	Monatsname (`Jan`, ..., `Dec`)
`%j`	Tag des Jahres (1..366)
`%W`	Wochennummer des Jahres (1..53) [**week**]

8 Vi

8.1 Die wichtigsten Vi-Befehle

~/.exrc ~/.vimrc	Vi/Vim-Konfigurationsdatei [**ex/vim resource**]
ESC	Kommando-Modus beenden (2x → piepst + ignoriert)
ESC :q! CR	Vi sicher verlassen (ohne Änderung an Datei)
vi [FILE] CR	Datei FILE bzw. leere Datei (:w FILE *nötig!*) editieren
:wq CR :x CR ZZ	Editierte Datei schreiben + Vi verlassen [**write+quit/exit**]
:q CR :q! CR	Editierte Datei nicht schr. + Vi verlassen (!*=erzwingen!*) [**quit**]
:w CR :w FILE CR	Editierten Text (auf Datei FILE) schreiben [**write**]
:e[!] FILE CR	Datei FILE statt aktueller editieren (!*=erzwingen!*) [**edit**]
:f FILE CR	Aktuelle Datei mit dem Namen FILE benennen [**file**]
:n CR	Nächste Datei editieren (falls mehrere angegeben) [**next**]
:prev :N VIM!	Vorherige Datei editieren [**previous/Next**]
:args CR	Alle beim Aufruf des Vi angeg. Dateien anzeigen [**arguments**]
:rew CR	Zur 1. beim Aufruf angegeb. Datei zurückspringen [**rewind**]
a TEXT... ESC	Nach Cursorposition TEXT... einfügen [**append**]
A TEXT... ESC	Am Zeilenende TEXT... einfügen [**Append**]
i TEXT... ESC	Vor Cursorposition TEXT... einfügen [**insert**]
I TEXT... ESC	Am Zeilenanfang TEXT... einfügen [**Insert**]
o TEXT... ESC	Neue Zeile TEXT... unter aktueller Zeile einfügen [**open**]
O TEXT... ESC	Neue Zeile TEXT... über aktueller Zeile einfügen [**Open**]
x X	Ein Zeichen unter/vor Cursor löschen [**crossout**]
dd dw dMOVE	Zeile/Wort/gemäß Bewegung MOVE löschen [**delete**]
rX	Ein Zeichen unter Cursor mit X übertippen (*kein* ESC*!*) [**replace**]
R TEXT... ESC	Ab Cursorposition mit TEXT... übertippen [**Replace**]
s TEXT... ESC	Ein Zeichent unter Cursor mit TEXT... übertippen [**substitute**]
cc cw cMOVE ESC	Zeile/Wort/gemäß Bewegung MOVE ändern [**change**]
yy yw yMOVE ESC	Zeile/Wort/gemäß Bew. MOVE in Zw.speicher kopieren [**yank/copy**]
p P	Zwischensp. nach/vor akt. Zeichen/Zeile einfügen [**put**]
h l	Zeichen links/rechts (auch <BACKSPACE>/<SPACE>)
j k	Zeile ab/auf (auch +/−/<RETURN>)
w b e	Wort vor/zurück/Wortende [**word/backword/endword**]
W B E	WORT vor/zurück/WORTende (*inkl. Ziffern, Satzzeichen!*)
0 ^ $ NN|	Zum Zeilenanfang/1.Buchstaben/Zl.ende/Spalte NN springen
<Strg-F> <Strg-B>	Bildschirmseite auf/abblättern [**forward/backward**]
G 1G NNG	Zum Dateiende/Dateianfang/Zeile NN springen [**Go**]
mX 'X `X	Marke X (a-z) setzen, Zeile/Zeichen mit M. X anspringen [**mark**]
/SUCH CR ?SUCH CR	Text SUCH vorwärts/rückwärts suchen
:%s/SUCH/TEXT/gc CR	Text SUCH überall in Datei durch TEXT ersetzen (%=1,$=alle Zeilen, s=substitute, g=global, c=confirm)
n N	Letzte Suche vor/zurück wiederholen [**next**]
u U	Letzte Änderung/alle Änd. einer Zeile zurücknehmen [**undo**]
<Strg-R> VIM!	Letzte Änderung wiederherstellen (2x u im Vi) [**redo**]
.	Letztes Edit-Kommando (Änderung) wiederholen [**Punkt**]

Mit VIM! markierte Befehle sind nur im Editor Vim möglich.

8.2 Weitere wichtige Vi-Befehle

%	Passende Klammer zu { [(< >)] } suchen
~	Groß/Kleinschreibung von aktuellem Zeichen vertauschen
J	Nachfolgende Zeile an aktuelle anhängen [**join**]
<< >>	Aktuelle Zeile einrücken/ausrücken (Option shiftwidth)
xp dwwP ddp	Zwei Zeichen/Worte/Zeilen vertauschen
v V <Strg-V> VIM!	Block zeichen/zeilen/spaltenorientiert markieren [**visual**]
:e# <Strg-6>	Vorherige Datei editieren
"XyMOVE	Von MOVE ausgewählten Text in Puffer X (a-z) kopieren [**yank**]
"XdMOVE	Von MOVE ausgewählten Text in Puffer X (a-z) verschieben [**del**]
"XP	Inhalt von Puffer X (a-z) vor Cursor/aktueller Zeile einfügen [**put**]
"Xp	Inhalt von Puffer X (a-z) nach Cursor/aktueller Zeile einfügen
:map MACRO TEXT	MACRO mit Ersatztext TEXT für Kommando-Modus definieren
:unmap MACRO	MACRO mit Ersatztext TEXT für Kommando-Modus löschen
:map	Alle Makros für Kommando-Modus anzeigen
:map! MACRO TEX	MACRO mit Ersatztext TEXT für Eingabe-Modus definieren
:unmap! MACRO	MACRO mit Ersatztext TEXT für Eingabe-Modus löschen
:map!	Alle Makros für Eingabe-Modus anzeigen
:ab ABBREV TEXT	Abkürzung ABBREV für Eingabe-Modus definieren (*Leerzeichen!*)
:unab ABBREV	Abkürzung ABBREV für Eingabe-Modus löschen
:ab	Alle Abkürzung für Eingabe-Modus anzeigen

8.3 Wichtige allgemeine Vi-Optionen

errorbells	Bei Eingabefehler piepsen
ignorecase	Groß/Kleinschreibung bei der Suche ignorieren
list	Zeilenenden durch $ und Tabulatoren durch ^I anzeigen
number	Zeilen numeriert anzeigen
report=NN	Änderungen ab NN Zeilen in Statuszeile anz. (*1=ab 1 Zeile, 0 geht n.!*)
showmode	Vi-Modus in Statuszeile anzeigen (INSERT, APPEND, ...)
wrapmargin=NN	Automatischer Umbruch NN Zeichen vor Zeilenende (*0=kein Umbruch!*)
wrapscan	Suche über Dateiende/anfang hinweg fortsetzen

Optionen mit :set OPTION setzen und mit :set noOPTION (*zusammengeschrieben!*) löschen. In Konfigurations-Datei .exrc/.vimrc den Doppelpunkt vor set weglassen. Kommentare in .exrc/.vimrc durch " einleiten (*keine Leerzeilen erlaubt!*).

8.4 Wichtige Vi-Optionen für Programmierer

`autoindent`	Automatisch einrücken (mit `<Strg-D/T>` ein/ausrücken)
`shiftwidth=NN`	Breite `NN` für nachträgliches Einrücken setzen (`<< >>`)
`showmatch`	Passende öffnende Klammer bei Eingabe der schließenden anzeigen
`tabstop=NN`	Tabulatorbreite auf `NN` Zeichen setzen

8.5 Wichtige Vim-Optionen

`nocompatible`	Nicht im Vi-Kompatibilitätsmodus arbeiten
`ruler`	Koordinatenanzeige (Zeile + Spalte) in Statuszeile [**Lineal**]
`:syntax on`	Syntaxcoloring an (abhängig von Datei-Extension)
`wrap`	Zu lange Zeilen umgebrochen darstellen
`incsearch`	Inkrementell suche [**incremental**]
`hlsearch`	Zur Suche passende Textteile markieren [**highlight**]
`showcmd`	Unvollständige Kommandos in Statuszeile anzeigen [**command**]
`ff=unix`	Dateiformat (`dos`, `unix`, `mac`) [**fileformat**]

8.6 Nützliche Vi-Makros

`K`	`!}fmt -80 -u^M`	Absatz (bis Leerzeile) auf 80 Zeichen Breite formatieren
`D`	`:%s/ *$//^M`	Alle Leerzeichen am Zeilenende entfernen
`Strg-W`	`/\<`	Suche nach Wort einleiten (danach Wort eintippen)
`Strg-W`	`\>/^M`	Suche nach Wort durchführen (*mit* `:map!` *definieren!*)
`+`	`:n^M`	Zur nächsten Datei springen (ohne Abspeichern)
`#`	`:w^M:n^M`	Zur nächsten Datei springen (aktuelle vorher speichern)
`-`	`:prev^M VIM!`	Zur vorherigen Datei springen (ohne Abspeichern)

Mit `:map NAME MACRO` definieren, `^M` ist einzugeben durch `<Strg-V> <RETURN>` oder im Vim durch `<CR>`.

9 Drucken

9.1 BSD-Variante der Druck-Kommandos

`lpr FILE`	Datei `FILE` drucken [**line print**]
`CMD | lpr`	Ergebnis von Kommando `CMD` drucken
`lpq`	Abgesetzte Druckaufträge anzeigen [**line print queue**]
`lprm JOBNR`	Druckauftrag `JOBNR` löschen [**line print remove**]
`lprm -`	Alle Druckaufträge löschen (*nur* `root`*!*)
`lpc [CMD]`	Drucker verwalten [**line print control**]
`PRINTER`	Umgebungsvariable zur Definition des Standard-Druckers
`lp`	Name des Standard-Druckers (falls Variable `PRINTER` leer)
`-P PRINTER`	Statt Standard-Drucker den Drucker `PRINTER` ansprechen

9.2 System V-Variante der Druck-Kommandos

`lp FILE`	Datei `FILE` drucken [**line print**]
`CMD | lp`	Ergebnis von Kommando `CMD` drucken
`lpstat`	Abgesetzte Druckaufträge anzeigen [**line print status**]
`cancel JOBNR`	Druckauftrag `JOBNR` löschen
`-d PRINTER`	Drucker festlegen [**destination**]
`LPDEST`	Umgebungsvariable zur Definition des Standard-Druckers
`lp`	Name des Standard-Druckers (falls Variable `LPDEST` leer)
`-d PRINTER`	Statt Standard-Drucker den Drucker `LPDEST` ansprechen [**destination**]

9.3 lpc-Kommandos

`status [QUEUE]`	Status von Warteschlange `QUEUE`/allen Warteschlangen ausgeben
`help [CMD]`	Befehlsliste/Hilfe zu `CMD` anzeigen
`quit/exit`	`lpc` verlassen
`enable QUEUE`	Warteschlange `QUEUE` aktivieren
`disable QUEUE`	Warteschlange `QUEUE` deaktivieren
`start QUEUE`	Ausdruck von `QUEUE` starten
`stop QUEUE`	Ausdruck von `QUEUE` stoppen
`up QUEUE`	`enable` + `start`
`down QUEUE`	`disable` + `stop`
`restart QUEUE`	Dämon der Warteschlange `QUEUE` stoppen + neu starten
`clean QUEUE`	Zwischendateien der Warteschlange `QUEUE` entfernen
`topq QUEUE NR`	Job `NR` an die Spitze der Warteschlange `QUEUE` setzen

10 Reguläre Ausdrücke

10.1 Standard-Metazeichen

`.`	Ein beliebiges Zeichen
`X*`	0 oder mehr Wiederholungen des Zeichens `X`
`^`	Zeilenanfang
`$`	Zeilenende
`\X`	Zeichen `X` quotieren
`[abc] [a-z]`	Menge von Zeichen (`[a-z]` = ASCII-Zeichenbereich)
`[^abc] [^a-z]`	Negierte Menge von Zeichen (auch `[^A-Z]`, `[^0-9]`)

10.2 Erweiterte Metazeichen

`?`	0 oder 1 Wiederholung des Teils davor (optional)
`+`	1 oder mehr Wiederholungen des Teils davor
`|`	Entweder-Oder (alternativ)
`(...)`	Klammerung mehrerer Zeichen
`\{M,N\}`	`M` bis `N` Wiederholungen des Teils davor
`\{M,\}`	`M` oder mehr Wiederholungen des Teils davor
`\{M\}`	Genau `M` Wiederholungen des Teils davor
`\n`	Zeilenvorschub [**newline**]
`\(...\)`	Zeichenkette merken (in `\1`..`\9`)
`\<  \>`	Wortanfang, Wortende (`\b` `\B` in `perl`)

10.3 Metazeichen im Ersetzungsmuster

`\N`	`N`-te per `\(...\)` gemerkte Zeichenkette (`N`=1..9)
`&`	Kompletten zu Muster gefundenen Text wieder einsetzen
`~`	Vorheriges Suchmuster verwenden
`\u \l`	Nächstes Zeichen in Groß/Kleinschrift umwandeln [**up**/**lowcase**]
`\U \L`	Alle folgenden Zeichen in Groß/Kleinschrift umwandeln
`\E`	Durch `\U` oder `\L` begonnene Umwandlung beenden [**end**]

10.4 Escape-Sequenzen

\a	[**alert**]	Akustisches Signal
\b	[**backspace**]	Zeichen vorher löschen
\f	[**form feed**]	Seitenvorschub
\n	[**newline**]	Zeilenvorschub
\r	[**carriage return**]	Wagenrücklauf
\t	[**tabulator**]	Tabulator
\v	[**vtab**]	Vertikaler Tabulator
\OOO	[**octal**]	Zeichen mit oktalem Wert OOO−377
\xHH	[**hexadecimal**]	Zeichen mit hexadezimalem Wert 00−FF

10.5 perl-Metazeichen

\A \Z		Zeilenanfang / -ende (bei mehreren Zeilen)
\b \B	[**break/no break**]	Wortgrenze/keine Wortgrenze
\s \S	[**space/no space**]	Leerraum / Kein Leerraum (Leerz., horiz. + vert. Tab. Zeilenvorschub, Wagenrücklauf, Seitenvorschub)
\d \D	[**digit/no digit**]	Ziffer (0−9) / Keine Ziffer
\w \W	[**word/no word**]	Buchstabe (a−zA−Z_0−9) / Kein Buchstabe

11 UNIX-Werkzeuge

11.1 strings, wc, head, tail, tee

strings FILE	In FILE nach allen ASCII-Textstücken suchen und sie ausgeben
strings −a	Alle Textstücke suchen (auch im Codesegment)
wc FILE	Zeilen, Wörter und Zeichen in FILE zählen
wc −l FILE	Nur Zeilen in FILE zählen [**lines**]
wc −w FILE	Nur Wörter in FILE zählen [**words**]
wc −c FILE	Nur Zeichen in FILE zählen [**characters**]
head FILE	Die 10 ersten Zeilen von FILE ausgeben
head −NN FILE	Die NN ersten Zeilen von FILE ausgeben
tail FILE	Die 10 letzten Zeilen von FILE ausgeben
tail −NN FILE	Die NN letzten Zeilen von FILE ausgeben
tail +NN FILE	FILE ab der Zeile NN ausgeben
tail −f FILE	Neue Zeilen am Ende von FILE permanent ausgeben [**follow**]
tee FILE	stdin auf FILE schreiben und an stdout weitergeben
tee −a FILE	stdin an FILE anhängen und an stdout weitergeben [**append**]

11.2 cmp, diff

`cmp FILE1 FILE2`	`FILE1` und `FILE2` binär vergleichen (Exit-Status)
`diff FILE1 FILE2`	`FILE1` und `FILE2` zeilenw. vergl. und Unterschiede ausgeben
`diff -i ...`	Groß/Kleinschreibung ignorieren [**ignore case**]
`diff -r ...`	Dateien in Unterverz. ebenfalls vergleichen [**recursive**]
`diff -s ...`	Gleiche Dateien ebenfalls anzeigen [**same**]
`diff -q ...`	Nur Dateinamen, nicht die Unterschiede anzeigen [**quiet**]

11.3 cut, paste, join, tr, split

`cut FILE`	Spalten oder Felder aus `FILE` schneiden (vertikal)
`cut -dC -fN-M`	Felder durch Trennzeichen `C` getrennt (Std: `<TAB>`)
	Felder `N-M` ausgeben [**delimiter,fields**]
`cut -cL,M,N`	Spalten `L`, `M` und `N` ausgeben [**columns**]
`cut -cNN-MM`	Spalten `NN-MM` ausgeben [**columns**]
`paste FILE1 FILE2`	Zeilen aus `FILE1` und `FILE2` nebeneinanderstellen
`paste -d":@;" ...`	Spaltentrenner `:@;` festlegen (`\0`=keiner) [**delimiter**]
`paste -s FILE`	Alle Zeile von `FILE` in einer Zeile zusammenfassen [**squeeze**]
`paste -s - - -`	Jeweils 3 Zeilen aus `stdin` in einer Zeile zusammenfassen
`join FILE1 FILE2`	Zeilen in `FILE1` und `FILE2` über Schlüsselspalte verknüpfen
	(*beide Dateien müssen alphabetisch sortiert sein!*)
`join -tC ...`	Feldtrenner `C` festlegen (Std: `<SPACE>`) [**terminator**]
`join -1 M -2 N`	Feld `M` der 1. Datei mit Feld `N` der 2. verknüpfen
`tr SET1 [SET2]`	Zeichen aus `SET1` in korresp. aus `SET2` umsetzen [**translate**]
`tr -c SET1`	Zeichen nicht(!) in `SET1` ersetzen [**complement**]
`tr -d SET1`	Zeichen in `SET1` löschen [**delete**]
`tr -s SET1 SET2`	Gleiche Zeichen hintereinander zusammenziehen [**squeeze**]
`split FILE [PREFIX]`	`FILE` in Stücke `xaa`, `xab`, `...` zerlegen (Std-Präfix: `x`)
`split -l NN ...`	Stücke der Länge `NN` Zeilen erstellen (Std: 1000) [**lines**]
`split -b NN ...`	Stücke der Länge `NN` Bytes erstellen [**bytes**]

`tr` liest nur von Standard-Eingabe und schreibt auf Standard-Ausgabe, eine Datei kann beim Aufruf nicht angegeben werden.

11.4 sort, uniq

`sort FILE`	Zeilen von `FILE` alphabetisch aufsteigend sortieren
`sort -n FILE`	Numerisch sortieren [**numeric**]
`sort -d FILE`	Lexikografisch sortieren (Std) [**dictionary**]
`sort -f FILE`	Groß/Kleinschreibung ignorieren [**fold**]
`sort -r FILE`	Sortierreihenfolge absteigend [**reverse**]
`sort +N -M FILE`	Von Feld `N+1` bis Feld `M` sortieren (`N` Felder überspringen)
`sort -k N,M`	Von Feld `N` bis Feld `M` sortieren (*neue Schreibweise!*) [**key**]
`sort +0.N -0.M`	Von Spalte `N+1` bis Spalte `M` sortieren
`sort -tC FILE`	Feldtrenner ist Zeichen `C` (Std: Tabulator) [**terminator**]
`sort -oFILE FILE`	Datei `FILE` direkt sortieren (ohne Kopie) [**output**]
`uniq FILE`	Doppelte Zeilen aus `FILE` entfernen (*muss sortiert sein!*) [**unique**]
`uniq -c FILE`	Vor Zeilen ihre Häufigkeit ausgeben [**count**]
`uniq -d FILE`	Nur mehrfach vorkommende Zeilen ausgeben [**duplicates**]
`uniq -u FILE`	Nur einfach vorkommende Zeilen ausgeben [**uniques**]

11.5 grep

`grep "TEXT" FILE`	Zeilen mit Muster `TEXT` in `FILE` suchen (Standard-Regexp)
`egrep "TEXT" FILE`	Zeilen mit Muster `TEXT` in `FILE` suchen (Erweiterte-Regexp)
`fgrep "TEXT" FILE`	Zeilen mit Text `TEXT` in `FILE` suchen (Feste Zeichenkette)

11.5.1 Die wichtigsten Optionen von grep

`-c`	Nur die Anzahl der passenden Zeilen ausgeben [**count**]
`-h`	Dateinamen nicht ausgeben (bei mehr als einer Datei) [**hide**/**head**]
`-i`	Groß/Kleinschreibung ignorieren [**ignore case**]
`-l`	Nur Dateinamen ausgeben (Muster paßt auf mind. eine Zeile) [**list**]
`-n`	Zeilennummer den passenden Zeilen voranstellen [**number**]
`-v`	Nach nicht(!) passenden Zeilen suchen [**vice versa**]

11.6 ed, ex, sed, awk, perl

`cat SCRIPT | ed FILE`	Ed-Kommandos `SCRIPT` auf `FILE` anwenden [**editor**]
`cat SCRIPT | ex FILE`	Ex-Kommandos `SCRIPT` auf `FILE` anw. [**extended editor**]
`sed [-e] 'CMD' FILE`	Sed-Kommandos `CMD` auf `FILE` anwenden [**stream editor**]
`sed -f SCRIPT FILE`	Sed-Kommandos aus `SCRIPT` auf `FILE` anwenden [**file**]
`-n`	Zeilen nicht automatisch ausgeben [**noprint**]
`awk 'CMD' FILE`	Awk-Befehle `CMD` auf `FILE` anw. [**aho**/**weinberger**/**kernighan**]
`awk -f SCRIPT FILE`	Awk-Kommandos aus `SCRIPT` auf `FILE` anwenden [**file**]
`-F "TEXT"`	Feldtrenner festlegen
`-v VAR="VALUE"`	Variable `VAR` mit Wert `VALUE` belegen
`perl -e 'CMD;' FILE`	Perl-Befehle `CMD` auf `FILE` anwenden [**execute**]
`perl -f SCRIPT FILE`	Perl-Kommandos aus `SCRIPT` auf `FILE` anwenden [**file**]
`-a`	`$_` automatisch in `@F` zerlegen [**autosplit**]
`-n`	`while (<>) {...}`-Schleife ohne Ausgabe [**noprint**]
`-p`	`while (<>) {...}`-Schleife mit Ausgabe [**print**]
`-W`	Alle Warnungen einschalten

11.6.1 Sed-Kommandos

`a\`	Nachfolgende Zeilen nach aktueller Zeile einfügen [**append**], alle Zeilen außer der letzten sind mit `\` abzuschließen
`c\`	Aktuelle Zeile durch nachfolgende Zeilen ersetzen [**change**], alle Zeilen außer der letzten sind mit `\` abzuschließen
`i\`	Nachfolgende Zeilen vor aktueller Zeile einfügen [**insert**], alle Zeilen außer der letzten sind mit `\` abzuschließen
`d`	Aktuelle Zeile löschen [**delete**]
`s/REGEX/SUBST/`	In aktueller Zeile `REGEX` durch `SUBST` ersetzen [**substitute**]
`y/abc/ABC/`	In aktueller Zeile Zeichen `a` durch `A`, `b` durch `B` und `c` durch `C` ersetzen (analog dem Kommando `tr`) [**yank**]
`n`	Aktuelle Zeile ausgeben und die nächste Zeile einlesen [**next**]
`l`	Aktuelle Zeile ausgeben (Steuer-Zeichen im ASCII-Code) [**list**]
`p`	Aktuelle Zeile ausgeben [**print**]
`q`	Aktuelle Zeile ausgeben und Sed-Skript abbrechen [**quit**]
`=`	Nummer der aktuellen Zeile ausgeben
`r FILE`	Inhalt der Datei `FILE` nach aktueller Zeile einfügen [**read**]
`w FILE`	Aktuelle Zeile auf Datei `FILE` ausgeben [**write**]
`b[LABEL]`	Zu Marke `LABEL` (oder zum Skriptende) springen [**branch**]
`t[LABEL]`	Zu Marke `LABEL` (oder zum Skriptende) springen, wenn seit dem letzten Einlesen oder seit dem letzten `t`-Kommando eine Ersetzung erfolgte [**test**]
`:LABEL`	Marke für `b`- oder `t`-Kommando (*max. 7 Zeichen!*) [**number**]
`{`	Kommandos bis zu `}` als Gruppe behandeln [**group**]
`g`	Aktuelle Zeile durch Zwischenpuffer ersetzen [**get**]
`h`	Aktuelle Zeile in Zwischenpuffer kopieren [**hold**]
`x`	Aktuelle Zeile und Zwischenpuffer vertauschen [**exchange**]
`D`	Aktuelle Zeile bis zum ersten Newline löschen [**delete**]
`G`	Zwischenpuffer an aktuelle Zeile anhängen [**get**]
`H`	Aktuelle Zeile an Zwischenpuffer anhängen [**hold**]
`N`	Nächste Zeile an aktuelle Zeile anhängen [**next**]
`P`	Akt. Zeile bis zum ersten Newline ausgeben (+ löschen) [**print**]

11.7 find, locate

`find PATH COND ACTION`	`ACTION` auf Dateien ab `PATH` mit Eigenschaft `COND` anwenden
`locate "MUSTER"`	Alle Dateien mit zu `MUSTER` passendem Namen suchen (LINUX)
`updatedb`	`locate`-Datenbank aktualisieren (LINUX)

11.7.1 find-Bedingungen

`-group NAME/ID`	Besitzergruppe
`-mtime [+-]DAYS`	Änderung vor mehr/weniger/genau `DAYS` 24h-Tagen [**modification**]
`-name "PATTERN"`	Dateinamen-Muster (`* ? [...] ...`, vor der Shell schützen)
`-perm [-]RIGHTS`	(Oktale) Rechte (`NNN`=exakt, `-NNN`=mindestens) [**permission**]
`-size [+-]NUM[c]`	Dateigröße mehr/weniger/genau `NUM` Blöcke (bzw. Zeichen)
`-type TYPE`	Dateityp (`f`=file, `d`=directory, `c`=character device, `b`=block device, `l`=symlink, `p`=named pipe, `s`=socket)
`-user NAME/ID`	Besitzer
`-atime [+-]DAYS`	Lesender Zugriff vor mehr/weniger/genau 24h-Tagen [**access**]
`-ctime [+-]DAYS`	Status-Änderung (Inode) vor mehr/... 24h-Tagen [**change**]
`-iname "PATTERN"`	Analog `-name`, aber Groß/Kleinschreibung egal [**ignorecase**]
`-inum [+-]NUM`	Inode-Nummer mehr/weniger/gleich `NUM` [**inode number**]
`-links [+-]NUM`	Link-Anzahl mehr/weniger/gleich `NUM`
`-nogroup`	Keiner Gruppe aus `/etc/groups` zugeordnet (ID ohne Name)
`-nouser`	Keinem Benutzer aus `/etc/passwd` zugeordnet (ID ohne Name)
`\(...\)`	Klammerung (quotieren wegen Shell)
`\!`	Negation (quotieren wegen Shell)
`-a`	UND-Verknüpfung (Standard falls keine Verknüpfung angegeben)
`-o`	ODER-Verknüpfung

Die Tagesgrenze bei `-a/c/mtime` liegt auf dem Startzeitpunkt des `find`-Kommandos, der Tag `0` entspricht 0 bis 24 Stunden vorher, 1 entspricht 24-48h, ...

11.7.2 find-Aktionen

`-print`	Ausgabe der gefundenen Dateinamen (*Standard!*)
`-ls`	Ausführliche Ausgabe der gefundenen Dateinamen (`ls -lsdi`)
`-exec CMD {} \;`	Kommando `CMD` auf allen gefundenen Dateien ausführen (`{}` = gef. Dateiname, `\;` = Kommandoende, quotiert wegen Shell)
`-ok CMD {} \;`	Analog `-exec`, verlangt aber vorher Bestätigung mit `y`(es)

11.7.3 find-Beispiele

`find / -name "*.c" -print`	C-Dateien ab Root-Verzeichnis
`find .. -mtime 1 -print`	Gestern modifizierte Dateien ab `..` (Eltern-Verz.)
`find / -mtime -7 -print`	In der letzten Woche modifizierte Dateien
`find / -mtime +100 -print`	Vor mehr als 100 Tagen modifizierte Dateien
`find / -user tsbirn -print`	Dateien des Anwenders `tsbirn`
`find /usr -type d -name "*man*"`	Verzeichnisse in `/usr` mit `man` im Namen [**directory**]
`find / -size 0 -ok rm {} \;`	Leere Dateien löschen (mit Abfrage)
`find / -type f \ -exec grep -l "made" {} \;`	Namen der Dateien die Text `made` enthalten [**file**]
`find / -user root -perm -002`	Gehören `root` und sind für andere schreibbar

11.8 tar, cpio

`tar`	Archivdatei verwalten [**tape archiver**]
`tar cf ARCH DIR/FILE`	`ARCH` erzeugen [**create/file**]
`tar xvf ARCH`	`ARCH` in aktuellem Verz. auspacken [**extract/verbose**]
`tar tf ARCH`	`ARCH` mit Anzeige testen [**test**]
`tar tzf ARCH`	Komprimiertes Archiv `ARCH` testen [**test,zipped**]
`cpio`	Archivdatei verwalten [**copy input output**]
`... | cpio -o ARCH`	`ARCH` (Dateiliste von `stdin`) erz. auf `stdout` [**output**]
`cpio -id < ARCH`	`ARCH` ausp. (von `stdin`) mit Verz. erz. [**input/directory**]
`cpio -tv < ARCH`	`ARCH` mit Anzeige testen (von `stdin`) [**test/verbose**]

11.9 compress, gzip

`compress FILE`	`FILE` komprimieren und ersetzen (Endung `.Z`, veraltet)
`uncompress FILE`	`FILE` dekomprimieren und ersetzen
`uncompress -c`	Ausgabe auf `stdout`, nicht ersetzen [**console**]
`gzip FILE`	`FILE` komprimieren und ersetzen (Endung `.gz`, Standard)
`gunzip FILE`	`FILE` dekomprimieren und ersetzen
`gunzip -c`	Ausgabe auf `stdout`, nicht ersetzen [**console**]
`bzip2 FILE`	`FILE` komprimieren und ersetzen (Endung `.bz2`, noch besser)
`bunzip2 FILE`	`FILE` dekomprimieren und ersetzen
`bunzip2 -c`	Ausgabe auf `stdout`, nicht ersetzen [**console**]
`zcat FILE...`	Komprimierte Dateien `FILE...` aneinanderhängen
`zmore FILE`	Komprimierte Datei `FILE` anzeigen
`zcmp FILE1 FILE2`	Komprimierte Dateien `FILE1` und `FILE2` vergleichen (binär)
`zdiff FILE1 FILE2`	Komprimierte Dateien `FILE1` und `FILE2` vergleichen (zeilenw.)
`zgrep "TEXT" FILE`	Zeilen mit Muster `TEXT` in komprimiertem `FILE` suchen

12 System-Administration

12.1 fdisk, fdformat, mkfs, fsck, dd

`fdisk DEV`	Gerät `DEV` menügesteuert partitionieren [**format disk**]
`fdisk -l DEV`	Alle Partitionen von Gerät `DEV` auflisten [**list**]
`fdformat DEV`	Floppy-Disk `DEV` formatieren [**floppy disk format**]
	(in Spuren + Sektoren einteilen)
`fdformat /dev/fd0u1440`	1,44 MByte-Diskette formatieren (80 Sp, 18 Sekt)
`fdformat /dev/fd0u1722`	1,72 MByte-Diskette formatieren (82 Sp, 21 Sekt)
`mkfs -t TYPE DEV`	Dateisystem `TYPE` auf Gerät `DEV` anlegen [**make filesystem**]
`mkfs -t msdos /dev/fd0`	MSDOS-Dateisystem auf Diskette anlegen
`mkfs -t ext2 /dev/hda1`	Ext2-Dateisystem auf 1. Partition der 1. Platte anlegen
`fsck DEV`	Dateisystem überprüfen [**file system check**]
`fsck /dev/hdb4`	4. Partition der 2. Festplatte überprüfen
`dd`	Sektoren lesen/schreiben (Platte/Floppy) [**disk duplicate**]
`if=FILE`	Eingabedatei [**input file**]
`of=FILE`	Ausgabedatei [**output file**]
`bs=NN`	Blockgröße `NN` [**block size**]
`count=NN`	Anzahl Blöcke `NN`
`skip=NN`	`NN` Blöcke der Eingabe überspringen
`seek=NN`	`NN` Blöcke der Ausgabe unterdrücken

12.2 mount, umount

`mount DEV PATH`	Gerät `DEV` in Pfad `PATH` des Dateibaumes einhängen [**montieren**]
`mount PATH`	Gerät gemäß `/etc/fstab` in Pfad `PATH` einhängen
`mount DEV`	Gerät `DEV` in Pfad gemäß `/etc/fstab` einhängen
`mount`	Alle montierten Geräte auflisten
`mount /floppy`	Diskettenlaufwerk automatisch einhängen (`/etc/fstab`*!*)
`mount /cdrom`	CDROM automatisch montieren (`/etc/fstab`*!*)
`mount -a`	Alle Geräte in `/etc/fstab` einhängen [**all**]
`mount -t TYPE`	Alle Geräte vom Typ `TYPE` in `/etc/fstab` einhängen [**type**]
`umount PATH`	Gerät mit Mount-Point `PATH` aushängen [**unmount**]
`umount DEV`	Gerät `DEV` aushängen
`umount -a`	Alle Geräte in `/etc/fstab` aushängen (bis auf `root`-Verz) [**all**]
`mount -t TYPE`	Alle Geräte vom Typ `TYPE` in `/etc/fstab` aushängen [**type**]

12.2.1 Mount-Optionen

`defaults`	Steht für: `rw`, `suid`, `dev`, `exec`, `auto`, `nouser`, `async`
`ro`	`read only` einhängen (write-Recht ignorieren)
`rw *`	`read write` einhängen (Zugriffsrecht `write` zählt)
`(no)auto *`	Während dem Bootvorgang (nicht) automatisch einhängen
`(no)user *`	Normale User dürfen (nicht) einhängen
`(no)atime`	Inode-Access-Zeit (nicht) updaten bei Lesezugriffen (*langsam!*)
`(no)dev *`	Block- und Zeichen-Gerätedateien (nicht) ignorieren
`(no)exec *`	Programme können (nicht) ausgeführt werden
`(no)suid *`	Set-UID-Programme können (nicht) ausgeführt werden
`(a)sync *`	Ein/Ausgabe synchron (sofort) bzw. asynchron (gepuffert)
`remount`	Eingehängte Partition erneut einhängen (*zum Optionen ändern!*)

Die mit * gekennzeichneten Optionen sind in `defaults` enthalten.

12.2.2 Samba-Mount-Optionen

`username=TEXT`	Benutzer-Name für Anmeldung
`password=TEXT`	Benutzer-Paßwort für Anmeldung
`uid=NUM`	User-ID festlegen
`gid=NUM`	Gruppen-ID festlegen
`fmask=NUM`	Auszumaskierende Rechte für Dateien
`dmask=NUM`	Auszumaskierende Rechte für Verzeichnisse
`umask=NUM`	Auszumaskierende Rechte festlegen

12.3 at, crontab

12.3.1 at-Befehle

`at HH:MM`	Auftrag um `HH:MM` starten (Eingabe auf `stdin`)
`atq`	Alle Aufträge anzeigen [**at queue**]
`atrm JOBNR`	Auftrag mit `JOBNR` löschen [**at remove**]

12.3.2 crontab-Befehle

`crontab -e`	Vorhandene `crontab`-Tabelle editieren [**edit**]
`crontab -l`	Vorhandene `crontab`-Tabelle anzeigen [**list**]
`crontab -r`	Vorhandene `crontab`-Tabelle löschen [**remove**]
`crontab FILE`	`crontab`-Tabelle aus `FILE` erstellen

12.3.2.1 crontab-Felder

1	Minute	0-59	Kleinste Einheit
2	Stunde	0-59	
3	Tag	1-31	
4	Monat	1-12	Oder `jan`, `feb`, ..., `dec`
5	Wochentag	0-6	`0`=Sonntag, ..., `6`=Samstag oder `sun`, ... `sat`
6-	Kommando		Ausgabe auf Datei umlenken (sonst erfolgt `mail`)

In den zentralen `crontab`-Dateien kommt als 6. Feld der Benutzername dazu, unter dem das Kommando ablaufen soll. Erst ab dem 7. Feld folgt dann das Kommando.

12.3.2.2 Erlaubte crontab-Feldwerte

`*`	Alle möglichen Werte
`NN`	Wert `NN`
`N1,N2,...`	Werteliste `N1`, `N2`, ...
`NN-MM`	Wertebereich `NN-MM`
`NN-MM/SS`	Werte `NN`, `NN+SS`, `NN+2*SS`, ... (Schrittweite `SS`)

12.4 mail

`mail`	Empfangene Mails lesen
`mail USER... < brief`	Mail `brief` an die Benutzer `USER...` verschicken
`mail -s "SUBJECT" USER`	Mail mit Titel `SUBJECT` an `USER` verschicken (von `stdin`)

12.4.1 Die wichtigsten Mail-Befehle

`h`	Kopfzeilen der (letzten 10) Mails anzeigen [**head**]
`NN`	Mail Nummer `NN` anzeigen
`d`	Aktuelle Mail löschen [**delete**]
`d NN-MM`	Mails `NN-MM` löschen [**delete**]
`q`	Mailprogramm verlassen [**quit**]
`x`	Mailprogramm ohne jede Änderung verlassen [**quit**]
`<RETURN>`	Nächste Mail anzeigen

13 Shell-Programmierung

13.1 Shell-Aufrufarten

	PATH durch-sucht	x-Recht nötig	Sub-Shell	Neuer Prozeß	Aliase+ Funkt. vererbt	Rückkehr zum Aufrufer	Option angebbar
`cmd.sh`	ja	ja	ja	ja	nein	ja	nein
`sh [OPT] cmd.sh`	nein	nein	ja	ja	nein	ja	ja
`./cmd.sh`	nein	ja	ja	ja	nein	ja	nein
`. cmd.sh`	ja	nein	nein	nein	ja	ja	nein
`. ./cmd.sh`	nein	nein	nein	nein	ja	ja	nein
`exec cmd.sh`	ja	ja	nein	nein	nein	nein	nein
`exec ./cmd.sh`	nein	ja	nein	nein	nein	nein	nein

Das Kommando `.` heißt `source`-Kommando, es liest die angegebene Datei in der aktuellen Shell ein (include-Anweisung).

13.2 Shell-Optionen

`-n`	Kommandos lesen aber nicht ausführen (Syntaxcheck) [**noexec**]
`-u`	Abbruch bei Zugriff auf nicht initialisierte Variable [**unset**]
`-v`	Gelesene Kommandos ausgeben [**verbose**]
`-x`	Wirklich ausgeführte Kommandos vorher ausgeben [**execute**]
`-o OPT`	Option `OPT` setzen
`+o OPT`	Option `OPT` zurücksetzen
`-o ignoreeof`	`<Strg-D>` ignorieren (d.h. `exit` notwendig)
`-o noclobber`	Umlenkung auf bereits existierende Dateien verhindern (`>|`)

13.3 Spezielle Shell-Variablen (Parameter)

`$1`	1. Kommandozeilen-Argument
`$2`	2. Kommandozeilen-Argument
`...`	...
`$9`	9. Kommandozeilen-Argument (Zugriff auf `${10}...` mit `shift`)
`$#`	Anzahl Kommandozeilen-Argumente
`$*`	Alle Kommandozeilen-Argumente
`$@`	Alle Kommandozeilen-Argumente (einzeln quotiert bei `"$@"`)
`$0`	Skript-Name (Aufrufform oder kompletter Pfad)
`$?`	Exit-Code des letzten ausgeführten Kommandos
`$$`	Prozess-ID des Shell-Skripts
`$!`	Prozess-ID des letzten im Hintergrund gestarteten Kommandos (`CMD &`)
`$-`	Shell-Optionen

13.3.1 Bedingte Auswertung von Shell-Variablen

`${VAR-TEXT}`	`VAR` zurück falls `VAR` definiert; sonst `TEXT` zurück
`${VAR=TEXT}`	Analog + Zuweisung von `TEXT` an `VAR`
`${VAR+TEXT}`	`TEXT` zurück falls `VAR` definiert; sonst nichts zurück
`${VAR?}`	Ausgabe `VAR: parameter not set` + Abbruch falls `VAR` nicht definiert
`${VAR?TEXT}`	Analog Ausgabe von `VAR: TEXT`; sonst `VAR` zurück

Ein Doppelpunkt `:` nach `VAR` verlangt, dass die Variable `VAR` nicht leer sein darf (`not null`). Ohne Doppelpunkt darf sie leer sein, muss aber definiert sein.

13.4 Bedingungen des test-Kommandos

Das Zeichen `K` heißt, der entsprechende Test ist nur in der `ksh` und in der `bash` vorhanden.
Das Zeichen `B` heißt, der entsprechende Test ist nur in der `bash` vorhanden.

13.4.1 Textvergleiche

`"TEXT"`	`TEXT` ist nicht leer (*nicht mit* `! -a -o` *kombinierbar!*)
`-n "TEXT"`	`TEXT` ist nicht leer (*mit* `! -a -o` *kombinierbar!*) [**nonzero**]
`-z "TEXT"`	`TEXT` ist leer [**zero**]
`"TEXT1" = "TEXT2"`	`TEXT1` und `TEXT2` sind gleich
`"TEXT1" != "TEXT2"`	`TEXT1` und `TEXT2` sind verschieden
`"TEXT1" \< "TEXT2" K`	`TEXT1` alphabetisch kleiner als `TEXT2` (quot. wg. Shell)
`"TEXT1" \> "TEXT2" K`	`TEXT1` alphabetisch größer als `TEXT2` (quot. wg. Shell)

13.4.2 Dateiattribut-Vergleiche

`-e FILE`	K	Datei `FILE` existiert [**exists**]
`-d FILE`		Datei `FILE` ist Verzeichnis [**directory**]
`-f FILE`		Datei `FILE` ist normale Datei [**file**]
`-r FILE`		Datei `FILE` ist lesbar [**readable**]
`-s FILE`		Datei `FILE` nicht leer [**size**]
`-w FILE`		Datei `FILE` schreibbar [**writable**]
`-x FILE`		Datei `FILE` ausführbar [**executable**]
`-b FILE`		Datei `FILE` ist blockorientiert [**block device**]
`-c FILE`		Datei `FILE` ist zeichenorientiert [**character device**]
`-L FILE`	K	Datei `FILE` ist ein symbolischer Link [**link**]
`-p FILE`		Datei `FILE` ist eine Named Pipe [**pipe**]
`-S FILE`	K	Datei `FILE` ist ein Socket [**socket**]
`-t DESC`		Deskriptor `DESC` ist ein Terminal (Std: 0=stdin) [**terminal**]
`-g FILE`		Datei `FILE` hat group id Recht gesetzt [**group**]
`-k FILE`		Datei `FILE` hat sticky Recht gesetzt [**sticky**]
`-u FILE`		Datei `FILE` hat user id Recht gesetzt [**user**]
`-G FILE`	B	Datei `FILE` gehört effektiver group id [**Group**]
`-N FILE`	B	Datei `FILE` seit letztem Lesen verändert [**New**]
`-O FILE`	B	Datei `FILE` gehört effektiver user id [**Owner**]
`-o OPTION`	B	Option `OPTION` eingeschalten

13.4.3 Datei-Vergleiche

`FILE1 -nt FILE2`	K	Datei `FILE1` neuer als `FILE2` [**newer than**]
`FILE1 -ot FILE2`	K	Datei `FILE1` älter als `FILE2` [**older than**]
`FILE1 -ef FILE2`	K	Dateien haben ident. device/inode-Nummer [**equal file**]

13.4.4 Numerische Vergleiche

`NUM1 -eq NUM2`	Zahl `NUM1` gleich `NUM2` [**equal**]
`NUM1 -ne NUM2`	Zahl `NUM1` nicht gleich `NUM2` [**not equal**]
`NUM1 -le NUM2`	Zahl `NUM1` kleiner gleich `NUM2` [**less equal**]
`NUM1 -lt NUM2`	Zahl `NUM1` kleiner `NUM2` [**less than**]
`NUM1 -ge NUM2`	Zahl `NUM1` größer gleich `NUM2` [**greater equal**]
`NUM1 -gt NUM2`	Zahl `NUM1` größer `NUM2` [**greater than**]

13.4.5 Logische Verknüpfungen

`\( ... \)`	Klammerung (*quotiert wg. Shell!*)
`\! EXPR`	Negation von `EXPR` (*quotiert wg. Shell!*) [**not**]
`EXPR1 -a EXPR2`	`EXPR1` und `EXPR2` [**and**]
`EXPR1 -o EXPR2`	`EXPR1` oder `EXPR2` [**or**]

Die Reihenfolge von oben nach unten entspricht dem Vorrang.

13.4.6 Shell-Optionen testen

`-o OPT`	K	Option `OPT` gesetzt [**option**]

13.5 Kontrollstrukturen

13.5.1 Benutzereingabe einlesen

`echo -n "Bitte eingeben: "`
`read EINGABE`
`echo "Eingegeben wurde: $EINGABE"`
	`-n`=kein Zeilenvorschub
	Benutzereingabe (bis `RETURN`)
	(Testausgabe)

13.5.2 Verzweigung

`if [ $ZAHL -gt 2 ]`
`then`
`  echo "$ZAHL ist größer als 2"`
`elif [ $ZAHL -lt 2 ]`
`then`
`  echo "$ZAHL ist kleiner als 2"`
`else`
`  echo "$ZAHL ist gleich 2"`
`fi`
	Leerzeichen um `[`, `]` und `-gt`
	`then` muss auf neuer Zeile stehen
	(Testausgabe)
	`else if/elsif` geht nicht
	`then` muss auf neuer Zeile stehen
	(Testausgabe)
	else-Zweig (alle sonstigen Fälle)
	(Testausgabe)
	`fi` nicht vergessen (*neue Zeile!*)

13.5.3 case-Mehrfachverzweigung (Text-Vergleich mit Shell-Metazeichen)

`TEXT="Babel"`
`case "$TEXT" in`
`  yes|ja) echo "ja gefunden" ;;`
`  n*)  echo "nein gefunden" ;;`
`  *)  echo "Fehler" ;;`
`esac`
	Zum Testen Zeilen vertauschen
	`in` und `...` nicht vergessen!
	`;;` nicht vergessen!
	`;;` nicht vergessen!
	else-Zweig (alle sonstige Fälle)
	`esac` nicht vergessen (*neue Zeile!*)

13.5.4 for-Schleife (Liste bzw. Shell-Aufrufargumente abarbeiten)

`for ELEM in abc defghi jklmnopqr`
`do`
`  echo "for: ELEM=$ELEM"`
`done`
	`ELEM` enthält Werte aus Liste
	`do` muss auf neuer Zeile stehen
	(Testausgabe)
	`done` nicht vergessen (*neue Zeile!*)

`for ARG`
`do`
`  echo "for: ARG=$ARG"`
`done`
	`ELEM` enthält Shell-Argumente
	`do` muss auf neuer Zeile stehen
	(Testausgabe)
	`done` nicht vergessen (*neue Zeile!*)

13.5.5 while und until-Schleife (Bedingung prüfen)

`ZAHL=3`
`while [ $ZAHL -gt 0 ]`
`do`
`  echo "while: ZAHL=$ZAHL"`
`  ZAHL=\`expr $ZAHL - 1\``
`done`
	`ZAHL` initialisieren
	Durchlauf solange `ZAHL` größer 0
	`do` muss auf neuer Zeile stehen
	(Testausgabe)
	Kommando-Subst (Leerz. um `-` nötig)
	`done` nicht vergessen (*neue Zeile!*)

`until [ $ZAHL -eq 3 ]`
`do`
`  echo "until: ZAHL=$ZAHL"`
`  ZAHL=\`expr $ZAHL + 1\``
`done`
	Durchlauf bis `ZAHL` Wert 3 hat
	`do` muss auf neuer Zeile stehen
	(Testausgabe)
	Kommando-Subst (Leerz. um `+` nötig)
	`done` nicht vergessen (*neue Zeile!*)

13.5.6 Funktions-Definition und -Aufruf

`func1()`
`{`
`  echo "Funktion func1() aufgerufen"`
`  echo "1.Parameter $1"`
`  echo "2.Parameter $2"`
`}`
	Definition (*vor Aufruf!*)
	Extra Zeile!
	Einrücken sinnvoll
	Zugriff auf 1.Parameter
	Zugriff auf 2.Parameter
	Extra Zeile!

`func1` `func1 100 Test`	Aufruf (*nach Definition!*) Aufruf mit Parametern

14 Awk

14.1 AWK-Optionen

Option	Bedeutung
`-F` *sep*	Feldtrenner `FS` auf *sep* festlegen (**Default:** "␣")
`-f` *pgm*	Programmdatei *pgm* einlesen (*mehrfach erlaubt*)
`-v` *var=text*	Variabler *var* den Wert *text* vor Programmstart zuweisen
`-mf` *n*	Maximale Anzahl Felder *n* festlegen (*im Gawk überflüssig*)
`-mr` *n*	Maximale Recordlänge *n* festlegen (*im Gawk überflüssig*)
`--`	Beendet Optionen-Liste (erlaubt Argumente mit führendem `-`)

14.2 GNU-AWK-spezifische Optionen

Gawk-Option	Bedeutung
`--field-separator` *sep*	Analog `-F`
`--assign` *var=val*	Analog `-v`
`--file=`*file*	Analog `-f` *mehrfach erlaubt*)
`--compat`	*Gawk*-Erweiterungen abschalten
`--traditional`	*Gawk*-Erweiterungen abschalten
`--posix`	Nur POSIX-Umfang zulassen (siehe unten).
`--copyleft`	GPL (GNU General Public Licence) ausgeben
`--copyright`	GPL (GNU General Public Licence) ausgeben
`--help`	*Gawk*-Usage-Meldung ausgeben
`--usage`	*Gawk*-Usage-Meldung ausgeben
`--version`	*Gawk*-Versionsnummer ausgeben
`--source` *program*	Programmtext *program* auf der Kommandozeile angeben (*mehrfach erlaubt, in Hochkomma setzen*)
`--exec` *pgm*	Analog `-f`, aber letzte verarbeitete Option (für CGI-Skripte)
`--lint`[=*wert*]	Warnung bei gefährlichen/nicht portablen Konstrukten (*wert*=`fatal` führt zu fatalem Fehler statt Warnung) (*wert*=`invalid`: nur Warnung bei wirklich ungültigen Dingen)
`--lint-old`	Warnung bei Konstrukten abweichend vom Ur-*Awk*
`--dump-variables`[=*file*]	Sortierte Liste globaler Var. mit Typ + Endwert ausgeben
`--gen-po`	Aus lokalisierbaren Strings `.po`-Datei erzeugen (`gentext`)
`--profile`[=*file*]	Profildaten in Datei `awkprof.out` bzw. *file* ablegen (`pgawk`)
`--disable-directories-fatal`	Verz. als Kmdozeilen-Argument ignorieren (statt abzubrechen)
`--enable-switch`	`switch-case-default`-Anweisung aus *C* erlauben
`--non-decimal-data`	Oktal/Hexadezimalwerte in Eingabedaten erkennen (Vorsicht!)
`--re-interval`	`{n,m}`-Wiederholung in Regulären Ausdrücken erlauben
`--use-lc-numeric`	Dezimaltrennzeichen gemäß Locale-Einstellung beim Einlesen

14.3 Konstanten

14.3.1 Zahlen

```
123   -1   -3.141592   +.0125   1e12   -987.654E-321
0123   0777     07654321             # Oktale Werte
0XFF   0xAFFE   0X1234567890ABCDEF   # Hexadezimale Werte
0xff   0Xaffe   0x1234567890abcdef   # Hexadezimale Werte
```

14.3.2 Zeichenketten

```
"abc"                    # Ohne Zeilenvorschub
"Dies ist ein Text.\n"   # Mit Zeilenvorschub
""                       # Leere Zeichenkette
```

14.3.3 Escape-Sequenzen

Gawk	Escape	Bedeutung
*	\a	Akustisches Signal (**alert**)
	\b	Backspace (Zeichen zurück)
	\f	Seitenvorschub (**formfeed**)
	\n	Zeilenvorschub (**newline**)
	\r	Wagenrücklauf (**carriage return**)
	\t	Tabulator
*	\v	Vertikaler Tabulator
	\ddd	Zeichen mit oktalem Wert ddd (Zahlen zwischen 000 und 377)
*	\x$dd$	Zeichen mit hexadezimalem Wert dd (Zahlen zwischen 00 und ff/FF)
	\"	Anführungszeichen
	\\	Backslash
	\/	Slash

14.3.4 Reguläre Ausdrücke

```
/abc/          # Enthält "abc"
/^abc$/        # Enthält exakt "abc"
/(abc|def)+/   # Enthält "abc" oder "def" mind. 1x nacheinander
```

14.3.4.1 Metazeichen in Regulären Ausdrücken

Gawk	Metazeichen	Bedeutung
	`(`r`)`	**Gruppierung**: Jede Zeichenkette, auf die r paßt (höchster Vorrang)
	c	Zeichen c (kein Metazeichen)
	`\`c	Escape-Sequenz oder (Meta)Zeichen c wörtlich
	`^`	Zeichenketten/Zeilen-Anfang
	`$`	Zeichenketten/Zeilen-Ende
	`.`	1 beliebiges Zeichen
	`[`abc`]`, `[`a`-`z`]`	**Zeichenklasse**: 1 Zeichen aus Menge abc bzw. a–z
	`[^`abc`]`, `[^`a`-`z`]`	**Invertierte Zeichenklasse**:1 Zeichen **nicht** aus Menge abc bzw. a–z
*	`\y \B`	Wortgrenze / Wortinneres (**Break**)
*	`\< \>`	Wortanfang / Wortende
*	`\w \W`	1 Wortzeichen (`[A-Za-z_0-9]`) / kein Wortzeichen (`[^A-Za-z_0-9]`)
*	`` \` \' ``	Zeichenketten/Zeilen-Anfang/Ende (alternative Form)
	r`*`	$0-\infty$ aufeinanderfolgende Zeichenketten, auf die r paßt (**Closure**)
	r`+`	$1-\infty$ aufeinanderfolgende Zeichenketten, auf die r paßt (**Positive Closure**)
	r`?`	$0/1$ Zeichenketten, auf die r paßt (**Option**)
*	r{m`,`n}	$m-n$ Wiederholungen der Zeichenkette, auf die r paßt
*	r{m`,`}	$m-\infty$ Wiederholungen der Zeichenkette, auf die r paßt
*	r{m}	m Wiederholungen der Zeichenkette, auf die r paßt (genau)
	$r_1\,r_2$	**Verkettung**: Jede Zeichenkette `xy`, wo r_1 zu `x` **und** r_2 zu `y` paßt
	r_1`|`r_2	**Alternative**: Jede Zeichenkette, die zu r_1 **oder** r_2 paßt (niedrigster Vorrang)

14.3.4.2 POSIX-Zeichenklassen ([:*class*:])

Klasse	Bedeutung
`alnum`	Alphanumerische Zeichen (Buchstaben + Ziffern)
`alpha`	Buchstaben
`blank`	Leerzeichen oder Tabulator
`cntrl`	Control-Zeichen
`digit`	Dezimalziffern
`graph`	Alle druckbaren und sichtbaren Zeichen
`lower`	Kleine Buchstaben
`print`	Druckbare Zeichen (keine Kontroll-Zeichen)
`punct`	Satzzeichen
`space`	Whitespace (Leerzeichen, Tabulator, Zeilenvorschub, ...)
`upper`	Große Buchstaben
`xdigit`	Hexadezimalziffern

14.3.5 Ausdrücke

```
1 + (2 * exp(3) + sqrt(100)) ^ 2                        # Numerisch
substr($0, 1, 4) "String"                               # Zeichenkette
i <= 10                                                 # Logisch
user !~ /Dieter/ && (age >= 18 || permission == "yes")  # Logisch
```

14.4 Programmaufbau

14.4.1 Regel

```
MUSTER { AKTION }
```

14.4.1.1 Muster

Muster	Bedeutung
`BEGIN` `END`	Preprocessing vor dem Einlesen der Dateien Postprocessing nach dem Einlesen aller Dateien
`Ausdruck` `/REGEXP/`	Logische Bedingung Matcht Eingabezeile
`Ausdruck, Ausdruck` `/REGEXP/, /REGEXP/`	Bereichsmuster auf Basis logischer Bedingungen Bereichsmuster auf Basis Regulärer Ausdrücke

14.4.1.2 Aktionen

!C	Anweisung

	`break`
	`continue`
	`do` *Anweisung* `while (`*Ausdruck*`)`
	`exit` [*Ausdruck*]
	`for (`*Ausdruck1*`;` *Ausdruck2*`;` *Ausdruck3*`)` *Anweisung*
	`for (`*Variable* `in` *Array*`)` *Anweisung*
	`if (`*Ausdruck*`)` *Anweisung1* [`else` *Anweisung2*]

*	`next`
*	`nextfile` (*Gawk*)
	`return` [*Ausdruck*] `while (`*Ausdruck*`)` *Anweisung*
	Ein/Ausgabe-Anweisung (`getline`, `print` und `printf`)
*	`delete` *Arrayelement/Array*
	Ausdruck (mit Konstanten, Variablen, Zuweisungen, Funktionsaufrufen, ...) *Funktion*`(`*Argumente*`)` `{` *Anweisung*`;` `...` `}` (**Block**)

	`switch (`*Ausdruck*`) {`
	`case` *Wert*`|`*Regex*`:` *Anweisung*
	`...`
	`default:` *Anweisung*
	`}`

14.4.2 Funktionen

14.4.2.1 Definition

```
function FUNCNAME(PARAM1, PARAM2, ...)
{
    Anweisung1
    Anweisung2
    ...
    return RESULT
}
```

14.4.2.2 Aufruf

```
FUNCNAME(EXPR1, EXPR2, ...)
```

14.4.3 Zeilenumbrücken sind erlaubt nach

Token	Bedeutung
`,`	Komma (in `print`, `printf`, Funktionsdefinition und -aufruf)
`{`	Linke geschweifte Klammer (Blockbeginn)
`?`	Bedingte Operation (nur *Gawk*)
`:`	Bedingte Operation (nur *Gawk*)
`&&`	Logisch Und
`||`	Logisch Oder
`if (...)`	Bedingung
`else`	Alternative
`for (...)`	Zählschleife
`while (...)`	Abweisende Schleife
`do`	Nicht abweisende Schleife
`}`	Rechte Klammer in einer `if`-, `for`- oder `while`-Anweisung

14.4.4 Felder, Variablen und Arrays

14.4.4.1 Defaultwerte für Feld- und Satztrenner

Variable	Default	Bedeutung
`FS`	␣	**Whitespace** = beliebig lange Folge von Leerzeichen und/oder Tabulatoren; am Zeilenanfang/ende vorkommender Whitespace wird ignoriert [**field separator**]
`RS`	`\n`	Zeilenvorschub [**record separator**]

14.4.4.2 Felder

Variable	Bedeutung
`$0`	Aktueller Eingabesatz
`$`n	Feld n des aktuellen Eingabesatzes (`$1..$NF`)

14.4.4.3 Variablen

```
v  v123  eine_variable  EineVariable  _var_  _VAR_  _Var_
```

14.4.4.4 Arrays

```
arr[123]   = "Text1"
arr["abc"] = 789


i = "A"; j = "B"; k = "C"
arr[i,j,k] = "hello world\n";   # Element "A\034B\034C" belegen


arr[i]
arr[i,j,k]


for (i in arr) ...       # OK
for (i,j,k in arr) ...   # Geht nicht!


if (i in arr) ...
if ((i,j,k) in arr) ... # OK


delete arr[i]
delete arr[i,j,k]


delete arr
```

14.4.4.5 Vergleichstypen

	str	**strnum**	**num**
str	str	str	str
strnum	str	num	num
num	str	num	num

14.4.5 Operatoren

!C	Operator	Bedeutung
	`(...)`	Klammerung
⋆	`$`	**Feld-Zugriff** (`$1`, ...)
	`++ --`	Inkrementieren, Dekrementieren (*Präfix* und *Postfix*)
⋆r	`^ **`	Potenzierung x^y (`**` nur *Gawk*)
	`+ - !`	*Unäres* Plus, *unäres* Minus, logisch NICHT
	`* / %`	Multiplikation, Division, Modulo (Divisionsrest)
	`+ -`	Addition, Subtraktion
⋆	*(space)*	**Verkettung** von Zeichenket. (*kein expliziter Operator!*)
⋆	`| |&`	Ein/Ausgabe auf Pipe (`getline`, `print` und `printf`)
	`< >`	Vergleich (*Zahlen und Zeichenketten*)
	`<= >=`	Vergleich (*Zahlen und Zeichenketten*)
	`== !=`	Vergleich (*Zahlen und Zeichenketten*)
⋆	`~ !~`	Vergleich mit **Reg. Ausdruck**, negiert (*matched by*)
⋆	`in`	**Index** in Array testen
	`&&`	Logisch UND (*short-cut evaluation*)
	`||`	Logisch ODER (*short-cut evaluation*)
r	`?:`	Bedingte Operation ($cond$ `?` $true\text{-}case$ `:` $false\text{-}case$)
r	`= += -= *= /= %= ^= **=`	Zuweisung (inkl. Operation `+ - * / % ^ **`)

14.4.6 Vordefinierte Variablen und Arrays

Gawk	Variable	Bedeutung
	ARGC	Anzahl Kommandozeilenargumente (Programmname in ARGV[0])
*	ARGIND	Index der aktuell eingelesenen Eingabedatei in ARGV[1..ARGC-1]
	ARGV	Array der Kommandozeilenargumente (ARGV[0..ARGC-1])
*	BINMODE	Binäre Ein/Ausgabe für Dateien (1/r = nur Eingabe-Dateien, 2/w = nur Ausgabe-Dateien, 3/rw = Ein+Ausgabe-Dateien)
*	CONVFMT	Format f. autom. Umwandlung Zahl → Zeichenkette (**Default:** "%.6g")
*	ENVIRON	Assoziatives Array mit Umgebungsvar. und ihrem Wert (**Environment**)
*	ERRNO	Systemfehlermeldung bei getline und close
*	FIELDWIDTHS	Liste mit durch Leerz. getrennten festen Spaltenbreiten (statt FS)
	FILENAME	Name der aktuellen Eingabedatei
	FNR	Satznummer der aktuellen Eingabedatei
	FS	Eingabefeldtrenner (**Default:** "␣" = Whitespace)
*	IGNORECASE	Groß/Kleinschreibung beim Vergleich mit Regulärem Ausdruck und in Zeichenketten-Funktionen ignorieren (**Default:** *false*)
*	LINT	Schalter --lint dyn. steuern (true=Lint Warnungen, false=keine)
	NF	Anzahl Felder im aktuellen Satz
	NR	Anzahl bisher eingelesener Sätze ingesamt
	OFMT	Ausgabeformat für Zahlen in print (**Default:** "%.6g")
	OFS	Ausgabe-Feldtrenner in print (**Default:** "␣")
	ORS	Ausgabe-Satztrenner in print (**Default:** "\n")
*	PROCINFO	Array mit Informationen zum Awk-Prozess (gid, uid, ...)
	RLENGTH	Länge des passenden Textes nach match (-1 falls nicht gefunden)
	RSTART	Beginn des passenden Textes nach match (0 falls nicht gefunden)
	RS	Eingabesatztrenner (**Default:** "\n")
*	RT	Zu RS passender Text nach dem Einlesen jedes Satzes
	SUBSEP	Trennzeichen für mehrdim. Index [i,j,...] (**Default:** "\034")
*	TEXTDOMAIN	Textdomäne des Programms für Übersetzung von Texten

14.4.7 Vordefinierte Funktionen

14.4.7.1 Arithmetik-Funktionen

Funktion	Bedeutung
sin(x)	Sinus von x (x in Radiant)
cos(x)	Cosinus von x (x in Radiant)
atan2(y,x)	Arcustangens von y/x im Bereich $-\pi$ bis π (in Radiant)
exp(x)	Exponentialfunktion e^x von x
log(x)	Natürlicher Logarithmus (Basis e) von x
int(x)	Ganzzahliger Teil von x
sqrt(x)	Quadratwurzel von x
rand()	Zufallszahl x, mit $0 \leq x < 1$
srand([x])	x ist neuer Startwert von rand, gibt alten Startwert zurück (verwendet Uhrzeit ohne x)

14.4.7.2 Bit-Funktionen

Funktion	Bedeutung
`and(`a`,`b`)`	a und b bitweise UND-verknüpfen
`or(`a`,`b`)`	a und b bitweise ODER-verknüpfen
`xor(`a`,`b`)`	a und b bitweise XOR-verknüpfen
`compl(`a`)`	a bitweise invertieren (**Komplement**)
`lshift(`a`,`n`)`	a um n Bits nach links schieben
`rshift(`a`,`n`)`	a um n Bits nach rechts schieben

14.4.7.3 Zeichenketten-Funktionen

Gawk	Funktion	Bedeutung
*	`gensub(`r`,`s`[,`h`[,`t`]])`	Ersetzt r in t durch s, gibt Ergebnis zurück; h=`"g/G"`(global) oder Zahl(=Position), sonst nur 1. Position; t bleibt unverändert, ohne t wird `$0` verwendet; `()` ist in r und `\1-\9` ist in s verwendbar
	`gsub(`r`,`s`[,`t`])`	Ersetzt r überall in t durch s; gibt Anzahl Ersetzungen zurück ohne t wird `$0` verwendet [**global**]
	`sub(`r`,`s`[,`t`])`	Wie `gsub`, ersetzt aber nur die erste Teilkette
	`length([`s`])`	Gibt Länge von s zurück, ohne s wird `$0` verwendet;
*	`length(`a`)`	*Gawk*: Anzahl Elemente des Arrays a ermitteln
	`index(`s`,`t`)`	Gibt erste Position (1..n) von t in s zurück; oder 0 wenn t nicht in s enthalten ist
	`match(`s`,`r`[,`a`])`	Überprüft, ob s eine zu r passende Teilkette enthält; gibt Position (1..n) oder 0 (kein Treffer) zurück, wenn r nicht auf s paßt; belegt `RSTART` und `RLENGTH` (0 und -1 wenn kein Treffer); füllt Array a mit in r geklammerten Treffern
	`split(`s`,`a`[,`r`])`	Zerlegt s gemäß r in Arrayelemente, gibt ihre Anzahl zurück; legt diese in den Elementen 1 bis n des Arrays a ab; ohne r wird `FS` verwendet; r=`""` zerlegt s in Buchstaben
*	`ord(`s`)`	Liefert ASCII-Code der 1. Zeichens von s
	`sprintf(`fmt`,`$list$`)`	Gibt $list$ formatiert gemäß fmt zurück (siehe `printf`)
*	`strtonum(`s`)`	Zeichenkette s in Dezimalzahl umwandeln (führende `0`=als Oktalzahl, `0x`/`0X`=als Hexadezimalzahl interpretiert)
	`substr(`s`,`i`[,`n`])`	Gibt Teilstück von s der Länge n ab Position i zurück; ohne n wird der Suffix ab Position i zurückgegeben; das erste Zeichen hat die Position 1
*	`toupper(`s`)`	Wandelt s in Großschreibung um
*	`tolower(`s`)`	Wandelt s in Kleinschreibung um

14.4.7.4 Array-Funktionen

Funktion	Bedeutung
`length(`arr`)`	Anzahl Elemente von Array arr
`delete` arr `delete` arr`[`ind`]`	Array arr vollständig löschen Arrayelement arr`[`ind`]` löschen
`asort(`src`[,`$dest$`])`	**Werte** von Array src direkt bzw. nach Array $dest$ je nach Typ **numerisch oder alpabetisch** sortieren, Anzahl Elemente von Array src zurückgeben
`asorti(`src`[,`$dest$`])`	**Indices** von Array src direkt bzw. nach Array $dest$ grundsätzlich **alphabetisch** sortieren, Anzahl Elemente von Array src zurückgeben

14.4.7.5 Ein/Ausgabe-Funktionen

Gawk	Funktion	Bedeutung
	`close(`$file$`)`	Datei $file$ schließen
	`close(`cmd`)`	Kommando cmd (Pipe) schließen
*	`close(`cmd`)`	Koprozess cmd (Pipe) schließen
*	`close(`cmd`, "to")`	Koprozess cmd (Pipe hin) schließen
*	`close(`cmd`, "from")`	Koprozess cmd (Pipe zurück) schließen
*	`fflush([`$file$`])`	Gepufferte Daten der Datei $file$ schreiben
*	`fflush([`cmd`])`	Gepufferte Daten des Kommandos cmd schreiben
	`getline`	`$0` mit nächster Eingabezeile belegen; setzt `NF`, `NR`, `FNR`
	`getline <` $file$	`$0` mit nächster Eingabezeile aus $file$ belegen; setzt `NF`
	`getline` var	var mit nächster Eingabezeile belegen; setzt `NR`, `FNR`
	`getline` var `<` $file$	var mit nächster Eingabezeile aus $file$ belegen
	cmd `|` `getline`	Ergebnis von BS-Kommando cmd in `$0` ablegen; setzt `NF`
	cmd `|` `getline` var	... in var ablegen
*	cmd `|&` `getline`	Ergebnis von BS-Koprozess cmd in `$0` ablegen; setzt `NF`
*	cmd `|&` `getline` var	... in var ablegen (Koprozess)
	`print`	Aktuelle Eingabezeile `$0` ausgeben (mit Return)
	`print` $list$	Ausdrücke $list$ ausgeben
	`print` $list$ `>` $file$	... auf $file$ ausgeben ($file$ vorher löschen)
	`print` $list$ `>>` $file$	... an $file$ anhängen ($file$ vorher *nicht* löschen)
	`print` $list$ `|` cmd	... an BS-Kommando cmd übergeben
*	`print` $list$ `|&` cmd	... an BS-Koprozess cmd übergeben (läuft parallel)
	`printf` fmt`,`$list$	Ausdrücke $list$ gemäß fmt formatieren und ausgeben
	`printf` fmt`,`$list$ `>` $file$	... und auf $file$ ausgeben
	`printf` fmt`,`$list$ `>>` $file$	... und an $file$ anhängen
	`printf` fmt`,`$list$ `|` cmd	... und an BS-Kommando cmd übergeben
	`printf` fmt`,`$list$ `|&` cmd	... und an BS-Koprozess cmd übergeben (parallel)
	`system(`cmd`)`	BS-Kommando cmd ausführen, Exit-Status zurückgeben

14.4.7.6 getline-Rückgabewerte

Code	Bedeutung
1	Beim erfolgreichen Lesen eines Satzes
0	Am Dateiende
-1	Bei Fehler (z.B. Datei existiert nicht)

14.4.7.7 printf-Formatumwandlungen

Gawk	Format	Bedeutung
	%c	Zeichen [**character**]
i	%d/i	Dezimalzahl mit Vorzeichen [**decimal/integer**]
E	%e/E	Gleitkommazahl [-]d.dddddde[+-]dd [**exponent**]; e klein oder E groß geschrieben
	%f	Gleitkommazahl [-]ddd.dddddd [**float**]
G	%g/G	Verwendet e/E- oder f-Umwandlung, je nachdem welche kürzer ist; nicht signifikante Nullen werden unterdrückt [**general**]
	%o	Oktalzahl ohne Vorzeichen [**octal**]
*	%u	Dezimalzahl ohne Vorzeichen [**unsigned**]
	%s	Zeichenkette [**string**]
*	%x/X	Hexadezimalzahl ohne Vorzeichen [**hexadecimal**]; a-f klein oder A-F groß geschrieben
	%%	Gibt ein % aus; verwendet kein Argument

14.4.7.8 printf-Zusatzangaben zu Formatumwandlungen

Gawk	Parameter	Bedeutung
	-	Ausdruck linksbündig ausrichten (Std: rechtsbündig)
*	␣	Leerzeichen vor positiven Zahlen, - vor negativen
*	+	+ vor positiven Zahlen, - vor negativen
	0	Führende Nullen statt Leerzeichen (nur bei $width$ relevant)
*	#	Alternatives Format bei:
*		o = Führende Null ausgeben
*		x/X = Führendes 0x oder 0X ausgeben
*		eEf = Dezimalpunkt immer ausgeben
*		gG = Nullen am Ende nicht entfernen
	$width$	Ausgabe in Breite $width$ (rechtsbündig mit Leerzeichen), eine führende 0 verwendet dazu Nullen
	$.prec$	Anzahl Nachkommastellen $prec$ bei Zahlen, maximale Breite $prec$ bei Zeichenketten
*	*	Statt $width$ oder $prec$ den nächsten Parameter in der printf-Argumentliste verwenden (nur *Gawk*)
*	$n$$	**Positionsparameter**: n-tes Argument der Argumentliste einsetzen

14.4.7.9 Zeit-Funktionen

Funktion	Bedeutung
`systime()`	Aktuelle Zeit in Sekunden (sec) seit 1.1.1970 1:00 UTC
`strftime(`[fmt[`,` sec]]`)`	Sekunden sec gemäß fmt formatieren Defaultformat: `"%a %b %d %H:%M:%S %Z %Y"` Defaultzeit: `systime()`
`mktime(`$datespec$`)`	Datum der Form `"YYYY MM DD HH MM SS [DST]"` (6 oder 7 Elemente) in Zeitstempel (Anzahl Sek. seit 1.1.1970 1:00 UTC) konvertieren.

14.4.7.10 strftime-Formatangaben

Format	Bedeutung
`%A` / `%a`	Wochentagname vollständig / abgekürzt auf 3 Zeichen
`%B` / `%b`	Monatsname vollständig / abgekürzt auf 3 Zeichen
`%c`	Datum + Zeit gemäß lokalem Standard
`%d`	Monatstag (01-31)
`%H` / `%I`	24 / 12-Stunde (00-23)
`%j`	Jahrestag (001-366)
`%m`	Monat (01-12)
`%M`	Minute (00-59)
`%p`	AM/PM
`%S`	Sekunden (00-61, wegen Korrektursekunden)
`%U` / `%W`	Wochennummer (00-53, erster Sonntag/Montag ist erster Wochentag 1)
`%w`	Wochentagnummer (0-6, 0=Sonntag)
`%X` / `%x`	Zeit / Datum gemäß lokalem Standard
`%Y` / `%y`	Jahr 4- / 2-stellig (1999/00-99)
`%Z`	Zeitzone
`%%`	`%` wörtlich

14.4.7.11 Internationalisierung-Funktionen

Funktion	Bedeutung
`bindtextdomain(`$dirpath$[`,` dom`])`	Verzeichnis $dirpath$ für `.mo`-Dateien festlegen gemäß Textdomäne dom (Std: `TEXTDOMAIN`)
`dcgettext(`s[`,` dom[`,` cat]]`)`	Übersetzung von s gemäß Textdomäne dom (Std: `TEXTDOMAIN`) und Localekategorie cat (Std: `LC_MESSAGES`)
`dcngettext(`s`,` p`,` n[`,` dom[`,` cat]]`)`	Singularform s ($n = 1$) oder Pluralform p ($n > 1$) gemäß Zahl n übersetzen gemäß Textdomäne dom (Std: `TEXTDOMAIN`) und Localekategorie cat (Std: `LC_MESSAGES`)

14.5 GAWK-Spezialdateien

Datei	Inhalt
`/dev/pid`	Aktuelle Prozeß-ID
`/dev/ppid`	Prozeß-ID des Elternprozesses
`/dev/pgrpid`	Aktuelle Prozeß-Gruppen-ID
`/dev/user`	Folgende 4 oder mehr Arrayelemente: 1. Real User-ID 2. Effective User-ID 3. Real Group-ID 4. Effective Group-ID 5.-n: Group-IDs