
Betriebssystem SINIX

Buch 1

Ausgabe Juli 1985 (SINIX V1.0C)

Bestell-Nr. U1901-J-Z95-3
Printed in the Federal Republic of Germany
8800 AG 8853. (11000)

SINIX ist der Name der Siemens-Version des Softwareproduktes XENIX.
XENIX ist ein Warenzeichen der Microsoft Corporation.
XENIX ist aus dem UNIX System III unter Lizenz der Firma AT & T
entstanden.

Copyright © an der Übersetzung Siemens AG, 1984, alle Rechte
vorbehalten.

Vervielfältigung dieser Unterlage sowie Verwertung ihres Inhalts
unzulässig, soweit nicht ausdrücklich zugestanden.

Im Laufe der Entwicklung des Produktes können aus technischen
oder wirtschaftlichen Gründen Leistungsmerkmale hinzugefügt
bzw. geändert werden oder entfallen. Entsprechendes gilt für andere
Angaben in dieser Druckschrift.

Siemens Aktiengesellschaft

Vorwort

Dieses Buch beschreibt SINIX. Es ist gedacht für Benutzer mit Programmierkenntnissen und Systemverwalter. Dieses Buch beschreibt alle Möglichkeiten, die Ihnen SINIX bietet, wenn Sie nicht über Menüs arbeiten.

Dieses Buch ist sowohl für SINIX-Anfänger geeignet, als auch für erfahrene SINIX- oder UNIX-Kenner. Sie benötigen lediglich allgemeine Vorkenntnisse aus der Datenverarbeitung, keine speziellen Systemkenntnisse. Das bedeutet, Ihnen sollten Begriffe, wie z.B.: Prozedur, Parameter, Datei oder Prozeß schon einmal begegnet sein.

Sind Sie SINIX-Anfänger?

Dann beginnen Sie am besten gleich mit dem Kapitel: Einführung. Arbeiten Sie sich durch, bis zum Kapitel: Der Ced-Editor. Eine Beispielsitzung. Anschließend kennen Sie die "SINIX-Philosophie". Sie haben sich eine Basis geschaffen, um mit allen im Kapitel 6 beschriebenen Kommandos arbeiten zu können.

Zum Bearbeiten von Dateien gibt es den bildschirmorientierten Editor CED. Die Beschreibung des CED-Editors sollten Sie sich erst anschauen, wenn Sie die Einführung in SINIX gelesen haben. Es wird dann ein leichtes für Sie sein, dieses komfortable Werkzeug zum Bearbeiten von Dateien zu benutzen.

Sind Sie SINIX/UNIX-Kenner?

SINIX ist eines der vielen UNIX-Kinder. Ein besonders gut gelungenes. Sicherlich haben Sie sich am Anfang Ihres Weges zum UNIX-Kenner oft über die Originaldokumentation geärgert. Wir, die Autoren dieses Buches, auch. Natürlich wollten wir alles besser machen. Und es ist uns wohl auch zum größten Teil gelungen. Wenn Sie nach Studium und Gebrauch dieses Buches anderer Meinung sein sollten, bitte schreiben Sie uns.

Die originalen UNIX-Begriffe haben wir weitgehend ins Deutsche übersetzt. Wir waren der Meinung, daß z.B. der Begriff *Dateiverzeichnis* selbsterklärender ist als *Directory*. Wir sind sicher, dass sich jemand unter dem Begriff *Prozedur* mehr vorstellen kann als unter dem Begriff *Script*. Sie finden am Ende dieses Buchs ein Fachwortverzeichnis deutsch/englisch und englisch/deutsch.

Eine Bitte an Sie

Keine erklärende Dokumentation kann perfekt sein. Eine Dokumentation lebt. Sie lebt auch von Ihren Anregungen, Ideen oder Verbesserungsvorschlägen. Helfen Sie uns, indem Sie uns Ihre Stolpersteine mitteilen, damit wir sie aus dem Weg räumen können.

Manualredaktion K D ST PM 2
Otto-Hahn-Ring 6, 8 München 83

Inhalt

1	Einführung	1-1
	Wie schließen Sie sich an SINIX an?	1-2
	Was passiert durch das Anschließen an SINIX ?	1-3
	Was passiert, wenn Sie über die Tastatur etwas eingeben?	1-3
	Wie können Sie ein Kommando eingeben?	1-4
	Wie können Sie Eingabefehler korrigieren?	1-5
	Wie können Sie Ihre Benutzerkennung durch ein Kennwort schützen	1-6
	Wie können Sie SINIX verlassen?	1-6
2	Das Dateisystem	2-1
	Was erlaubt die Baumstruktur?	2-2
2.1	Dateiverzeichnisse	2-3
	Was kennzeichnet ein Dateiverzeichnis?	2-3
	Wie kann man ein Dateiverzeichnis erzeugen?	2-3
	Wie kann man ein Dateiverzeichnis löschen?	2-4
	Bezeichnungen für Dateiverzeichnisse	2-4
2.2	Dateien	2-5
	Was kennzeichnet eine Datei?	2-5
	Wie kann man eine Datei erzeugen?	2-5
	Wie kann man eine Datei löschen?	2-6
2.2.1	Die Dateien .profile und /etc/profile	2-7
2.2.2	Dateien für Geräte	2-9
2.2.3	Sonderzeichen für Dateinamen	2-10
	Entwerten von Sonderzeichen	2-12
2.3	Indexnummern für Dateien und Dateiverzeichnisse	2-13
	Anzahl der Verweise für ein Dateiverzeichnis	2-14
	Anzahl der Verweise für eine Datei	2-14
2.4	Pfadnamen für Dateien und Dateiverzeichnisse	2-17
	Wie kommt ein Pfadname zustande?	2-17
	Beispiele für Pfadnamen	2-18
2.5	Zugriffsschutz für Dateien und Dateiverzeichnisse	2-19
	Warum sind bei SINIX Dateien und Dateiverzeichnisse geschützt?	2-19
	Wodurch sind bei SINIX Dateien und Dateiverzeichnisse geschützt?	2-19
2.5.1	Benutzerkennung, Kennwort und Gruppennummer	2-19
2.5.2	Schutzbits für Dateien und Dateiverzeichnisse	2-20
	Wie kann man Schutzbits setzen ?	2-21
	Schutzbits gelten nur für ihre Benutzerklasse	2-22
	Beim x-Bit muß man aufpassen	2-24
	Kontrollierter Dateizugriff über Programme durch das s-Bit	2-25

3	Die Kommandoebene Shell	3-1
3.1	Grundsätzliches über Kommandos	3-3
	Wann kann man ein Kommando eingeben?	3-3
	Wie muß bzw. kann man ein Kommando eingeben?	3-3
	Wie verarbeitet die Shell Kommandos?	3-4
3.2	Umleiten der Standard-Ein-/Ausgabe	3-6
	Wie leitet man eine Ausgabe um?	3-6
	Wie leitet man eine Eingabe um?	3-7
3.3	Pipeline	3-8
	Welche Kommandos kann man in einer Pipeline benutzen?	3-9
	Wozu kann man eine Pipeline benutzen?	3-10
3.4	Apostrophiermechanismus	3-12
3.5	Variablen für die Shell	3-15
	Standard-Variablen der Shell	3-16
3.6	Shell-Prozeduren	3-18
3.6.1	Parameter für Shell-Prozeduren	3-19
	Wie kann man in einer Prozedur mit Stellungsparametern arbeiten?	3-21
	Wie kann man in einer Prozedur mit Kennwortparametern arbeiten	3-23
	Wie übergibt man Variablen über die Kommandozeile an eine Prozedur?	3-27
	Das Kommando set -k hat zwei Effekte	3-28
	Umwandeln von Kennwort- in Stellungsparameter	3-29
	Parameter vordefinieren	3-30
3.6.2	Shell-Prozeduren und Prozesse	3-33
	Shell-Prozeduren im Hintergrund ablaufen lassen	3-36
3.7	Ablaufanweisungen für die Shell	3-36
3.7.1	break- und continue	Schleifen steuern 3-37
3.7.2	case	Abfragen und Verzweigen 3-39
3.7.3	for	Liste in Schleifen abarbeiten 3-41
3.7.4	if	Abfragen und Kommandos ausführen 3-43
3.7.5	while- und until	Schleife mit Abbruchbedingung 3-45

3.8	Kommandos der Shell	3-49
3.8.1	:	Ein Kommando, das "nichts" macht 3-50
3.8.2	#	Kommentare einfügen 3-50
3.8.3	() und {}	Kommandos zusammenfassen 3-51
3.8.4	eval	Kommandos übergeben und ausführen 3-53
3.8.5	exec	Kommando ausführen und Shell ersetzen 3-54
3.8.6	exit	Beenden einer Shell-Prozedur 3-57
3.8.7	export	Variablen weiterreichen 3-62
3.8.8	punkt	Datei starten ohne neuen Prozeß zu erzeugen 3-64
3.8.9	read	Prozedur anhalten und etwas einlesen 3-65
3.8.10	readonly	Variablen schützen 3-67
3.8.11	set und sh	Schalter für die Shell setzen 3-68
3.8.12	shift	Verschieben von Operanden 3-72
3.8.13	times	Prozedurzeiten auflisten 3-74
3.8.14	trap	Shell-Prozedur unterbrechen 3-75
3.8.15	umask	Schutzbits voreinstellen 3-77
3.8.16	wait	Auf Prozeßabschluß warten 3-78

4	Der CED-Editor. Eine Beispielsitzung	4-1
	Welches Ziel hat diese Beispielsitzung?	4-1
	Was ist ein Dokument?	4-1
	Was heißt bildschirmorientiert?	4-2
	Was können Sie mit dem CED machen?	4-2
4.1	Welche Tasten können Sie im CED benutzen?	4-3
4.1.1	Die Funktionstasten F9 bis F17	4-3
4.1.2	Löschen und Einfügen von Zeichen	4-4
4.1.3	Löschen und Einfügen von Zeilen	4-4
4.1.4	Wie bewegen Sie die Schreibmarke?	4-4
4.1.5	Der CED arbeitet in verschiedenen Modi	4-5
4.1.6	Die HELP-Taste	4-6
4.1.7	Ein Tastendruck-ein Kommando	4-6
4.2	Beginnen Sie!	4-7
4.2.1	Wie geben Sie Text ein?	4-9
4.2.2	Wie blättern Sie in einem Dokument?	4-10
	'Schnelle Schreibmarkenbewegungen'	4-10
	Wie springen Sie an den Rand des Arbeitsbereichs?	4-10
	Wie springen Sie ans Ende oder den Anfang Ihrer Datei?	4-11
4.2.3	Wie verschieben Sie einen Zeilenbereich?	4-12
4.2.4	Können Sie mit dem CED Rechtecke verschieben?	4-15
	Warum horizontal einfügen?	
	Kann man auch vertikal einfügen?	4-18
4.2.5	Wie können Sie einzelne Zeilen verschieben?	4-19
4.2.6	Wie können Sie SINIX-Kommandos aufrufen?	4-20
4.2.7	Wie können Sie einzelne Zeilen bearbeiten?	4-24
4.2.8	Wie können Sie Daten zwischen Dateien hin-und herschieben?	4-28
4.2.9	Beenden Sie Ihre CED-Sitzung!	4-32
4.3	Sonstige Kommandos	4-33
4.3.1	Wie können Sie Texte in einem Dokument suchen?	4-33
4.3.2	Wie können Sie das Fenster über das Dokument verschieben?	4-35
4.3.3	Wechseln in die Shell	4-36
4.3.4	Tasten mit beliebiger Zeichenfolge belegen	4-37
4.3.5	Tastenbelegung anzeigen	4-40
4.3.6	Bildschirm neu aufbauen	4-41

5	Systemverwaltung	5-1
	Die Rolle des Systemverwalters	5-1
	Die Console	5-1
5.1	Dateiverzeichnisse für die Systemverwaltung	5-2
	/etc	5-3
	/usr/lib	5-2
5.2	Das Betriebssystem hochfahren	5-4
5.3	Das Betriebssystem stoppen	5-4
	/etc/poweroff	5-4
	/etc/haltsys	5-4
5.4	Nachrichten an alle Benutzer senden	5-5
	/etc/wall	5-5
5.5	Login-Nachricht an alle Benutzer senden	5-6
	/etc/motd	5-6
5.6	Daten sichern	5-6
	tar	5-6
5.7	Dateisystem überprüfen	5-7
	/etc/fsck	5-7
	/etc/dcheck	5-10
5.8	Plattenbelegung überprüfen	5-11
	df, du, qout, find	5-11
5.9	Neue Benutzer definieren	5-12
	/etc/passwd	5-12
5.10	Benutzergruppe definieren	5-14
	/etc/group	5-14
5.11	Disketten	5-16
	/etc/flformat	5-17
	/etc/flchk	5-18
	/etc/fldisp	5-18
	/etc/flinit	5-17
5.12	Dateisystem auf Disketten	5-18
	/etc/mkfs	5-20
	/etc/mount	5-25
	/etc/umount	5-26
5.13	Datensichtstationen aktivieren und deaktivieren	5-27
	/etc/disable	5-27
	/etc/enable	5-27
5.14	Datei für Geräte anlegen	5-28
	/etc/mknod	5-28
5.15	Periodische Tätigkeiten anstoßen	5-29
	/etc/cron	5-29
5.16	Dateisystem auf aktuellem Stand halten	5-31
	/etc/update	5-31

5.17	Prozesse, die beim Systemstart ablaufen	5-32
	/etc/init	5-32
	/etc/inir	5-34
	/etc/ttys	5-32
	/etc/ttytype	5-32
	/etc/rc	5-32
5.18	Systemuhr stellen	5-35
	/etc/asktime	5-35
	/etc/mc	5-35
5.19	Begrüßungsbildschirm ändern	5-37
	/etc/herald/*	5-37
5.20	Neue Software installieren	5-38
	/etc/superinstall	5-38
5.21	Die Druckerverwaltung	5-39
	/etc/qdaemon	5-39
5.22	Datensichtstationsdefinitionen	5-44
	/etc/termcap	5-44
5.23	Tips für den Systemverwalter	5-46
	– Der Drucker läuft nicht mehr?	5-46
	– Ein Benutzer hat sein Passwort vergessen?	5-46
	– Der Systemverwalter hat sein Passwort vergessen?	5-47
	– Die Platte ist voll?	5-47
	– Es läuft nichts mehr?	5-47
	– Das System ist abgestürzt?	5-49
	– Definition neuer Benutzer führte zu Inkonsistenzen?	5-50

6	Die Kommandos	6-1
6.1	Kommandos eingeben, aber richtig	6-1
6.2	Was Sie zu jedem Kommando wissen sollten	6-4
6.3	Welches Kommando für welche Aufgabe?	6-6
6.4	Vollständige Beschreibung der Kommandos in alphabetischer Reihenfolge	6-10
at	Prozedur-Dateien zu einer bestimmten Zeit ausführen, Datum englisch	6-11
awk	Dateien durchsuchen und bearbeiten	6-14
basename	Dateinamen vom Pfad trennen	6-27
bc	Arithmetische Sprache	6-29
cal	Kalender ausgeben	6-33
calendar	Erinnerungsdienst, Datum in englischer Schreibweise	6-35
cat	Dateien ausgeben	6-37
cd	Dateiverzeichnis wechseln	6-39
ced	Bildschirmorientierter Editor	6-41
chgrp	Gruppennummer für eine Datei ändern	6-57
chmod	Zugriffsrechte ändern	6-58
chown	Eigentümer einer Datei ändern	6-62
cmp	Dateien zeichenweise vergleichen	6-63
comm	Sortierte Dateien vergleichen	6-65
copy	Dateien gruppenweise kopieren	6-67
cp	Datei kopieren	6-71
crypt	Dateien verschlüsseln	6-73
date	Datum und Uhrzeit ausgeben, englische Schreibweise	6-75
dateityp	Art einer Datei bestimmen	6-77
datum	Datum und Uhrzeit ausgeben, deutsche Schreibweise	6-79
dc	Tischrechner	6-81
df	Dateisystem auf freien Platz prüfen	6-85
diff	Dateien zeilenweise vergleichen und ed-Skript erstellen	6-87
diff3	Drei Dateien zeilenweise vergleichen	6-91
du	Belegten Speicherplatz ausgeben	6-95
echo	Zeichenfolgen ausgeben	6-97
ed	Zeilenorientierter Editor im Dialogbetrieb	6-100
egrep	Erweiterte Muster suchen	6-116
enroll	Schlüssel für geheime Post festlegen	6-119
expr	Ausdrücke auswerten	6-120
false	Leeres Kommando mit Endestatus 1	6-123
far	Archivieren auf Diskette	6-125
fgrep	Einfache Muster schnell suchen	6-126
file	Art einer Datei bestimmen	6-129
find	Dateiverzeichnisse durchsuchen	6-131

grep	Muster in Dateien suchen	6-135
head	Anfangszeilen von Dateien ausgeben	6-138
join	Dateien verbinden nach Vergleichsfeldern	6-139
kalender	Erinnerungsdienst, Datum in deutscher Schreibweise	6-142
kill	Prozesse beenden, Signale senden	6-144
ln	Verweis auf eine Datei eintragen	6-146
login	Benutzerkennung wechseln	6-148
look	Zeilen mit bestimmtem Anfang suchen	6-150
lpr	Dateien ausdrucken und Druckaufträge steuern . . .	6-152
ls	Informationen über Dateiverzeichnisse und Dateien	6-160
mail	Post senden und empfangen	6-164
make	Gruppen von Dateien verwalten	6-168
mesg	Ausgabe von Meldungen verhindern oder erlauben .	6-174
mkdir	Dateiverzeichnis einrichten	6-176
more	Bildschirmausgabe steuern	6-178
mv	Dateien umbenennen oder übertragen	6-183
newgrp	Benutzergruppe wechseln	6-186
nice	Priorität von Kommandos ändern	6-188
nohup	Signale ignorieren	6-190
num	Datei ausgeben mit Zeilennummern	6-192
page	Bildschirmausgabe steuern	6-194
passwd	Kennwort für Benutzerkennung eintragen oder ändern	6-195
pr	Dateien aufbereiten zum Ausdrucken	6-197
prep	Text statistisch aufbereiten	6-201
print	Dateien ausdrucken am Drucker	6-203
printenv	Variablenwerte ausgeben	6-205
ps	Prozeßdaten abfragen	6-207
pstat	Systeminformation ausgeben	6-212
pwd	Pfadnamen des aktuellen Dateiverzeichnisses ausgeben	6-217
quot	Dateisystem prüfen auf Belegung pro Benutzer . . .	6-218
rev	Reihenfolge von Zeichen umkehren	6-220
rm	Dateien löschen	6-221
rmdir	Dateiverzeichnisse löschen	6-223
script	Sitzung protokollieren	6-224
sed	Editor im Prozedurbetrieb	6-226
settime	Zeit des letzten Zugriffs oder der letzten Änderung einer Datei setzen	6-236
sleep	Prozesse zeitweise stilllegen	6-238
sort	Sortieren und mischen von Dateien	6-239
split	Datei aufteilen auf mehrere Dateien	6-244
stty	Eigenschaften der Datensichtstation ändern	6-246

su	Benutzerkennung vorübergehend wechseln	6-249
sum	Prüfsumme einer Datei berechnen	6-252
sync	Systempuffer zurückschreiben	6-254
tail	Endabschnitt einer Datei ausgeben	6-256
tar	Archivieren auf Band oder Diskette	6-258
tcout	Einträge aus der Datei /etc/termcap lesen	6-266
tee	Gleichzeitig auf Standard-Ausgabe und in eine Datei ausgeben	6-269
test	Bedingungen prüfen	6-271
time	Laufzeit eines Kommandos messen	6-276
touch	Zeit der letzten Änderung einer Datei auf aktuelles Datum setzen	6-277
tr	Zeichen durch andere ersetzen	6-278
true	Leeres Kommando mit Endestatus 0	6-280
tty	Pfadname Ihrer Datensichtstation ausgeben	6-281
um	Prozedur-Dateien zu einer bestimmten Zeit ausführen, Datum deutsch	6-283
uniq	Mehrfache Zeilen suchen	6-285
units	Einheiten umrechnen	6-287
wc	Zeilen, Worte und Zeichen zählen	6-290
what	Versionsnummern ausgeben	6-292
who	Aktive Benutzerkennungen anzeigen	6-293
write	Dialog mit anderen Benutzern	6-295
xd	Dateiinhalte hexadezimal ausgeben	6-297
xget	Geheime Post lesen	6-299
xsend	Geheime Post senden	6-301

A	Anhang	A-1
	Ausdrücke	A-2
	Reguläre Ausdrücke	A-3
	Erweiterte reguläre Ausdrücke	A-4
	Die ASCII-Zeichen	A-5

Fachwörter, deutsch - englisch

Fachwörter, englisch - deutsch

Literatur

Stichwörter

1 Einführung

Diese Einführung in SINIX soll Ihnen helfen, für das SINIX-System "ein Gefühl zu bekommen". An Hand einer einfachen Beispielsitzung wird Ihnen folgendes erklärt:

- Wie schließen Sie sich an SINIX an?
- Was passiert durch das Anschließen an SINIX?
- Was passiert, wenn Sie über die Tastatur etwas eingeben?
- Wie können Sie ein Kommando eingeben?
- Wie können Sie Eingabefehler korrigieren?
- Wie können Sie Ihre Benutzerkennung durch ein Kennwort schützen?
- Wie können Sie SINIX verlassen?

Am besten lernen Sie SINIX kennen, wenn Sie praktisch damit arbeiten. Setzen Sie sich deshalb an eine Datensichtstation Ihres Mehrplatzsystems oder an die Datensichtstation Ihres Einplatzsystems und probieren direkt aus, was Ihnen in diesem Buch erklärt wird. Sollten Sie sich im Laufe dieser Beispielsitzung "verirren" und in eine Situation kommen, in der Sie nicht mehr weiter wissen, drücken Sie die Taste `END`. Anschließend wird Ihnen wieder der Begrüßungsbildschirm ausgegeben und Sie können es erneut versuchen.

Wenn Sie länger als 10 Minuten nichts eingeben, verdunkelt sich der Bildschirm. Dadurch wird verhindert, daß sich Zeichen am Bildschirm einbrennen. Drücken Sie die Taste `J`, dann wird am Bildschirm sein letzter Inhalt wieder sichtbar. Den gleichen Effekt erzielen Sie durch jede andere Taste (z.B.: `a`, `x`, `Q`, `p` usw.); allerdings wird der Inhalt dieser Tasten dann am Bildschirm abgebildet.

In diesem Buch wird Ihnen sehr viel mit Beispielen erklärt. Einfache Beispiele, die Ihnen erklären, wie etwas prinzipiell funktioniert oder anwendbar ist. Es ist notwendig in den Beispielen zum Teil Kommandos zu benutzen, deren Bedeutung Ihnen zu diesem Zeitpunkt noch nicht klar sein kann. In den Beispielen wurde auf eine Erklärung dieser (einfachen) Kommandos verzichtet, um nicht vom eigentlichen Beispiel abzulenken. Die Bedeutung und Funktion dieser Kommandos ist im Kapitel 6 erklärt.

Wie schließen Sie sich an SINIX an?

Wenn Sie an der Datensichtstation sitzen, müssen Sie sie als erstes einschalten (Schalter oben rechts). Warten Sie einen Augenblick. Wenn Sie jetzt nicht gleich den Begrüßungsbildschirm, sondern stattdessen eine blinkende Schreibmarke ausgegeben bekommen, drücken Sie bitte die Taste `END`. Anschließend bekommen Sie den Begrüßungsbildschirm ausgegeben.

Bevor Sie mit SINIX arbeiten können, müssen Sie sich mit ihm verbinden. Diesen Vorgang nennt man: Login. Dazu benötigen Sie eine Benutzerkennung. Haben Sie noch keine, dann lassen Sie sich von Ihrem Systemverwalter eine einrichten oder benutzen Sie die Standard-Benutzerkennung: `gast`. Die Benutzerkennung: `gast` ist durch das Kennwort: `SIEMENS` geschützt und steht jedem Benutzer des SINIX-Systems zur Verfügung. Verwenden Sie bitte deshalb diese Benutzerkennung nur zum "Kennenlernen" des SINIX-Systems.

Tippen Sie jetzt Ihre Benutzerkennung in Kleinbuchstaben ein. Ihre Benutzerkennung wird automatisch ab der Position der blinkenden Schreibmarke am Bildschirm angezeigt. Schließen Sie Ihre Eingabe ab, durch Drücken der Taste `↓`.

Falls SINIX Sie jetzt nicht auffordert ein Kennwort einzugeben, das der Systemverwalter für Sie definiert hat, sind Sie jetzt mit SINIX verbunden. Am Bildschirm erscheint ein `$`-Zeichen.

Beachten Sie: Für SINIX ist es ein Unterschied, ob Sie Ihre Benutzerkennung in GROSSBUCHSTABEN oder in kleinbuchstaben eingeben. Die "normale" Eingabe der Benutzerkennung sollte in kleinbuchstaben erfolgen. Eine Eingabe in GROSSBUCHSTABEN hat zur Folge, daß jede folgende Eingabe sofort in GROSS-BUCHSTABEN umgesetzt wird. SINIX antwortet Ihnen dann auch nur noch in GROSSBUCHSTABEN, d.h. Sie können keine kleinbuchstaben mehr eingeben. Außerdem werden bei diesem Modus einige Sonderzeichen anders interpretiert, als bei einer Eingabe in kleinbuchstaben. Geben Sie also Ihre Benutzerkennung nur dann in GROSSBUCHSTABEN ein, wenn Sie bewußt in diesem Modus arbeiten wollen.

Was passiert durch das Anschließen an SINIX?

Wenn Sie sich wie oben beschrieben mit SINIX verbinden, werden Sie von der sogenannten Shell empfangen. Die Shell ist ein Programm, das von nun an alle Ihre Eingaben empfängt, bewertet und entsprechende Aktionen entweder selbst ausführt oder anstößt.

Was passiert, wenn Sie über die Tastatur etwas eingeben?

Erscheint am Bildschirm das \$-Zeichen, können Sie dahinter eine Eingabe schreiben. Geben Sie z.B. über die Tastatur ein Kommando ein, wird Ihre Eingabe am Bildschirm abgebildet. Optisch bekommen Sie den Eindruck, als würden Sie Ihre Eingabe vom Bildschirm aus abschicken. In Wirklichkeit läuft Ihre Eingabe jedoch einen anderen Weg.

Ihre SINIX-Datensichtstation besitzt keine eigene Intelligenz. Jedes Zeichen, das Sie über die Tastatur eingeben, geht sofort an den Rechner. Von dort wird es lediglich zur Kontrolle als "echo" auf Ihren Bildschirm zurückgeschrieben (siehe Bild 1-1).

Wenn Sie Ihre Eingabe abschließen (Taste), steht sie bereits schon im Rechner und wird nicht mehr vom Bildschirm gelesen. Weil eine Eingabe den "Umweg" über den Rechner geht, bevor sie am Bildschirm angezeigt wird, kann es zu einer geringen zeitlichen Verzögerung zwischen einer Eingabe und ihrer Anzeige am Bildschirm kommen. Wenn Sie während dieser Zeitspanne "nervös" werden und etwas eingeben, werden diese Eingaben auch vom Rechner empfangen und bewertet.

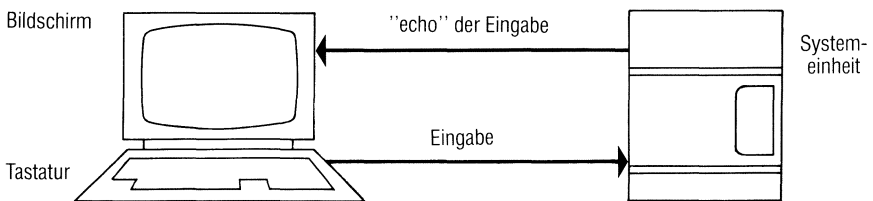


Bild 1-1 Ablauf einer Eingabe

Wie können Sie ein Kommando eingeben?

Nach dem Login erscheint am Bildschirm das \$-Zeichen. Damit signalisiert Ihnen SINIX, daß es bereit ist, Eingaben zu empfangen. Hinter das \$-Zeichen können Sie jetzt ein Kommando schreiben. Wenn Sie Ihre Benutzerkennung in Kleinbuchstaben eingegeben haben, müssen Sie beachten, daß SINIX bei einer Eingabe zwischen Groß- und Kleinbuchstaben unterscheidet. Wenn Sie z.B. das datum-Kommando so eingeben: DATUM, dann ist das für SINIX eine andere Eingabe als: datum. Die Eingabe eines Kommandonamens in Großbuchstaben "versteht" SINIX nicht. Kommandonamen muß man immer in Kleinbuchstaben eingeben. Probieren Sie einmal folgende Eingabe:

```
datum
```

Beenden Sie diese Eingabe durch drücken der Taste . Mit dieser Taste signalisieren Sie dem Rechner im Klartext: Meine Eingabe ist beendet. Wie Sie sehen, wird Ihnen jetzt das aktuelle Datum ausgegeben. Wenn Sie jetzt die Eingabe so machen:

```
DATUM
```

bekommen Sie die Meldung ausgegeben:

```
DATUM: nicht gefunden
```

Ein anderes einfaches Kommando ist das echo-Kommando. Damit können Sie z.B. einen Text eingeben, der auf den Bildschirm als "echo" zurückgeschrieben wird. Wenn sie folgendes eingeben:

```
echo Hallo
```

wird als Ausgabe am Bildschirm ausgegeben:

```
Hallo
```

Merke: Eine Kommandoeingabe muß man mit der Taste abschließen. Anschließend wird die entsprechende Aktion ausgeführt.

Wie können Sie einen Eingabefehler korrigieren?

Machen Sie einmal eine "falsche" Eingabe. Geben Sie das datum-Kommando einmal so ein:

```
datum
```

als Antwort bekommen Sie jetzt ausgegeben:

```
datum: nicht gefunden
```

Diese Eingabe versteht SINIX nicht. Solange Sie eine Eingabe noch nicht mit der Taste abgeschlossen haben, können Sie sie korrigieren. Benutzen Sie dazu bitte die Taste und **nicht** die Tasten des farbigen Tastaturblocks.

Korrigieren Sie eine Eingabe mit den Tasten dieses Tastenblocks, erscheint Ihre Eingabe am Bildschirm zwar optisch richtig, wird aber anders interpretiert, als Sie es auf den ersten Blick annehmen. Warum? Erinnern Sie sich. Jedes Zeichen, das Sie von der Tastatur aus eingeben, geht als erstes an den Rechner, anschließend bekommen sie es als "echo" am Bildschirm angezeigt. Positionieren Sie die Schreibmarke mit einer Taste des Tastaturblocks, wird dafür ein Steuerzeichen an den Rechner geschickt. Dort wird es als Bestandteil der Eingabe interpretiert und nicht als Korrektur Ihrer Eingabe. Am Bildschirm erscheint Ihre Eingabe "optisch richtig", in Wirklichkeit besteht für den Rechner Ihre Eingabe jedoch aus den am Bildschirm angezeigten Zeichen plus Steuerzeichen. Schließen Sie Ihre Eingabe jetzt mit der Taste ab, übergeben Sie damit an den Rechner eine andere Eingabe, als Sie am Bildschirm angezeigt bekommen. Die Tasten des Tastaturblocks können Sie "gefährlos" benutzen, wenn Sie mit dem CED-Editor oder mit Menüs arbeiten (siehe auch: Der CED-Editor. Eine Beispielsitzung).

Wie können Sie Ihre Benutzerkennung durch ein Kennwort schützen?

Eventuell werden Sie von SINIX nach der Eingabe Ihrer Benutzerkennung aufgefordert, ein Kennwort einzugeben. Das ist der Fall, wenn der Systemverwalter Ihre Benutzerkennung durch ein Kennwort geschützt hat. Dieses Kennwort können Sie nach dem Login selbst ändern. Dazu müssen Sie das passwd-Kommando benutzen. Ein Beispiel. Geben Sie das passwd-Kommando ein:

```
passwd
```

SINIX meldet sich mit:

Altes Kennwort: (geben Sie Ihr altes Kennwort ein)

Neues Kennwort:

geben Sie jetzt Ihr neues Kennwort ein. Diese Eingabe wird nicht am Bildschirm abgebildet. Nach Ihrer Eingabe fordert Sie SINIX auf:

Neues Kennwort wiederholen:

Aus Sicherheitsgründen müssen Sie jetzt das neue Kennwort noch einmal eingeben. Anschließend haben Sie zum Schutz Ihrer Benutzerkennung ein Kennwort Ihrer Wahl definiert. SINIX fordert Sie beim Login jetzt immer auf, dieses Kennwort einzugeben.

Wie können Sie SINIX verlassen?

Das System verlassen Sie durch Drücken der Taste `END`. Die aktuelle Kommandozeile am Bildschirm muß dazu leer sein, d.h. die Schreibmarke muß hinter dem \$-Zeichen stehen. Anschließend erscheint am Bildschirm wieder der Begrüßungsbildschirm.

2 Das Dateisystem

Das Dateisystem sieht aus wie ein Baum der nach unten wächst (siehe Bild 2-1). Es besteht aus Dateien und Dateiverzeichnissen. Von Dateiverzeichnissen aus sind Verzweigungen in andere Dateiverzeichnisse oder Dateien möglich. Von einer Datei aus ist keine Verzweigung mehr möglich. Dateiverzeichnisse sind Knotenpunkte des Dateisystems, in denen Namen von Dateien oder anderen Dateiverzeichnissen stehen. Um im Bild zu bleiben: Dateiverzeichnisse sind wie Astgabeln eines Baumes, Dateien sind die Blätter.

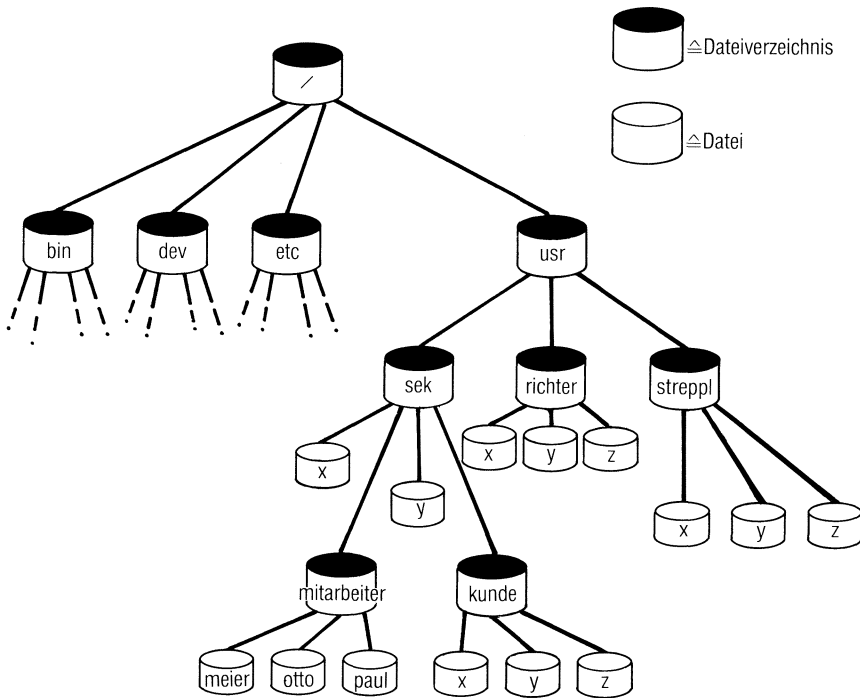


Bild 2-1 Beispiel für ein Dateisystem

Was erlaubt die Baumstruktur?

- An einem Mehrplatzsystem können mehrere Benutzer leicht auf gleiche Dateien zugreifen. Sie sollten jedoch nicht zur gleichen Zeit mit ihnen arbeiten
(siehe: Zugriffsschutz für Dateien und Dateiverzeichnisse).
- Theoretisch kann jeder Benutzer mit jeder Datei oder jedem Dateiverzeichnis des gesamten Dateisystems arbeiten. Dazu braucht er lediglich eine Zugriffsberechtigung für die entsprechende Datei oder das entsprechende Dateiverzeichnis (siehe: Zugriffsschutz für Dateien und Dateiverzeichnisse).
- Eine Datei läßt sich leicht in ein anderes Dateiverzeichnis übertragen. Dazu gibt es zwei Möglichkeiten. Man kann eine Datei physikalisch (d.h. Name plus Inhalt) in ein anderes Dateiverzeichnis kopieren (siehe: copy-Kommando). Die Datei ist dann physikalisch mehrfach vorhanden. Man kann aber auch nur den Namen einer Datei in ein anderes Dateiverzeichnis übertragen (siehe: link-Kommando). Auf so eine Datei bestehen dann entsprechend mehrere Verweise, die Datei ist physikalisch aber nur einmal vorhanden.
- Ein Benutzer kann seine Dateien in ein oder mehrere Dateiverzeichnisse schreiben. Dadurch kann man Dateien übersichtlich und zusammenhängend organisieren.
- Ein Benutzer kann mehreren Dateien den gleichen Namen geben. Bedingung: Die einzelnen Dateien müssen in unterschiedlichen Dateiverzeichnissen eingetragen sein.

2.1 Dateiverzeichnisse

Was kennzeichnet ein Dateiverzeichnis?

Dateiverzeichnisse braucht man, um ein baumartiges Dateisystem aufbauen zu können. Ein Dateiverzeichnis kann nur zwei Informationsarten enthalten:

- Namen von Dateien plus Indexnummern
- Namen von Dateiverzeichnissen plus Indexnummern

Die Indexnummer ist im Kapitel 2.3 erklärt.

Wie kann man ein Dateiverzeichnis erzeugen?

Durch das `mkdir`-Kommando kann jeder Benutzer ein Dateiverzeichnis erzeugen. Ein neu eingerichtetes Dateiverzeichnis enthält standard-mäßig zwei Einträge:

- den eigenen Namen. Er wird durch einen Punkt dargestellt (`.`).
- den Namen des übergeordneten Dateiverzeichnisses. Er wird durch zwei aufeinanderfolgende Punkte dargestellt (`..`).

Mit den Zeichen: `..` und `.` kann man arbeiten. Wie? Wollen Sie z.B. aus dem Dateiverzeichnis, in dem Sie sich gerade befinden, in das übergeordnete Dateiverzeichnis wechseln, können Sie das so erreichen:

Eingabe: `cd ..`

Das Zeichen: `..` bezeichnet den Namen des übergeordneten Dateiverzeichnisses.

Wie kann man ein Dateiverzeichnis löschen?

Löschen kann ein Dateiverzeichnis nur sein Eigentümer. Dazu muß er das `rmdir`-Kommando benutzen. Hängen an einem Dateiverzeichnis beschriebene Dateien, müssen zuerst die Dateien gelöscht werden. Man kann ebenfalls kein Dateiverzeichnis löschen, solange an ihm noch weitere Dateiverzeichnisse hängen. Damit wird verhindert, daß man z.B. unbeabsichtigt einen Teil-Baum "abknipst".

Bezeichnungen für Dateiverzeichnisse

Im Dateibaum gibt es für vier Dateiverzeichnisse Bezeichnungen, die Ihnen die Orientierung erleichtern. Sie heißen:

- Login-Dateiverzeichnis
- Home-Dateiverzeichnis
- Aktuelles-Dateiverzeichnis
- Root-Dateiverzeichnis

Was bedeuten diese Bezeichnungen?

Als **Login-Dateiverzeichnis** wird das Dateiverzeichnis bezeichnet, in dem Sie sich automatisch nach dem Login befinden. Für jeden Benutzer definiert der Systemverwalter in der `/etc/passwd`-Datei ein Login-Dateiverzeichnis.

Als **Home-Dateiverzeichnis** wird das Dateiverzeichnis bezeichnet, das mit der Shell-Variable: `HOME` definiert ist. In dieses Dateiverzeichnis werden Sie automatisch gesetzt, wenn Sie das `cd`-Kommando ohne weitere Angabe eingeben.

Als **Aktuelles-Dateiverzeichnis** bezeichnet man das Dateiverzeichnis, in dem Sie sich jeweils aktuell befinden.

Als **Root-Dateiverzeichnis** wird das Dateiverzeichnis bezeichnet, von dem aus die Baumstruktur des Dateisystems beginnt. Es hat den Namen: `/`.

2.2 Dateien

Was kennzeichnet eine Datei?

Eine Datei kann man als einen "Behälter" betrachten, in den man beliebige Einträge machen kann. Einträge können sein:

- Texte
- Shell-Prozeduren
- Programmcodes
- ausführbare Programme

Dateiinhalte speichert das Betriebssystem sequentiell ab. Solange man über die Shell mit Dateien arbeitet, kann man auf Dateiinhalte nur "ganz oder garnicht" zugreifen. Dateien und Dateiverzeichnisse kann man vor unberechtigtem Zugriff schützen (siehe: Zugriffsschutz für Dateien und Dateiverzeichnisse).

Wie kann man eine Datei erzeugen?

Es ist sehr einfach eine Datei zu erzeugen. Entweder man benutzt dazu einen Editor oder das >-Zeichen. Es gibt kein spezielles Kommando, um eine Datei zu erzeugen. Jeder Dateiname wird in einem Dateiverzeichnis eingetragen. Er wird immer in dem Dateiverzeichnis eingetragen, in dem man sich gerade befindet, wenn man die Datei erstellt. Wenn sie z.B. mit dem ced-Editor eine Datei erstellen wollen, um anschließend gleich etwas in sie einzugeben, geben Sie folgendes ein:

```
ced xyz
```

Jetzt wird der CED-Editor aufgerufen und automatisch eine Datei mit dem Namen xyz erzeugt. Sie können jetzt etwas in sie eintragen.

Durch folgende Eingabe erzeugen Sie mit dem >-Zeichen eine Datei:

```
> eric
```

Jetzt haben Sie eine Datei ohne Inhalt mit dem Namen *eric* erzeugt. Allerdings sollten Sie eine Datei nur dann erzeugen "wenn Sie sie brauchen", d.h. wenn Sie etwas in Sie hineinschreiben wollen. Das ist z.B. der Fall, wenn Sie eine Ausgabe anstatt auf die Standard-Ausgabe (z.B. den Bildschirm der Datensichtstation) in eine Datei schreiben wollen. Das können Sie mit dieser Eingabe erreichen:

```
echo hallo > xyz
```

Die Ausgabe *hallo* des echo-Kommandos wird jetzt automatisch in die Datei *xyz* geschrieben. Existierte die Datei *xyz* bereits, wird sie überschrieben. Gab es die Datei *xyz* noch nicht, wird sie automatisch erzeugt (siehe auch: Umleiten der Standard-Ein-Ausgabe).

Ein Dateiname wird immer automatisch in das Dateiverzeichnis eingetragen, in dem man sich gerade befindet. Noch ein Beispiel. Erzeugt man mit dem cat-Kommando eine Datei, geht das wie folgt:

```
cat > abcd
```

```
hallo    (Taste  drücken)  
         (Taste  drücken)
```

Anschließend meldet sich die Shell mit dem \$-Zeichen zurück. Man hat jetzt eine Datei mit dem Namen *abcd* erzeugt, die den Inhalt *hallo* hat. Existierte die Datei *abcd* bereits, wurde ihr Inhalt überschrieben.

Wie kann man eine Datei löschen?

Eine Datei kann man mit dem rm-Kommando löschen (siehe auch Kommandobeschreibung). Ein Beispiel. Die Eingabe:

```
rm eric
```

löscht die Datei: eric.

Beachten Sie: Eine Datei kann man nur dann durch Eingabe ihres Namens löschen, wenn man sich in dem Dateiverzeichnis befindet, in dem die Datei eingetragen ist. Ansonsten muß man ihren Gesamt-Pfadnamen angeben.

2.2.1 Die Dateien `.profile` und `/etc/profile`

Die Dateien `.profile` und `/etc/profile` sind Dateien für "fortgeschrittene SINIX-Kenner". Um verstehen zu können, was es mit diesen Dateien auf sich hat, sollten Sie vorher die Beschreibung der Shell gelesen haben und mit dem Umgang von Variablen vertraut sein. Nach beiden Dateien sucht die Shell beim Login und führt sie aus, wenn sie vorhanden sind. Das läßt sich als komfortables Hilfsmittel nutzen, um z.B. Variablen zu setzen, zu verändern oder beim Login Kommandos auszuführen.

.profile

Wie funktioniert die Datei `.profile`? Die Datei `.profile` müssen Sie sich selbst erstellen- und zwar in Ihrem Login-Dateiverzeichnis. Das Besondere ist, daß die Shell bei Ihrem Login nach diesem Dateinamen sucht. Wie läuft das ab? Nach dem Login befinden Sie sich in Ihrem Login-Dateiverzeichnis. Bevor die Shell Sie dort "absetzt", durchsucht sie dieses Dateiverzeichnis nach der Datei `.profile`. Enthält Ihr Login-Dateiverzeichnis eine Datei mit Namen `.profile`, führt die Shell diese Datei aus. Was kann man dadurch erreichen? Sie können durch diese Datei z.B. Variablen an die Shell übergeben, mit denen Sie während Ihrer Sitzung arbeiten wollen oder Shell-Variablen verändern. Wenn Sie in Prozeduren mit Kennwortparametern arbeiten wollen, können Sie z.B. das Kommando: `set -k` in die Datei `.profile` schreiben und ausführen lassen. Ein Beispiel für eine `.profile`-Datei:

```
HOME = /usr/lager
PATH = :/bin:/usr/bin:/etc
set -k
name = $USER
echo Hallo
export HOME PATH name
```

Wenn das Ihre `.profile`-Datei wäre, würde bei Ihrem Login folgendes passieren: Sie würden die Standardwerte der Shell-Variablen `HOME` und `PATH` Ihrer Shell umdefinieren, das `set`-Kommando ausführen, eine Variable `name = $USER` definieren sowie am Bildschirm ausgegeben bekommen: `Hallo`. Außerdem hätten Sie die Variablen `HOME`, `PATH` und `name` exportiert.

Die .profile-Datei bietet sich auch während einer Sitzung an, wenn man z.B. Variablen der Shell verändert und nicht will, daß diese Änderungen nach dem Ende der Sitzung verloren sind. Schreiben Sie dann einfach Ihre gewünschten Änderungen in die .profile-Datei und übergeben Sie diese mit dem Punkt-Kommando an die Shell. Damit stehen Ihnen die Änderungen sofort in Ihrer aktuellen Sitzung zur Verfügung und werden bei Ihrem nächsten Login gleich automatisch gesetzt.

/etc/profile

Diese Datei kann nur der Systemverwalter erstellen, weil "normale Benutzer" im Dateiverzeichnis /etc keine Dateien erstellen dürfen. Die /etc/profile-Datei funktioniert im Prinzip genauso wie die .profile-Datei. Einziger Unterschied: Die /etc/profile-Datei wird für jeden Benutzer beim Login ausgeführt.

2.2.2 Dateien für Geräte

Jedes Gerät ist über eine Datei mit SINIX verbunden. Diese Dateien sind alle im Dateiverzeichnis: /dev eingetragen (siehe Bild 2-2).

Dateien für Geräte kann nur der Systemverwalter erstellen oder löschen (siehe: mknod-Kommando). Solange Sie sich jedoch nicht auf der Programmierenebene befinden, sollten Sie Disketten oder Drucker nur über folgende Kommandos ansprechen: print, lpr, far oder tar (siehe Kommandobeschreibung).

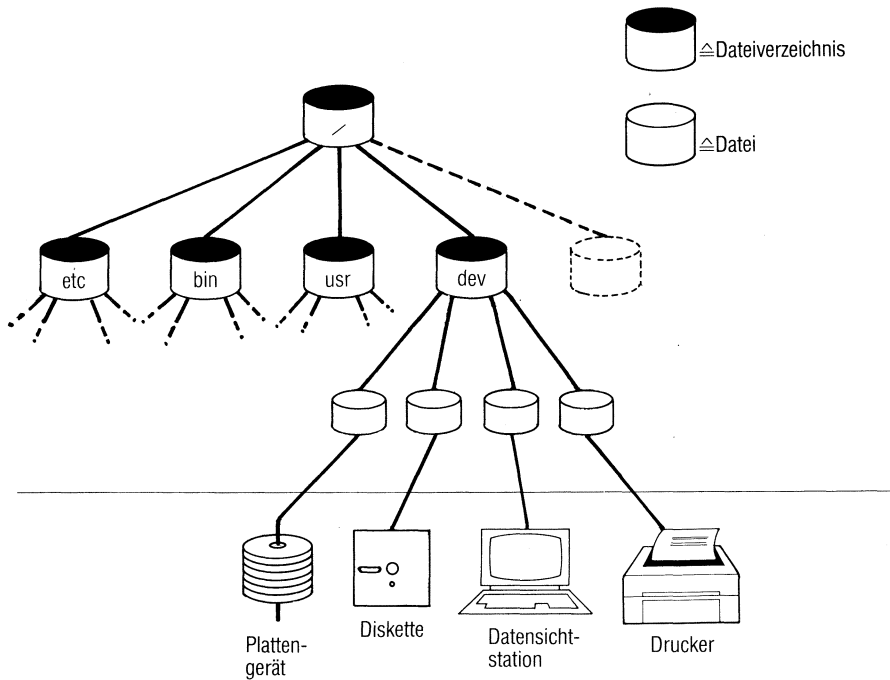


Bild 2-2 Anschluß von Geräten an SINIX

2.2.3 Sonderzeichen für Dateinamen

Es gibt Sonderzeichen, mit denen man sich das Arbeiten mit Dateien erleichtern kann: * , [] und ?. Um sie optimal auszunutzen, können Sie bereits bei der Definition von Dateinamen etwas tun. Ein Beispiel: Angenommen, Sie wollen ein längeres Dokument erfassen (z.B. ein Buch). Logisch kann man es in viele kleine Teile unterteilen (z.B. Kapitel). Physikalisch muß man es ebenfalls unterteilen, weil der Editor mit großen Dateien eventuell nicht umgehen kann. Sie sollten deshalb ihr Dokument nicht in eine Datei schreiben, sondern es in einem Dateiverzeichnis z.B. auf folgende Dateien aufteilen:

```
kap1.1
kap1.2
kap1.3
kap1.4
.
.
usw.
```

Wie man sieht, sind die Dateinamen nach einem Schema aufgebaut. Das bringt zwei Vorteile. Einmal wird die ganze Sache übersichtlicher und außerdem kann man leicht mit Sonderzeichen arbeiten. Eine Folge von Dateien nach dem oben skizzierten Muster, kann man durch folgende Eingabe ausdrucken:

```
cat kap1.1 kap1.2 kap1.3 kap1.4
```

Das ist die umständlichste Art. Mit Sonderzeichen geht es einfacher und schneller. Wenn Sie das cat-Kommando so eingeben:

```
cat kap*
```

bekommen Sie die Inhalte aller Dateien ausgedruckt, deren Dateinamen mit kap beginnen und mit beliebig vielen anderen Zeichen enden. Das wird durch das Sonderzeichen: * erreicht. Dieses Sonderzeichen kann man auf jede Position eines Dateinamens setzen und innerhalb einer Eingabe auch mehrfach angeben. Wird das Zeichen: * ohne Zusatz eingegeben, werden dafür alle Dateien des aktuellen Dateiverzeichnisses eingesetzt.

Aber Achtung: Die Eingabe

`rm *`

löscht alle Dateien Ihres aktuellen Dateiverzeichnisses, deren Namen nicht mit einem Punkt beginnen.

Ein weiteres Beispiel: Das Kommando

`ls *.c`

listet alle Einträge des aktuellen Dateiverzeichnisses auf, deren Namen mit dem Zeichen: `.c` enden.

Mit den Sonderzeichen: `[]` und `[!...]` lassen sich Zeichenfolgen "von-bis" zusammenfassen. So listet z.B. die Eingabe:

`ls [a-k]*`

alle Einträge des aktuellen Dateiverzeichnisses auf, die mit einem der Buchstaben a bis k beginnen.

Die Eingabe:

`ls [!a-k]*`

listet alle Einträge des aktuellen Dateiverzeichnisses auf, die nicht mit einem der Buchstaben a bis k beginnen. Die Negierung wird erreicht durch das Sonderzeichen: `!`.

Mit dem Sonderzeichen: `?` kann man ein beliebiges einzelnes Zeichen in einer Eingabe ersetzen. Die Eingabe:

`cat kap1.?`

gibt die Inhalte aller Dateien aus, deren Namen mit *kap1.* beginnen und denen ein einzelnes beliebiges Zeichen folgt.

Um ein Gefühl für den Umgang mit den Sonderzeichen: * , [] und ? zu bekommen, sollte man sie ausprobieren. Dazu eignet sich besonders das echo-Kommando, denn es hat keine verändernde Aktion zur Folge. Angenommen, Sie haben ein aktuelles Dateiverzeichnis mit folgenden Dateien *kap1 kap2 kap3 ed.doc man.doc*, dann kann man durch folgende Eingaben mit den Sonderzeichen untenstehende Ausgaben erreichen:

echo *	Ausgabe: ed.doc kap1 kap2 kap3 man.doc
echo *.doc	Ausgabe: ed.doc man.doc
echo kap?	Ausgabe: kap1 kap2 kap3
echo kap[2-3]	Ausgabe: kap2 kap3
echo ???.doc	Ausgabe: ed.doc
echo ????	Ausgabe: kap1 kap2 kap3
echo ??	Ausgabe: ?? (weil es keinen Dateinamen mit nur zwei Zeichen gibt)

Wenn Sie die gleichen Eingaben mit dem cat-Kommando machen, bekommen Sie die Inhalte der Dateien ausgegeben.

Wie (fast) immer, gibt es auch für Sonderzeichen eine Ausnahme. Beginnt nämlich ein Dateiname mit einem Punkt, muß man dieses Zeichen immer angeben. Einen führenden Punkt kann man durch kein Sonderzeichen ersetzen. Deshalb bekommt man bei der Eingabe:

```
echo *
```

nur die Dateinamen aufgelistet, die nicht mit einem Punkt beginnen. Durch folgende Eingabe bekommt man auch solche Dateien aufgelistet:

```
echo .*
```

Entwerten von Sonderzeichen

Neben den Sonderzeichen: * , [] und ? haben für die Shell auch noch folgende Zeichen eine spezielle Bedeutung: < , , | , & , && , () und {}.

Will man eines dieser Zeichen z.B. in einem Text als "normales" Zeichen angeben, d.h. ohne daß es die Shell als Sonderzeichen interpretiert, muß man es vorher entwerten. Das macht man mit dem Zeichen: \. Ein Beispiel:

```
echo \?
```

schreibt ein Fragezeichen auf den Bildschirm der Datensichtstation. Das \-Zeichen hat die spezielle Bedeutung des Fragezeichens für die Shell entwertet (siehe auch Kapitel: Apostrophier-Mechanismus).

2.3 Indexnummern für Dateien und Dateiverzeichnisse

Für jede Datei und jedes Dateiverzeichnis wird ein 64 Byte langer Indexeintrag erzeugt. Ein Indexeintrag enthält Verwaltungsinformationen über die entsprechende Datei oder das entsprechende Dateiverzeichnis. Jeder Indexeintrag hat eine Indexnummer, unter der er vom Betriebssystem verwaltet wird. Die Indexnummer wird zusammen mit dem dazugehörigen Dateiverzeichnis- oder Dateinamen im entsprechenden Dateiverzeichnis eingetragen (siehe Bild 2-3).

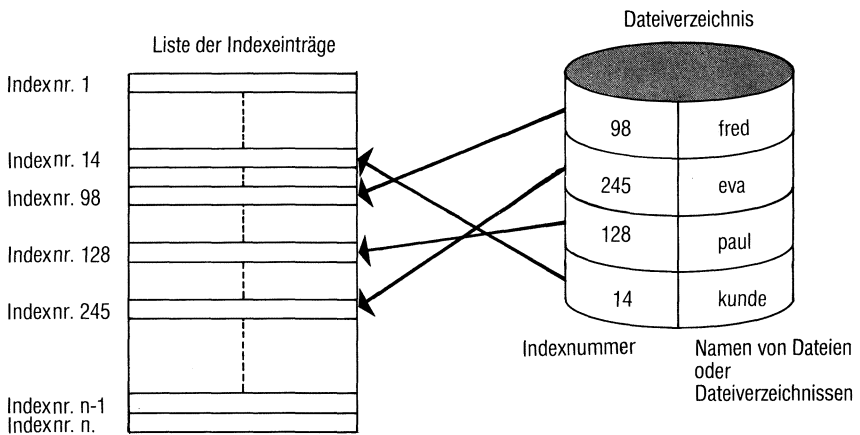


Bild 2-3 Zuordnung von Indexnummer zu Indexeintrag-Liste

Ein vollständiger Indexeintrag enthält folgende Informationen:

- Schutzbits sowie eine Identifikation, ob es sich um einen Eintrag für ein Dateiverzeichnis oder eine Datei handelt.
- Wieviele Verweise auf die Datei oder das Dateiverzeichnis existieren, für die dieser Indexeintrag gilt (Erklärung: siehe weiter unten)
- Benutzernummer
- Gruppennummer
- Größe der Datei oder des Dateiverzeichnisses in Byte
- Plattenblockadresse
- Die Zeit, wann das letzte mal zugegriffen wurde
- Die Zeit, wann das letzte mal modifiziert wurde
- Erstellungszeit des Indexeintrag's

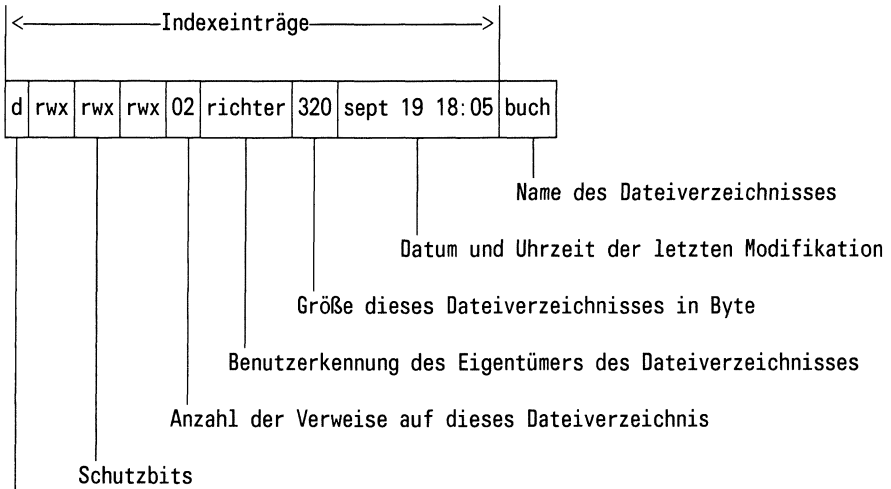
Anzahl der Verweise für ein Dateiverzeichnis

Bei diesem Eintrag wird folgendes eingetragen: Wieviele andere Dateiverzeichnisse direkt an ihm hängen plus 2. Warum: plus 2? Weil ein Dateiverzeichnis immer mindestens zwei Einträge enthält: Seinen eigenen Namen (Abkürzung: .) und den Namen vom Dateiverzeichnis, in dem es selbst eingetragen ist (Abkürzung: ..).

Anzahl der Verweise für eine Datei

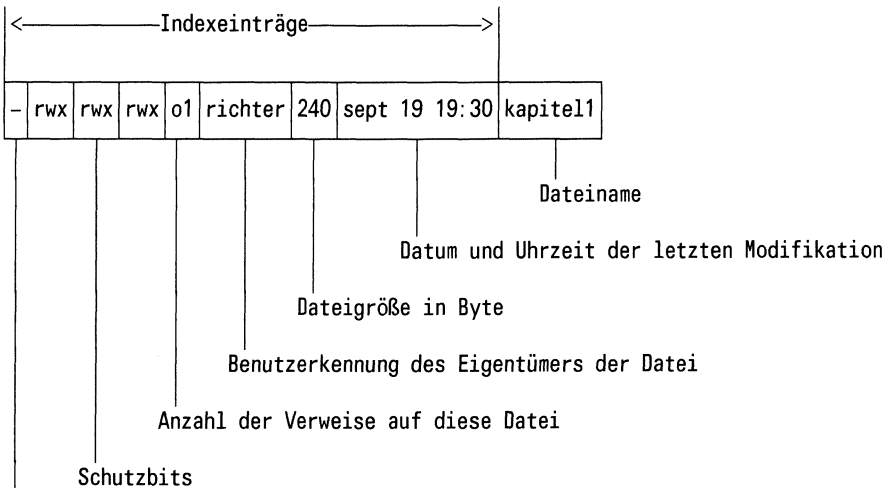
Bei diesem Eintrag wird eingetragen, wie oft eine Datei in irgendeinem Dateiverzeichnis eingetragen ist. Auf eine Datei gibt es mehrere Verweise, wenn es mehrere Links auf sie gibt (siehe ln-Kommando). Indexeinträge kann man mit dem ls-Kommando lesen. Die Ausgaben für Dateien und Dateiverzeichnisse sind unterschiedlich. Ein paar Beispiele.

Beispiel dafür, welche Indexeinträge für ein Dateiverzeichnis mit dem Namen *buch* ausgegeben werden (Kommando: `ls -l`):



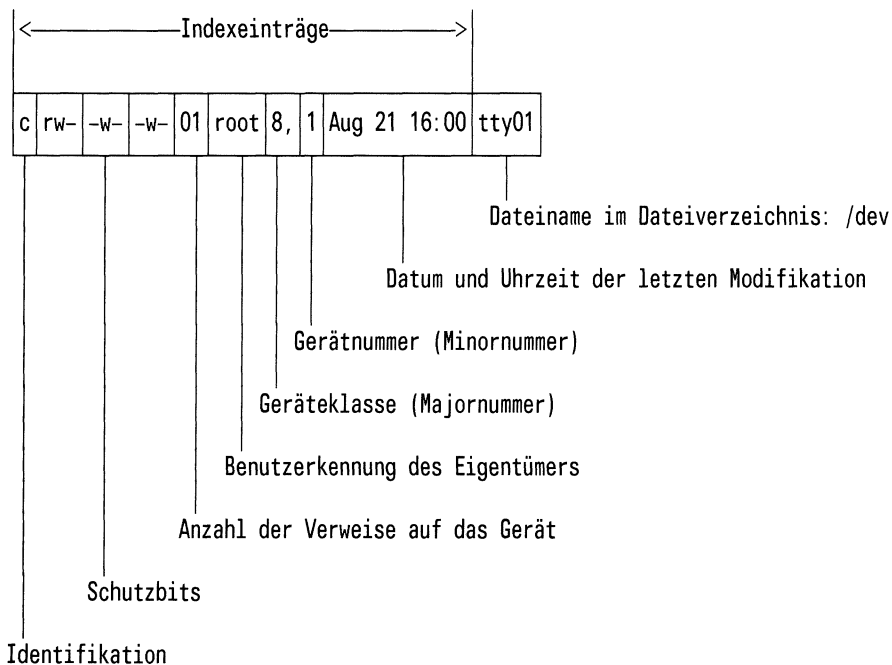
Identifikation (d $\hat{=}$ Dateiverzeichnis)

Beispiel dafür, welche Indexeinträge für eine Datei mit dem Namen *kapitel1* ausgegeben werden (Kommando: `ls -l`):



Identifikation (- $\hat{=}$ Datei)

Beispiel für Indexeinträge einer Datei, über die ein Gerät definiert ist
 (Kommando: `ls -l`):



2.4 Pfadnamen für Dateien und Dateiverzeichnisse

Jede Datei und jedes Dateiverzeichnis hat einen Namen. Zu beiden führt ein Pfad. Der Pfad, der zu einer Datei oder einem Dateiverzeichnis führt, ergibt sich automatisch durch die Lage im Dateisystem.

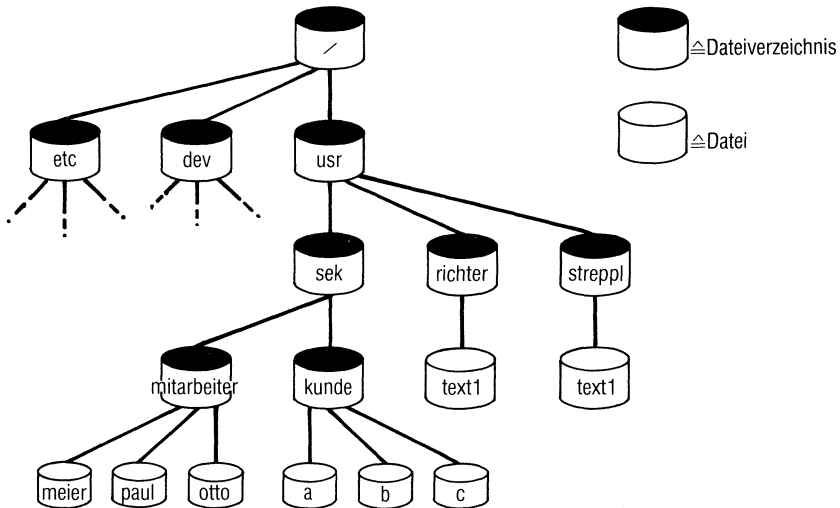


Bild 2-4 Pfadnamen im Dateisystem

Wie kommt ein Pfadname zustande?

Erzeugt man eine Datei oder ein Dateiverzeichnis, wird der Name immer automatisch in dem Dateiverzeichnis eingetragen, in dem man sich befindet. Der Pfadname für eine Datei oder ein Dateiverzeichnis beschreibt den Weg durch die Baumstruktur des Dateisystems, der zur Datei oder zum Dateiverzeichnis führt. Es gibt zwei Arten von Pfadnamen:

- Gesamt-Pfadname
- Relativ-Pfadname

Was sind die Unterschiede?

Ein Gesamt-Pfadname beschreibt den Weg vom Anfang des Dateisystems bis zu einer Datei oder einem Dateiverzeichnis. Er beginnt immer mit einem Schrägstrich: / (das ist der Name des Root-Dateiverzeichnisses).

Ein Relativ-Pfadname beschreibt den Weg zwischen dem gerade aktuellen Dateiverzeichnis und einer Datei oder einem Dateiverzeichnis. Er beginnt immer mit dem Namen eines Dateiverzeichnisses.

Die einzelnen Elemente eines Pfadnamens muß man durch einen Schrägstrich voneinander trennen.

Beispiele für Pfadnamen

Angenommen, einem Benutzer wird nach dem Login das Dateiverzeichnis *mitarbeiter* zugewiesen (siehe Bild 2-4). Dann kann er durch Eingabe der Namen *meier*, *paul* oder *otto* mit den Dateien dieses Dateiverzeichnisses arbeiten. Der Gesamt-Pfadname für die Datei *meier* sieht so aus:

```
/usr/sek/mitarbeiter/meier
```

Will der Benutzer mit der Datei *a* im Dateiverzeichnis *kunde* arbeiten, kann er das über die Eingabe des Gesamt- oder Relativ- Pfadnamens erreichen. Der Gesamt-Pfadname heißt:

```
/usr/sek/kunde/a
```

Der Relativ-Pfadname, ausgehend vom Dateiverzeichnis *mitarbeiter*, heißt:

```
../kunde/a
```

Was bedeutet das Zeichen: .. ? Damit sagt man "gehe ein Dateiverzeichnis zurück", in diesem Fall in das Dateiverzeichnis *sek*. Vom Dateiverzeichnis *mitarbeiter* ausgehend, sieht der Relativ-Pfadname für die Datei *text1* im Dateiverzeichnis *richter* so aus:

```
../../richter/text1
```

2.5 Zugriffsschutz für Dateien und Dateiverzeichnisse

Warum sind bei SINIX Dateien und Dateiverzeichnisse geschützt?

Theoretisch könnte man auf einen Zugriffsschutz für Dateien und Dateiverzeichnisse verzichten. Jeder Benutzer könnte dann auf alle Dateien und Dateiverzeichnisse des Dateisystems zugreifen und mit ihnen arbeiten. Die Folge wäre wahrscheinlich bald ein Chaos. Denn jeder Benutzer könnte -versehentlich oder absichtlich- unkontrolliert das gesamte Dateisystem verändern. Jeder Benutzer könnte die Dateien anderer Benutzer lesen und verändern. Um das zu verhindern, gibt es Schutzmechanismen. Damit kann man Dateien und Dateiverzeichnisse vor unbefugtem Zugriff schützen. Es läßt sich festlegen, mit welchem Teil des Dateibaums welcher Benutzer arbeiten darf.

Wodurch sind bei SINIX Dateien und Dateiverzeichnisse geschützt?

Durch:

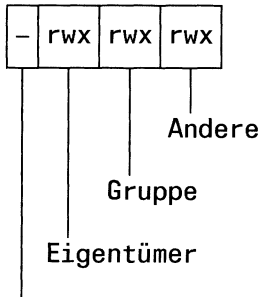
- Benutzerkennungen
- Kennworte für Benutzerkennungen
- Zusammenfassen von Benutzerkennungen zu Gruppen
- Schutzbits für Dateien und Dateiverzeichnisse
- Verschlüsselung von Dateiinhalten (siehe: crypt-Kommando)

2.5.1 Benutzerkennung, Kennwort und Gruppennummer

Jeder Benutzer, der sich ans System anschließen will (Login), braucht eine Benutzerkennung. Die muß für ihn der Systemverwalter definieren. Jeder Benutzer kann selbst ein Kennwort definieren oder verändern, um seine Benutzerkennung zu schützen. Das Kennwort wird beim Login abgefragt und muß nach der Benutzerkennung eingegeben werden (siehe: passwd-Kommando). Benutzer kann man zu Gruppen zusammenfassen. Damit kann man Dateien oder Dateiverzeichnisse einer Gruppe von Benutzern zugänglich machen. Dafür muß der Systemverwalter eine Gruppennummer vergeben (siehe: /etc/group- und /etc/passwd-Datei).

2.5.2 Schutzbits für Dateien und Dateiverzeichnisse

Jeder Datei und jedem Dateiverzeichnis werden beim Erstellen Schutzbits automatisch mitgegeben. Diese Schutzbits kann der Eigentümer verändern und damit seine Dateien oder Dateiverzeichnisse schützen (siehe: umask-Kommando). Es gibt jeweils 3 Schutzbits für die 3 Benutzerklassen: Eigentümer, Gruppe und Andere.



Identifikationszeichen

Wie man sieht, steht vor den Schutzbits noch ein Identifikationszeichen. Je nachdem ob man eine Datei oder ein Dateiverzeichnis erstellt, wird dort automatisch ein entsprechender Eintrag gemacht. Folgende Einträge sind möglich:

- d $\hat{=}$ Dateiverzeichnis
- $\hat{=}$ Datei
- b $\hat{=}$ Datei für ein Gerät, auf das man blockweise (b) zugreifen kann.
- c $\hat{=}$ Datei für ein Gerät, auf das man zeichenweise (c) zugreifen kann.

Für jede der 3 Benutzerklassen kann man folgende Berechtigungen festlegen:

- r $\hat{=}$ Leseberechtigung (read)
- w $\hat{=}$ Schreibberechtigung (write)
- x $\hat{=}$ Ausführberechtigung (execute)

Die Berechtigungen werden für Dateien und Dateiverzeichnisse unterschiedlich interpretiert. Und zwar so:

	Datei	Dateiverzeichnis
read	lesen	Einträge lesen
write	schreiben	Einträge löschen/anlegen
execute	ausführen	durchlaufen/durchsuchen

Wie kann man Schutzbits setzen?

Schutzbits kann man mit dem `chmod`-Kommando setzen (siehe: Kommandobeschreibung). Der Systemverwalter darf die Schutzbits aller Dateien oder Dateiverzeichnisse verändern. Der Eigentümer darf nur die Schutzbits seiner eigenen Dateien oder Dateiverzeichnisse verändern. Auch wenn jemand aus der Gruppe oder Andere volle Zugriffsberechtigung auf eine Datei oder ein Dateiverzeichnis besitzt, darf er die Schutzbits nicht verändern.

Die Schutzbits werden vom Betriebssystem als ein Teil des Indexeintrags abgespeichert. Deshalb kann man sie auch am Bildschirm nicht alleine lesen, sondern nur zusammen mit anderen Indexeinträgen (siehe: `ls`-Kommando). Für jede Datei und jedes Dateiverzeichnis gibt es eine Standardeinstellung der Schutzbits, die beim Erstellen mitgegeben wird. Die Standardeinstellung kann man mit dem `umask`-Kommando verändern (siehe: Kommandos der Shell). Die Standardeinstellung sieht so aus:

Datei

-	rw-	rw-	r--
---	-----	-----	-----

Dateiverzeichnis

d	rx	rx	r-x
---	----	----	-----

Wie man sieht, kann mit einer Datei standardmäßig der Eigentümer und die Gruppe: lesen und schreiben. Andere können nur: lesen.

Ein Dateiverzeichnis kann standardmäßig vom Eigentümer und der Gruppe durchlaufen werden. Beide dürfen Einträge lesen und löschen bzw. anlegen. Andere dürfen ein Dateiverzeichnis nur durchlaufen und Einträge lesen.

Schutzbits gelten nur für ihre Benutzerklasse

Die Schutzbits beziehen sich streng auf die Benutzerklassen: Eigentümer, Gruppe, Andere. Was heißt das? Hat z.B. der Eigentümer einer Datei keine Zugriffsberechtigung für seine Datei aber die Gruppe und Andere (Einstellung `---rwxrwx`), dann darf er auch nicht mit ihr arbeiten. Haben z.B. nur die Anderen eine Zugriffsberechtigung auf eine Datei (Einstellung `-----rwx`), dann darf weder der Eigentümer noch ein Mitglied seiner Gruppe mit der Datei arbeiten. Die Schutzbits für: Eigentümer, Gruppe und Andere schließen einander also aus. Der Eigentümer einer Datei bekommt damit z.B. keine Zugriffsberechtigung über seine Gruppenzugehörigkeit.

Beispiele dafür, wie man Schutzbits definieren kann

Beispiel 1

Eigentümer, Gruppe und Andere sollen volles Zugriffsrecht auf die Datei *text* haben.

Eingabe: `chmod 777 text`

Eingabe: `ls -l`

Ausgabe: `-rwxrwxrwx 1 richter 33 oct 19 10:00 text`

Beispiel 2

Eigentümer und Gruppe sollen auf die Datei *text* volles Zugriffsrecht haben. Andere sollen aus ihr nur lesen können.

Eingabe: `chmod 774 text`

Lesen der Änderung.

Eingabe: `ls -l`

Ausgabe: `-rwxrwxr-- 1 richter 33 oct 19 10:00 text`

Beispiel 3

Nur der Eigentümer der Datei *text* soll Lese- und Schreibberechtigung haben.

Eingabe: `chmod 600 text`

Lesen der Änderung.

Eingabe: `ls -l`

Ausgabe: `-rw----- 1 richter 33 oct 19 10:00 text`

Beispiel 4

Nur der Eigentümer des Dateiverzeichnisses *buch* soll volles Zugriffsrecht haben.

Eingabe: `chmod 700 buch`

Lesen der Änderung.

Eingabe: `ls -l`

Ausgabe: `drwx - - - - - 2 richter 33 oct 19 10:00 buch`

Beispiel 5

Nur der Eigentümer des Dateiverzeichnisses *buch* soll volles Zugriffsrecht haben. Die Gruppe und Andere sollen das Dateiverzeichnis durchlaufen können.

Eingabe: `chmod 711 buch`

Lesen der Änderung.

Eingabe: `ls -l`

Ausgabe: `drwx - - x - - x 2 richter 33 oct 19 11:00 buch`

Beispiel 6

Der Eigentümer der Datei *text*, die eine Shell-Prozedur enthält, soll volles Zugriffsrecht haben. Gruppe und Andere sollen sie nur ausführen können.

Eingabe: `chmod 755 text`

Lesen der Änderung.

Eingabe: `ls -l`

Ausgabe: `-rwxr - xr - x 1 richter 33 oct 19 10:00 text`

Beim x-Bit muß man aufpassen

Das Schutzbit für die Ausführberechtigung (execute) wird für Dateien und Dateiverzeichnisse unterschiedlich interpretiert. Da ein Dateiverzeichnis nicht ausgeführt werden kann, wird das x-Bit so interpretiert, daß die entsprechende Benutzerklasse das Dateiverzeichnis durchlaufen darf. Will man mit dem x-Bit eine Datei, die eine Shell-Prozedur enthält, als ausführbar kennzeichnen, muß man gleichzeitig durch das r-Bit eine Leseerlaubnis vergeben. Das x-Bit alleine reicht nicht, um eine Prozedurdatei für eine Benutzerklasse ausführbar zu machen.

Achtung

Ein nicht gesetztes x-Bit garantiert noch keinen Schutz gegen unbeberechtigtes Ausführen einer Prozedurdatei. Wieso? Eine Datei kann man auf zwei Arten ausführen:

- durch Angabe ihres Namens
- durch das sh-Kommando

Will man eine Datei durch Angabe ihres Namens ausführen, kann man das nur, wenn das entsprechende r- und x-Bit gesetzt ist (siehe oben). Für das sh-Kommando gilt dieser Mechanismus nicht. Was macht das sh-Kommando? Damit ruft man die Shell auf, um eine Datei zu lesen und ihren Inhalt zu interpretieren. Enthält diese Datei eine Shell-Prozedur, führt die Shell diese auch aus. Das x-Bit (=Ausführberechtigung) braucht dafür nicht gesetzt zu sein. Es reicht, wenn das r-Bit (= Leseberechtigung) gesetzt ist. Was hat das zur Folge? Da jeder Benutzer das sh-Kommando aufrufen darf, sollte man sich als Eigentümer einer Datei darüber im klaren sein, daß man mit Vergabe einer Leseberechtigung auch gleichzeitig eine Ausführberechtigung vergibt. Will man eine Datei, die einen ausführbaren Programmcode enthält, als ausführbar kennzeichnen, reicht es, wenn man das x-Bit alleine setzt.

	Datei mit Shell-Prozedur als Inhalt	Datei mit ausführbarem Programmcode	Dateiverzeichnis (ausführen \cong durchlaufen)
wird ausführbar durch	x-Bit mit r-Bit oder r-Bit alleine	x-Bit	x-Bit

Kontrollierter Dateizugriff über Programm durch das s-Bit

Das s-Bit bindet ein Programm an definierte numerische Identifikationen. Was bedeutet das? Für jeden Benutzer gibt es zwei numerische Identifikationen (siehe auch: Bild 2-5):

- Eine Benutzernummer (UID). Darüber ist jeder Benutzer im SINIX-System eindeutig identifiziert.
- Eine Gruppennummer (GID). Darüber ist jeder Benutzer im SINIX-System einer Gruppe zugeordnet.

Beide Identifikationen vergibt der Systemverwalter.

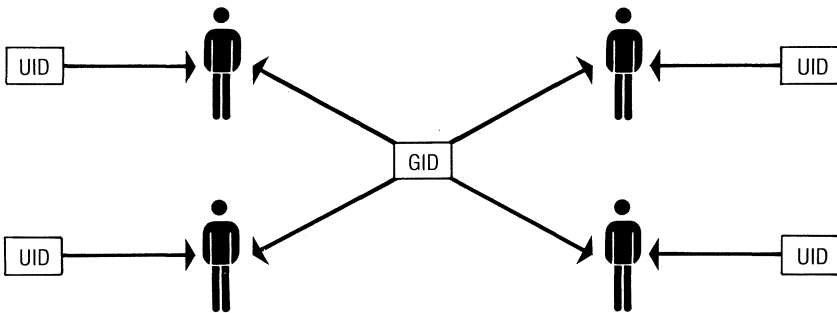


Bild 2-5 Zuordnung von UID und GID zu Benutzern

Startet ein Benutzer ein Programm, läuft es unter diesen numerischen Identifikationen (UID und GID) ab (vorausgesetzt natürlich, der Benutzer hat eine Ausführberechtigung für das entsprechende Programm). Das kann man bewusst ausnutzen- es kann aber auch zu Problemen führen.

Ein Beispiel. Angenommen, Sie haben als Eigentümer ein Programm, das auf eine Datei zugreift, die ebenfalls Ihnen gehört. Die Schutzbiteinstellung soll so sein (siehe auch: Bild 2-6):

- für das Programm: `-rwxr-xr-x`
- für die Datei: `-rwx-----`

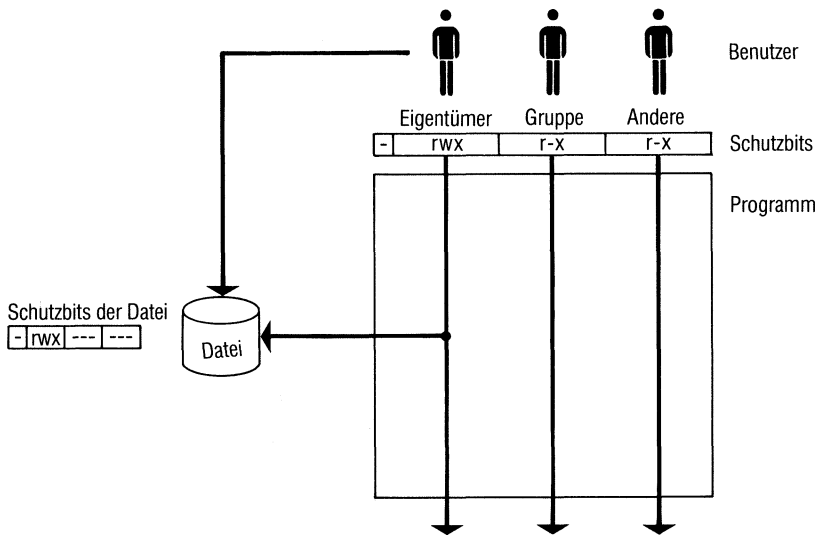


Bild 2-6 Dateizugriff über ein Programm

Damit können Sie als Eigentümer Ihre Datei und Ihr Programm lesen, schreiben (d.h. verändern) und ausführen. Die Mitglieder Ihrer Gruppe und Andere können nur Ihr Programm ausführen und nicht auf die Datei zugreifen. Mitglieder Ihrer Gruppe und Andere können damit zwar Ihr Programm ausführen, jedoch nicht mit diesem Programm auf Ihre Datei zugreifen. Wie kann man den Benutzern: Gruppe/Andere einen Zugriff auf die Datei erlauben? Erst einmal natürlich dadurch, daß Sie die Schutzbits für Ihre Datei so einstellen: `-rwxr-xr-x`. Damit können die Benutzer: Gruppe/Andere jedoch auch auf die Datei zugreifen, ohne Ihr Programm zu benutzen. Der Zugriff auf die Datei ist dann nicht mehr über Ihr Programm kontrollierbar. Soll der Zugriff auf die Datei nur über Ihr Programm möglich sein, muß man das `s`-Bit setzen. Was macht das `s`-Bit? Es hat zur Folge, daß ein Programm immer nur unter einer fest definierten UID oder GID abläuft- egal welcher Benutzer es aufruft. Ein Beispiel: Angenommen, Sie haben als Eigentümer folgende Schutzbits gesetzt:

- für das Programm: `-rwsr-xr-x`
- für die Datei: `-rwx-----`

Dann haben Sie für sich als Eigentümer das `s`-Bit gesetzt (siehe auch: `chmod`-Kommando). Jetzt läuft das Programm immer unter Ihrer Benutzernummer (UID). Egal ob es ein Mitglied Ihrer Gruppe oder Andere ausführen. Das Programm kann damit unabhängig davon, welcher Benutzer es ausführt, auch auf die Datei zugreifen.

Noch ein Beispiel für das s-Bit. Angenommen, Sie haben als Eigentümer folgende Schutzbits gesetzt (siehe auch Bild 2-7):

- für das Programm: - rwx - - s - - x
- für die Datei1: - rwxrw - - - -
- für die Datei2: - rwx - - - - - -

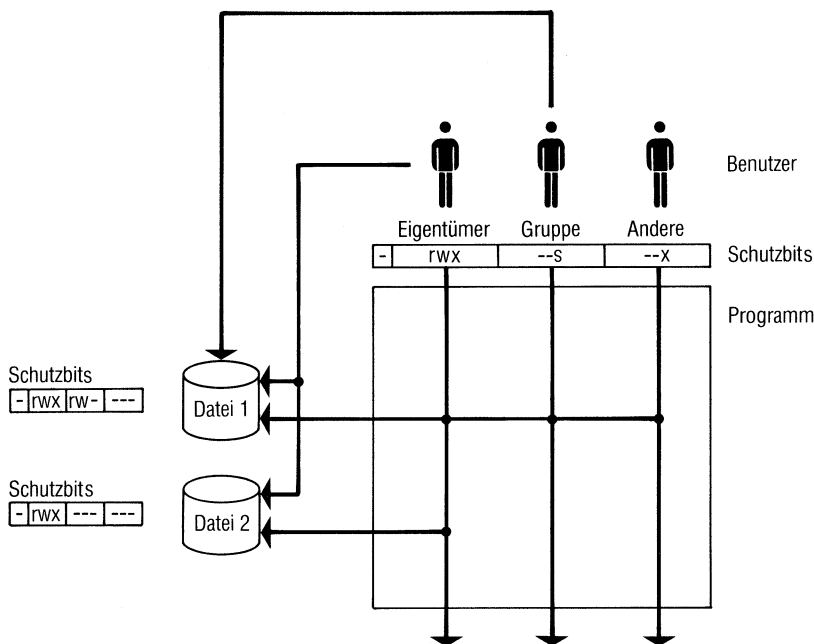


Bild 2-7 Dateizugriff über ein Programm mit s-Bit

Jetzt können Sie als Eigentümer, die Mitglieder Ihrer Gruppe sowie Andere das Programm ausführen und darüber auf die Datei1 zugreifen. Das wird durch Setzen des s-Bits bei der Gruppe erreicht. Andere können auf die Datei1 nur über das Programm zugreifen, auf die Datei2 haben sie weder direkt noch über das Programm einen Zugriff. Sie als Eigentümer können, ohne das Programm zu benutzen, die Datei1 und Datei2 lesen, schreiben und ausführen. Mitglieder Ihrer Gruppe können, ohne das Programm zu benutzen, die Datei1 lesen und schreiben.

3 Die Kommandoebene Shell

Die Kommandoebene Shell verbindet die Benutzer des Systems mit dem Systemkern (= Betriebssystem). Die Shell und der Systemkern sind voneinander getrennt und kommunizieren über Systemaufrufe miteinander (siehe auch: Bild 3-1). Auf den Systemkern haben Benutzer keinen direkten Zugriff. Die Hauptkomponenten des Systemkerns sind:

- das Dateiverwaltungssystem
- die Prozeßverwaltung
- das Ein-/Ausgabesystem
- die Speicherverwaltung

Die Kommandoebene Shell bietet dem Benutzer eine umfangreiche Kommandosprache. Im Gegensatz zu anderen dialogorientierten Kommandosprachen hat sie den Vorteil, daß sie sich wie eine Programmiersprache anwenden läßt. Gleichzeitig erlaubt es die Shell, eigene neue Kommandos zu erstellen bzw. Kommandos miteinander zu verknüpfen.

Die Shell liest Kommandos entweder von einer Datensichtstation oder aus einer Datei und startet entsprechende Programme. In einer Datei stehende Kommandos heißen Shell-Prozeduren. Wie Sie solche Prozeduren erstellen können und auf was dabei zu achten ist, wird in diesem Kapitel beschrieben. Ebenso alle anderen Möglichkeiten, die Ihnen die Shell im Umgang mit Kommandos bietet.

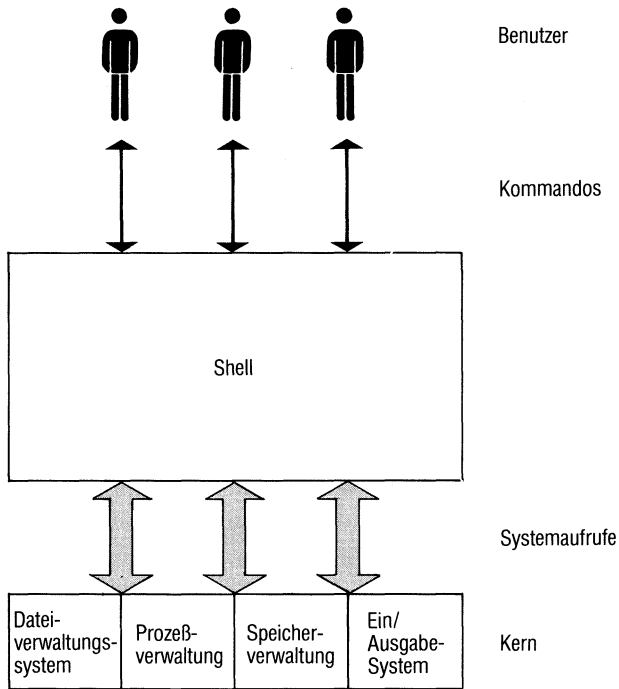


Bild 3-1 Die Kommandoebene Shell im SINIX-System

3.1 Grundsätzliches über Kommandos

Wann kann man ein Kommando eingeben?

Nach dem Login (= anschließen an SINIX) können Sie grundsätzlich jederzeit ein Kommando eingeben. Ihre Eingabe wird von SINIX immer entgegengenommen, aber nur dann verarbeitet, wenn die Shell empfangsbereit ist. Das ist sie, wenn Sie am Bildschirm das Empfangsbereitsymbol ausgegeben bekommen. Das ist in der Regel das Zeichen: \$. Geben Sie nach einer Eingabe sofort wieder etwas ein, ohne vorher das \$-Zeichen bekommen zu haben, so wird Ihre Eingabe zwar von SINIX angenommen und am Bildschirm abgebildet, jedoch erst verarbeitet, wenn die Shell das nächste \$-Zeichen auf den Bildschirm schreibt. Eine Ausnahme ist Taste DEL, sie wird immer sofort ausgeführt. Mit ihr kann man z.B. eine laufende Prozedur abbrechen.

Merke: Die Shell verarbeitet ein Kommando erst, wenn sie empfangsbereit ist. Das ist sie, wenn am Bildschirm das \$-Zeichen erscheint.

Erscheint an Ihrem Bildschirm das Zeichen: > , erwartet die Shell weitere Eingaben zum gerade eingegebenen Kommando. Das passiert, wenn Sie:

- mit Kommandos für Shell Prozeduren im Dialog arbeiten
- eine Zeichenkette in der Kommandozeile mit einem Apostroph begonnen haben, ohne sie mit einem Apostroph zu beenden.
- eine Kommandozeile mit dem Sonderzeichen: * abschließen. Dadurch entwertet Sie das Zeichen <Neue Zeile> und die Shell nimmt an, daß Sie mit der Kommandoingabe in der nächsten Zeile fortfahren wollen.

Wie muß bzw. kann man ein Kommando eingeben?

Standardmäßig folgt nach dem Empfangsbereitsymbol ein Leerzeichen. Direkt an das Leerzeichen anschließend, kann ein Kommando folgen. Abschließen muß man eine Kommandoingabe mit der Taste ↵.

Mehrere Kommandos kann man zusammen in einer Kommando-Liste eingeben. Wie kann eine Kommando-Liste aussehen? Sie kann bestehen aus:

- einem oder mehreren Kommandos
- einer oder mehreren Pipelines

Mehrere Angaben in einer Kommando-Liste müssen durch eines der folgenden Zeichen voneinander getrennt sein: ; , & , && oder ||.

Die Zeichen && und || haben eine höhere Priorität als die Zeichen ; und &. Die einzelnen Zeichen haben folgende Wirkung:

- Das ;-Zeichen bewirkt, daß die einzelnen Kommandos oder Pipelines einer Kommando-Liste nacheinander ausgeführt werden.
- Das &-Zeichen bewirkt, daß die einzelnen Kommandos oder Pipelines einer Kommando-Liste als Hintergrundprozeß ablaufen.
- Das &&- und ||-Zeichen fragen den Ende-Status eines Kommandos oder einer Pipeline ab. Auf das &&-Zeichen folgende Kommandos oder Pipelines werden nur dann ausgeführt, wenn der Ende-Status Null gemeldet wird (d.h. das Kommando oder die Pipeline wurde erfolgreich ausgeführt).
- Auf das ||-Zeichen folgende Kommandos oder Pipelines werden nur dann ausgeführt, wenn der Ende-Status ungleich Null gemeldet wird (d.h. das Kommando oder die Pipeline wurde nicht erfolgreich ausgeführt).

Wie verarbeitet die Shell Kommandos?

Bis auf "eingebaute Kommandos" (siehe: Kommandos der Shell) ist jedem Kommando eine Datei zugeordnet. Die Namen der meisten Kommando-dateien sind in den Dateiverzeichnissen /bin und /usr/bin eingetragen. Zur Ausführung liest die Shell das Kommando und sucht in diesen Dateiverzeichnissen nach dem Dateinamen, der identisch mit dem Kommandonamen ist. Die gesuchte Datei enthält ein Programm, das das eingegebene Kommando ausführt.

Um ein Kommando auszuführen, erzeugt die Shell meist einen neuen Prozeß und wartet anschließend, bis er fertig ist. Danach gibt die Shell das \$-Zeichen aus und kann ein neues Kommando verarbeiten.

Ein Beispiel: Auf die Eingabe

`datum`

bekommt man umgehend das aktuelle Datum ausgegeben, anschließend schickt die Shell das `$`-Zeichen auf den Bildschirm und ist damit bereit, ein neues Kommando zu verarbeiten.

Bei einem Kommando wie `datum` ist die Zeitspanne klein, die zwischen der Eingabe des Kommandos und der erneuten Empfangsbereitschaft der Shell vergeht. Es könnte jedoch zu längeren Wartezeiten an der Datensichtstation führen, wenn man z.B. eine Datei übersetzt. Denn die Shell meldet sich ja erst zurück, nachdem sie ein Kommando ausgeführt hat. Um längere Wartezeiten zu vermeiden, gibt es das Sonderzeichen: `&`. Wenn man mit diesem Zeichen eine Kommandoingabe beendet, meldet sich die Shell sofort mit dem Empfangsbereitschaftszeichen: `$` am Bildschirm zurück, anstatt erst auf die Beendigung des eingegebenen Kommandos zu warten. Man kann dann sofort ein neues Kommando eingeben und braucht nicht zu warten. Warum? Weil das gestartete Kommando als Hintergrundprozeß gestartet wurde, um den Sie sich nicht weiter zu kümmern brauchen.

Beispiel. Die Eingabe:

`cc pgm.c&`

ruft den C-Compiler auf, um die Datei `pgm.c` als Hintergrundprozeß zu übersetzen. Mit dem `ps`-Kommando kann man sich eine Liste der aktiven Prozesse ausgeben lassen. Auf diese Art kann man Hintergrundprozesse im Auge behalten.

3.2 Umleiten der Standard-Ein-/Ausgabe

Die Shell nimmt an, daß im Normalfall Ein- und Ausgaben mit der Datensichtstation verbunden sind, von der aus ein Kommando eingegeben wurde. Im Klartext heißt das: Die Eingabe kommt von der Tastatur und die Ausgabe geht auf den Bildschirm. Diesen Ein-/Ausgabefluß kann man ändern. Durch Umleitungen. So kann z.B. eine Eingabe an die Shell anstatt von der Tastatur aus einer Datei kommen. Eine Ausgabe kann man ebenfalls anstatt auf den Bildschirm in eine Datei oder auf einen Drucker leiten.

Wie leitet man eine Ausgabe um?

Ein Beispiel: Gibt man das folgende Kommando ein:

```
ls
```

bekommt man die Dateinamen des aktuellen Dateiverzeichnisses am Bildschirm ausgegeben. Wenn man die folgende Eingabe macht:

```
ls > dateiliste
```

werden die Einträge des aktuellen Dateiverzeichnisses in die Datei *dateiliste* geschrieben. Diese Datei wird erzeugt, falls es sie noch nicht gab oder aber überschrieben, wenn sie schon existierte. Die Umleitung der Ausgabe wird durch das Zeichen: > erreicht. Das >-Zeichen bedeutet: Schreibe die durch dieses Kommando erzeugte Ausgabe in die dem >-Zeichen folgende Datei, anstatt auf den Bildschirm der Datensichtstation. Mit dem cat- Kommando und dem >-Zeichen kann man auch mehrere Dateien in einer Datei zusammenfassen:

```
cat d1 d2 d3 > sammel
```

Durch dieses Kommando werden die Dateien *d1*, *d2* und *d3* in der Datei *sammel* zusammengefaßt. Existierte die Datei *sammel* bereits, wird ihr alter Inhalt überschrieben. Soll ein alter Inhalt erhalten bleiben, muß man die Eingabe so machen:

```
cat d1 d2 d3 >> sammel
```

Das Zeichen: >> bedeutet: Schreibe die durch das cat-Kommando zusammengefaßten Dateien *d1*, *d2* und *d3* hinter einen in der Datei *sammel* stehenden Inhalt.

Wie leitet man eine Eingabe um?

Eine Eingabe leitet man um mit dem Zeichen: < . Es bedeutet: Nimm die Eingabe für das Kommando aus der dem < Zeichen folgenden Datei.

Ein Beispiel: Das Kommando

```
wc <eric
```

liest seine Eingabedaten aus der Datei *eric* und gibt am Bildschirm die Anzahl der gefundenen Zeilen, Worte und Zeichen aus.

Hinweis

Die Standard-Eingabe läßt sich auch mit dem `exec`-Kommando umleiten (siehe Kapitel 3.8.5).

3.3 Pipeline

Man kann mehrere Kommandos zu einer Pipeline ketten. Was heißt das? Ein Beispiel. Angenommen, Sie wollen wissen, wieviele Einträge ein Dateiverzeichnis enthält. Das können Sie dadurch erreichen, indem Sie folgende Kommandos nacheinander eingeben:

```
ls > temp  
wc -l < temp  
rm temp
```

Jetzt haben Sie die Ausgabe des `ls`-Kommandos in die Datei `temp` geschrieben, anschließend durch das `wc`-Kommando die Anzahl der Zeilen (\cong Anzahl der Einträge) ermittelt und als drittes mit dem `rm`-Kommando die "Schmierdatei": `temp` wieder gelöscht. Wenn man die gleichen Kommandos durch eine Pipeline miteinander verbindet, geht das ganze schneller und einfacher.

```
ls|wc -l
```

hat die gleiche Wirkung. Erreicht wird sie durch das Pipe-Zeichen: `|`. Damit erzeugt man eine Pipeline (=Verbindung) zwischen den beiden Kommandos. Die vom `ls`-Kommando erzeugte Ausgabe wird in diesem Fall dem `wc`-Kommando als Eingabe übergeben. Allgemein ausgedrückt: In einer Pipeline wird die Standard-Eingabe eines Kommandos, durch das Pipe-Zeichen, mit der Standard-Ausgabe des vorhergehenden Kommandos verbunden. Auf diese Art lassen sich in der Praxis bis zu 10 Kommandos verbinden. Durch das Pipe-Zeichen verbundene Kommandos brauchen für die Datenübergabe keine eigene Datei. Für je zwei Kommandos wird im Kernspeicher ein Puffer eröffnet. Dadurch wird eine größere Effizienz erreicht, als bei einer Datenübergabe mittels einer Datei.

Alle in einer Pipeline stehenden Kommandos arbeiten gleichzeitig und jedes als eigener Prozeß. Ein Kommando liest und verarbeitet Daten, sobald das vorhergehende Kommando sie auf seine Standard-Ausgabe schreibt. Dadurch braucht ein Kommando nicht darauf zu warten, bis das vorhergehende Kommando beendet ist. Wenn Sie das folgende Kommando eingeben:

```
more /etc/termcap
```

wird am Bildschirm der Inhalt der Datei */etc/termcap* ausgegeben. In der letzten Bildschirmzeile steht eine Angabe, wieviel Prozent der Datei bis jetzt angezeigt wurde (z.B. -- MORE 27%).

Geben Sie jetzt folgendes ein:

```
cat /etc/termcap|more
```

Sobald jetzt das *cat*-Kommando über seine Standard-Ausgabe etwas in die Pipeline eingibt, liest es das *more*-Kommando und gibt es bildschirmweise aus. In der letzten Bildschirmzeile steht jetzt jedoch lediglich:

-- MORE -- , d.h. die Prozentangabe fehlt. Warum? Weil das *more*-Kommando zum Zeitpunkt der ersten Ausgabe noch nicht "weiß", wieviele Daten es aus der Pipe noch "geliefert bekommt". Es kann folglich auch keine Prozentangabe darüber machen, wieviel Prozent der Datei es am Bildschirm ausgegeben hat.

Welche Kommandos kann man in einer Pipeline benutzen?

In einer Pipeline kann man alle Kommandos benutzen, die von der Standard-Eingabe lesen und auf die Standard-Ausgabe ausgeben. Am Anfang oder Ende können auch andere Kommandos stehen (z.B. *ls,who,date*).

Wozu kann man eine Pipeline benutzen?

Grundsätzlich kann man mit einer Pipeline nichts machen, was man nicht auch anders machen könnte. Jedes in einer Pipeline verwendbare Kommando kann man auch einzeln eingeben. Allerdings kann eine Pipeline ein sehr komfortables Hilfsmittel sein. Wenn Sie z.B. aus einer Eingangsinformation eine bestimmte Information herausfiltern wollen, können Sie das durch viele Einzeleingaben erreichen- oder sie durch eine Pipeline schicken. Ein Beispiel. Die beiden Dateien *kunde1* und *kunde2* sollen folgenden Inhalt haben:

Datei: *kunde1*

artmann	muenchen	kundennummer:160349
richter	muenchen	kundennummer:300350
noack	muenchen	kundennummer:190459
bap	koeln	kundennummer:310846
borg	koeln	kundennummer:301462

Datei: *kunde2*

strunck	muenchen	kundennummer:160856
strepel	muenchen	kundennummer:300461
blumann	muenchen	kundennummer:190851
zelting	koeln	kundennummer:311249
buchber	koeln	kundennummer:302368

Beispiel 1: Es sollen alle münchner Kunden aus beiden Dateien alphabetisch aufgelistet am Bildschirm erscheinen.

```
Eingabe:  cat kunde1 kunde2 | grep muenchen | sort
Ausgabe:  artmann  muenchen  kundennummer:160349
          blumann  muenchen  kundennummer:190851
          noack    muenchen  kundennummer:190459
          richter  muenchen  kundennummer:300350
          strepl   muenchen  kundennummer:300461
          strunck  muenchen  kundennummer:160856
```

Beispiel 2: Es sollen alle Kunden aus beiden Dateien, deren Kundennummer mit der Ziffer 30 beginnt, alphabetisch sortiert am Bildschirm erscheinen.

```
Eingabe:  cat kunde1 kunde2 | grep :30 | sort
Ausgabe:  borg      koeln      kundennummer:301462
          buchber  koeln      kundennummer:302368
          richter  muenchen  kundennummer:300350
          strepel  muenchen  kundennummer:300461
```

Was wurde gemacht?

In beiden Fällen wurden die Dateien *kunde1* und *kunde2* durch das `cat`-Kommando gelesen. Durch Angabe des Pipe-Zeichens: `|` wurden sie an das `grep`-Kommando übergeben und dort nach den gewünschten Begriffen durchsucht. Jede Zeile, die den entsprechenden Suchbegriff enthielt, wurde an das `sort`-Kommando übergeben und von diesem in alphabetischer Reihenfolge ausgegeben.

3.4 Der Apostrophier-Mechanismus

Durch den Apostrophier-Mechanismus kann man der Shell befehlen, wie sie eine zu verarbeitende Zeichenkette interpretieren soll. Es gibt 3 Möglichkeiten, eine Zeichenkette durch Apostrophe zu kennzeichnen:

- ”....” Steht eine Zeichenkette zwischen doppelten Apostrophen, dann:
- liest die Shell den Inhalt,
 - ersetzt die Shell vorhandene Stellungs- oder Kennwortparameter,
 - führt die Shell diese Zeichenkette im Sinn des davorstehenden Kommandos aus,
 - bleibt für die Shell die Bedeutung folgender Sonderzeichen erhalten: \$, ` und \.
- ’....’ Steht eine Zeichenkette zwischen einfachen Apostrophen, dann:
- liest die Shell den Inhalt, ignoriert jedoch seine Bedeutung.
- `....` Steht eine Zeichenkette innerhalb Akzent Gravis, dann:
- interpretiert die Shell den Inhalt als Kommando,
 - führt die Shell das Kommando aus und setzt das Ergebnis des Kommandos an die Stelle dieser Zeichenkette.

Beispiele für den Apostrophier-Mechanismus:

Angenommen, es wurde folgende Eingabe gemacht: name=Meier. Die Shell merkt sich diese Eingabe in ihrem aktuellen Umfeld. Damit kann man mit dieser Eingabe als Kennwortparameter arbeiten. Die folgenden Beispiele zeigen, wie man den Apostrophier-Mechanismus darauf anwenden kann.

Eingabe: echo "\$name"
Ausgabe: Meier

Eingabe: echo '\$name'
Ausgabe: \$name

Eingabe: echo ` \$name `
Ausgabe: Meier: nicht gefunden

Eingabe: echo "Der Name des Kunden ist: \$name"
Ausgabe: Der Name des Kunden ist: Meier

Eingabe: echo "\$name `date`"
Ausgabe: Meier plus aktuelles Datum

Eingabe: echo date oder echo "date" oder echo 'date'
Ausgabe: date

Eingabe: echo `date`
Ausgabe: Aktuelles Datum

In den bisherigen Beispielen liest die Shell von der Standard-Eingabe (Tastatur) und gibt auf die Standard-Ausgabe (Bildschirm) aus. Die Shell behandelt die eingegebenen Zeichenketten entsprechend dem angegebenen Apostrophier-Mechanismus.

Weitere Beispiele mit Kennwortparameter: name = Meier

Die Eingabe:

```
echo "echo Der Name ist: $name">Text
schreibt in die Datei: Text
echo Der Name ist: Meier
```

Wie man sieht, wird der Kennwortparameter \$name durch seinen definierten Wert (Meier) ersetzt und in die Datei *Text* geschrieben. Bei Aufruf dieser Datei würde ausgegeben: Der Name ist: Meier

Dieser Aufruf entspricht den Konventionen, wenn man eine Zeichenkette in doppelte Apostrophe "...." einschließt. Gibt man die gleiche Zeichenkette mit einfachen Apostrophen '....' ein:

```
echo 'echo Der Name ist: $name' > Text
```

wird in die Datei *Text* folgendes eingetragen:

```
echo Der Name ist $name
```

Wie man sieht, wird jetzt der Kennwortparameter \$name nicht sofort durch seinen definierten Wert ersetzt. Erst wenn man die Datei *Text* ausführt, wird \$name durch den Wert: Meier ersetzt (substituiert).

Bei Eingabe von: echo `date` > Text wird in die Datei *Text*

```
date
```

eingetragen. Beim Ausführen der Datei *Text* wird date als Kommando erkannt und am Bildschirm das aktuelle Datum angezeigt.

Bei Eingabe von: echo `date` > Text

wird in die Datei: Text

```
`date`
```

eingetragen. Ruft man die Datei *Text* auf, erwartet die Shell ausführbare Kommandos. Was passiert? Als erstes liest die Shell die Zeichenkette `date` und substituiert darin das date-Kommando. Für `date` steht dann das aktuelle Datum. Das Ergebnis dieser Kommando-Substitution versucht die Shell nun aus der Datei zu lesen und als ausführbares Kommando zu behandeln. Was natürlich nicht funktioniert. Das Ergebnis des date-Kommandos wird bei dieser Schreibweise nicht am Bildschirm ausgegeben.

3.5 Variablen für die Shell

Die Shell erlaubt die Verwendung von Variablen. Variablen kann man z.B. dazu benutzen, um Abläufe zu steuern. Eine Variable besteht aus einem Variablennamen und einem Wert. So definiert man eine Variable in der Shell:

Eingabe: name = Hans

Durch diese Eingabe hat man in der Shell eine Variable mit dem Wert *Hans* definiert. Will man mit dem Wert einer Variablen arbeiten (z.B. im Dialog oder in einer Prozedur), muß man ihn so aufrufen: \$Variablenname (siehe auch: export- und readonly-Kommando). Ein Beispiel.

Eingabe1: name = fred

Eingabe2: echo \$name

Ausgabe : fred

Was wurde gemacht? Es wurde die Variable *name* mit dem Wert *fred* definiert. Anschließend wurde der Variablenwert *fred* durch das echo-Kommando aufgerufen und am Bildschirm ausgegeben.

Den Namen für Variablen kann man frei wählen. Festgelegt sind lediglich die Namen einiger Standard-Variablen. Bis auf die MAIL-Variable weist die Shell ihnen Standardwerte zu. Alle Standardwerte kann der Benutzer ändern.

Aber Achtung: Ändert man den Standardwert einer Standard-Variablen, kann man mit dem geänderten Wert in einer anderen als der Login-Shell nur dann arbeiten, wenn man ihn vorher exportiert hat (siehe export-Kommando). Andernfalls arbeitet man außerhalb der Login-Shell immer nur mit den zugewiesenen Standardwerten der Standard-Variablen, d.h. mit den "nicht geänderten" Werten.

Standard-Variablen für die Shell

Variable	Bedeutung
HOME= ...	Dieser Variablen weist die Shell den in der /etc/passwd-Datei definierten Namen des Login-Dateiverzeichnisses zu. Gibt man das cd-Kommando ohne weitere Angaben an, benutzt es den durch diese Variable definierten Wert.
IFS= ...	Durch IFS kann man Feld-Trennzeichen definieren. Das sind die Zeichen, die die Shell als Leerzeichen interpretiert. Standardwerte sind: Leerzeichen, Tabulator und Neue Zeile
MAIL= ...	Durch diese Variable gibt man den Pfadnamen einer Post-Datei an. Jedesmal wenn etwas in diese Post-Datei geschrieben wird, bekommt der Eigentümer während der Sitzung eine Nachricht am Bildschirm ausgegeben. Hinweis: Die Variable hat keinen Einfluß auf das mail-Kommando. Das mail-Kommando schreibt Post immer in die Standard-Post-Datei (Name: /usr/spool/mail/\$USER).
PATH= ...	Durch diese Variable gibt man einen oder mehrere Suchpfade an, die von der Shell durchsucht werden sollen, um ein Kommando zu finden. Die Suchpfade werden von der Shell in der Reihenfolge ihrer Angabe durchsucht. Die folgenden Suchpfade werden von der Shell als Standardwerte definiert: :/usr/bin/local:/bin:/usr/bin. Die Doppelpunkte trennen die einzelnen Suchpfade voneinander. Beginnt eine Suchpfadangabe mit einem Doppelpunkt, wird dieser Doppelpunkt jedoch als aktuelles Dateiverzeichnis interpretiert, d.h. die Shell beginnt die Suche nach einem Kommando im aktuellen Dateiverzeichnis. Soll das aktuelle Dateiverzeichnis als letztes durchsucht werden, muß man es so schreiben: PATH=/usr/bin/local:/bin:/usr/bin::
PS1= ...	Durch PS1 wird ein Zeichen oder eine Zeichenfolge festgelegt, womit die Shell einen Benutzer zur Eingabe auffordert. Als Standardwert trägt die Shell ein \$-Zeichen mit anschließendem Leerzeichen ein. Diese Angabe kann man jederzeit ändern.
PS2= ...	Durch PS2 wird ein Fortsetzungszeichen definiert, das die Shell an den Anfang der nächsten Zeile setzt, falls eine Kommandoeingabe länger als eine Zeile ist. Als Standardwert trägt die Shell ein >-Zeichen mit anschließendem Leerzeichen ein.
TERM=...	Durch diese Variable wird ein Datensichtstationstyp definiert. Als Standardwert trägt die Shell den in der Datei: ttytype eingetragenen Wert ein (z.B. 97801).
USER=...	Durch diese Variable gibt man eine Benutzerkennung an. Als Standardwert trägt die Shell die aktuell an der Datensichtstation eingegebene Benutzerkennung ein.

Die Shell versteht auch einige Sonderzeichen. Gibt man sie zusammen mit einem \$-Zeichen im Dialog an der Datensichtstation ein oder ruft man sie in einer Prozedur auf, ersetzt sie die Shell durch definierte Werte. Das sind sie:

Zeichen	Bedeutung
#	Gibt die Anzahl der angegebenen Stellungsparameter dezimal an.
-	Schalter, mit denen die Shell aufgerufen wurde oder durch das set-Kommando gesetzt wurden.
?	Der zurückgemeldete Wert (dezimal) des zuletzt ausgeführten Kommandos (Ende-Status eines Kommandos).
\$	Die Prozeßnummer dieser Shell.
!	Die Prozeßnummer des zuletzt aufgerufenen Hintergrundkommandos.
* oder @	Dafür werden an der angegebenen Stelle alle vorhandenen Stellungsparameter eingesetzt, beginnend bei \$1 und voneinander durch ein Leerzeichen getrennt.

3.6 Shell Prozeduren

Als Programmierer sind Sie es gewohnt, Probleme mit Hilfe einer Programmiersprache durch Programme zu lösen. Bevor Sie ein Programm ablaufen lassen können, müssen Sie Tätigkeiten ausführen, die mit der eigentlichen Problemlösung nichts mehr zu tun haben (z.B. Programme übersetzen und binden). Das kostet Zeit, erfordert Organisationsaufwand und ist meist recht langweilig.

Die Shell bietet Ihnen Kommandos und Ablaufanweisungen, die sich wie eine Programmiersprache nutzen lassen (siehe nachfolgende Kapitel). Gegenüber "normalen" Programmiersprachen besitzt sie jedoch den großen Vorteil, daß Sie von "lästigem" Administrationsaufwand (z.B. Übersetzen) befreit sind. Sie können sich voll auf Ihre Problemlösung konzentrieren.

Zu einem Programm zusammengefaßte Kommandos bezeichnet man als Shell-Prozedur. Um eine Shell-Prozedur zu erstellen, müssen Sie einfach die gewünschte Kommandofolge in eine Datei schreiben. Anschließend können Sie diese Kommandofolge sofort ausführen.

Beispiel für eine Shell-Prozedur:

Setzen Sie sich am besten an eine Datensichtstation und probieren das folgende einfache Beispiel.

1. Schritt: Geben Sie ein

```
cat > auswahl  (Taste  drücken)
who            (Taste  drücken)
date          (Taste  drücken)
mail          (Taste  drücken)
              (Taste  drücken)
```

2. Schritt: Datei *auswahl* ausführen

Dazu muß für die Datei *auswahl* eine Ausführungserlaubnis bestehen. Die vergibt man durch das `chmod`-Kommando. Geben Sie folgendes ein:

```
chmod +x auswahl
```

Wenn Sie jetzt den Dateinamen eingeben:

```
auswahl
```

werden die in der Datei *auswahl* stehenden Kommandos ausgeführt. Am Bildschirm erscheint eine Liste aller mit dem System verbundenen Benutzer, das aktuelle Datum sowie die für Sie vorhandene Post.

Sie haben damit 3 Kommandos durch eine Eingabe gestartet.

Besteht für die Datei *auswahl* keine Ausführungserlaubnis, kann man das in ihr enthaltene Programm durch folgende Eingabe ebenfalls starten:

```
sh auswahl
```

Das `sh`-Kommando ist u.a. im Kapitel: Dateischutz erklärt. Jetzt wissen Sie, wie man grundsätzlich eine einfache Shell-Prozedur erstellt und zum Ablauf bringt. Die folgenden Kapitel enthalten detaillierte Informationen darüber, was man mit Shell-Prozeduren alles machen kann.

3.6.1 Parameter für Shell-Prozeduren

Jede Prozedur hat einen Variablen- und Parameterbereich (siehe: Bild 3-2). In diesen Bereich können Sie:

- Stellungsparameter eintragen
- Variablen eintragen, deren Werte man als Kennwortparameter aufrufen kann

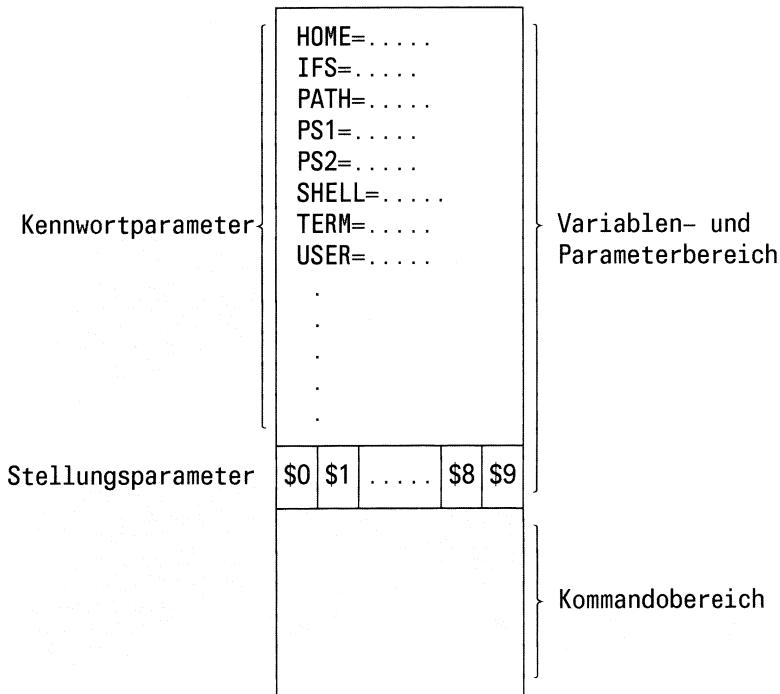


Bild 3-2 Der Variablen- und Parameterbereich für eine Prozedur

In der Login-Shell sind für jeden Benutzer Variablen mit Standardwerten vorhanden (siehe: Variablen der Shell). Einige dieser Variablen stehen mit ihren Standardwerten jeder Prozedur in ihrem Variablen- und Parameterbereich automatisch zur Verfügung. Welche das sind, können Sie mit dem `printenv`-Kommando abfragen.

Wie kann man in einer Prozedur mit Stellungsparametern arbeiten?

Wird eine Prozedur von einer Datensichtstation aus oder aus einer Prozedur heraus aufgerufen, numeriert die Shell die Angaben in der Kommandozeile. Der Name der Prozedur bekommt die Nummer 0 zugewiesen. Die erste folgende Angabe die Nummer 1, die zweite Angabe die Nummer 2 usw.. Die durchnummerierten Werte einer Kommandozeile kann man als Stellungsparameter aufrufen. Dazu muß man das `$`-Zeichen benutzen. Den Namen einer Prozedur kann man durch die Angabe `$0` aufrufen, die erste folgende Angabe durch `$1` usw.. Man kann auf bis zu 9 Stellungsparameter direkt zugreifen. Will man mit mehr als 9 Stellungsparametern arbeiten, muß man das `shift`-Kommando (siehe: Kommandos der Shell) oder `$*` benutzen. Mit `$*` liest man alle Stellungsparameter einer Kommandozeile gleichzeitig.

Nur die erste Ziffer hinter dem `$`-Zeichen wird als Stellungsparameter interpretiert. Gibt man z.B. `$10` an, interpretiert die Shell diese Angabe im Klartext so: Lese Stellungsparameter `$1` und hänge eine 0 an.

Stellungsparameter sind nicht exportierbar, d.h. in einer Prozedur gesetzte Stellungsparameter können nicht durch das `export`-Kommando für eine andere Prozedur verfügbar gemacht werden. Das ist ein Gegensatz zu Kennwortparametern. Direkt kann man mit Stellungsparametern nur in der Prozedur arbeiten, für die sie bestimmt sind.

Beispiele, wie Sie beim Aufruf einer Prozedur über die Kommandozeile Stellungsparameter übergeben können.

Beispiel 1: Angenommen, Sie wollen von einer Datenstation aus, an eine Prozedur einen Stellungsparameter übergeben. Name der Datei in der die Prozedur steht: *kunde*. Inhalt der Datei:

```
echo Der Name des Kunden ist: $1
```

Wenn Sie diese Prozedur durch folgendes Kommando aufrufen:

```
kunde Huber
```

erscheint am Bildschirm folgende Ausgabe:

```
Der Name des Kunden ist: Huber
```

Für den in der Prozedurdatei *kunde* angegebenen Stellungsparameter `$1`, wurde beim Aufrufen der Prozedur der in der Kommandozeile in Position 1 stehende Parameter (d.h. Huber) eingesetzt und durch das `echo`-Kommando ausgegeben.

Beispiel 2: Angenommen, Sie wollen aus einer Prozedur heraus eine andere Prozedur aufrufen und dabei einen Stellungsparameter übergeben. Namen der Dateien, in denen die Prozeduren stehen:

Aufrufende Prozedur: *kunde*

Aufgerufene Prozedur: *ausgabe*

Inhalt von: *kunde*

echo Der Name des Kunden ist: \$1

ausgabe 21479

Inhalt von: *ausgabe*

echo Seine Kundennummer ist: \$1

Wenn Sie jetzt folgende Eingabe machen:

kunde Meier

erscheint am Bildschirm folgende Ausgabe:

Der Name des Kunden ist: Meier

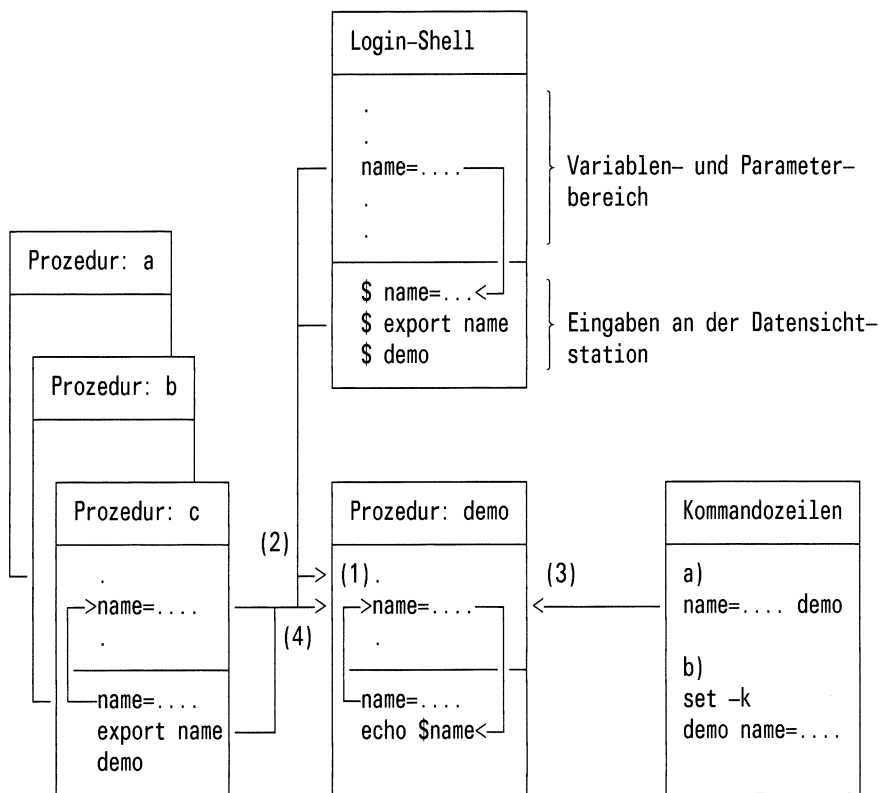
Seine Kundennummer ist: 21479

Was ist passiert? An die Prozedurdatei *kunde* wurde aus der an der Daten-sichtstation eingegebene Kommandozeile der Stellungsparameter \$1 übergeben (d.H. Meier) und durch das echo-Kommando auf den Bildschirm geschrieben. Durch die daran anschließende Kommandozeile wird die Prozedur *ausgabe* aufgerufen. Diese Kommandozeile enthält zusätzlich noch die Angabe: 21479, die in der Prozedur *ausgabe* durch den Stellungsparameter \$1 gelesen und durch das echo-Kommando ausgegeben wird.

Wie kann man in einer Prozedur mit Kennwortparametern arbeiten?

Einer Prozedur stehen beim Aufruf alle Variablen zur Verfügung, die in ihrem Variablen- und Parameterbereich eingetragen sind. In den Variablen- und Parameterbereich einer Prozedur werden automatisch alle Variablen eingetragen, die: (siehe auch: Bild 3-3)

- (1) in einer Prozedur direkt definiert werden
- (2) in der Login-Shell definiert und exportiert wurden
- (3) beim Aufruf der Prozedur über die Kommandozeile an die Prozedur übergeben wurden
- (4) in einer Aufrufhierarchie von Prozeduren (Prozedur1 ruft Prozedur2 auf, Prozedur2 ruft Prozedur3 auf usw.) in einer "unterhalb" der aktuellen Prozedur abgelaufenen Prozedur definiert und exportiert wurden.



a) und b) kann man in einer Prozedur oder an einer Datensichtstation angeben.

Bild 3-3 Eintrag von Variablen in den Variablen- und Parameterbereich

Den Wert einer Variablen kann man als Kennwortparameter lesen. Dazu muß man den Variablennamen mit den \$-Zeichen aufrufen. Auf den folgenden Seiten gibt es dazu einige Beispiele.

Wie kann man in einer Prozedur eine Variable definieren und mit ihrem Wert als Kennwortparameter arbeiten?

Ein Beispiel. In der Prozedur *register* soll die Variable *name=SINIX* definiert werden (siehe auch: Bild 3-4).

Prozedurdatei: *register*
Inhalt: *name=SINIX*
 echo \$name
Aufruf: *sh register*
Ausgabe: *SINIX*

Was wurde gemacht? In den Variablen- und Parameterbereich der Prozedur *register* wurde die Variable *name=SINIX* eingetragen. Durch den Kennwortparameter *\$name* wurde ihr Wert gelesen und durch das *echo*-Kommando am Bildschirm ausgegeben.

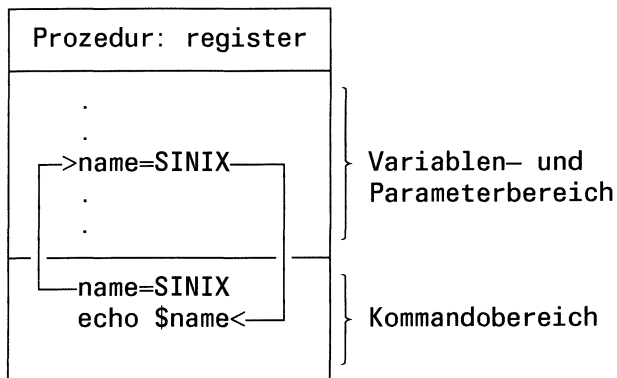


Bild 3-4 Definition einer Variablen in einer Prozedur

Wie kann man in einer Prozedur mit Variablen arbeiten, die in der Login-Shell definiert sind?

In der Login-Shell definieren Sie eine Variable, indem Sie an der Datensichtstation z.B. folgendes eingeben:

```
name = eric
```

Jetzt haben Sie eine Variable *name = eric* definiert. Um mit dieser Variable auch außerhalb der Login-Shell arbeiten zu können, müssen Sie sie exportieren (siehe auch: *export*-Kommando). Geben Sie folgendes ein:

```
export name
```

Jetzt steht jeder Prozedur, die Sie aufrufen, in ihrem Variablen- und Parameterbereich die Variable *name* zur Verfügung. Angenommen, Sie haben folgende Prozedur:

```
Prozedurdatei: kartei  
  Inhalt: echo $name  
  Aufruf: kartei  
  Ausgabe: eric
```

Was wurde gemacht? Beim Aufruf der Prozedur *kartei* wurde in ihren Variablen- und Parameterbereich die in der Login-Shell definierte und exportierte Variable *name = eric* eingetragen. Durch den Kennwortparameter *\$name* wurde ihr Wert gelesen und durch das *echo*-Kommando am Bildschirm ausgegeben (siehe auch: Bild 3-5).

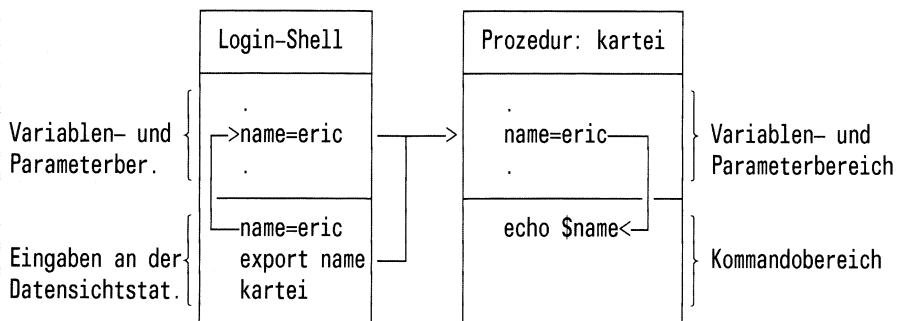


Bild 3-5 Übergabe von Variablen der Login-Shell an eine Prozedur

Wie übergibt man Variablen über die Kommandozeile an eine Prozedur?

Eine Prozedur wird durch eine Kommandozeile aufgerufen, die Sie entweder im Dialog von der Datensichtstation aus eingeben oder die aus einer Prozedur heraus ausgeführt wird. Eine Kommandozeile kann Variablen enthalten, die in den Variablen- und Parameterbereich der aufgerufenen Prozedur geschrieben werden sollen. Das können Sie auf zwei Arten erreichen. Ein Beispiel. Angenommen, Sie haben folgende Prozedur: (siehe auch: Bild 3-6)

```
Prozedurdatei: archiv
             Inhalt: echo $kunde
```

Wenn Sie die Prozedurdatei *archiv* so aufrufen:

```
kunde = miller archiv
```

erhalten Sie als Ausgabe am Bildschirm: *miller*.

Wenn Sie die Prozedurdatei *archiv* so aufrufen:

```
archiv kunde = miller
```

erhalten Sie am Bildschirm keine Ausgabe. Warum?

Was wurde gemacht? Beim ersten Aufruf beginnt die Kommandozeile mit der Angabe der Variablen *kunde = miller* gefolgt vom Namen der Prozedur *archiv*. Bei dieser Eingabeform wird die Variable automatisch exportiert, d.h. in den Variablen- und Parameterbereich der aufgerufenen Prozedur *archiv* eingetragen. Der Kennwortparameter *\$kunde* kann den Wert lesen. Das *echo*-Kommando gibt ihn am Bildschirm aus.

Beim zweiten Aufruf wurde als erstes der Name der Prozedur (*archiv*) eingegeben und anschließend die Variable (*kunde = miller*). Eine solche Eingabeform "versteht" die Shell nicht. Man muß der Shell mitteilen, wie sie diese Eingabe "zu verstehen hat". Dazu muß man das *set*-Kommando benutzen. Geben Sie ein:

```
set -k
archiv kunde = miller
```

erhalten Sie als Ausgabe am Bildschirm: *miller*

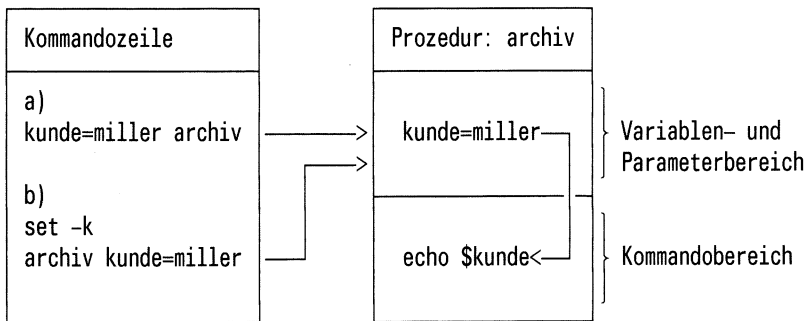
Das Kommando: set -k hat zwei Effekte:

- Alle Angaben in der Kommandozeile, die ein Gleichheitszeichen enthalten, interpretiert die Shell als Variablen.
- Alle Variablen, die Sie in einer Kommandozeile angeben, werden automatisch exportiert. Und zwar für alle Prozeduren, die direkt oder indirekt durch diese Kommandozeile ausgeführt werden.

Aber beachten Sie beim set-Kommando:

Die Wirkung des set-Kommandos ist auf die Shell begrenzt, in der es angegeben wird. Was bedeutet das? Geben Sie das set-Kommando im Dialog von der Datensichtstation aus ein, bezieht sich seine Wirkung nur auf Kommandozeilen, die Sie direkt von der Datensichtstation aus eingeben. Wenn Sie innerhalb einer Prozedur mit Kommandozeilen arbeiten wollen, müssen Sie in die entsprechende Prozedur ein set-Kommando schreiben. Ein an der Datensichtstation eingegebenes set-Kommando, hat also keine Wirkung auf Kommandozeilen innerhalb einer Prozedur.

Ein im Dialog an einer Datensichtstation eingegebenes set-Kommando gilt nur für die Dauer einer Sitzung. Anschließend ist seine Wirkung aufgehoben. Nach einem erneuten Login muß man deshalb das set-Kommando neu eingeben, wenn man mit Kennwortparametern arbeiten will. Schreibt man das set-Kommando in die Datei *.profile*, wird es bei jedem Login automatisch ausgeführt.



a) und b) kann man in einer Prozedur oder im Dialog an einer Datensichtstation angeben.

Bild 3-6 Übergabe von Variablen über die Kommandozeile

Umwandeln von Kennwort-in Stellungsparameter

Man kann Kennwortparameter in Stellungsparameter umwandeln. Dazu muß man das `set`-Kommando benutzen. Das kann sehr nützlich sein, wenn man aus dem Wert einer Variablen etwas herausfiltern möchte. Ein einfaches Beispiel. Angenommen, Sie wollen aus der Ausgabe des `datum`-Kommandos nur die Uhrzeit ausgegeben bekommen. Über eine Shell-Prozedur (`zeit`) können Sie das wie folgt erreichen:

```
Inhalt von zeit: datum >x  
                  exec <x  
                  read xyz  
                  set $xyz  
                  echo $3
```

Was passiert, wenn Sie die Shell-Prozedur `zeit` aufrufen? (Eingabe: `sh zeit`). Als erstes wird das Ergebnis des `datum`-Kommandos in die Datei `x` geschrieben. Es könnte z.B. so aussehen: `Di 5.Jun.1984, 12:37:02 MEZ`. Anschließend wird durch das `exec`-Kommando die Standard-Eingabe auf die Datei `x` umgelenkt. Von dort liest das `read`-Kommando und weist das Ergebnis der Variablen `xyz` als Wert zu. Das `set`-Kommando macht diesen Wert für Stellungsparameter zugreifbar. Mit dem `echo`-Kommando wird der Stellungsparameter `$3` gelesen und ausgegeben. Am Bildschirm erscheint die gewünschte Uhrzeit: `12:37:02`.

Parameter vordefinieren

- Standardwerte setzen: `${parameter-wort}` oder `${parameter:-wort}`

Für die Angabe - gilt:

Ist für **parameter** beim Aufruf der Prozedur ein Wert gesetzt, wird er eingesetzt, wenn nicht, wird das angegebene **wort** eingesetzt.

Geben Sie :- an, wird zusätzlich abgefragt, ob `parameter=""` ist.

Beispiel 1 (mit Stellungsparameter)

Inhalt der Datei `test1`: `echo Alex ist: ${1-klein}`

Aufruf : `test1`

Ausgabe : `Alex ist: klein`

Aufruf : `test1 jung`

Ausgabe : `Alex ist: jung`

Beispiel 2 (mit Kennwortparameter)

Inhalt der Datei `test2`: `echo Das Auto ist: ${farbe-schwarz}`

Aufruf : `test2`

Ausgabe : `Das Auto ist: schwarz`

Aufruf : `test2 farbe=rot`

Ausgabe : `Das Auto ist: rot`

- Standardwerte setzen: `${parameter=wort}` oder `${parameter:=wort}`

Der Wert von `wort` wird eingesetzt wenn:

- bei der Angabe = kein parameter angegeben wird
- bei der Angabe := kein parameter oder `parameter=""` angegeben wird

Beispiel

Inhalt der Datei `test3`: `echo Der Kunde ist: ${a=Neu}`
`echo Die Kundennummer ist: $a`

Aufruf : `test3`

Ausgabe : `Der Kunde ist: Neu`
`Die Kundennummer ist: Neu`

Aufruf : `test3 a=21479`

Ausgabe : `Der Kunde ist: 21479`
`Die Kundennummer ist: 21479`

- Prüfen, ob ein Wert gesetzt ist: `${parameter?wort}` oder `${parameter:?wort}`

Für die Angabe `? gilt:`

Der für **parameter** gesetzte Wert wird ersetzt. Ist kein Wert gesetzt, wird **wort** ausgedruckt und die Prozedur beendet. Geben Sie `:?` an, wird zusätzlich abgefragt, ob `parameter = ""` ist. Wird kein **wort** angegeben bekommt man eine Standardmeldung ausgegeben.

Beispiel 1 (mit Stellungsparameter)

```
Inhalt der Datei test4: echo Flugzeuge fliegen: ${1?Abgebrochen}
Aufruf                  : test4 hoch
Ausgabe                 : Flugzeuge fliegen: hoch

Aufruf                  : test4
Ausgabe                 : test4: 1: Abgebrochen
```

Beispiel 2 (mit Kennwortparameter)

```
Inhalt der Datei test5: echo Flugzeuge fliegen: ${speed?Abbruch}
Aufruf                  : test5 speed=schnell
Ausgabe                 : Flugzeuge fliegen: schnell

Aufruf                  : test5
Ausgabe                 : test5: speed: Abbruch
```

- Parameter durch feste Werte ersetzen: `${parameter+ wort}` oder `${parameter:+ wort}`

Für die Angabe `+ gilt:`

Ist ein **parameter** gesetzt, wird der Wert von **wort** eingesetzt. Ist kein **parameter** gesetzt, passiert garnichts. Geben Sie `:+` an, wird zusätzlich abgefragt, ob `parameter = ""` ist.

Beispiel

```
Inhalt der Datei test6:
    echo Diese Datei ist ein: ${name+Dateiverzeichnis}
    echo Sie haben für das Dateiverzeichnis: $name keine Leseerlaubnis

Aufruf : test6 name=Kundendatei
Ausgabe: Diese Datei ist ein: Dateiverzeichnis
        Sie haben für das Dateiverzeichnis: Kundendatei keine Leseerlaubnis

Aufruf : test6
Ausgabe: Diese Datei ist ein:
        Sie haben für das Dateiverzeichnis.....keine Leseerlaubnis
```

3.6.2 Shell-Prozeduren und Prozesse

Dieser Abschnitt beschreibt Ihnen die Auswirkungen des SINIX-Prozeßkonzepts, wenn Sie mit der Shell arbeiten, d.h. wenn Sie Kommandos eingeben oder mit Shell-Prozeduren arbeiten.

Ein laufendes SINIX-System (Ein- oder Mehrplatz), an dem Benutzer arbeiten, besteht aus einer baumartigen Struktur von Prozessen (siehe Bild 3-7).

Ein- oder Mehrplatzsysteme haben ein prinzipiell gleiches Prozeßkonzept. Einziger Unterschied: Da an einem Einplatzsystem nur ein Benutzer arbeiten kann, gibt es nur eine Login-Shell und damit auch nur einen Prozeßbaum.

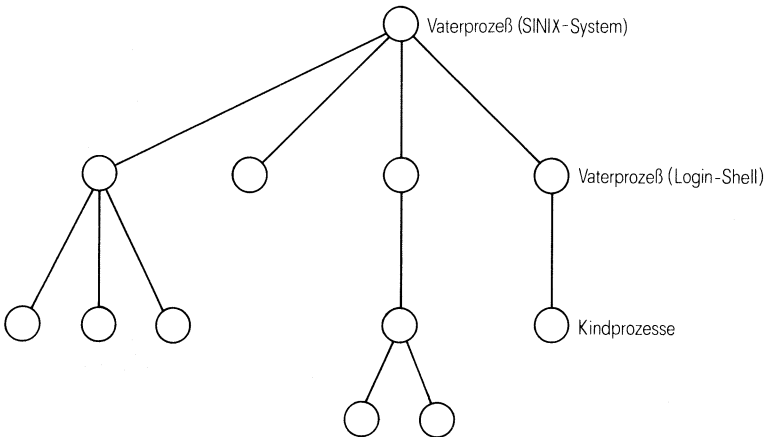


Bild 3-7 SINIX-Prozeßstruktur

Beim Starten des Systems wird ein Vaterprozeß mit der Prozeßnummer 1 (PID 1) erzeugt. Er ist der Ursprung für alle weiteren Prozesse im SINIX-System. Direkt an ihn angeschlossen ist für jeden Benutzer ein Kindprozeß, in dem seine Login-Shell abläuft. Alle Eingaben, die ein Benutzer macht, nimmt die Login-Shell in Empfang und führt entsprechende Aktionen aus. Bis auf wenige Ausnahmen (siehe: Kommandos der Shell), wird für jedes eingegebene Kommando ein eigener Prozeß erzeugt. Der Prozeß in dem die Login-Shell abläuft, wird damit selbst zu einem Vaterprozeß.

Grundsätzlich gilt: Jeder Prozeß kann einen neuen Prozeß erzeugen. Solange ein Prozeß keinen anderen Prozeß erzeugt, ist er der Kindprozeß eines Vaterprozesses. Sobald ein Kindprozeß einen anderen Prozeß erzeugt, wird er selbst vom Kind- zum Vaterprozeß.

Die Vater-Kind-Prozeßhierarchie müssen Sie beachten, wenn Sie in Shell-Prozeduren Variablen weiterreichen wollen. Warum? Jeder Benutzer kann für seine Login-Shell (= Vaterprozeß) ein Umfeld definieren, d.h. Variablen festlegen, mit denen die Login-Shell arbeiten soll (siehe auch: Kapitel 3.6). Dieses Umfeld ist aber grundsätzlich erst einmal nur der Login-Shell bekannt. Soll in einem Kindprozeß mit Variablen aus dem Umfeld des Vaterprozesses (= Login-Shell) gearbeitet werden, muß dieses Umfeld an den Kindprozeß exportiert werden. Dazu muß man das `export`-Kommando benutzen (siehe: Kapitel 3.8.7). Dieser Mechanismus gilt sinngemäß auch für Prozesse, die von einem Kindprozeß aufgerufen werden.

Beachten Sie: Variablen lassen sich durch Exportieren nur vom Vater- an einen Kindprozeß weiterreichen. Will man vom Kind- an seinen Vaterprozeß Variablen zurückgeben, geht das nur mit "Tricks".

Ein Beispiel dafür, wie man Variablen "zurückgeben" kann

Aus der Datei */etc/passwd* sollen die Benutzerkennungen herausgefiltert und auf den Bildschirm geschrieben werden. Dazu soll aus einer Prozedur (Name: *start*) eine andere Prozedur (Name: *filter*) aufgerufen werden, in der die Benutzerkennungen herausgefiltert werden. Die Benutzerkennungen sollen als Variablen an die aufrufende Prozedur zurückgegeben und von ihr ausgegeben werden.

```
Inhalt von start: xyz=`filter`  
                  echo $xyz
```

```
Inhalt von filter: IFS=:  
                    exec </etc/passwd  
                    while read lines  
                    do    set $lines  
                        echo $1  
                    done
```

Was passiert, wenn man die Prozedur *start* aufruft? (Eingabe: `sh start`) Die Prozedur *filter* wird gestartet und ihr Ergebnis wird der Variablen *xyz* zugewiesen. Das sind die gesuchten Benutzerkennungen. Was passiert in der Prozedur *filter*? Als erstes wird durch die Variable *IFS* das Feldtrennzeichen umdefiniert auf das Zeichen: `:`. Das ist notwendig, um später auf die einzelnen Elemente jeder Zeile in der Datei: */etc/passwd* zugreifen zu können. Durch das `exec`-Kommando wird die Standard-Eingabe auf die Datei */etc/passwd* umgeleitet. Das in der `while`-Schleife angegebene `read`-Kommando liest zeilenweise von der Standard-Eingabe, d.h. aus der Datei */etc/passwd*, und weist das Ergebnis der Variablen *lines* als Wert zu. Das `set`-Kommando macht diese Werte so verfügbar, daß man auf sie über Stellungparameter zugreifen kann. Das `echo`-Kommando liest mit dem Stellungparameter `$1` das erste Element. Das ist die gesuchte Benutzerkennung, die als Ergebnis der Prozedur *filter* an die Prozedur *start* übergeben wird.

Shell-Prozeduren im Hintergrund ablaufen lassen

Wenn man nach dem Aufruf einer Prozedur nicht auf die Beendigung der Prozedur warten will, kann man sie im Hintergrund ablaufen lassen. Dazu muß man den Prozeduraufruf mit folgendem Zeichen abschliessen: `&` . Jetzt bekommt man unmittelbar nach dem Aufruf der Prozedur, die Prozeßnummer des Prozesses ausgegeben, in dem die Prozedur im Hintergrund abläuft.

Anschließend erscheint am Bildschirm das Eingabeaufforderungszeichen. Man kann sofort weitere Kommandos eingeben und braucht nicht auf die Beendigung des Hintergrundprozesses warten. Die ausgegebene Prozeßnummer braucht man, um mit dem `kill`-Kommando den Hintergrundprozeß eventl. abbrechen zu können (siehe auch: `!`, Variablen für die Shell).

3.7 Ablaufanweisungen für die Shell

Mit den Anweisungen: `break`, `continue`, `case`, `for`, `if` und `while` können Sie eine Shell-Prozedur strukturieren. Alle Ablaufanweisungen können Sie auch im Dialog von der Datensichtstation aus eingeben. Wie Sie innerhalb einer Ablaufanweisung eine Kommando-Liste angeben können, ist im Kapitel 3.1 beschrieben.

3.7.1 Schleifen steuern: Die break- und continue-Anweisung

Formate der Anweisungen:

```
break[n]      continue [n]
```

Mit der break-Anweisung kann man eine **while**- oder **for**-Schleife beenden. Mit der continue-Anweisung gibt man an, daß die Schleife noch einmal durchlaufen werden soll. Beide Anweisungen funktionieren nur, wenn sie zwischen **do** und **done** stehen.

Eine break-Anweisung beendet die Schleife, in der sie angegeben wird. Die Prozedur wird hinter dem nächsten, nicht erreichten done fortgesetzt. Will man n-Schleifen beenden, muß man eingeben: break n.

Eine continue-Anweisung führt noch einmal die Schleife aus, in der sie gegeben wurde, d.h. die Verarbeitung geht an den Anfang einer for-, while- oder until-Schleife zurück. Bei verschachtelten Schleifen kann man auch bei der n-ten einschließenden Schleife fortsetzen, dazu muß man angeben: continue n.

Beispiel für die break- und continue-Anweisung:

Eine Prozedur soll ein gewünschtes Dateiverzeichnis nach leeren Dateien durchsuchen. Sobald die erste leere Datei gefunden wird, soll eine Meldung am Bildschirm erscheinen. Name der Prozedurdatei *konti*. Der Inhalt sieht so aus:

```
letzte='ls |l | tail -1'      #letzte Datei im Dateiverzeichnis merken
cd $1
for x in *
do
if test -d $x
then if test $x = $letzte
    then echo "Das Dateiverzeichnis $1 enthält keine leeren Dateien"
    else continue
    fi
elif test ! -s $x
then echo "Das Dateiverzeichnis $1 enthält leere Dateien"
break
else if test $x = $letzte
#ist das die letzte Datei im Dateiverzeichnis?
    then echo "Das Dateiverzeichnis $1 enthält keine leeren Dateien"
fi
fi
done
```

3.7.2 Abfragen und Verzweigen: Die case-Anweisung

Mit der case-Anweisung kann man eine Eingabe prüfen und entsprechend dem Ergebnis etwas ausführen. Die case-Anweisung hat das allgemeine Format:

```
case name in  
muster) Kommando-Liste;;  
.  
.  
.  
.  
muster) Kommando-Liste  
esac
```

Die unterstrichenen Begriffe sind Konstanten der case-Anweisung.

Die Shell vergleicht den für *name* stehenden Wert mit den unter *muster* stehenden Werten.

Dafür gelten dieselben Konventionen wie für Dateinamen. Bei einer Übereinstimmung zwischen *name* und *muster* wird eine hinter *muster* stehende Kommando-Liste ausgeführt. Bis auf die letzte Kommando-Liste muß jede andere mit den Zeichen `;;` abgeschlossen sein. Es kann immer nur ein *muster* ausgeführt werden. Die *muster* werden der Reihe nach geprüft.

Beachten Sie: Wird das erste *muster* mit `*` angegeben, wird kein weiteres *muster* mehr geprüft. Es ist möglich, vor eine Kommando-Liste mehrere *muster* zu setzen. Mehrere *muster* müssen durch `|` voneinander getrennt sein.

Beispiel für die case-Anweisung:

Es soll entsprechend einer bestimmten Eingabe jeweils ein Text ausgegeben werden. Die Prozedurdatei *meldung* soll dazu folgendes enthalten:

```
case $1 in
    alex) echo "alex ist klein";;
    pascal) echo "pascal ist jung";;
    peter|werner) echo "$1"ist klug";;
    *) echo "Diese Parametereingabe ist nicht erlaubt"
esac
```

Eingabe : meldung alex
Ausgabe : alex ist klein

Eingabe : meldung pascal
Ausgabe : pascal ist jung

Eingabe : meldung peter oder meldung werner
Ausgabe : peter ist klug oder werner ist klug
Jede andere Eingabe z.B. : meldung auto
erzeugt die Ausgabe : Diese Parametereingabe ist nicht erlaubt

3.7.3 Liste in Schleifen abarbeiten: Die for-Anweisung

Mit der for-Anweisung kann man für jeden Eintrag in einer Liste, die gleiche Kommando-Liste ausführen. Mit der for-Anweisung kann man z.B. mehrere unterschiedliche Dateien mit dem gleichen Kommando bearbeiten. Die for-Anweisung hat das allgemeine Format:

```
for name [in Liste]
do Kommando-Liste
done
```

Die unterstrichenen Begriffe sind Konstanten der for-Anweisung. Um erkannt zu werden, müssen sie entweder (außer in) am Beginn einer Zeile oder nach einem `;`-Zeichen stehen.

name Name einer Variablen, die erzeugt werden soll (z.B. `x` oder `eric`). Dieser Variablen werden nacheinander die Werte zugewiesen, die in der hinter in angegebenen Liste stehen. Mit diesen Werten kann man in der Kommando-Liste als Kennwortparameter arbeiten.

Liste Dafür kann man eine Liste von Werten angeben, die durch Leerzeichen voneinander getrennt sein müssen. Für jeden Wert in dieser Liste wird die hinter do angegebene Kommando-Liste ausgeführt. Fehlt die Liste-Angabe, wird `$@` als Standard angenommen, d.h. es werden alle vorhandenen Stellungsparameter (`$1`, `$2`, `$3`, usw.) eingesetzt.

Kommando-Liste

Dafür kann man eine abzuarbeitende Kommando-Liste angeben.

done Damit muß jede for-Schleife abgeschlossen sein.

Beispiele für die for-Anweisung:

Angenommen, Sie haben 3 Dateien mit Kundennamen *kunde1*, *kunde2*, *kunde3*. Durch eine for-Schleife können Sie alle 3 Dateien nach gleichen Kundennamen durchsuchen. Erstellen Sie dazu als erstes eine Prozedurdatei, z.B. *kundenname*, mit folgendem Inhalt:

```
for x in kunde1 kunde2 kunde3
do grep $1 $x
done
```

Wenn Sie jetzt folgende Eingabe machen:
kundenname huber

passiert im Klartext folgendes: Der Variablen *x* wird der erste Wert aus der hinter in folgenden Liste zugewiesen, d.h. der Dateiname *kunde1*. Anschliessend wird das nach do angegebene Kommando ausgeführt. Als erstes ersetzt die Shell die hinter dem *grep*-Kommando stehenden Parameter. Für *\$1* wird der erste Stellungsparameter aus der Kommandozeile eingesetzt (d.h. *huber*) und für *\$x* wird der erste Wert der der Variablen *x* zugewiesen wurde eingesetzt (d.h. der Dateiname *kunde1*).

Das *grep*-Kommando durchsucht jetzt die Datei *kunde1* nach dem Kundennamen *huber*. Anschließend wird die *for*-Schleife erneut durchlaufen. Der Variablen *x* wird als Wert der Dateiname *kunde2* zugewiesen. Das *grep*-Kommando übernimmt mit *\$x* diesen Wert, d.h. die Datei *kunde2* wird nach dem Kundennamen *huber* durchsucht. das gleiche passiert mit der Datei *kunde3*. Am Bildschirm bekommen Sie eine Auflistung darüber, welche der Dateien den Kundennamen *huber* enthält.

Das gleiche Ergebnis können Sie mit einer anderen Schreibweise einer *for*-Anweisung erreichen:

```
for eingabe
do grep $eingabe kunde1 kunde2 kunde3
done
```

Was ist daran anders? Die *for*-Zeile enthält keine Liste-Angabe. Dafür wird als Standardwert *\$@* eingesetzt. Alle in der Kommandozeile angegebenen Stellungsparameter werden jetzt "unsichtbar" an diese Stelle gesetzt.

Damit sieht die *for*-Zeile in diesem Fall eigentlich so aus:

```
for eingabe in $1 $2
```

Die Eingabe:
\$ kundenname huber meier

bewirkt, daß der hinter *for* stehende Name *eingabe* auf *huber* gesetzt wird und das *grep*-Kommando *huber* übernimmt, um die 3 Dateien *kunde1*, *kunde2* und *kunde3* danach zu durchsuchen. Das Gleiche passiert mit dem Kundennamen *meier*.

Am Bildschirm wird aufgelistet, welche Dateien die Kundennamen *huber* oder *meier* enthalten.

3.7.4 Abfragen und Kommando ausführen: Die if-Anweisung

Mit der if-Anweisung kann man eine Kommando-Liste abarbeiten und entsprechend dem Ergebnis in zwei alternative Richtungen verzweigen. Wie kann das Ergebnis aussehen? Jedes Kommando meldet nach seiner Ausführung einen Ende-Status zurück, den die if-Anweisung abfragt. Der Ende-Status kann entweder null oder ungleich null sein. Null bedeutet *wahr*, ungleich null bedeutet *unwahr*. Ist der Ende-Status gleich null, wird in die folgende then-Abfrage verzweigt. Ist der Ende-Status ungleich null wird zum nächsten elif verzweigt. Mit elif kann man erneut eine Bedingung abfragen und entsprechend dem Ergebnis zum nächsten then oder else verzweigen. Das if-Kommando hat das allgemeine Format:

```
if Kommando-Liste  
then Kommando-Liste  
elif Kommando-Liste  
then Kommando-Liste  
else Kommando-Liste  
fi
```

Die unterstrichenen Begriffe sind Konstanten der if-Anweisung. Um erkannt zu werden, müssen sie entweder am Beginn einer Zeile oder nach einem ;-Zeichen stehen.

Beispiel für die if-Anweisung:

Es soll abgefragt werden, ob ein Datei- oder ein Dateiverzeichnisname eingegeben wurde.

Bei Eingabe eines Dateinamens soll der Inhalt der Datei ausgegeben werden. Bei Eingabe eines Dateiverzeichnisnamens sollen die Einträge des Dateiverzeichnisses ausgegeben werden. Wird weder ein Datei- noch ein Dateiverzeichnisname eingegeben, soll eine Meldung am Bildschirm erscheinen. Name der Prozedurdatei *check*

```
Inhalt: if test -f "$1"
        then cat $1
        elif test -d "$1"
        then (cd $1;ls -l)
        else echo $1 "ist weder ein Datei- noch ein Dateiverzeichnisname"
        fi
```

Angenommen, es gibt eine Datei mit dem Namen *richter* und ein Dateiverzeichnis mit dem Namen *kundendatei*, dann kann man mit der Prozedurdatei *check* wie folgt arbeiten:

Eingabe: `check richter`

Ausgabe: Inhalt der Datei richter

Eingabe: `check kundendatei`

Ausgabe: Alle Einträge des Dateiverzeichnisses kundendatei

Eingabe: `check`

Ausgabe: Alle Einträge des HOME-Dateiverzeichnisses. Warum? Weil dann an das in der Prozedurdatei stehende `cd`-Kommando kein Parameter übergeben wird und es dann automatisch das HOME-Dateiverzeichnis annimmt (siehe auch: Kapitel 3.5, HOME-Variable).

3.7.5 Schleife mit Abbruchbedingung: Die while- und until-Anweisung

Mit der while-Anweisung kann man den Ende-Status eines Kommandos oder einer Kommando-Liste auf null (d.h. *wahr*) und mit der until-Anweisung auf ungleich null (d.h. *unwahr*) abfragen. Anschließend kann man eine zweite Kommando-Liste solange ausführen, wie die Bedingung erfüllt ist. Der Ende-Status einer while- oder until-Anweisung ist der des letzten Kommandos in der zweiten Kommando-Liste.

Die **while-Anweisung** fragt den Ende-Status null ab und hat das allgemeine Format:

```
while Kommando-Liste  
do Kommando-Liste  
done
```

Die unterstrichenen Begriffe sind Konstanten der while-Anweisung. Um erkannt zu werden, müssen sie entweder am Beginn einer Zeile oder nach einem ;-Zeichen stehen.

Die Kommandos in der nach while stehenden Kommando-Liste werden ausgeführt. Solange die hinter while stehende Kommando-Liste den Ende-Status null (d.h. *wahr*) meldet, wird die hinter do stehende Kommando-Liste ausgeführt. Diese Kommando-Liste wird solange in einer Schleife ausgeführt, wie die erste Kommando-Liste den Ende-Status null meldet.

Beispiel für die while-Anweisung:

Es soll die Datei *datei1* in die Dateien *datei2* *datei3* und *datei4* kopiert werden. Die angenommene Prozedurdatei *Kopiere* muß folgenden Inhalt haben:

```
while test "$2" != ""
do    cp $1 $2
      shift
done
```

Durch die Eingabe:

```
$ Kopiere datei1 datei2 datei3 datei4
```

wird nun die *datei1* in die Dateien *datei2*, *datei3* und *datei4* kopiert. Diese while-Anweisung liest sich im Klartext so: Solange ("while") in der Kommandozeile unter Stellungsparameter \$2 ein Wert steht, führe die hinter do angegebene Kommando-Liste aus. Der Stellungsparameter \$2 wird durch das test-Kommando abgefragt. Solange der Stellungsparameter \$2 ungleich null ist, ist die im test-Kommando abgefragte Bedingung erfüllt, d.h. das test-Kommando hat den Ende-Status null. Da die while-Anweisung auf den Ende-Status null abfragt, wird die hinter do stehende Kommando-Liste ausgeführt. Nach dem ersten Lesen der Kommandozeile steht für \$1 der Dateiname *datei1* und für \$2 der Dateiname *datei2*. Die while-Bedingung ist erfüllt, also kopiert das cp-Kommando die *datei1* in *datei2*. Anschließend verschiebt das shift-Kommando die Werte in der Kommandozeile.

Unter dem Stellungsparameter \$1 steht dann *datei2*, unter \$2 *datei3* und unter \$3 *datei4*. Die while-Bedingung ist immer noch erfüllt, denn der Wert unter \$2 ist immer noch ungleich null. Also kopiert das cp-Kommando \$1 nach \$2. In diesem Fall *datei2* nach *datei3*. Da in *datei2* beim ersten Durchlauf der Inhalt von *datei1* kopiert wurde, steht in *datei3* ebenfalls der Inhalt von *datei1*. Dieser Ablauf wiederholt sich solange, bis das test-Kommando unter \$2 null liest, die Abfragebedingung deshalb nicht mehr erfüllt ist und es den Ende-Status eins (d.h. *unwahr*) meldet.

Die **until-Anweisung** fragt den Ende-Status auf ungleich null ab und hat das allgemeine Format:

```
until Kommando-Liste  
do Kommando-Liste  
done
```

Die unterstrichenen Begriffe sind Konstanten der until-Anweisung. Um erkannt zu werden, müssen sie entweder am Beginn einer Zeile oder nach einem ;-Zeichen stehen.

Die Kommandos in der nach until stehenden Kommando-Liste werden ausgeführt. Solange ("until") diese Kommando-Liste einen Ende-Status ungleich null meldet, wird die hinter do stehende Kommando-Liste ausgeführt. Die hinter do stehende Kommando-Liste wird solange ausgeführt, wie die erste Kommando-Liste einen Ende-Status ungleich null meldet.

Beispiel für die until-Anweisung:

Angenommen, Sie wollen aus einem Dateiverzeichnis alle leeren Dateien löschen. Vorausgesetzt, Sie sind der Eigentümer des Dateiverzeichnisses und der entsprechenden Dateien, dann können Sie das durch eine Prozedur machen, die eine until-Anweisung enthält. Die Prozedurdatei *loesche* muß dann folgendes enthalten:

```
for x in *
do until test -s "$x"
do    rm $x
    echo Die Datei $x wurde geloescht
    break
done
done
```

Was passiert, wenn Sie jetzt die Prozedurdatei *loesche* aufrufen?

Eingabe: sh loesche

Die for-Anweisung stellt der until-Anweisung jeden Eintrag des aktuellen Dateiverzeichnisses zur Abfrage zur Verfügung. Das test-Kommando hinter der until-Anweisung fragt ab: "Die Datei existiert und ist nicht leer". Handelt es sich um eine leere Datei, wird der Ende-Status 1 gemeldet (d.h. *unwahr*). Die Abfragebedingung für die until-Anweisung ist erfüllt und die folgenden Kommandos werden ausgeführt. Das rm-Kommando löscht die durch \$x gelesene Datei. Das echo-Kommando gibt eine Meldung aus, welche Datei gelöscht wurde und das break-Kommando beendet die until-Schleife. Durch das for-Kommando wird der nächste Eintrag gelesen, an die until-Anweisung zur Abfrage übergeben usw..

3.8 Kommandos der Shell

Neben den im Kapitel 6 beschriebenen Kommandos stellt die Shell noch folgende "eingebaute" Kommandos und Anweisungen zur Verfügung:

break	case	cd	continue	display
eval	exec	exit	export	for
if	newgrp	read	readonly	set
shift	times	trap	umask	until
wait	while	.	: und #	()
{	[...]			

Der Unterschied zwischen "eingebauten" Kommandos bzw. Anweisungen und allen anderen Kommandos ist der, daß die Shell "eingebaute" Kommandos und Anweisungen selbst interpretiert und verarbeitet. Alle anderen Kommandos führt die Shell ungeprüft aus. Für "eingebaute" Kommandos oder Anweisungen erzeugt die Shell keinen eigenen Prozeß, deshalb sind sie schneller. Diese Kommandos geben Sie genauso ein, wie alle anderen Kommandos. Mit den Anweisungen: break, continue, case, for, if und while kann man eine Shell-Prozedur steuern. Diese Anweisungen sind im Kapitel 3.7 beschrieben. Bis auf das cd- und newgrp-Kommando (siehe: Kommandobeschreibung) sind alle Shell-Kommandos, in diesem Kapitel beschrieben.

3.8.1 : Ein Kommando, das "nichts" macht

Das Leere-Kommando tut "nichts" und sein Ende-Status ist 0 (d.h. *wahr*).
Das Leere-Kommando hat das Format:

:

Beispiel für das :-Kommando:

Man kann das :-Kommando in einer Prozedur dazu benutzen, um "nichts" zu machen, um z.B. den Zweig einer if- oder case-Anweisung zu füllen.

```
if test -f $1
then :
else echo "Guten Tag"
fi
```

In diesem einfachen Beispiel wird jede Eingabe, die kein Dateiname ist, mit der Meldung *Guten Tag* quittiert.

3.8.2 # Kommentare einfügen

Mit dem #-Zeichen kann man in einer Prozedur Kommentarzeile schreiben. Das #-Zeichen hat das Format:

```
#_
```

Beispiel für das #-Kommando:

```
if test -f $1          # Ist die Eingabe ein Dateiname?
```

Hinweis

Beachten Sie, daß dem # ein Leerzeichen folgen muß. Der Kommentar endet mit dem Zeilenende.

3.8.3 () und { } Kommandos zusammenfassen

Durch runde und geschweifte Klammern kann man Kommandos zusammenfassen. Die Auswirkungen sind unterschiedlich. Für runde und geschweifte Klammern gilt: Ihr Ende-Status ist der des zuletzt ausgeführten Kommandos.

Stehen Kommandos zwischen zwei runden Klammern (), erzeugt die Shell für diese Kommandofolge eine Sub-Shell (d.h. einen eigenen Prozeß). Was kann man damit anfangen? Das kann man beim Arbeiten mit Variablen ausnutzen. Werden Kommandos in einer Sub-Shell ausgeführt, werden Variablen in der eigenen aktuellen Shell davon nicht berührt. Verändert man also in einer Sub-Shell Variablen (die z.B. vorher von der aktuellen Shell an die Sub-Shell übergeben wurden), haben diese Änderungen keine Rückwirkung auf die aktuelle Shell.

Beispiel für runde Klammern:

Beispiel 1: Angenommen, man hat für das Umfeld der aktuellen Shell die Variable *name* = *meier* definiert. Durch folgende Eingabe sieht man, daß diese Variable in einer Sub-Shell verändert werden kann ohne daß sie deshalb in der aktuellen Shell verändert wird:

```
(echo $name;name = richter;echo $name);echo $name
```

Die Ausgabe sieht so aus:

```
meier
richter
meier
```

Was wurde gemacht? Die Shell liest die erste runde Klammer und erzeugt eine Sub-Shell. Das erste echo-Kommando liest *\$name* und gibt *meier* aus. Die Angabe *name = richter* ändert die Variable *name*, sie wird durch das zweite echo-Kommando ausgegeben. Die zweite runde Klammer beendet die Sub-Shell, man ist wieder in der eigenen aktuellen Shell. Dort liest das dritte echo-Kommando die Variable *name* und gibt *meier* aus.

Beispiel 2: Runde Klammern kann man auch noch dazu benutzen, um Kommandos in einem anderen, als dem eigenen Dateiverzeichnis auszuführen; ohne daß man dafür extra einen Rücksprung angeben muß.

Die Eingabe:

```
(cd ../;ls -l)
```

gibt alle Einträge des dem eigenen Dateiverzeichnis übergeordneten Dateiverzeichnis aus. Anschließend befindet man sich wieder automatisch in seinem aktuellen Dateiverzeichnis. Die Eingabe:

```
cd ../;ls -l
```

hätte zwar die gleiche Ausgabe zur Folge, gleichzeitig würde man jedoch auch das Dateiverzeichnis wechseln.

Stehen Kommandos zwischen geschweiften Klammern { }, werden die Ausgaben aller Kommandos zusammengefaßt. Die Summe dieser Ausgaben kann man zur Weiterverarbeitung an ein anderes Kommando übergeben. Die eingeschlossenen Kommandos werden von der Shell gelesen und verarbeitet. Arbeitet man mit geschweiften Klammern, muß die öffnende Klammer als erstes Zeichen in der Kommandozeile angegeben werden, gefolgt von einem Blank. Vor einer schließenden geschweiften Klammer muß ein Semikolon stehen. Eine Eingabe für geschweifte Klammern:

```
{ ls -l;ls -l ../;}|grep meier
```

Was wird gemacht? Mit dieser Eingabe werden alle Dateien ausgegeben, die *meier* heißen oder *meier* gehören. Durchsucht wird dafür das aktuelle und das übergeordnete Dateiverzeichnis.

3.8.4 eval Kommandos übergeben und ausführen

Mit dem eval-Kommando kann man der Shell Kommandos übergeben und ausführen. Ausgaben des eval-Kommandos lassen sich nicht umleiten. Es wird eine Kommando- und/oder Parameter-Substitution durchgeführt. Das eval-Kommando hat das Format:

```
eval [Kommando]
```

Beispiel für das eval-Kommando:

```
Dateiname: testeval
Inhalt    : kommando1=who
           : kommando2=';date'
           : eval $kommando1 $kommando2
Aufruf    : $ testeval
Ausgabe   : 1) Alle z.Zt. ans System angeschlossenen Benutzer
           : 2) Das aktuelle Datum
```

Was wurde gemacht? Die in der eval-Kommandozeile angegebenen Kennwortparameter *\$kommando1* und *\$kommando2* wurden ersetzt durch *who;date*. Diese beiden Kommandos hat die Shell ausgeführt und deren Ausgabe auf die Standard-Ausgabe geschrieben.

3.8.5 exec Kommando ausführen und Shell ersetzen

Mit dem exec-Kommando kann man:

- ein Kommando oder eine Shell-Prozedur ausführen, wodurch die Shell ersetzt wird.
- die Standard-Eingabe für eine Shell-Prozedur auf eine Datei umlenken.

Das exec-Kommando hat das Format:

```
exec name
```

name Name einer Shell-Prozedur, eines Kommandos oder einer Datei.

Beispiele für das exec-Kommando:

Beispiel 1: Was passiert, wenn man das exec-Kommando im Dialog von der Datensichtstation aus eingibt?

```
exec date
```

Durch das exec-Kommando wird als erstes die Shell beendet und anschließend das date-Kommando ausgeführt. Am Bildschirm erscheint als Ausgabe das aktuelle Datum. Da die Shell beendet wurde, sind Sie anschließend nicht mehr mit dem System verbunden. Sie werden automatisch aufgefordert, ihre Benutzerkennung einzugeben.

Beispiel 2: Was passiert, wenn man mit dem exec-Kommando in einer Prozedur eine andere Prozedur aufruft? Angenommen, es gibt folgende Dateien:

Datei1

Name : prozedur1
Inhalt : echo prozedur1 hat die Prozessnummer \$\$
sh prozedur2

Datei2

Name : prozedur2
Inhalt : echo prozedur2 hat die Prozessnummer \$\$
exec prozedur3

Datei3

Name : prozedur3
Inhalt : echo prozedur3 hat die Prozessnummer \$\$

Wenn jetzt *prozedur1* gestartet wird durch: `sh prozedur1`, bekommt man am Bildschirm eine Ausgabe, die so aussehen kann:

```
prozedur1 hat die Prozessnummer 16834
prozedur2 hat die Prozessnummer 16840
prozedur3 hat die Prozessnummer 16840
```

Was zeigt diese Ausgabe? Sie zeigt, daß *prozedur1* unter einer eigenen Prozeßnummer läuft und *prozedur2* sowie *prozedur3* unter einer gemeinsamen Prozeßnummer ablaufen. Warum? Weil *prozedur3* aus der *prozedur2* heraus durch `exec` aufgerufen wurde. Deshalb wird für *prozedur3* kein neuer Prozeß erzeugt, er bekommt die gleiche Prozeßnummer wie *prozedur2*.

Beachten Sie, daß eine durch das `exec`-Kommando aufgerufene Prozedur als ausführberechtigt gekennzeichnet sein muß (siehe: `chmod`-Kommando).

Beispiel 3: Dieses Beispiel zeigt Ihnen, wie man mit dem `exec`-Kommando die Standard-Eingabe für eine Prozedur auf eine Datei umlenkt. Eine Prozedur soll ihre Eingabe also nicht mehr von der Datensichtstation, sondern aus einer Datei lesen. Name der Prozedur: `extest`. Wenn Sie für diese Prozedur eine Eingabe von der Datensichtstation lesen wollen, müssen Sie folgendes in sie eintragen:

```
read baba echo $baba
```

Was passiert, wenn Sie die Prozedur aufrufen? Eingabe: `sh extest`. Die Prozedur wird durch das `read`-Kommando gestoppt und wartet auf eine Eingabe von Ihnen, die Sie an der Datensichtstation eingeben müssen. Angenommen, Sie geben ein `auto`, dann wird diese Eingabe dem hinter `read` angegebenen Variablennamen (`baba`) als Variablenwert zugewiesen. Durch das `echo`-Kommando wird dieser Wert am Bildschirm ausgegeben. Es erscheint `auto`.

Wollen Sie für die gleiche Prozedur eine Eingabe anstatt von der Datensichtstation aus einer Datei lesen, müssen Sie folgendes in die Datei `extest` eintragen:

```
exec </etc/passwd  
read baba  
echo $baba
```

Wenn Sie jetzt diese Prozedur aufrufen, passiert folgendes. Durch das `exec`-kommando wird die Standard-Eingabe für die Prozedur `extest` auf die Datei `/etc/passwd` umgelenkt, d.h. das dem `exec`-Kommando folgende `read`-Kommando liest jetzt nicht mehr von der Datensichtstation, sondern aus der Datei `/etc/passwd`. Konkret liest es den ersten Satz und weist ihn der Variablen `baba` als Wert zu. Das `echo`-Kommando gibt diesen Satz aus. Im Normalfall müßten das die Benutzer-Einträge für den Systemverwalter sein.

3.8.6 exit Beenden einer Shell-Prozedur

Mit dem `exit`-Kommando kann man eine Shell-Prozedur beenden und wahlweise einen Ende-Status setzen. Als Ende-Status kann man eine Zahl von 0 bis 32768 angeben. Format des `exit`-Kommandos:

```
exit [n]
```

Wie arbeitet das `exit`-Kommando? Das `exit`-Kommando beendet immer die Shell, in der es angegeben wird. Was heißt das? Gibt man an der Datensichtstation z.B. ein Kommando ein, um eine Shell-Prozedur zu starten, wird von der Login-Shell dafür eine Sub-Shell gestartet. Die Sub-Shell verarbeitet die aufgerufene Shell-Prozedur. Wird in der aufgerufenen Prozedur ein `exit`-Kommando ausgeführt, wird die Sub-Shell beendet und man befindet sich wieder in der Login-Shell. Den Mechanismus Shell/Sub-Shell muß man verstehen, um einen durch das `exit`-Kommando gesetzten Ende-Status anwenden zu können.

Was ist der Ende-Status? Es gibt zwei Möglichkeiten. Wurde ein Kommando erfolgreich ausgeführt, wird eine null (*wahr*) gesetzt. Konnte ein Kommando nicht erfolgreich ausgeführt werden, wird eine Ziffer ungleich null (*unwahr*) gesetzt. Beendet man eine Prozedur durch das `exit`-Kommando, kann man einen beliebigen Ende-Status setzen. Den Ende-Status kann man auswerten. Wie? Mit Hilfe des Zeichens: `$?`.

Beispiel für das exit-Kommando:

Probieren Sie einmal folgendes: Lesen Sie Ihr Dateiverzeichnis

Eingabe: ls

Ausgabe: Einträge des aktuellen Dateiverzeichnisses

Wenn Sie jetzt den Ende-Status des ls-Kommandos abfragen, durch die Eingabe:

echo \$?

erhalten Sie als Ausgabe: 0 , d.h. das ls-Kommando wurde erfolgreich ausgeführt. Machen Sie jetzt einmal irgendeine "Phantasieeingabe", z.B. :

unsinn

dann erscheint als Ausgabe: unsinn: nicht gefunden Wenn Sie jetzt den Ende-Status abfragen, durch die Eingabe:

echo \$?

erhalten Sie als Ausgabe: 1, d.h. die "Phantasieeingabe": unsinn konnte nicht ausgeführt werden. Probieren Sie jetzt einmal folgende Eingaben hintereinander:

1. Eingabe: unsinn
Ausgabe: unsinn: nicht gefunden
2. Eingabe: echo \$?
Ausgabe: 1
3. Eingabe: echo \$?
Ausgabe: 0

Dieses Beispiel zeigt, daß für die Shell-Variable: \$? immer der Ende-Status des zuletzt ausgeführten Kommandos eingetragen wird. Deshalb kann man den von der "Phantasieeingabe" erzeugten Ende-Status auch nur einmal abfragen. Anschließend wird für \$? der Ende-Status der zweiten Eingabe gesetzt, d.h. der des erfolgreich ausgeführten echo-Kommandos. Die dritte Eingabe hat demzufolge die Ausgabe: 0 und nicht mehr die 1 des "Phantasiekommandos". Will man einen Ende-Status auswerten, muß man darauf achten, daß zwischen Setzen des Ende-Status und Auswertung, er durch kein anderes Kommando verändert wird.

Wie kann man den Ende-Status in einer Prozedur auswerten?

Als erstes ein Beispiel dafür, wie es nicht geht. In einer Prozedur soll abgefragt werden, ob eine Eingabe ein Datei- oder Dateiverzeichnisname ist. Entsprechend dem Abfrageergebnis soll durch ein `exit`-Kommando verzweigt werden. Die Prozedur sieht so aus:

```
Dateiname: abfrage
Inhalt  : if test -f "$1"
          then exit 2;
          elif test -d "$1";
          then exit 3;
          else exit 4;
          fi
          case $? in
            2) cat $1;echo "Das war der Inhalt der Datei: "$1;;
            3) cd $1;ls -l;echo "Das war das Dateiverzeichnis: "$1;;
            4) echo "Die Eingabe hat für diese Prozedur keinen Sinn";;
          esac
```

Was passiert, wenn man diese Prozedur aufruft?

Gibt man z.B. ein: `abfrage kunde` (*kunde* soll der Name einer Datei sein), passiert folgendes: Die Login-Shell erzeugt für diese Eingabe eine Sub-Shell um die Prozedur auszuführen. Die Sub-Shell fragt durch das `if`-Kommando die Datei *kunde* ab und beendet sich durch das Kommando: `exit 2`. Der Ende-Status der Prozedur ist auf `$?=2` gesetzt. Weil die Sub-Shell durch das `exit`-Kommando beendet wurde, wird das dem `if`-Kommando folgende `case`-Kommando nicht mehr ausgeführt. Der Ende-Status `$?=2` wird in dieser Prozedur nicht mehr weiterverarbeitet. Den Ende-Status kann man jetzt nur noch im Dialog an der Datensichtstation abfragen (z.B. durch das `echo`-Kommando) oder weiterverarbeiten. Wobei man, wie bereits erwähnt, aufpassen muß, daß man ihn nicht durch die Eingabe eines anderen Kommandos verändert.

Was kann man tun, um den Ende-Status in einer Datei auswerten zu können?

Antwort: Man setzt im obigen Beispiel das if-Kommando in runde Klammern. Die Prozedur sieht dann so aus:

Dateiname: *abfrage*

```
Inhalt : (if test -f "$1"
        then exit 2;
        elif test -d "$1";
        then exit 3;
        else exit 4;
        fi)
        case $? in
            2) cat $1;echo "Das war der Inhalt der Datei: "$1;;
            3) cd $1,ls -l;echo "Inhalt des Dateiverzeichnisses: "$1;;
            4) echo "Eingabe hat keinen Sinn für diese Prozedur"
        esac
```

Was ist daran anders? Für ein in runde Klammern gesetztes Kommando wird eine Sub-Shell erzeugt (siehe auch: Kapitel 3.8.3).

Welche Auswirkungen hat das in unserem Beispiel? Macht man wieder die Eingabe: `$ abfrage kunde`, erzeugt die Login-Shell als erstes wieder eine Sub-Shell, um die Prozedur *abfrage* auszuführen. Diese Sub-Shell liest als erstes die runde Klammer und erzeugt dadurch eine weitere Sub-Shell, um das if-Kommando auszuführen. Die "zweite" Sub-Shell wird durch das Kommando `exit 2` beendet. Der Ende-Status wird auf `$? = 2` gesetzt. Dieser Ende-Status wird an die "erste" Sub-Shell weitergereicht. Da durch das `exit`-Kommando nur die "zweite" Sub-Shell und nicht gleichzeitig die "erste" Sub-Shell beendet wurde, wird nun als nächstes von der "ersten" Sub-Shell das dem if-Kommando folgende `case`-Kommando ausgeführt. Das `case`-Kommando kann dann den Ende-Status `$? = 2` auswerten. Wie man an diesem Beispiel sieht, kann man in einer Prozedur einen durch ein `exit`-Kommando gesetzten Ende-Status, nur in der aufrufenden und nicht in der den Ende-Status erzeugenden Sub-Shell weiterverarbeiten.

Eine andere Möglichkeit dieses Beispiel zu realisieren, kann so aussehen: Man schreibt die beiden Kommandos `if` und `case` in zwei Dateien eines Dateiverzeichnisses:

Datei 1

```
Name : abfrage1
Inhalt: if test -f "$1"
        then exit 2;
        elif test -d "$1";
        then exit 3;
        else exit 4;
        fi
```

Datei 2

```
Name : abfrage2
Inhalt: abfrage1 $1
        case $? in
            2) cat $1;echo "Das war der Inhalt der Datei: "$1";;
            3) cd $1;ls -l;echo "Inhalt des Dateiverzeichnis: "$1";;
            4) echo "Die Eingabe hat für diese Prozedur keinen Sinn"
        esac
```

Ruft man jetzt die Datei: *abfrage2* zusammen mit dem Namen einer Datei als Parameter auf (z.B. durch: `abfrage2 kunde`), erzeugt die Login-Shell als erstes wieder eine Sub-Shell um die Prozedur *abfrage2* auszuführen. Diese Sub-Shell liest die erste Zeile (Inhalt: `abfrage1 $1`). Nachdem die Prozedur *abfrage1* ebenfalls in einer neuen Sub-Shell ausgeführt wurde, bekommt die erste Sub-Shell den Ende-Status `$? = 2` zurückgemeldet (er wurde durch `exit 2` in der Prozedur: *abfrage1* gesetzt). Mit diesem Ende-Status wird jetzt in der Prozedur *abfrage2* das `case`-Kommando ausgeführt.

3.8.7 export Variablen weiterreichen

Mit dem export-Kommando kann man Variablen von einer Shell an eine Sub-Shell weiterreichen.

Das export-Kommando hat das Format:

```
export [variable]
```

Beispiel für das export-Kommando:

Angenommen, Sie haben in Ihrer Login-Shell eine Variable definiert, z.B. durch die Eingabe:

```
gruss=hallo
```

Auf diese Variable können Sie jetzt z.B. mit dem echo-Kommando zugreifen:

```
Eingabe: echo $gruss Ausgabe: hallo
```

Warum funktioniert das? Weil der Login-Shell die Variable *\$gruss* bekannt ist und deshalb von ihr substituiert werden kann. Schreiben Sie das gleiche echo-Kommando in eine Datei, z.B.:

```
Dateiname: text  
Inhalt    : echo $gruss
```

und rufen diese Datei auf, durch die Eingabe:

```
text
```

wird nichts ausgegeben. Warum? Weil die Login-Shell eine so aufgerufene Datei (und damit das in ihr stehende echo-Kommando) immer in einer Sub-Shell ausführt. Dieser Sub-Shell ist jedoch die Variable *\$gruss* nicht bekannt, weil sie nur für die Login-Shell definiert wurde. Was tun? Man macht die Variable systemweit bekannt, indem man sie durch das export-Kommando "exportiert".

```
export gruss
```

Nach dieser Eingabe wird in der Prozedur *text* die Variable *\$gruss* ersetzt. Die Eingabe: *text* erzeugt die Ausgabe: hallo.

Hinweis 1

Im Dialog eingegebene Variablen sind nach dem Abmelden vom System gelöscht. Will man Variablen dauerhaft definieren, muß man sie in die Datei *.profile* schreiben und exportieren. Durch die Eingabe: `$ export` erhält man eine Liste von allen als exportierbar deklarierten Variablen, sowie eine Liste der Variablen, die "nur lesbar" sind (siehe auch: `readonly`-Kommando).

Durch die Eingabe: `set`, kann man sich eine Liste der "aktuellen Umwelt" ausgeben lassen (siehe auch: `set`-Kommando).

Hinweis 2

Über weitere Möglichkeiten Variable zu exportieren, informiert Sie Kapitel 3.6.1 .

3.8.8 punkt Datei starten ohne neuen Prozeß zu erzeugen

Mit dem Punkt-Kommando veranlassen Sie, daß die Shell eine Datei ausführt ohne das sie dafür einen eigenen Prozeß erzeugt. An eine durch das Punkt-Kommando auszuführende Datei, kann man aus der Kommandozeile nur Kennwort-Parameter übergeben.

Das punkt-Kommando hat das Format:

. dateiname

dateiname Name einer Datei, die ausgeführt werden soll.

Das Punkt-Kommando bietet sich an, wenn Sie z.B. in der Datei *.profile* Variablen verändern und diese Änderungen an die Shell übergeben möchten.

Beispiel für das *.-*Kommando:

Eingabe: *..profile*

Was passiert? Die aktuellen (eventl. geänderten) Variablen der Datei *.profile* werden an die Shell übergeben.

Achtung

Das Punkt-Kommando ist etwas anderes als ein Punkt vor einem Dateinamen (z.B. *.profile*).

3.8.9 read Prozedur anhalten und etwas einlesen

Mit dem read-Kommando kann man eine Prozedur anhalten, um sie entsprechend einer Eingabe von der Standard-Eingabe aus, weiterlaufen zu lassen.

Das read-Kommando hat das Format:

read [name...]

name Dafür kann man einen oder mehrere Namen angeben, denen in der Standard-Eingabe angegebene Eingaben zugewiesen werden. Auf name kann man durch \$name zugreifen.

Wie arbeitet das read-Kommando? Das read-Kommando liest eine Zeile der Standard-Eingabe. Die einzelnen Werte der Eingabe werden der Reihe nach den Variablen: name zugewiesen. Überzählige Werte in der Standard-Eingabe werden der letzten Variablen des read-Kommandos zugewiesen.

Beispiel für das read-Kommando:

Prozedurdatei: *readtest*

```
Inhalt: echo Bitte geben Sie Parameter ein:
        read kunde1 kunde2 kunde3
        echo $kunde1
        echo $kunde2
        echo $kunde3
```

Eingabe: sh readtest

Ausgabe: Bitte geben Sie Parameter ein:

Eingabe: Meier Miller Beck

Ausgabe: Meier

Miller

Beck

Was wurde gemacht? Die Prozedur *readtest* wurde gestartet und nach der Ausgabe des Textes des ersten echo-Kommandos gestoppt. Die Eingabe der Worte *Meier Miller Beck* wurde vom read-Kommandos gelesen und den Namen *kunde1 kunde2 kunde3* zugewiesen. Anschließend wurde jeder Name mit einem *\$*-Zeichen aufgerufen und durch ein echo-Kommando ausgegeben. Gibt man für die gleiche Prozedur folgendes ein:

Eingabe: Meier Miller Beck Eric Joe

erhält man folgende Ausgabe:

Meier

Miller

Beck Eric Joe

Warum? Der letzten Variablen des read-Kommandos (*kunde3*) werden jetzt die überzähligen Werte der Standard-Eingabe zugewiesen und damit auch vom Kommando: *echo \$kunde3* ausgegeben.

3.8.10 readonly Variablen schützen

Mit dem readonly-Kommando kann man Variablen als "nur lesbar" deklarieren. Die Werte dieser Variablen lassen sich durch Zuweisungen nicht mehr ändern. Wird readonly alleine eingegeben, bekommt man eine Liste aller readonly-Variablen ausgegeben.

Das readonly-Kommando hat das Format:

```
readonly [variable ... ]
```

3.8.11 set und sh Schalter für die Shell setzen

Durch das set-Kommando kann man Schalter setzen, die das Verhalten der Shell ändern. Besonders die beiden Schalter -x und -v können sehr nützlich sein. Diese beiden Schalter kann man auch wieder zurücksetzen- alle anderen nur durch Beenden der Sitzung. Außerdem kann man die Reihenfolge von in der Kommandozeile stehenden Angaben ändern oder eine Kommandozeile neu aufbauen (siehe auch Kapitel: 3.6.1 sowie: Beispiele). Das set-Kommando hat das Format:

```
set [-xvекntu] [name]
```

Schalter kann man z.B. so eingeben: set -xv Gibt man das set-Kommando alleine ein (Eingabe: set), bekommt man alle Variablen ausgegeben, die für die aktuelle Shell gesetzt sind.

Bedeutung der Schalter:

- v Alle Eingabe-Zeilen, die von der Shell gelesen werden, erscheinen am Bildschirm. Mit diesem Schalter kann man Syntax-Fehler finden. Die Kommandos jeder Zeile werden ausgeführt, nachdem diese Zeile ausgegeben ist.
- x Die Kommandos und ihre Parameter werden so ausgegeben (mitgeschrieben), wie sie ausgeführt werden (Shell-Kommandos, wie z.B. for oder while werden allerdings nicht ausgegeben).
- e Die Shell wird sich sofort beenden (exit), sobald ein von ihr ausgeführtes Kommando mit einem Ende-Status ungleich null abschließt.
- k Diesen Schalter muß man setzen, wenn man in einer Kommandozeile hinter dem Prozeduraufruf stehende Kennwortparameter übergeben will.
- n Dieser Schalter überprüft eine Prozedur auf Syntaxfehler, ohne die Prozedur-Kommandos auszuführen.
- t Die Shell soll die Kommandos der aktuellen Eingabezeile lesen, ausführen und sich anschließend beenden.

- u Ist dieser Schalter gesetzt, behandelt die Shell eine nicht zugewiesene Variable als Fehler, sobald sie den Wert dieser Variable substituieren soll. Mit diesem Schalter kann man Variable global prüfen.
- Schaltet die -x und -v Schalter aus.

name Angaben für die Kommandozeile (siehe Beispiele). Diese Angaben gelten nur für die Prozedur, in der man sie angibt, d.h. sie sind nicht exportierbar (siehe: export-Kommando).

Die obenstehenden Schalter kann man auch alle beim Aufruf einer neuen Shell (durch das sh-Kommando) direkt mit angeben.

Die folgenden Schalter lassen sich nur durch das sh-Kommando setzen:

- c pfadname Mit diesem Schalter kann man der Shell einen Dateinamen oder Pfadnamen einer ausführbaren Datei mitteilen, von dem sie Kommandos lesen soll.
- s Ist dieser Schalter gesetzt, liest die Shell Kommandos von der Standard-Eingabe. Ausgaben erfolgen über Dateideskriptor 2 (Standard-Fehlerausgabedatei) auf den Bildschirm. Der -s Schalter ist für die nach Login geladene Shell automatisch gesetzt.
- i Die Shell befindet sich im Dialog, wenn dieser Schalter gesetzt oder die Ein-/Ausgabe der Datensichtstation zugeordnet ist. Von so einer Shell wird das Signal 2 (INTERRUPT) angenommen aber nicht bewertet. Die Signale 3 (QUIT) und 15 (TERMINATE) werden ignoriert.

Beispiele für das set-Kommando:

Beispiel 1

An die Prozedurdatei *settest* soll eine Kommandozeile übergeben werden. Die Reihenfolge der Angaben in der Kommandozeile soll in der Prozedur geändert werden.

```
Inhalt der Prozedurdatei:  echo $1 $2 $3
                           set $3 $2 $1
                           echo $1 $2 $3
Aufruf:                   settest SINIX IST TOLL
Ausgabe:                  SINIX IST TOLL
                           TOLL IST SINIX
```

Was wurde gemacht? Beim Aufruf der Prozedurdatei: *settest* wurde an die Prozedur übergeben: *SINIX IST TOLL*. Durch das erste *echo*-Kommando wurde diese Zeichenfolge ausgegeben. Durch das *set*-Kommando wurde die Reihenfolge der Angaben in der Kommandozeile geändert in: *TOLL IST SINIX* und anschließend durch das zweite *echo*-Kommando ausgegeben.

Beispiel 2

An die Prozedurdatei *settest1* soll eine Kommandozeile übergeben werden. Der Inhalt der Kommandozeile soll in der Prozedur durch einen anderen Inhalt ersetzt werden.

```
Inhalt der Prozedurdatei:  echo $1 $2 $3
                           set WER IST DA
                           echo $1 $2 $3
Aufruf:                   settest1 HALLO IHR DORT
Ausgabe:                  HALLO IHR DORT
                           WER IST DA
```

Was wurde gemacht? Die Prozedur wurde gestartet durch die Eingabe: *settest*. Das erste *echo*-Kommando liest die Eingabe: *HALLO IHR DORT* und gibt sie aus. Das *set*-Kommando ersetzt den Inhalt der Kommandozeile durch: *WER IST DA*. Das zweite *echo*-Kommando liest die Kommandozeile und gibt den neuen Inhalt aus.

Beispiel 3

Die Eingabe für das in der Prozedurdatei *settest2* stehende `read`-Kommando, soll als Stellungparameter verfügbar gemacht werden.

```
Inhalt der Prozedurdatei: read namen
                          set $namen
                          echo $1 $2 $3
                          echo $3 $1 $2
                          echo $2 $3 $1
    Aufruf: settest2
    Eingabe: Miller Beck Plant
    Ausgabe: Miller Beck Plant
            Plant Miller Beck
            Beck Plant Miller
```

Was wurde gemacht? Die Prozedur wurde gestartet durch die Eingabe: `settest2`. Das `read`-Kommando erwartet eine Eingabe von der Tastatur. Eingegeben wurde *Miller Beck Plant*. Diese Eingabe wurde als Wert der Variablen *namenzugewiesen*. Das `set`-Kommando macht diesen Wert durch *\$namen* im Variablen- und Parameterbereich der Prozedur als Stellungparameter verfügbar (siehe auch: Kapitel 3.6.1). Die drei `echo`-Kommandos gaben entsprechende Kombinationen aus.

3.8.12 shift Verschieben von Operanden

Am Bildschirm kann man in der Kommandozeile maximal 9 Stellungparameter (\$1-\$9) direkt angeben und abarbeiten. Will man mehr als 9 Stellungparameter bearbeiten, kann man das mit dem shift-Kommando machen. Das shift-Kommando verschiebt die Zuordnung der Werte zu den Stellungparametern nach links. Der Wert von \$1 fällt heraus und \$1 erhält den Wert von \$2, \$2 erhält den Wert von \$3, \$3 den Wert von \$4 usw.. \$0 wird durch das shift-Kommando nicht beeinflusst.

Das shift-Kommando hat das Format:

shift

Beispiele für das shift-Kommando:

Beispiel 1

Dateiname: *testshift*

```
Inhalt : while test "$1" != ""
        do
            echo $1
            shift;
        done
```

Aufruf : testshift 1 2 3 4 5 6 7 8 9 10 11

```
Ausgabe : 1
          2
          3
          4
          5
          .
          .
          11
```

Beispiel 2

```

Dateiname: testshift
Inhalt   : while test "$1" != ""
           do
               echo $1 $2 $3 $4 $5 $6 $7 $8 $9
           shift;
           done
Aufruf   : testshift 1 2 3 4 5 6 7 8 9 10 11
Ausgabe  : 1 2 3 4 5 6 7 8 9
           2 3 4 5 6 7 8 9 10
           3 4 5 6 7 8 9 10 11
           4 5 6 7 8 9 10 11
           5 6 7 8 9 10 11
           6 7 8 9 10 11
           7 8 9 10 11
           8 9 10 11
           9 10 11
           10 11
           11
    
```

Beispiel 3

```

Dateiname: testshift
Inhalt   : while test "$1" != ""
           do
               echo $*
           shift;
           done
Aufruf   : testshift 1 2 3 4 5 6 7 8 9 10 11
Ausgabe  : 1 2 3 4 5 6 7 8 9 10 11
           2 3 4 5 6 7 8 9 10 11
           3 4 5 6 7 8 9 10 11
           4 5 6 7 8 9 10 11
           5 6 7 8 9 10 11
           6 7 8 9 10 11
           7 8 9 10 11
           8 9 10 11
           9 10 11
           10 11
           11
    
```

3.8.13 times Prozeßzeiten auflisten

Für die von der Shell aus gestarteten Prozesse werden die aufsummierten Benutzer- und Systemzeiten ausgegeben. Das times-Kommando hat das Format:

times

3.8.14 trap Shell-Prozedur unterbrechen

Eine Shell-Prozedur kann durch das Betriebssystem oder von der Datensichtstation aus durch den Benutzer unterbrochen werden. Dazu muß mit dem kill-Kommando ein Signal an die Shell-Prozedur gesendet werden. Mit dem trap-Kommando kann man Unterbrechungssignale empfangen und darauf reagieren. Das trap-Kommando hat das Format:

trap Kommando-Liste Signal-Liste

Kommando-Liste An diese Stelle kann man ein oder mehrere Kommandos schreiben, die ausgeführt werden sollen, wenn ein in der Signal-Liste stehendes Signal eintrifft. Die Shell liest die angegebene Kommando-Liste, sobald sie das erste Mal das trap-Kommando liest. Das passiert beim Start der Shell-Prozedur. Damit die Kommando-Liste aber erst beim Eintreffen eines Signals ausgeführt wird und nicht bereits dann, wenn die Shell die Prozedur startet, muß man sie in einfache Apostrophe eingeschlossen schreiben. Einfache Apostrophe verbieten der Shell eine sofortige Ausführung der Kommandos (siehe auch Kapitel: Der Apostrophier-Mechanismus). Das wird z.B. wichtig, wenn man temporäre Dateien löschen will. Denn zum Zeitpunkt zu dem die Shell das trap-Kommando erstmals liest, stehen die Namen temporärer Dateien meist noch garnicht fest. Ist keine Kommando-Liste angegeben, wird für jedes in der Signal-Liste angegebene und empfangene Signal eine Standardroutine durchlaufen. Ist die Kommando-Liste eine Leermenge, z.B.: () oder " ", werden die Signale in der Signal-liste ignoriert. Damit kann man eine Shell-Prozedur gegen bestimmte Signale immun machen.

Signal-Liste In der Signal-Liste kann man eine oder mehrere Signalnummern angeben. Trifft eines der definierten Signale ein, wird die Kommando-Liste ausgeführt. Welche Signalnummern man angeben kann, ist beim kill-Kommando beschrieben.

Beispiel für das trap-Kommando:

Angenommen, es gibt folgende Prozedur:

```
Dateiname: traptest
Inhalt   : trap 'echo Prozedurabbruch durch ein Signal;exit' 2 3
          while true
          do echo hallo
          done
```

Ruft man diese Prozedur durch die Eingabe: *traptest* auf, wird am Bildschirm solange die Meldung: hallo ausgegeben, bis das Signal 2 oder 3 eintrifft. Die Signale kann man von der Datensichtstation aus durch folgende Tasten senden:

- Die Tasten `CTRL` und `I` gleichzeitig gedrückt, erzeugen das Signal 3
- Die Taste `DEL` erzeugt das Signal 2

Die Kommando-Liste muß ein `exit`-Kommando enthalten, weil sonst nach Ausführung der Kommando-Liste die folgenden Kommandos noch einmal ausgeführt würden. Was in diesem Beispiel eine Endlosschleife zur Folge hätte. Startet man die Prozedur *traptest* als Hintergrundprozeß (Eingabe: *traptest&*), kann man sie von der Datensichtstation aus nur noch durch das `kill`-Kommando abbrechen. Dazu muß man die Prozeßnummer mit angeben. Die Prozeßnummer eines Hintergrundprozesses wird sofort nach seinem Start am Bildschirm ausgegeben. Die Eingabe des `kill`-Kommandos kann in eine laufende Ausgabe eingetippt werden. In unserem Beispiel z.B.: `kill -2` (Prozeßnummer). Funktionstasten haben bei Hintergrundprozessen keine Wirkung.

3.8.15 umask Standardeinstellung der Schutzbits ändern

Mit dem umask-Kommando kann man die Standardeinstellung der Schutzbits ändern. Die geänderte Einstellung der Schutzbits wird dann einer neu erstellten Datei oder einem neu erstellten Dateiverzeichnis mitgegeben. Die Wirkung des umask-Kommandos gilt nur für den aufrufenden Benutzer und für die Dauer einer Sitzung.

Das umask-Kommando hat das Format:

```
umask [nnn]
```

nnn Dreistelliger oktaler Wert (siehe chmod-Kommando). Jedes n gilt für eine der Benutzergruppen: Eigentümer, Gruppe oder Andere.
Standardeinstellung: 066

Geben Sie das umask-Kommando ohne Parameter ein, wird die Standardeinstellung ausgegeben.

Wie funktioniert das umask-Kommando? Das binäre Komplement der umask-Angabe wird der Standardeinstellung der Schutzbits logisch UND verknüpft und ergibt damit die neue Schutzbiteinstellung.

Beispiel für das umask-Kommando:

Die Eingabe

```
umask 333
```

setzt die Standardeinstellung der Schutzbits für eine Datei auf:

```
r - - r - - r - - .
```

Warum? Die oktale Eingabe 333 entspricht binär: 011011011. Das Komplement davon ist: 100100100.

Wenn Sie jetzt eine neue Datei erstellen, wird dieses Komplement durch ein logisches UND zur alten Standardeinstellung addiert. Die Standardeinstellung für eine Datei ist: rw-rw-rw-, das entspricht binär: 110110110. Die Addition ergibt:

```

  110110110 (Standardeinstellung für eine Datei)
& 100100100 (Komplement der umask-Angabe)
-----
  100100100
```

Binär 100100100 entspricht folgender Einstellung: r - - r - - r - - .

3.8.16 wait Auf Prozeßabschluß warten

Das wait-Kommando wartet auf die Beendigung aller vor ihm laufenden Prozesse. Das wait-Kommando hat das Format.

wait [_prozeßnummer...]

prozeßnummer Nummer eines oder mehrerer Prozesse, auf deren Beendigung wait warten soll.

Beispiel für das wait-Kommando:

```
Prozedurdatei: waittest
  Inhalt: who&
         date&
         wait
         echo Die Prozedur ist beendet
Aufruf: sh waittest
Ausgabe: (aktuelles Datum)
         (Liste aller angeschlossenen Benutzer)
         Die Prozedur ist beendet
```

Was wurde gemacht? In der Prozedur *waittest* wurden das *who*- und *date*-Kommando als Hintergrundprozesse gestartet. Durch das *wait*-Kommando wurde in der Prozedur solange gewartet, bis beide Hintergrundprozesse beendet waren. Anschließend wurde das *echo*-Kommando ausgeführt.

Würde man die gleiche Prozedur ohne das *wait*-Kommando ablaufen lassen, würde am Bildschirm als erstes die Meldung des *echo*-Kommandos ausgegeben: Die Prozedur ist beendet. Warum? Weil *who* und *date* als Hintergrundprozesse ablaufen und das *echo*-Kommando sofort ausgeführt wird.

Hinweis

Wird *wait* unter *script* aufgerufen, müssen Sie eine Prozeßnummer angeben, um auf einen bestimmten Prozeß zu warten. Sonst wartet *wait* auch auf die Beendigung des *script*-Prozesses.

4 Der CED-Editor. Eine Beispielsitzung

Welches Ziel hat diese Beispielsitzung?

Sie sollen ein Gefühl für die Arbeit mit dem CED bekommen. Anhand eines durchgehenden Beispiels (Kapitel 4.2) werden die verschiedenen Modi erklärt, in denen der CED arbeitet.

Es sind bei weitem nicht alle Möglichkeiten des CED aufgeführt. Eine vollständige Kommandobeschreibung des CED finden Sie in Kapitel 6.

Was ist ein Dokument?

Sie werden sehr häufig Dokumente am Datensichtgerät bearbeiten. Dokumente können z.B. enthalten:

- Text (z.B. ein Brief, Dokumentation)
- den Quellcode eines Programms
- eine Shell-Prozedur
- Daten, z.B. eine Liste von Telefonnummern.

Ein Dokument ist in einer Datei gespeichert.

Editoren können nur Dokumente bearbeiten, die aus Zeilen aufgebeut sind.

Zur Erinnerung: Eine Zeile ist abgeschlossen durch das Neue-Zeile-Zeichen (hexadezimal x'0A'). Enthält eine mit dem CED gelesene Datei als letztes Zeichen kein Neue-Zeile-Zeichen, wird das letzte Zeichen in der Datei nicht abgebildet.

Ausführbare Programme (Maschinencode), sind keine Dokumente, da sie keine Zeilen enthalten.

Ist im folgenden das Wort Datei verwendet, so ist damit immer eine Datei gemeint, die ein Dokument enthält.

Im folgenden steht das Wort Text für beliebige Zeichenketten, die aus Buchstaben, Ziffern und den auf der Tastatur enthaltenen Sonderzeichen bestehen.

Zur Bearbeitung von Dokumenten steht Ihnen ein komfortabler bildschirmorientierter Editor zur Verfügung: CED.

Was heißt bildschirmorientiert:

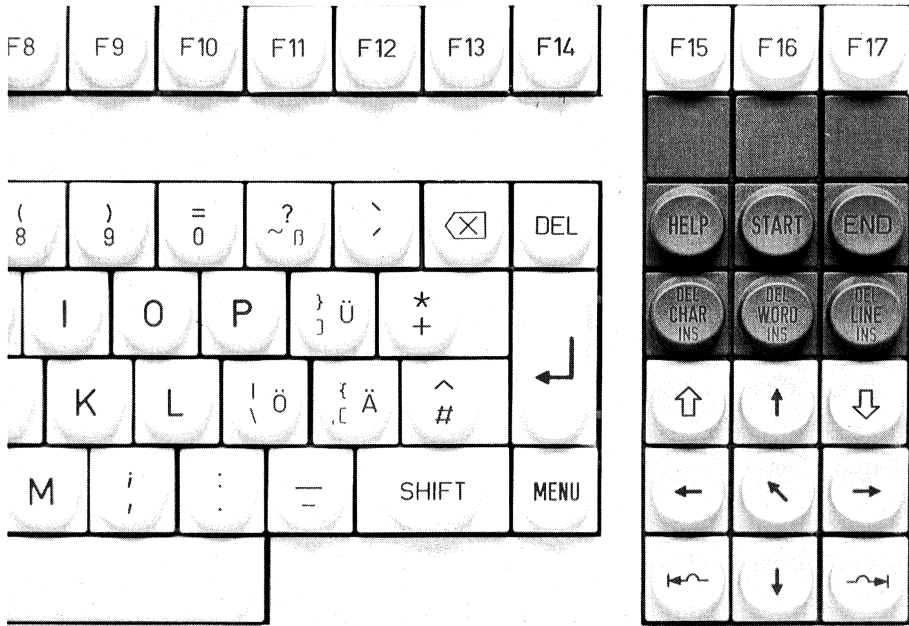
- Sie können die Schreibmarke frei im Dokument bewegen.
- Sie sehen immer den Dokumentenausschnitt, den Sie gerade bearbeiten, auf Ihrem Bildschirm.
- Sie tippen Ihre Änderungen an die Stellen, an die sie hin sollen, und haben sofort visuelle Kontrolle.
- Sie brauchen keine komplizierte Kommandosyntax zu lernen, um den CED zu benutzen, sondern wählen einfach aus einem Menü das nächste Kommando.

Was können Sie mit dem CED machen?

- Text eingeben.
- Text einfügen.
- Zeilen einfügen, löschen, teilen, verbinden.
- frei markierte Blöcke von Zeilen kopieren, löschen, einfügen.
- frei markierte Rechtecke kopieren, löschen.
- SINIX-Kommandos aufrufen: Eingabe aus der Datei, Ausgabe in die Datei.
- nach Zeichenketten suchen.
- für die CED-Sitzung Tasten mit Text und Kommandos belegen.
- in andere Dateien umschalten.
- temporär in die Shell (siehe Kapitel 4.3.3) wechseln.

4.1 Welche Tasten können Sie im CED benutzen?

Zur Eingabe von Text stehen Ihnen alle alphanumerischen Tasten und alle Sonderzeichentasten zur Verfügung. Die im folgenden Bild abgebildeten Tasten haben für den CED besondere Bedeutung:



4.1.1 Die Funktionstasten F9 bis F17

Die Funktionstasten **F9** bis **F17** dienen zum einfachen Aufruf einiger CED-Funktionen. Ihre Bedeutung steht auf dem mitgelieferten Tastenstreifen. Legen Sie ihn bitte über die obere Tastenreihe der Tastatur.

4.1.2 Löschen und Einfügen von Zeichen

Die Tasten **CHAR** und **XL** dienen zum Korrigieren von Text. Mit der Taste **XL** setzen Sie die Schreibmarke um ein Zeichen nach links und löschen das Zeichen. Drücken Sie gleichzeitig die Taste **SHIFT** und die Taste **CHAR**, löschen Sie das Zeichen, auf dem die Schreibmarke steht. Drücken Sie nur die Taste **CHAR**, fügen Sie an der Schreibmarkenposition ein Leerzeichen ein. Wenn Sie allerdings mehrere Zeichen einfügen wollen, benutzen Sie besser den CED-Modus 'Text einfügen'.

4.1.3 Löschen und Einfügen von Zeilen

Um eine Zeile in Ihren Text einzufügen, drücken Sie die Taste **LINE**. Die neue Zeile wird oberhalb der Zeile, in der sich Ihre Schreibmarke befindet, eingefügt. Drücken Sie gleichzeitig die Taste **SHIFT** und die Taste **LINE**, löschen Sie die Zeile, in der sich die Schreibmarke befindet.

4.1.4 Wie bewegen Sie die Schreibmarke?

Mit den Tasten **↑** **←** **→** **↶** **↓** bewegen Sie die Schreibmarke auf dem Bildschirm hin und her. Die Taste **↶** verstärkt deren Wirkung. Ihre genaue Funktion ist im Kapitel 4.2.2 beschrieben.

Wie können Sie das Fenster über dem Dokument verschieben? oder Wie können Sie im Dokument blättern?

Mit den Tasten **↑** und **↓** verschieben Sie das Fenster jeweils um einen Bildschirm, d.h. 10 Zeilen nach oben, bzw. unten. Mit den beiden Funktionstasten **F15** und **F16** verschieben Sie das Fenster jeweils um einen ganzen Bildschirm nach unten, bzw. nach oben. Wollen Sie das Fenster auf den Anfang des Dokuments positionieren, benutzen Sie dazu die Taste **START**.

4.1.5 Der CED arbeitet in verschiedenen Modi

Je nachdem, in welchem Modus Sie sich gerade befinden, haben Ihre Eingaben eine andere Bedeutung für den CED. Wollen Sie in einen anderen Modus umschalten oder ein anderes CED-Kommando ausführen, müssen Sie zuerst in das CED-Hauptmenü umschalten. Drücken Sie dazu die Taste **MENU**. In der untersten Zeile des Bildschirms bietet Ihnen der CED alle nun möglichen Eingaben an.

Die Modi des CED und die dazugehörigen Kommandobuchstaben, die Sie benutzen müssen, um in den entsprechenden Modus umzuschalten, sind:

- Zeilenbereich bearbeiten Taste **B**
- SINIX-Kommando ausführen Taste **K**
- CED verlassen Taste **V**
- Text einfügen Taste **E**
- suchen Taste **S**
- eine Zeile bearbeiten Taste **Z**
- ein Rechteck bearbeiten Taste **R**
- neuen Text eingeben Taste **N**
- abspeichern Taste **A**
- Dokument wechseln Taste **D**
- Fenster positionieren Taste **F**
- Textmarkierung suchen Taste **T**
- Protokoll, Tasten belegen Taste **P**

Außerdem bietet Ihnen der CED noch zwei Kommandos an, die nicht in einen anderen Modus schalten, sondern direkt ausgeführt werden:

- Tastenbelegung anzeigen Taste **X**
- temporär in die Shell wechseln Taste **I**

4.1.6 Die HELP-Taste

Wenn Sie während einer CED-Sitzung nicht mehr weiter wissen, drücken Sie im CED-Hauptmenü die Taste **HELP**. Der CED schaltet dann in ein Dokument, das eine Kurzbeschreibung des CED enthält, um.

Wichtig: Um in das ursprüngliche Dokument zurückzugelangen, drücken Sie einfach die Funktionstaste **F10** oder, wenn Sie es genau machen wollen, die Tasten:

MENU

D

↓

4.1.7 Ein Tastendruck - ein Kommando!

Der CED verarbeitet Ihre Eingaben sofort. Deshalb brauchen Sie Ihre Kommandos nicht mit der Taste **↓** abzuschließen.

Ausnahmen: In vier Modi, in denen Sie Zusatzinformationen eintragen müssen, müssen Sie diese mit der Taste **↓** abschließen. Das sind:

- im Modus 'abspeichern' der Name der Datei, in die abgespeichert werden soll;
- im Modus 'Dokument wechseln' der Name des Dokuments, in das umgeschaltet werden soll;
- im Modus 'suchen' die Zeichenkette, die gesucht werden soll;
- im Modus 'Fenster positionieren' die Schrittweite oder die Zeilennummer.

4.2 Beginnen Sie !

Tippen Sie ein: `ced` , dann ein Leerzeichen und den Namen der zu editierende Datei.

Für unsere Beispielsitzung wählen wir den Dateinamen *brief*. Diese Datei soll in unserem aktuellen Dateiverzeichnis noch nicht existieren. Ihre Eingabezeile sollte jetzt so aussehen.

`$ ced brief`

Drücken Sie die nun Taste `↵` . Wie sieht Ihr Bildschirm jetzt aus?

```
** CED Texteditor V1.0           Zeile:   1 Spalte:   1 Name: brief
```

```
Sie wollen neuen Text eingeben?  
Bitte geben Sie Ihren Text ein!
```

Hinweis

Die Position der Schreibmarke ist auf dem Bildschirm durch das Zeichen `^` dargestellt.

Der CED hat Ihnen einen Editierahmen auf den Bildschirm geschrieben. Dieser Rahmen besteht aus drei Teilen:

- einer Kopfzeile
- dem Arbeitsbereich
- dem Bedienbereich.

Die Kopfzeile informiert Sie über

- den Namen und die Versionsnummer des Editors,
- die Position der Schreibmarke im Dokument,
- den Namen der Datei, die Sie gerade bearbeiten.

Die Angabe der Zeile und Spalte - hier 1 und 1, d.h. die Schreibmarke steht am Anfang der Datei - bezieht sich immer auf die absolute Position der Schreibmarke im Dokument und nicht auf ihre Position auf dem Bildschirm.

Der Arbeitsbereich, der von der Kopfzeile und dem Bedienbereich durch je eine durchgezogene Linie getrennt ist, ist leer, wenn Sie den Namen einer nicht existierenden Datei angegeben haben.

Der zweizeilige Bedienbereich am unteren Bildschirmrand zeigt Ihnen:

- in der ersten Zeile den Namen des Modus an, in dem Sie sich gerade befinden;
- in der zweiten Zeile alle in diesem Modus möglichen Kommandoingaben.

Zu Beginn einer CED-Sitzung befinden Sie sich immer im Modus 'neuen Text eingeben'.

4.2.1 Wie geben Sie Text ein?

Tippen Sie ihn wie auf einer Schreibmaschine ein. Zum Korrigieren benutzen Sie die Tasten:

←X

schiebt die Schreibmarke um eine Position nach links und löscht das dortige Zeichen,

SHIFT **CHAR**

löscht das Zeichen in der Schreibmarkenposition,

CHAR

fügt an der Schreibmarkenposition ein Leerzeichen ein.

Wenn Sie eine neue Zeile beginnen wollen, drücken Sie die Taste **↓**. Die Schreibmarke befindet sich jetzt in der ersten Spalte der nächsten Zeile.



Hinweis





Auf diese Weise können Sie sich neue Zeilen am Ende Ihres Dokuments schaffen.

Versuchen Sie, über den rechten Rand hinaus, d.h. über die 80. Spalte hinauszuschreiben. Der CED verschiebt den Arbeitsbereich um 41 Spalten nach rechts. Nun können Sie wieder weiterschreiben, bis Sie wieder am rechten Rand anstoßen. Dieses Mal verschiebt der CED das Fenster nur um 6 Spalten, denn die maximale Länge einer Zeile ist 127 Zeichen. Wenn Sie versuchen, darüber hinauszuschreiben, gibt der CED eine Fehlermeldung aus.





4.2.2 Wie blättern Sie im Dokument?

Mit diesem 'an-den-Rand-Stoßen' haben Sie eine interessante Eigenschaft des CED entdeckt. Sobald Sie auf dem Bildschirm an einen Rand stoßen, entweder mit der Schreibmarke oder beim Eintippen von Text, blättert der CED automatisch um einen halben Bildschirm in die entsprechende Richtung - falls möglich.







Eine andere Möglichkeit, im Dokument zu blättern, haben Sie durch die Tasten  und . Drücken Sie diese Tasten, bewegen Sie den Arbeitsbereich jeweils eine halbe Seite, d.h. 10 Zeilen nach oben bzw. unten.

Wollen Sie noch schneller blättern, benutzen Sie die beiden Funktionstasten  oder .  verschiebt das Fenster um eine Seite, d.h. 20 Zeilen, nach unten,  nach oben.

'Schnelle Schreibmarkenbewegungen'

Um die Schreibmarke in Sprüngen nach rechts oder links in einer Zeile zu bewegen, können Sie die Tasten  und  benutzen. Der CED benutzt intern Tabulatormarken, die alle 8 Spalten gesetzt sind. Betätigen Sie eine der Tasten  oder  springt die Schreibmarke zuerst auf die nächste Tabulatormarke rechts oder links; bei jedem weiteren Betätigen dieser Tasten auf die nächste Tabulatormarke. Sie können die Tabulatormarken nicht verändern.

Wie springen Sie an den Rand des Arbeitsbereichs?

Haben Sie schon einmal die Taste  benutzt? Sie haben sich sicherlich gewundert, daß sie nicht so wirkt, wie Sie es erwartet haben. Drücken Sie anschließend noch eine der vier Schreibmarkentasten    . Jetzt können Sie ihre Funktion ahnen. Die Taste  verstärkt die Wirkung der nachfolgend gedrückten Taste. Die Schreibmarke wird auf die oberste Zeile, den linken oder den rechten Rand oder die unterste Zeile des Arbeitsbereichs positioniert. Damit haben Sie nun die Möglichkeit, mit zwei Tastenanschlägen alle Ränder des Arbeitsbereiches zu erreichen.

Wie springen Sie ans Ende oder den Anfang Ihrer Datei?

Auch das ermöglicht die Taste `↵`. Drücken Sie sie zweimal und dann eine der Tasten `↑` `←` `→` oder `↓`. Das Fenster wird auf den entsprechenden Rand des Dokuments positioniert. Die Schreibmarke steht bei der Tastenfolge:

<code>↵</code> <code>↵</code> <code>↑</code>	auf der ersten Zeile,
<code>↵</code> <code>↵</code> <code>←</code>	auf der ersten Spalte,
<code>↵</code> <code>↵</code> <code>→</code>	auf der letzten Spalte,
<code>↵</code> <code>↵</code> <code>↓</code>	auf der letzten Zeile des Dokuments.

An den rechten oder linken Rand des Dokuments können Sie auch durch die Tastenfolge `↵` `↵` bzw. `↵` `↵` gelangen.

Mit der Taste `START` kommen Sie auch an den Anfang des Dokuments.

Der Beispieltext

Als Grundlage für die nächsten Schritte in unserer Beispielsitzung soll der folgende Text dienen. Die zweispaltige Tabelle können Sie mit Hilfe der Tabulatortasten leicht eingeben:

```
** CED Texteditor V1.0           Zeile:  11 Spalte:  39 Name: brief
```

```
Betr.: SINIX-Dokumentation.  
Die folgende Tabelle enthaelt Begriffe, wie sie aus der  
englischen Originaldokumentation fuer die SINIX-Dokumen-  
tation uebersetzt wurden.
```

```
process      Prozess  
shell        Shell  
special file  Geratedatei  
cursor       Schreibmarke  
directory    Dateiverzeichnis  
file         Datei_
```

```
Sie wollen neuen Text eingeben?  
Bitte geben Sie Ihren Text ein!
```

4.2.3 Wie verschieben Sie einen Zeilenbereich?

Nehmen Sie an, Sie wollen Ihre Tabelle alphabetisch sortieren. Schieben Sie die drei unteren Zeilen der Tabelle nach oben, und Ihre Tabelle ist sortiert. Aber wie? Drücken Sie folgende Tasten:

[MENU]

[B]

Schreibmarke auf die Zeile 9 positionieren

[M]

Schreibmarke auf die Zeile 11 positionieren

[M]

Jetzt schalten Sie in den Modus 'Zeilenbereich bearbeiten'. In diesem Modus können Sie zusammenhängende Bereiche von ganzen Zeilen bearbeiten. Diese Bereiche legen Sie durch Markieren der ersten und letzten Zeile fest. Dabei spielt es keine Rolle, in welcher Spalte Sie die Zeile markieren. Die untere Zeile des Bedienbereichs zeigt Ihnen die in diesem Modus möglichen Kommandos an. Mit der Taste **[M]** markieren Sie eine Zeile. Bewegen Sie also die Schreibmarke auf die Zeile 9 des zu verschiebenden Bereichs

cursor Schreibmarke

Drücken Sie die Taste **[M]**. Der CED meldet:

***** MARKE GESETZT

Genauso markieren Sie die Zeile 11:

file Datei

Drücken Sie die Taste **[L]**. Der markierte Bereich wird gelöscht. Aber keine Angst: Der CED hat die gelöschten Zeilen in einem Puffer zwischengespeichert. Wenn Sie die Schreibmarke jetzt auf die oberste Zeile der Tabelle positioniert haben, drücken Sie die Taste **[Z]**- zurückholen - : der gelöschte Bereich wird oberhalb der Zeile, in der sich die Schreibmarke befindet, eingefügt.

Jetzt kennen Sie eine der Möglichkeiten, wie Sie einen markierten Block verschieben können:

Löschen Sie im Modus 'Zeilenbereich bearbeiten' den markierten Bereich, positionieren Sie die Schreibmarke auf die neue Position und holen Sie den gelöschten Bereich zurück. Drücken Sie die Tasten:

MENU

B

Schreibmarke positionieren

M

Schreibmarke positionieren

M

L

Schreibmarke positionieren

Z

Aber Achtung: der Inhalt des Zwischenspeichers wird von der nächsten Löschoperation überschrieben.

Den Inhalt des Zwischenspeichers können Sie auch mit der Funktionstaste **F17** zurückholen.

Falls Sie dieser Methode mißtrauen, wählen Sie die zweite Möglichkeit: Nachdem Sie wie oben den Bereich markiert haben, positionieren Sie die Schreibmarke auf die erste Zeile der Tabelle, kopieren Sie den markierten Bereich vor diese Zeile, und löschen Sie den Bereich am Ende der Tabelle. Drücken Sie die Tasten:

.

.

.

Schreibmarke positionieren

K

L

In beiden Fällen muß Ihr Bildschirm so aussehen:

```
** CED Texteditor V1.0           Zeile:   6 Spalte:  39 Name: brief
```

Betr.: SINIX-Dokumentation.

Die folgende Tabelle enthaelt Begriffe, wie sie aus der
englischen Originaldokumentation fuer die SINIX-Dokumen-
tation uebersetzt wurden.

cursor	Schreibmarke
directory	Dateiverzeichnis
file	Datei
process	Prozess
shell	Shell
special file	Geraetedatei

Sie wollen einen Zeilenbereich bearbeiten?

Bitte waehlen Sie eine der folgenden Funktionen: (e,k,l,m,z,<MENU>)!)

Stellen Sie sich vor, Sie entschließen sich, die beiden Spalten der Tabelle zu vertauschen, d.h. die deutschen Begriffe vor die englischen zu stellen.

4.2.4 Können Sie mit dem CED Rechtecke verschieben?

Drücken Sie die Tasten:

MENU

R

schalten Sie in den Modus 'Rechtecke bearbeiten' um. Jetzt können Sie einen beliebigen rechteckigen Textblock verschieben. Die untere Zeile des Bedienbereichs zeigt Ihnen wieder alle möglichen Unterkommandos an. Mit der Taste **M** markieren Sie einen Eckpunkt eines Rechtecks. Um ein ganzes Rechteck festzulegen, müssen Sie zwei diagonal gegenüberliegende Ecken markieren. Bewegen Sie also die Schreibmarke zuerst auf den ersten Buchstaben des ersten Worts in der englischsprachigen Spalte und drücken Sie die Taste **M**. CED quittiert Ihre Markierung durch:

***** MARKE GESETZT.

Genauso markieren Sie nun das letzte Leerzeichen vor dem Wort Geraetedatei.

```
** CED Texteditor V1.0           Zeile:  11 Spalte:  33 Name: brief
```

```
Betr.: SINIX-Dokumentation.
      Die folgende Tabelle enthaelt Begriffe, wie sie aus der
      englischen Originaldokumentation fuer die SINIX-Dokumen-
      tation uebersetzt wurden.

      cursor           Schreibmarke
      directory        Dateiverzeichnis
      file             Datei
      process          Prozess
      shell            Shell
      special file     _Geraetedatei
```

Sie wollen ein Rechte ***** MARKE GESETZT
Bitte waehlen Sie eine der folgenden Funktionen: (h,l,m,u,v,<MENU>)?

Im obigen Bild ist das nun markierte Rechteck gekennzeichnet.

Wie machen Sie weiter?

Der grundsätzliche Ablauf ist wie folgt:

- Schreibmarke auf die Position der oberen linken Ecke des "neuen Rechtecks" positionieren
- Je nach Art des kopierens, **[H]** oder **[V]** oder **[U]** drücken.

Bewegen Sie jetzt die Schreibmarke auf eine Position hinter dem Wort Schreibmarke in der ersten Zeile der Tabelle. Die Position der Schreibmarke wird die linke obere Ecke des einzufügenden Rechtecks. Um die markierte Spalte hier einzufügen, drücken Sie die Taste **[H]** (horizontal einfügen). Ihr Bildschirm müßte jetzt so aussehen:

```
** CED Texteditor V1.0           Zeile:   6 Spalte:  55 Name: brief
```

```
Betr.: SINIX-Dokumentation.
```

```
Die folgende Tabelle enthaelt Begriffe, wie sie aus der
englischen Originaldokumentation fuer die SINIX-Dokumen-
tation uebersetzt wurden.
```

cursor	Schreibmarke	cursor
directory	Dateiverzeichnis	directory
file	Datei	file
process	Prozess	process
shell	Shell	shell
special file	Geraetedatei	special file

```
Sie wollen ein Rechteck bearbeiten?
```

```
Bitte waehlen Sie eine der folgenden Funktionen: (h,l,m,u,v,<MENU>)?
```

Haben Sie Ihre deutsche Spalte zerstört? Dann haben Sie die Schreibmarke vor dem Kopieren nicht weit genug nach rechts bewegt. Wenn Sie horizontal einfügen, und das markierte Rechteck

- z Zeilen und
- s Spalten umfaßt,

verschiebt der CED

- ab der Schreibmarkenposition
- z Zeilen - nach unten gezählt
- um s Spalten nach rechts

Er schafft also genau den Platz, den das einzufügende Rechteck benötigt. Nun kopiert er das markierte Rechteck in den freien Bereich und positioniert die Schreibmarke auf die obere linke Ecke des eingefügten Rechtecks.

Jetzt haben Sie drei Spalten auf dem Bildschirm. Drücken Sie die Taste um die erste Spalte zu löschen.

Sie wundern sich? Die von Ihnen gesetzten Markierungen auf der ersten Spalte waren noch immer gültig. Die Taste bezog sich also auf diese Spalte. Ihr Bildschirm hat jetzt wieder nur noch zwei Spalten:

```
** CED Texteditor V1.0           Zeile:   6 Spalte:  19 Name: brief
```

Betr.: SINIX-Dokumentation.

Die folgende Tabelle enthaelt Begriffe, wie sie aus der englischen Originaldokumentation fuer die SINIX-Dokumentation uebersetzt wurden.

Schreibmarke	cursor
Dateiverzeichnis	directory
Datei	file
Prozess	process
Shell	shell
Geraetedatei	special file

Sie wollen ein Rechteck bearbeiten?

Bitte waehlen Sie eine der folgenden Funktionen: (h,l,m,u,v,<MENU>)?

Warum horizontal einfügen? Kann man auch vertikal einfügen?

Im Modus 'Rechteck bearbeiten' stehen Ihnen zum Kopieren noch zwei weitere Tasten zur Verfügung: (vertikal einfügen) und (überschreiben):

Fügen Sie vertikal ein,

- fügt der CED oberhalb der Zeile, in der die Schreibmarke steht, so viele Leerzeilen ein, wie das markierte Rechteck lang ist,
- kopiert das markierte Rechteck in diese Leerzeilen rechts der Spalte, in der sich die Schreibmarke befindet,
- positioniert die Schreibmarke auf die obere linke Ecke des eingefügten Rechtecks.

Fügen Sie überschreibend ein, (Taste) überschreibt das kopierte Rechteck den an der Einfügestelle stehenden Text.

Hinweis

Das kleinste markierbare Rechteck ist ein Zeichen.

Die Tabelle ist jetzt natürlich wieder in einem unsortierten Zustand (s. letzter Bildschirm). Eine Möglichkeit, sie zu sortieren, wäre, per Hand die einzelnen Zeilen entsprechend zu verschieben. Das könnten Sie im Modus 'eine Zeile bearbeiten' durchführen.

4.2.5 Wie können Sie einzelne Zeilen verschieben?

Versuchen Sie als Beispiel die Zeile

Datei file

an den Anfang der Tabelle zu verschieben.

MENU

Z

Schalten Sie in den Modus 'eine Zeile bearbeiten' um.

Positionieren Sie die Schreibmarke auf die zu löschende Zeile.

L

Löschen Sie die Zeile

Positionieren Sie die Schreibmarke in die erste Zeile der Tabelle

Z

Holen Sie die gelöschte Zeile zurück.

An Stelle des letzten Kommandos hätten Sie wiederum die Funktionstaste

F17 benutzen können.

Sie haben einen ersten Schritt zur Sortierung Ihrer Tabelle getan. Diese müßte jetzt so aussehen:

```
kk CED Texteditor V1.0            Zeile:    6 Spalte:    19 Name: brief
```

Betr.: SINIX-Dokumentation.

Die folgende Tabelle enthaelt Begriffe, wie sie aus der
englischen Originaldokumentation fuer die SINIX-Dokumen-
tation uebersetzt wurden.

Datei	file
Schreibmarke	cursor
Dateiverzeichnis	directory
Prozess	process
Shell	shell
Geraetedatei	special file

Sie wollen eine Zeile bearbeiten?

Bitte waehlen Sie eine der folgenden Funktionen: (e,l,r,t,v,z,<MENU>)

Das ist natürlich eine sehr mühsame Art zu sortieren. Wie Sie vielleicht wissen, gibt es in SINIX ein Sortierprogramm `sort`. Das kann sogar einiges mehr als nur Zeilen zu sortieren. Aber für die Sortierung unserer Tabelle reicht der einfache Aufruf `sort` aus.

4.2.6 Wie können Sie SINIX-Kommandos aufrufen?

Der CED bietet die Möglichkeit, SINIX-Kommandos aufzurufen. Diesen können Teile der gerade editierten Datei durch Festlegung als Eingabebereich als Eingabe angeboten werden. Die Ausgabe des Kommandos fügt der CED an Stelle des Eingabebereiches in den Text ein.

Um Ihre Tabelle zu sortieren, müssen Sie:

- in den Modus 'Kommando ausführen' umschalten,
- die Tabelle als Eingabe spezifizieren, und
- das Kommando `sort` aufrufen.

Durch numerische Eingabe vor dem Kommando können Sie einen Eingabebereich für das Kommando festlegen.

Achtung

Der Eingabebereich beginnt immer mit der Zeile, in der die Schreibmarke steht.

Sie können den Eingabebereich entweder in Einheiten von Zeilen oder von Abschnitten festlegen. Ein Abschnitt reicht immer von der Zeile, in der sich die Schreibmarke befindet, bis zur nächsten Leerzeile. Zeilen spezifizieren Sie durch Angabe einer Dezimalzahl und entweder Anfügen des Zeichens 'l' oder Vorstellen eines '-'.
l
-

Beispiel

-10 bzw. 10l kennzeichnet die nächsten 10 Zeilen ab der Zeile, in der sich die Schreibmarke befindet, als Eingabebereich.

Geben Sie nur eine Dezimalzahl n an, so bezeichnen Sie damit die nächsten n Abschnitte als Eingabebereich.

Machen Sie keine Angaben, nimmt CED standardmäßig den aktuellen Abschnitt, ab der Zeile, in der sich die Schreibmarke befindet, als Eingabebereich an.

Um die Tabelle alphabetisch zu sortieren müssen Sie folgende Tasten drücken:

Positionieren Sie die Schreibmarke auf die erste Zeile der Tabelle

l sort

Also:

Nachdem Sie die Schreibmarke positioniert haben, schalten Sie in den Modus 'Kommando ausführen' um. Geben Sie jetzt Eingabebereich und Kommando in der ersten Zeile des Bedienbereichs ein: l sort

Drücken Sie die Taste . Der CED meldet:

***** KOMMANDO WIRD AUSGEFÜHRT.BITTE WARTEN

und bearbeitet das Kommando. Ist die Kommandoverarbeitung beendet, löscht der CED den Eingabebereich, d.h. die Tabelle, und fügt an seine Stelle die Ausgabe des Kommandos , d.h. die alphabetisch geordnete Tabelle ein. Anschließend schaltet er in den Modus 'neuen Text eingeben' um. Ihr Bildschirm muß mit der jetzt sortierten Tabelle so aussehen.

Betr.: SINIX-Dokumentation.

Die folgende Tabelle enthaelt Begriffe, wie sie aus der englischen Originaldokumentation fuer die SINIX-Dokumentation uebersetzt wurden.

Datei	file
Dateiverzeichnis	directory
Geraetedatei	special file
Prozess	process
Schreibmarke	cursor
Shell	shell

Sie wollen neuen Text eingeben?
Bitte geben Sie Ihren Text ein!

Achtung

Es sind vor allem solche SINIX-Kommandos sinnvoll, die:

- von der Standardeingabe lesen und auf die Standardausgabe schreiben, wie z.B. grep, sort, usw.(siehe Kapitel 6).
- nur auf die Standardausgabe schreiben, z.B. who, date (siehe Kapitel 6).

SINIX-Kommandos, die eine Veränderung der Umgebung zur Folge haben, können zu seltsamen Ergebnissen führen.

Nachdem Sie Ihre Tabelle sortiert haben, möchten Sie vielleicht noch einen Schlußsatz unter diesen Brief setzen, z.B.:

Diese Liste wurde am Di 8.Nov 1983, 14:05:00 MEZ erstellt und soll laufend ergaenzt werden.

Wie bekommen Sie das aktuelle Datum in den Text? Natürlich wieder durch Angabe eines SINIX-Kommandos wie oben beschrieben.

Tippen Sie Ihren Text zunächst ohne das Datum in die oberste Leerzeile.
Ihr Brief könnte jetzt so aussehen:

** CED Texteditor V1.0 Zeile: 13 Spalte: 65 Name: brief

Betr.: SINIX-Dokumentation.

Die folgende Tabelle enthaelt Begriffe, wie sie aus der
englischen Originaldokumentation fuer die SINIX-Dokumen-
tation uebersetzt wurden.

Datei	file
Dateiverzeichnis	directory
Geraetedatei	special file
Prozess	process
Schreibmarke	cursor
Shell	shell

Diese Liste wurde am erstellt und soll laufend ergaenzt werden._

Sie wollen neuen Text eingeben?
Bitte geben Sie Ihren Text ein!

4.2.7 Wie können Sie einzelne Zeilen bearbeiten?

Um das Datum an der richtigen Stelle einzufügen, müssen Sie die gerade eingegebene Zeile an der richtigen Stelle teilen.

Im Modus 'eine Zeile bearbeiten' können Sie eine Zeile teilen. Drücken Sie dazu folgende Tasten:

MENU

Z

Positionieren Sie die Schreibmarke auf die Stelle, an der die Zeile geteilt werden soll.

T

Also:

Schalten Sie in den Modus 'eine Zeile bearbeiten' um. Positionieren Sie die Schreibmarke auf den ersten Buchstaben des Wortes 'erstellt', und teilen Sie die Zeile an dieser Stelle durch drücken der Taste **T**.

Der Rest der Zeile ab der Schreibmarkenposition ist in eine neue, zwischen diese und die nächste eingefügte Zeile geschoben worden.

Ihr Brief hat jetzt folgende Form:

```
** CED Texteditor V1.0           Zeile: 13 Spalte: 23 Name: brief
-----
    Betr.: SINIX-Dokumentation.
           Die folgende Tabelle enthaelt Begriffe, wie sie aus der
           englischen Originaldokumentation fuer die SINIX-Dokumen-
           tation uebersetzt wurden.

           Datei                file
           Dateiverzeichnis     directory
           Geraetedatei        special file
           Prozess              process
           Schreibmarke        cursor
           Shell                shell

Diese Liste wurde am _____
erstellt und soll laufend ergaenzt werden.
```

Sie wollen eine Zeile bearbeiten?
Bitte waehlen Sie eine der folgenden Funktionen: (e,l,r,t,v,z,<MENU>!!)

Das Datum fügen Sie im Modus 'Kommando ausführen' mit Hilfe des SINIX-Kommandos *datum* (siehe Kapitel 6) in den Text ein.

Beachten Sie dabei, daß Sie explizit keinen Eingabebereich festlegen. Sonst löscht der CED Teile des Textes. Ihre Kommandoingabe muß also sein:

0 datum

Drücken Sie folgende Tasten:

Positionieren der Schreibmarke auf die zweite Zeile des geteilten Satzes

MENU

K

0 datum

Das Datum wird in eine über der aktuellen Zeile eingefügt geschrieben.

```
*** CED Texteditor V1.0           Zeile: 14 Spalte: 1 Name: brief
-----
    Betr.: SINIX-Dokumentation.
           Die folgende Tabelle enthaelt Begriffe, wie sie aus der
           englischen Originaldokumentation fuer die SINIX-Dokumen-
           tation uebersetzt wurden.

           Datei                file
           Dateiverzeichnis     directory
           Genaetodatei        special file
           Prozess              process
           Schreibmarke        cursor
           Shell                shell

Diese Liste wurde am
Do 24.Mai.1984, 11:14:25 MEZ
erstellt und soll laufend ergaenzt werden.

-----
Sie wollen neuen Text eingeben?
Bitte geben Sie Ihren Text ein!
```

Um das Datum richtig in Ihren Schlußsatz einzufügen, müssen Sie die Zeile, die den ersten Teil des Satzes enthält und die Zeile, die das Datum enthält, zusammenfassen.

Drücken Sie folgende Tasten:

MENU

Z

Positionieren der Schreibmarke auf der ersten Zeile des Nachsatzes

V

Nachdem Sie im Modus 'eine Zeile bearbeiten' die Schreibmarke auf der oberen Zeile des Nachsatzes positioniert haben, verbinden Sie die beiden Zeilen zu einer durch die Taste **V**. Eventuell überflüssige Leerzeichen können Sie jetzt entfernen - mit der Taste **CHAR** - bzw. fehlende einfügen - im Modus 'Text einfügen', so daß Ihr Brief jetzt so aussieht:

```
** CED Texteditor V1.0           Zeile: 13 Spalte: 23 Name: brief
```

```
Betr.: SINIX-Dokumentation.  
Die folgende Tabelle enthaelt Begriffe, wie sie aus der  
englischen Originaldokumentation fuer die SINIX-Dokumen-  
tation uebersetzt wurden.
```

```
       Datei           file  
Dateiverzeichnis     directory  
Geratedatei         special file  
Prozess              process  
Schreibmarke        cursor  
Shell                shell
```

```
Diese Liste wurde am Do 24.Mai.1984, 11:14:25 MEZ  
erstellt und soll laufend ergaenzt werden.
```

Sie wollen in den Text etwas einfüegen?
Bitte füegen Sie Ihren Text ein!

Hinweis

Zeilen können Sie in diesem Modus nicht kopieren.

Wie Sie sicherlich schon gesehen haben, bietet der Modus 'eine Zeile bearbeiten' nicht die Möglichkeit, eine Zeile zu kopieren. Wollen Sie doch einmal einzelne Zeilen kopieren, schalten Sie in den Modus 'Zeilenbereich bearbeiten' um. Dann können Sie eine Zeile durch doppeltes Markieren als Bereich kennzeichnen und die gewohnten Operationen auf ihn anwenden.

Oder einfacher:

Benutzen Sie die Funktionstaste **F11**, um eine Zeile auszuwählen. Positionieren Sie jetzt die Schreibmarke an die neue Position, und drücken Sie die Funktionstaste **F12**. Die mit **F11** ausgewählte Zeile wird oberhalb der Schreibmarkenzeile eingefügt.

Nehmen Sie an, Sie benötigen diese Tabelle auch noch in einer anderen Datei. Natürlich können Sie in der Shell den gesamten Inhalt der Datei *brief* in eine andere kopieren. Es geht aber auch viel einfacher. Der CED bietet Ihnen die Möglichkeit, in andere Dateien umzuschalten und Daten hin und her zu bewegen.

4.2.8 Wie können Sie Daten zwischen Dateien hin- und herschieben?

Markieren Sie im Modus 'Zeilenbereich bearbeiten' die erste und letzte Zeile der Tabelle. Schalten Sie nun in den Modus 'Dokument wechseln'. Tragen Sie jetzt den Namen des anderen Dokuments, z.B. *brief.kopie* in die obere Zeile des Bedienbereichs ein und drücken Sie die Taste . Existiert die Datei und können Sie auf sie zugreifen, zeigt der CED im Arbeitsbereich die ausgewählte Datei an. Die Titelzeile enthält den Namen der Datei, die Schreibmarke befindet sich am Anfang dieser Datei. Der CED ist im Modus 'neuen Text eingeben'.

Meldet der CED

***** DATEI ANLEGEN? (j,n):

konnte der CED die angegebene Datei nicht öffnen. Entweder ist sie noch nicht vorhanden oder Sie dürfen auf Sie nicht zugreifen.

Antworten Sie mit n, schaltet der CED in den Modus 'neuen Text eingeben' zurück.

Antworten Sie mit j, versucht der CED, die Datei anzulegen und schaltet bei Erfolg in die neue Datei um.

Kann der CED die Datei nicht anlegen, z.B. auf Grund fehlender Schreibberechtigung im angegebenen Dateiverzeichnis, meldet er:

DOKUMENT KANN NICHT GEOEFFNET/ERZEUGT WERDEN.

und schaltet zurück in den Modus 'neuen Text eingeben'.

Was heißt umschalten? Sie können jetzt in dieser Datei ohne Einschränkungen arbeiten. Sind die von Ihnen gesetzten Marken in der Datei *brief* noch vorhanden? Ja! Den markierten Bereich können Sie so bearbeiten, als hätten Sie die Datei *brief* nie verlassen. Was müssen Sie nun tun, um die Tabelle zu kopieren?

Die Tastenfolge nach dem Markieren der Tabelle in der Datei *brief* ist:

brief.kopie

Auf die Frage: DATEI ANLEGEN? (j,n): antworten:

Im Modus 'Zeilenbereich bearbeiten' im Dokument *brief.kopie* drücken Sie einfach die Taste und der im Dokument *brief* markierte Bereich - die Tabelle - wird in das Dokument *brief.kopie* kopiert.

Wie können Sie in das alte Dokument zurückschalten?

Am schnellsten geht es mit der Funktionstaste . Der Name des zuletzt aktiven Dokuments, d.h. des Dokuments, das Sie vor dem letzten Umschalten bearbeitet haben, ist intern abgelegt. Die Funktionstaste bewirkt den Aufruf der richtigen CED-Kommandos.

Wenn Sie auf die ausführliche Weise zurückschalten wollen, wählen Sie wieder den Modus 'Dokument wechseln' und drücken Sie die Taste . Sie befinden sich wieder im Dokument *brief* im Modus 'neuen Text eingeben'.

Auch die Funktionen, die den internen Zwischenspeicher benutzen, arbeiten zwischen Dateien. Ein Beispiel. Um eine Zeile aus der Datei *brief* in die Datei *brief.kopie* zu verschieben, müssen Sie:

- MENU in der Datei *brief* in den Modus 'eine Zeile bearbeiten'
- Z schalten,
Schreibmarke auf die zu löschende Zeile positionieren;
- L die Zeile löschen; sie befindet sich jetzt im Zwischenspeicher;
- MENU
- D in den Modus 'Dokument wechseln' umschalten;
brief.kopie ↵ in die Datei *brief.kopie* umschalten;
- MENU
- Z in den Modus 'eine Zeile bearbeiten' schalten,
die Schreibmarke positionieren,
- Z die Zeile aus dem Zwischenspeicher zurückholen.

Jetzt ist es an der Zeit, den so mühsam erstellten Brief zu sichern. Denn bis jetzt haben Sie nur auf einer temporären Datei gearbeitet.

Wie können Sie Ihre Datei sichern?

Wollen Sie Ihre Datei unter einem anderen Namen als den, den Sie beim Aufruf des CED angegeben haben, sichern oder einmal während der CED-Sitzung zwischensichern, so können Sie dies im Modus 'abspeichern' tun. Drücken Sie die Tasten:

- MENU
- A
- ↵

Nach dem Drücken der Taste **↵** , wird die Datei unter Ihrem aktuellen Namen abgelegt - das ist der in der Kopfzeile angegebene. Wollen Sie sie unter einem Namen sichern, geben Sie bevor Sie die Taste **↵** drücken, diesen Namen ein. Drücken Sie nun die Taste **↵**. Der CED rettet Ihre Datei unter diesem Namen. Nach erfolgreichem Abspeichern - Sicherung ist der CED wieder im Modus 'neuen Text eingeben'.

Wollen Sie nur unter dem aktuellen Namen abspeichern genügt auch schon ein Druck auf die Funktionstaste **F10**. Ihr Dokument wird dann automatisch abgespeichert.

4.2.9 Beenden Sie Ihre CED-Sitzung

Drücken Sie die Tasten:

jetzt meldet sich der CED mit der Frage:

Wollen Sie Ihren Text sichern? Geben Sie j = Ja oder n = Nein ein!

Achtung

Antworten Sie jetzt unvorsichtigerweise mit n, war Ihre ganze Arbeit umsonst. Alle Änderungen, die Sie während dieser CED-Sitzung gemacht haben, sind gelöscht, falls Sie sie nicht zwischendurch abgespeichert haben. Geben Sie dagegen ein j ein, schreibt der CED Ihre Änderungen in die Datei: *brief*.

Haben Sie während der CED-Sitzung im Modus 'Dokument wechseln' andere Dateien geöffnet und diese noch nicht geschlossen, schaltet der CED jetzt zu der zuletzt aktiven um; in unserem Fall also zu *brief.kopie*. Diese können Sie jetzt auf dieselbe Weise sichern. Haben Sie während der CED-Sitzung noch weitere Dokumente geöffnet **und** verändert, so wechselt der CED automatisch zu all diesen Dateien. Sie können dann von Fall zu Fall entscheiden, ob Sie die Veränderungen in der jeweiligen Datei retten wollen oder nicht. Damit der CED weiterschaltet, müssen Sie die einzelnen Dateien im Modus 'verlassen' abspeichern oder nicht.

4.3 Sonstige Kommandos

Was gibt es noch für Modi und Kommandos?

Unser Beispiel ist zu kurz, um alle Modi und Kommandos des CED zu besprechen. Deshalb suchen Sie sich am besten eine Übungsdatei, die etwas größer als unsere Briefdatei ist. Oder, tippen Sie einfach ein etwas längeres Dokument in eine Datei.

4.3.1 Wie können Sie Texte in einem Dokument suchen?

Natürlich gibt es im CED auch die Möglichkeit, nach Zeichenketten zu suchen. Schalten Sie in den Modus 'suchen' um. Drücken Sie die Tasten:

In der oberen Zeile des Bedienbereichs können Sie die zu suchende Zeichenfolge eingeben. Alle Zeichen, auch führende und nachfolgende Leerzeichen, sind signifikant.

Drücken Sie jetzt die Taste oder die Taste , beginnt der CED, ab der Schreibmarkenposition nach der eingegebenen Zeichenfolge zu suchen. Hat der CED diese nicht bis zum Erreichen des Dateiendes gefunden, gibt er die Meldung aus:

***** TEXT NICHT GEFUNDEN

Wird die Zeichenkette gefunden, positioniert der CED die Schreibmarke auf deren ersten Buchstaben. Wollen Sie nach weiterem Auftreten der angegebenen Zeichenkette suchen, drücken Sie noch einmal die Taste oder .

Wie können Sie in der Datei vor der Schreibmarkenposition suchen?

Entweder Sie positionieren zuerst die Schreibmarke an den Anfang der Datei und beginnen dann zu suchen. Oder Sie suchen rückwärts, indem Sie den Suchvorgang durch Drücken der Taste **[↑]** auslösen.

Hinweis

Haben Sie eine Zeichenkette im Modus 'suchen' als Suchzeichenkette eingetragen, bleibt diese während der gesamten CED-Sitzung als solche erhalten. Wollen Sie sie löschen, drücken Sie im Modus 'suchen' die Tasten **[CTRL]** und **[X]** gleichzeitig.

Ist die Suchzeichenkette noch eingetragen, können Sie auch, ohne in den Modus 'suchen' zu schalten, nach dieser Zeichenkette suchen. Durch Drücken der Funktionstasten **[F13]** bzw. **[F14]** schalten Sie automatisch in diesen Modus und starten das Suchen vorwärts - **[F13]** -, bzw. rückwärts - **[F14]**. Anschließend befinden Sie sich wieder in Ihrem Ausgangsmodus.

Achtung

Der CED versteht im Gegensatz zu vielen anderen Kommandos keine regulären Ausdrücke (siehe ed-Kommando, Kapitel 6).

4.3.2 Wie können Sie das Fenster über das Dokument verschieben?

Schalten Sie durch Drücken der Tasten **[MENU]** **[F]** in den Modus 'Fenster positionieren'.

Wollen Sie das Fenster auf eine bestimmte Zeile positionieren?

Tragen Sie die Zeilennummer - ohne Vorzeichen - im Bedienbereich ein und drücken Sie die Taste **[↓]** oder eine der Tasten **[↑]** oder **[↓]**. Das Fenster wird über den gewählten Bereich geschoben; die gewählte Zeile wird oberste Zeile im Fenster. Die Schreibmarke befindet sich in dieser Zeile. Ist die eingegebene Zahl größer als die Anzahl der Zeilen, springt der CED auf die letzte Zeile der Datei.

Wollen Sie eine bestimmte Spalte zur ersten, bzw. letzten im Fenster machen, benutzen Sie in diesem Modus die Taste **[←]** bzw. die Taste **[→]**.

Oder wollen Sie das Fenster in festen Schrittweiten zeilen- oder spaltenweise über dem Dokument verschieben? Dann tragen Sie die Schrittweite - mit einem Vorzeichen, die Art des Vorzeichens spielt keine Rolle - in das freie Feld im Bedienbereich ein. Mit den Tasten **[←]** bzw. **[→]** verschieben Sie das Fenster nach links bzw. rechts in der angegebenen Schrittweite; mit den Tasten **[↑]** bzw. **[↓]** nach oben bzw. unten.

Zur Erinnerung: Auch mit den Tasten **[↑]** und **[↓]** - halbseitenweise -, und den Funktionstasten **[F15]** und **[F16]** können Sie das Fenster über das Dokument verschieben.

4.3.3 Wechseln in die Shell

Haben Sie sich auch schon einmal geärgert, daß Sie erst eine Editorsitzung beenden mußten, um ein einfaches Shell-Kommando auszuführen? Den CED können Sie vorübergehend unterbrechen, ohne daß das gerade bearbeitete Dokument erst gerettet werden muß. Drücken Sie die Tasten:

MENU

!

Der CED verabschiedet sich jetzt mit der Meldung:

> > > Aufruf cedsh, Rueckkehr zum CED mit Taste: END

und die Shell meldet sich mit dem Empfangs-Bereitsymbol \$. In der Shell können Sie nun beliebige SINIX-Kommandos ausführen. Allerdings hat diese sog. temporäre Shell eine niedrigere Priorität, so daß Sie, wenn Sie umfangreichere Arbeiten in der Shell durchführen wollen, doch besser den CED ganz verlassen.

Wollen Sie die temporäre Shell wieder verlassen, drücken Sie zu Beginn einer Kommandozeile die Taste **END**. Sie befinden sich nun wieder im CED und können Ihr Dokument weiter bearbeiten.

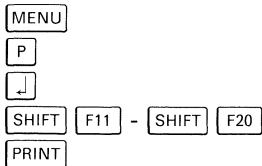
Haben Sie sich auch schon oft gewünscht, mit einem einzigen Tastendruck eine ganze Folge von Befehlen auslösen zu können? Der CED bietet Ihnen diese Möglichkeit.

4.3.4 Tasten mit beliebiger Zeichenfolge belegen

Hinweis

Tastenbelegungen sind nur für die aktuelle CED-Sitzung gültig.

Den folgenden Tasten kann man keinen Text zuweisen:



Wie können Sie einer Taste einen Text zuweisen?

Schalten Sie in den Modus 'Tasten belegen' um: **MENU** **P**, tippen Ihre gesamte Zeichen- und Befehlsfolge ein, schalten ein zweites Mal in den Modus 'Tasten belegen', und drücken die Taste, der Sie den gespeicherten Text zuweisen wollen.

Das klingt einfacher, als es meistens ist. Deshalb soll an zwei Beispielen die Belegung von Tasten erklärt werden.

Nehmen Sie an, Sie wollen einen Bericht über den Mississippi schreiben. In diesem Bericht wird der Name Mississippi oft vorkommen. Anstatt ihn nun jedes Mal neu einzutippen, können Sie auch die Zeichenkette Mississippi einer sonst nicht genutzten Taste zuordnen. Anschließend reicht dann ein Tastendruck, um das Wort Mississippi in den Text einzufügen.

Um eine Taste mit der Zeichenkette Mississippi zu belegen, schalten Sie zuerst in den Tastenbelegungsmodus um: **MENU** **P**. Der CED schaltet sofort wieder in den Modus 'neuen Text eingeben' oder 'einfügen' zurück. Daß Sie im Modus 'Tasten belegen' sind, erkennen Sie nur an dem Wort **PROTO**, das rechts neben der Versionsnummer des Editors in der Kopfzeile steht. Jedes von Ihnen jetzt eingegebene Zeichen wird vom CED gespeichert. Tippen Sie also Mississippi ein. Der Text Mississippi erscheint so im Dokument, als hätten Sie ihn im Texteingabemodus eingegeben. Jetzt wollen Sie den Text beenden und einer Taste zuweisen. Schalten Sie ein zweites Mal in den Modus 'Tasten belegen': **MENU** **P**

Der CED wird Sie auffordern:

***** GEWUENSCHTE TASTE DRUECKEN

Der Taste, die Sie als nächste drücken, wird die Zeichenkette, die Sie soeben eingegeben haben, zugeordnet. Üblicherweise benutzt man entweder selten gebrauchte Sonderzeichen, eine der vom CED nicht verwendeten Funktionstasten, außer den oben ausgeschlossenen, oder eine **CTRL**-Tastenkombination. Drücken Sie z.B. gleichzeitig die Tasten **CTRL** **A**. Jedes Mal, wenn Sie von nun an diese Tastenkombination drücken, wird dafür der Text Mississippi in Ihr Dokument eingetragen.

Achtung

Wie Sie gemerkt haben, führt der CED alle Ihre Eingaben sofort aus. D.h. im Modus 'Tasten belegen' können Sie sofort Ihre eingegebene Zeichenkette oder Kommandofolge überprüfen.

Das zweite Beispiel ist etwas komplexer. Hier sollen zu der Zeichenfolge auch einige CED-Kommandos gehören.

Stellen Sie sich vor, Sie wollten das Wort CED an einigen Stellen im Text durch das Wort Editor ersetzen. Ideal wäre, wenn Sie für die beiden Tätigkeiten - suchen und ersetzen - nur je einen Tastenanschlag benötigen.

Mit den bisher bekannten Kommandos ist das ziemlich umständlich: Sie müssen jedes Mal in das Menü Suchfunktionen umschalten, um das nächste Auftreten von CED zu finden. Wenn Sie dann entschieden haben, daß Sie es durch Editor ersetzen wollen, tippen Sie im Modus 'neuen Text eingeben' die ersten drei Buchstaben: Edi, schalten dann in den Modus 'einfügen' und tippen die restlichen drei Buchstaben: tor ein. Jetzt schalten Sie wieder in den Modus 'suchen' usw.

Durch Belegen einer Taste können Sie sich die Arbeit erheblich vereinfachen. Legen Sie zuerst die Suchfunktion auf die Tastenkombination **CTRL** **B**. Dabei müssen Sie folgendes beachten: Das Eintragen der Suchzeichenkette CED sollte nicht mit zur Tastenbelegung gehören! Da der CED den Sucheintrag nach dem Suchen nicht löscht, würde jedes weitere Eintragen die Zeichenkette verlängern. Stattdessen können Sie sich dieses Gedächtnis zunutze machen. Schalten Sie in den Modus 'suchen' und tragen Sie CED als Suchzeichenkette ein. Ohne die Taste **↓** zu drücken und damit den Suchvorgang auszulösen, schalten Sie jetzt in den Modus 'Tasten belegen' um.

In der Kopfzeile muß jetzt PROTO erscheinen, d.h. alles, was Sie von nun an eingeben, können Sie nachher einer Taste zuordnen. Schalten Sie zuerst in den Modus 'suchen' um. Der Sucheintrag CED ist noch vorhanden. Drücken Sie die Taste **J**. Der CED beginnt, nach dem nächsten Auftreten von CED zu suchen. Ungeachtet, ob Sie dabei CED im Text gefunden haben oder nicht, schalten Sie wieder in den Modus 'neuen Text eingeben' und beenden Ihre Befehlsfolge, indem Sie wieder in den Modus 'Tasten belegen' umschalten. Drücken Sie die Tasten:

MENU

P

Auf die Aufforderung

***** GEWUENSCHTE TASTE DRUECKEN

drücken Sie jetzt gleichzeitig die Tasten **CTRL** und **B**.

Sie können jetzt ausprobieren, ob Sie alles richtig gemacht haben, indem Sie mehrere Male **CTRL** **B** drücken. Der CED sollte dann nach dem jeweils nächsten Auftreten von CED suchen.

Die zweite Funktion, das Ersetzen von CED durch Editor legen Sie jetzt auf die Tasten **CTRL** **C**. Positionieren Sie die Schreibmarke auf das C von CED, so wie es die CED-Suchfunktion beim Auffinden der Zeichenkette CED machen würde. Schalten Sie wieder in den Modus 'Tasten belegen'. Überschreiben Sie jetzt das Wort CED mit Edi. Schalten Sie um in den Modus 'einfügen' und tippen Sie die restlichen drei Buchstaben: tor ein. Schalten Sie wieder in den Texteingabemodus zurück. Beenden Sie jetzt den Modus 'Tasten belegen' und weisen Sie Ihre Eingabe der Tastenkombination **CTRL** **C** zu.

Jetzt haben Sie die Möglichkeit, mit zwei Tastenanschlägen das Wort CED durch Editor zu ersetzen. Positionieren Sie zuerst die Schreibmarke auf den Anfang Ihres Dokuments. Mit der Tastenkombination **CTRL** **B** suchen Sie jetzt das nächste Auftreten des Wortes CED; wollen Sie es durch Editor ersetzen, drücken Sie **CTRL** **C**.

4.3.5 Tastenbelegung anzeigen

Wenn das nicht so funktioniert wie beschrieben, lassen Sie sich Ihre Tastenbelegungen anzeigen. Drücken Sie die Tasten:

`MENU`
`X`.

Die Ausgabe müßte so aussehen:

`LB: MENU s RETURN MENU n`

`LC: E d i MENU e t o r MENU n`

Die Ausgabe: `RETURN` bezieht sich auf die Taste `↓`, `LBX` bedeutet die Tastenkombination `CTRL` und `X`,

Schreibmarkenbewegungen werden durch einfache Pfeile oder

UP,	Taste <code>↑</code>
DOWN,	Taste <code>↓</code>
HOME,	Taste <code>↶</code>
PJUMP,	Taste <code>↵</code>
MJUMP	Taste <code>↷</code>

angezeigt. Funktionstasten werden als `Pn` - $1 < n < 20$ - bzw. `Fn`, wenn gleichzeitig mit der Funktionstaste die Taste `SHIFT` gedrückt wurde, bezeichnet.

Schon belegte Tasten können Sie jederzeit neu belegen. Wollen Sie eine Tastenbelegung löschen, schalten Sie direkt nach dem ersten Umschalten in den Modus 'Tasten belegen' ein zweites Mal in ihn um. Die so entstandene leere Zeichenfolge weisen Sie der Taste zu, deren Belegung Sie löschen wollen.

4.3.6 Bildschirm neu aufbauen

Schickt Ihnen während einer CED-Sitzung an einem Mehrplatzsystem ein anderer Benutzer eine Meldung mit dem write-Kommando, wird durch die auf dem Bildschirm erscheinende Meldung der Bildschirmaufbau des CED zerstört. Auch durch Übertragungsfehler in der Leitung kann der Bildschirmaufbau verändert werden. Um den alten Bildschirminhalt wieder abgebildet zu bekommen, drücken Sie die beiden Tasten **CTRL** und **R** gleichzeitig. Der Bildschirm wird gelöscht und so wieder aufgebaut, wie er ausgesehen hatte, bevor er zerstört wurde.

5 Systemverwaltung

Die Rolle des Systemverwalters

Vielleicht ist Ihnen schon der Begriff Super-User begegnet. Auf jedem SINIX-System ist eine Benutzerkennung mit dem Namen: root und der Benutzernummer: 0 eingerichtet. Der Benutzer dieser Kennung wird als Super-User bezeichnet. Er kann sich über alle Schutzbits hinwegsetzen. Er kann also durch falsche Eingaben eventl. beträchtlichen Schaden anrichten. Deshalb sollte möglichst nur ein Mitarbeiter das root-Kennwort kennen.

Melden Sie sich unter der Benutzerkennung root ans System an, meldet sich die Shell mit dem Bereitzeichen # an Stelle des sonst üblichen \$. Das soll Sie darauf hinweisen, daß Sie beim Arbeiten unter dieser Kennung eine besondere Verantwortung haben.

Der Super-User wird im folgenden als Systemverwalter bezeichnet. Er muß die Benutzerkennung root benutzen.

Der Systemverwalter ist verantwortlich für den zuverlässigen Betrieb des gesamten Systems. Seine Aufgaben umfassen:

- Einrichten neuer Benutzerkennungen
- Überwachen der Plattenbelegung
- Datensicherung
- Noteingriffe
- Konfigurierung des Systems
- Formatieren von Disketten
- Installation neuer Softwareprodukte

Die Console

Die Datensichtstation eines Einplatzsystems heißt: Console

An einem Mehrplatzsystem ist die Datensichtstation die Console, die an der Systemeinheit am Anschluß CRT angeschlossen ist. Das Betriebssystem eines Mehrplatzsystems können Sie durch Ein- oder Ausschalten der Console hochfahren oder stoppen.

Das Betriebssystem schreibt seine Fehlermeldungen auf die Console. Sie können die Console ansprechen über die Datei: /dev/console.

5.1 Dateiverzeichnisse für die Systemverwaltung

/etc
/usr/lib

Die meisten für die Systemverwaltung notwendigen Kommandos finden Sie im Dateiverzeichnis: /etc. Um diese Kommandos aufzurufen, müssen Sie entweder den Gesamt-Pfadnamen des Kommandos angeben, oder vorher die Shell-Variable: PATH zusätzlich mit: /etc belegen. Einige Systemdateien und Kommandos für den Systemverwalter befinden sich im Dateiverzeichnis: /usr/lib.

Die Ausführungserlaubnis für Systemverwaltungskommandos ist auf den Eigentümer beschränkt. Die in /etc und /usr/lib enthaltenen Systemdateien sollten gegen unbefugtes Verändern geschützt sein.

In den folgenden Tabellen finden Sie alle in den Dateiverzeichnissen: /etc und /usr/lib enthaltenen Dateien, Angaben über ihren Inhalt, welches Kommando welche Dateien benutzt oder welche anderen Kommandos aufruft, und, ob die Datei vom Systemverwalter verändert oder das Kommando aufgerufen werden kann.

/usr/lib

Shell-Prozedur (S) Systemdatei (D) Kommando (K)	wird benutzt beim oder aufgerufen vom	veränderbar (v) aufrufbar (a)	beschrieben im Kapitel
atrun	K i. a. von /etc/cron	nein	5.15
calendar	K /bin/calendar	nein	Kap. 6
ced/helpq	D /bin/ced	nein	Kap. 6
crontab	D /etc/cron	v	5.15
diff3	K /bin/diff3	nein	Kap. 6
diffh	K /bin/diffh	nein	Kap. 6
digest	K /bin/lpr	nein	Kap. 6
kalender	K /bin/kalender	nein	Kap. 6
lp9001	K /etc/qdaemon	nein	5.21
lp9004	K /etc/qdaemon	nein	5.21
makekey	K /bin/ed, /bin/crypt	nein	Kap. 6
qconfig	D /bin/lpr	v	Kap. 6
qconfig.bin	D /bin/lpr	nein	Kap. 6
units	D /bin/units	v	

/etc

Shell-Prozedur (S) Systemdatei (D) Kommando (K)	wird benutzt beim oder aufgerufen vom	veränderbar (v) aufrufbar (a)	beschrieben im Kapitel
asktime	S /etc/rc	nein	5.18
checklist	D /etc/fsck	v, nicht sinnvoll	5.7
cron	K /etc/rc	nein	5.15
dcheck	K -	a	5.7
disable	K -	a, nicht sinnvoll	5.13
enable	K -	a, nicht sinnvoll	5.13
flchk und fldisp und flinit	K Menüsystem, /etc/superinstall	a	5.11 5.11 5.11
flformat	K -	a	5.11
fsck	K /etc/init	a	5.7
getty	K /etc/init	nein	5.17
group	D /bin/ls, /bin/newgrp	v	5.10
haltsys	K /etc/poweroff	a, im Notfall	5.3
herald/*	D /etc/getty	v	5.19
inir	K Systemhochfahren	nein	5.17
init	K Systemhochfahren	nein	5.17
mc	K /etc/asktime	a, (Uhr falsch)	5.18
mkfs	K Installation	a	5.12
mknod	K Instalation	a, nicht sinnvoll	5.14
motd	D /bin/login	v	5.5
mount	K /etc/rc	a	5.12
mtab	D /etc/mount und umount	nein	5.12
ncheck	K -	a	5.12
passwd	D /bin/login, /bin/ls, .	v	5.9
poweroff	S Systemabschalten	v, (a im Notfall)	5.3
qdaemon	K /etc/rc	a, (Neustart des Drucker-Systems)	5.21
rc	S /etc/init	v (Vorsicht)	5.17
superinstall	S Installation	a bei Neuinstall.	5.20
termcap	D /etc/getty, /bin/ced	v	5.22
ttys	D /etc/init, /etc/*able	v, nicht sinnvoll	5.17
ttytype	D /etc/getty	v, nicht sinnvoll	5.17
umount	K -	a	5.12
update	K /etc/rc	nein	5.16
utmp	D /bin/who	nein	Kap. 6
wall	K -	a	5.4

5.2 Das Betriebssystem hochfahren

Das Betriebssystem können Sie hochfahren:

- bei einem Mehrplatzsystem durch den Ein-/Ausschalter an der Console.
- bei einem Einplatzsystem durch den Netzschalter an der Systemeinheit (unten links)

5.3 Das Betriebssystem stoppen

```
/etc/poweroff  
/etc/haltsys
```

Um das Betriebssystem eines Mehrplatzsystems zu stoppen, brauchen Sie nur die Console auszuschalten (Drehknopf oben rechts betätigen). Das Betriebssystem eines Einplatzsystems stoppen Sie durch den Netzschalter an der Systemeinheit. Dadurch wird die Shell-Prozedur `/etc/poweroff` aufgerufen. Diese ruft das Kommando `/etc/haltsys` mit dem Parameter `1` auf. `/etc/haltsys` schreibt noch die in den Puffern befindlichen Daten auf die Platte. Anschließend markiert es das Dateisystem als "in Ordnung", stoppt das Betriebssystem und schaltet das System aus. Das Kommando `/etc/haltsys` können Sie im Notfall auch direkt eingeben.

Das `/etc/haltsys`-Kommando schaltet das Betriebssystem ohne Rücksicht auf laufende Benutzerprozesse ab. Sie sollten deshalb, bevor Sie am Mehrplatzsystem die Console abschalten bzw. das `haltsys`-Kommando eingeben, alle Benutzer mit Hilfe des `/etc/wall`-Kommandos benachrichtigen. Durch einen entsprechenden Eintrag in der Datei `/etc/poweroff` können Sie ein anderes Abschaltverhalten definieren. Sie können dort z.B. durch eine Shell-Prozedur erreichen, daß das Betriebssystem erst gestoppt wird, nachdem sich der letzte Benutzer abgemeldet hat.

Achtung

Kann man an einem Mehrplatzsystem das Betriebssystem durch Ausschalten der Console nicht stoppen, dann ist der Wippschalter an der Systemeinheit nach oben gedrückt. Drücken Sie ihn nach unten, dann läuft alles wie oben beschrieben. Ein nach oben gedrückter Wippschalter "überbrückt" die Ausschaltfunktion der Console.

5.4 Nachricht an alle Benutzer senden

`/etc/wall`

Mit dem `/etc/wall`-Kommando können Sie Nachrichten an alle angeschlossenen Benutzer senden. Deshalb ist dieses Kommando bei einem Einplatzsystem sinnlos; da kein anderer Benutzer gleichzeitig arbeiten kann. Das `/etc/wall`-Kommando setzt sich über eine durch das `mesg`-Kommando (siehe Kapitel 6) gesetzte Schreibsperre hinweg.

`/etc/wall [datei]`

Das `/etc/wall`-Kommando schickt den Inhalt der angegebenen Datei an alle angeschlossenen Benutzer. Haben Sie keine Datei angegeben, liest `/etc/wall` von der Standardeingabe.

Wichtig: Geben Sie die Nachricht von der Datensichtstation ein, müssen Sie sie mit der Taste `END` abschließen.

Beispiel

`$ /etc/wall`

Achtung! In 5 Minuten wird das System vorübergehend wegen Neuinstallation von Software gesperrt. Bitte beenden Sie Ihre Sitzung.

`END` \$

5.5 Login-Nachricht für alle Benutzer

`/etc/motd`

In diese Datei können Sie eine Nachricht eintragen, die jedem Benutzer beim Login ausgegeben wird. Existiert diese Datei noch nicht, können Sie sie anlegen.

5.6 Daten sichern

`tar`

Sie sollten regelmäßig Ihre Daten sichern, d.h. auf externe Datenträger kopieren. Dadurch verhindern Sie, daß durch Benutzer-, Hardware- oder Softwarefehler wichtige Daten verloren gehen. Haben Sie Ihre Daten gesichert, sparen Sie sich viel Arbeit bei der Rekonstruktion Ihrer Daten.

Ihnen steht zur Datensicherung das `tar`-Kommando zur Verfügung (siehe Kapitel 6). Mit diesem Kommando können Sie Teile Ihres Dateisystems auf Diskette kopieren oder von der Diskette wieder einlesen.

5.7 Dateisystem überprüfen

```
/etc/fsck
/etc/dcheck
/etc/ncheck
```

Das /etc/fsck-Kommando

Mit dem /etc/fsck-Kommando kann man ein auf Platte oder Diskette stehendes physikalisches Dateisystem überprüfen und reparieren. Was ist ein physikalisches Dateisystem? Jede Platte oder Diskette ist durch Blöcke strukturiert. Diese Blöcke sind in zwei "Bereichen" zusammengefaßt, die als physikalisches Dateisystem bezeichnet werden, sie haben die Namen: /dev/root und /dev/usr. In diesen beiden physikalischen Dateisystemen ist das "logische Dateisystem" abgebildet, d.h. der Dateibaum bestehend aus Dateien und Dateiverzeichnissen. In den Blöcken des physikalischen Dateisystems: /dev/usr sind alle Dateien und Dateiverzeichnisse abgebildet, die im "logischen Dateisystem" vom Dateiverzeichnis: /usr direkt oder indirekt ausgehen. Das physikalische Dateisystem: /dev/root enthält alle anderen Dateien oder Dateiverzeichnisse des Dateibaums.

Andere Namen für die physikalischen Dateisysteme sind: /dev/rroot und /dev/rusr. Das r steht für: raw device. Über diese Namen ist ein zeichenweiser Zugriff möglich.

/etc/fsck [schalter] [dateisystem]

- schalter
- y fsck nimmt an, die Antwort auf alle Fragen sei "yes".
 - n fsck nimmt an, die Antwort auf alle Fragen sei "no".
 - t fsck benutzt eine Hilfsdatei, falls es im Hauptspeicher nicht genügend Platz für den Aufbau seiner Tabellen hat. fsck nimmt den nächsten Parameter als Namen dieser Hilfsdatei. Ist der Schalter -t nicht angegeben, erfragt fsck den Namen der zu benutzenden Hilfsdatei. Die Datei sollte sich nicht auf dem zu untersuchenden Dateisystem befinden. Die Hilfsdatei wird nach der Beendigung von fsck gelöscht, falls sie vorher noch nicht existierte oder es sich um eine Datei handelte, über die ein Gerät angesprochen wurde.
 - s Wiederherstellen der Liste der freien Blöcke.
Achtung: Das zu prüfende Dateisystem muß vorher mit dem umount-Kommando abgehängt werden und der Systemverwalter sollte alleine an der Anlage arbeiten.

dateisystem Name des zu überprüfenden physikalischen Dateisystems.
Auf Ihrer Festplatte sind zwei physikalische Dateisysteme eingerichtet: /dev/usr und /dev/root. Die Diskette hat den Namen: /dev/fl2 .

/etc/fsck überprüft:

1. Ob es Blöcke gibt, auf die mehrfach verwiesen wird.
2. Ob es Blöcke gibt, deren Adresse außerhalb der Grenzen des physikalischen Dateisystems liegt.
3. Falsche Verweiszähler.
4. Falsche Anzahl von Blöcken in einer Datei.
5. Ob die Größe eines Dateiverzeichnisses ein Vielfaches von 16 Byte ist.
6. Falsche Indexeintrag-Formate.
7. Ob es Blöcke gibt, auf die es keine Verweise gibt.
8. Ob es Indexnummern gibt, die außerhalb der Grenzen der Liste der Indexeinträge liegen.
9. Ob es im Superblock mehr als 65536 Indexnummern gibt.
10. Ob mehr Blöcke für Indexeinträge beansprucht werden, als im Dateisystem vorhanden sind.
11. Falsches Format der Freispeicherliste.
12. Falsche Gesamtanzahl der freien Blöcke und/oder der freien Indexnummern.

Findet fsck Dateien, deren Indexeintrag noch erhalten ist, die aber in keinem Dateiverzeichnis enthalten sind, schreibt es diese Indexeinträge, bei Zustimmung des Systemverwalters, in das Dateiverzeichnis: /lost+found. Der Name der Datei ist ihre Indexnummer. Das Dateiverzeichnis /lost+found muß vorher existieren und leere Plätze enthalten, in die Einträge gemacht werden können. Das erreichen Sie, indem Sie das Dateiverzeichnis anlegen, einige Dateien in dieses Dateiverzeichnis eintragen und diese wieder löschen.

Das /etc/ncheck-Kommando

Mit dem ncheck-Kommando kann man eine Liste erzeugen, die Pfadnamen und dazugehörige Indexnummer enthält.

/etc/ncheck **[-i inummer...][-a][-s][dateisystem]**

Schalter und Parameter

- i ncheck interpretiert die folgenden Parameter als Indexnummern, zu denen es die dazugehörigen Pfadnamen auflisten soll.
- a ncheck gibt neben der sonstigen Ausgabe auch Dateien aus, die mit "." oder ".." beginnen.
- s ncheck gibt nur Pfadnamen von Dateien aus, über die man entweder Geräte anspricht oder bei denen das s-Bit für den Eigentümer gesetzt ist. Mit dieser Funktion kann man gezielt Verletzungen der Sicherheitsregeln suchen.

Wird kein Schalter angegeben, gibt ncheck alle Pfadnamen aus, die zum angegebenen oder standardmäßig angenommenen Dateisystem gehören.

dateisystem Name eines physikalischen Dateisystems.
Wenn Sie keine Angabe machen, untersucht ncheck nacheinander /dev/root und /dev/rusr (siehe fsck-Kommando).

Das /etc/dcheck-Kommando

Das dcheck-Kommando überprüft die Dateiverzeichnisse eines Dateisystems. Es vergleicht die Verweiszähler jedes Indexeintrags mit der Häufigkeit des Auftretens seiner Indexnummer in allen Dateiverzeichnissen.

Unterschiede werden gemeldet.

Sie können das Kommando auch dazu benutzen, zu gegebenen Indexnummern den Dateinamen und das Dateiverzeichnis, in dem diese Datei enthalten ist, zu suchen.

/etc/dcheck **[-i inummer...] dateisystem**

Schalter

- i dcheck interpretiert die folgenden Parameter (bis auf Dateiname) als Indexnummern. Jedes Auftreten dieser Indexnummern in einem Dateiverzeichnis wird gemeldet mit: Indexnummer, Indexnummer des Dateiverzeichnisses, Name der Datei. Findet dcheck eine Datei, bei der Verweiszähler und Anzahl des Auftretens in Dateiverzeichnissen nicht übereinstimmt, gibt dcheck diese Information aus. Dateien, deren Verweiszähler auf Null steht und die in keinem Dateiverzeichnis eingetragen sind, werden auch aufgelistet. Die einzig gefährliche Situation ist, wenn die Anzahl der Einträge größer ist, als die Anzahl der Verweiszähler der Indexeinträge. Wenn dann Einträge gelöscht werden und der Verweiszähler dadurch auf Null gefallen ist, bleiben Einträge zurück, die auf nicht mehr gültige Einträge zeigen. Deshalb sollten diese Einträge gelöscht werden. Wenn der Verweiszähler größer als die Anzahl der Einträge ist, kann es höchstens dazu kommen, daß Plattenplatz vergeudet wird, wenn es keine Einträge mehr gibt, der Verweiszähler aber noch größer Null ist.

dateisystem

Der Name des zu überprüfenden physikalischen Dateisystems.
(siehe: fsck-kommando)

5.8 Plattenbelegung überprüfen

df, du, quot und find

Zur Überwachung der Plattenbelegung stehen die folgenden Kommandos zur Verfügung:

- df Damit können Sie den noch freien Platz in einem physikalischen Dateisystem abfragen.
- du Damit können Sie abfragen, wieviele Blöcke von bestimmten Dateien oder Dateiverzeichnissen belegt werden.
- quot Damit können Sie den von jedem Benutzer belegten Platz auf allen Dateisystemen abfragen.
- find Damit finden Sie Dateien, die bestimmte Bedingungen erfüllen, z.B. Dateien, auf die seit einer bestimmten Zeit nicht mehr zugegriffen wurde.

Eine vollständige Beschreibung dieser Kommandos finden Sie im Kapitel 6.

Hinweis

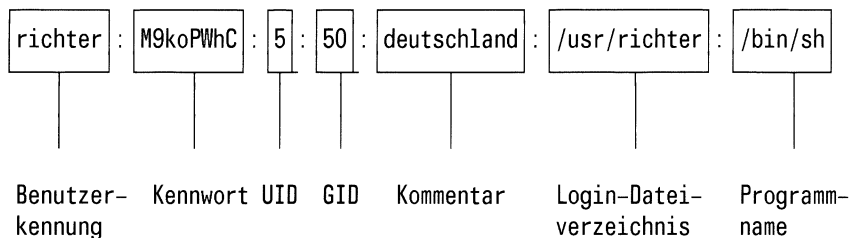
Achten Sie darauf, daß mindestens immer 15 Prozent des physikalischen Dateisystems: /dev/usr frei sein sollten.

5.9 Neue Benutzer definieren

`/etc/passwd`

Neue Benutzer dürfen Sie nur über das Systemverwaltermenü: Login-Administration definieren. Wenn Sie einen Benutzer lediglich in der Datei `/etc/passwd` definieren, erzeugen Sie damit Inkonsistenzen im System (siehe auch: Kapitel 5.23, Tips für den Systemverwalter). Auf die Datei: `/etc/passwd` sollten Sie nur direkt zugreifen, um eventl. ein Kennwort zu löschen oder den Kommentareintrag zu ändern. Dazu benutzen Sie am besten den CED-Editor.

Das folgende Beispiel zeigt einen vollständigen Eintrag für einen Benutzer in der Datei `/etc/passwd`.



Bedeutung der Einträge:

Jeder Eintrag muß vom anderen durch einen Doppelpunkt getrennt sein. Wird ein Eintrag weggelassen (z.B. Kommentar), muß man den folgenden Doppelpunkt trotzdem angeben.

Benutzerkennung

Hier wird eine Benutzerkennung definiert, die der Benutzer beim Login eingeben muß.

Kennwort

Diesen Eintrag können Sie nicht selbst machen (außer sie schreiben ein vorhandenes verschlüsseltes Kennwort ab). Ein Kennwort definieren Sie mit dem `passwd`-Kommando (siehe Kapitel 6). Das `passwd`-Kommando trägt ein Kennwort verschlüsselt an dieser Stelle ein.

UID Hier wird eine Benutzernummer (UID) eingetragen. Durch sie ist die im ersten Feld eingetragene Benutzerkennung im System eindeutig identifiziert. Jede Benutzerkennung bekommt deshalb eine eigene UID zugewiesen.

GID Hier wird eine Gruppennummer (GID) eingetragen. Durch sie ist jeder Benutzer einer Gruppe von Benutzern zugeordnet (siehe auch: `/etc/group`-Datei).

Kommentar Dieser Eintrag kann entfallen. Üblicherweise wird hier Information über den Benutzer eingetragen.

Login-Dateiverzeichnis

Hier wird der Pfadname eingetragen, der zum Login-Dateiverzeichnis des Benutzers führt.

Programmname

Hier wird der Name des Programms eingetragen, das nach dem Login des Benutzers gestartet werden soll.
Standard: `/bin/sh` .

Die Datei `/etc/passwd` enthält standardmäßig folgende Einträge:

```
root:chUkyKx8z60P.:0:0:Systemverwalter und super user:/:bin/sh
admin:PvmNsgLmv:0:3:Menu System Administrator:/usr/admin:/usr/menus/sabin/ums
daemon:x:1:0:Manager fuer automatischen Aufruf:/usr/spool/lpd:/bin/sh
cron:x:1:0:Manager fuer automatischen Aufruf:/usr/lib:/bin/sh
sys:x:2:2:System Manager:/usr/sys:/bin/sh
bin:x:3:3:Kommando Manager:/usr:/bin/sh
gast:x:4:1:SINIX Gastbenutzer:/usr/gast:/bin/sh
mgast::5:1:einzige Kennung ohne Kennwort:/usr/mgast:/usr/menus/sabin/ums
```

An diese Einträge können sich beliebig viele Einträge für Benutzer anschließen.

Hinweis

Die Datei `/etc/passwd` muß für alle Benutzer lesbar sein, da das `ls`-Kommando sie benötigt, um einer UID Benutzerkennungen zuzuordnen zu können.

5.10 Benutzergruppe definieren

`/etc/group`

Jeder Benutzer ist durch die Angabe einer Gruppennummer (in der Datei `/etc/passwd`) einer Benutzergruppe zugeordnet. Dieser Nummer wird in der Datei `/etc/group` ein Name zugeordnet.

Ein Benutzer kann vorübergehend in eine andere Benutzergruppe wechseln, wenn er in der Datei `/etc/group` für diese andere Gruppe als Berechtigter eingetragen ist. Zum temporären Wechseln der Benutzergruppe können Sie das `newgrp`-Kommando (siehe Kapitel 6) benutzen. Dieses Kommando benötigt die Datei `/etc/group`.

Das folgende Beispiel zeigt einen vollständigen Eintrag für die Datei `/etc/group`.

<code>programmierer</code>	<code>:</code>	<code>M5t9oK</code>	<code>:</code>	<code>50</code>	<code>:</code>	<code>richter,coyne,ryder</code>
Gruppenname		Kennwort		GID		Benutzerkennungen

Bedeutung der Einträge: Jeder Eintrag muß vom anderen durch einen Doppelpunkt getrennt sein. Wird ein Eintrag weggelassen, muß man den folgenden Doppelpunkt trotzdem angeben.

Gruppenname Dieser Eintrag ist Pflicht. Damit wird ein Gruppenname vergeben.

Kennwort Dieser Eintrag kann entfallen. Ist ein Kennwort definiert, muß es der Benutzer angeben, der in diese Gruppe wechseln will. Wie können Sie dieses Kennwort definieren? Nur mit Tricks. Es gibt kein Kommando, um ein Gruppenkennwort direkt definieren zu können. Was tun? Es gibt zwei Möglichkeiten:

- Man kann ein verschlüsseltes Kennwort abschreiben, dessen Bedeutung man im Klartext kennt.
- Man definiert mit dem `passwd`-Kommando (siehe Kapitel 6) in der Datei: `/etc/passwd` unter einer "Dummy-Kennung" ein Kennwort und schreibt das verschlüsselte Passwort in die Datei: `/etc/group`.

GID Dieser Eintrag ist Pflicht. Dieser Eintrag entspricht der Gruppennummer (GID), wie sie in der Datei `/etc/passwd` steht. Für diese GID definieren Sie im Feld: Gruppenname einen Namen.

Benutzer An dieser Stelle brauchen Sie nur die Kennungen der Benutzer eintragen, denen es möglich sein soll, sich vorübergehend dieser Gruppe anzuschließen (siehe `newgrp`-Kommando, Kapitel 6).

5.11 Disketten

/etc/flformat
/etc/flchk
/etc/fldisp
/etc/flinit

Allgemeines über Disketten

An Ihrem System verwendeten Disketten müssen folgende Eigenschaften haben:

- double density mit doppelter Dichte beschreibbar
- 5 1/4 Zoll Durchmesser
- 96 tpi (tracks per inch) Spuren pro Inch
- two sided doppelseitig beschreibbar

Hinweis

Durch Verkleben der Aussparung an der Diskette oben rechts können Sie Ihre Diskette gegen Überschreiben schützen. Dieser Mechanismus ist umgekehrt wie bei anderen Disketten.

Eine Diskette ist in drei Bereiche eingeteilt:

- den Labelbereich, der Informationen über Eigentümer und Version enthalten kann;
- den Bootbereich, der ein Programm enthalten kann, das beim Systemstart abläuft; das ist allerdings nur für die Bootdisketten erforderlich;
- den Datenbereich zur Speicherung von Daten.

Auf die folgenden Bereiche können Sie zugreifen über Dateien im Dateiverzeichnis: /dev

- auf den Labelbereich über /dev/fl0 oder - zeichenweise - /dev/rfl0;
- auf den Bootbereich über /dev/fl1;
- auf den Datenbereich über /dev/fl2.

Die meisten Kommandos, die mit Disketten arbeiten, haben den Namen der entsprechenden Datei eingebaut, über die sie ein Gerät ansprechen.

Bevor Sie eine neue Diskette benutzen können, müssen Sie diese mit dem /etc/flformat-Kommando formatieren.

Das /etc/flformat-Kommando

Mit dem /etc/flformat-Kommando können Sie eine Diskette formatieren. Legen Sie, bevor Sie das /etc/flformat-Kommando aufrufen, die Diskette in das Diskettenlaufwerk.

Das /etc/flformat-Kommando teilt die Spuren einer Diskette in Sektoren von 256 Byte ein. Formatierte 5 1/4 Zoll Disketten haben eine Kapazität von 0,65 MB = 650 kB.

Achtung

Das /etc/flformat-Kommando arbeitet ohne Rücksicht auf etwaige Inhalte der Diskette. Haben Sie auf der Diskette Daten gespeichert, werden diese überschrieben.

Labelbereich auf einer Diskette initialisieren, prüfen, ausgeben: flinit-, flchk- und fldisp-Kommando.

Zur softwaremäßigen Identifizierung Ihrer Disketten, ist auf ihnen ein besonderer Bereich, der Labelbereich eingerichtet. Den Labelbereich können Sie ansprechen über die Datei /dev/fl0. In diesen Bereich können Versionsnummer und Eigentümer eingetragen werden.

Zur Initialisierung, Überprüfung und Ausgabe dieses Bereichs stehen Ihnen die Kommandos flinit, flchk und fldisp zur Verfügung.

Diese Kommandos werden vom Menüsystem bei der Bearbeitung von Disketten benutzt. Das tar-Kommando benutzt den Labelbereich nicht.

Labelbereich initialisieren: flinit

/etc/flinit [-v vsn] [-o owner]

Wird /etc/flinit ohne Parameter aufgerufen, wird die Standardeinstellung für -v und -o ausgeführt.

- v vsn Hier müssen Sie eine max. sechsstellige alphanumerische Versionsnummer angeben, die in den Labelbereich eingetragen werden soll. Geben Sie keine Versionsnummer an, wird eine leere Zeichenkette eingetragen.
- o owner Hier können Sie einen maximal 14 Zeichen langen Eigentümernamen angeben. Geben Sie nichts an, werden 14 Leerzeichen eingetragen.

Labelbereich überprüfen: flchk

`/etc/flchk [-i][-v vsn][-o owner]`

- `-i` Bei Unterschieden zwischen Versionsnummern oder Eigentümern, fragt flchk nicht nach und beendet sich sofort mit dem exit-Status 1.
- `-v vsn` flchk vergleicht die angegebene Versionsnummer mit der auf der Diskette eingetragenen. Sind sie verschieden, fragt flchk, ob der Unterschied ignoriert werden soll. Antworten Sie mit nein, endet flchk mit exit-Status 1.
- `-o owner` flchk vergleicht den angegebenen Eigentümer mit dem auf der Diskette eingetragenen. Sind sie verschieden, fragt flchk, ob der Unterschied ignoriert werden soll. Antworten Sie mit nein, endet flchk mit exit-Status 1.

Labelbereich ausgeben: /etc/fldisp [-o][-v]

`/etc/fldisp` gibt Eigentümer und Versionsnummer der eingelegten Diskette aus.

- `-o` Namen des Eigentümers ausgeben
- `-v` Versionsnummer ausgeben

Standard: `-v` und `-o`

5.12 Dateisystem auf Disketten

`/etc/mkfs`
`/etc/mount`
`/etc/umount`

SINIX bietet die Möglichkeit, an das bestehende Dateisystem andere Dateisysteme anzuhängen. Diese anderen Dateisysteme bilden für sich ein eigenes baumartig strukturiertes Dateisystem. An das bestehende Dateisystem angehängt, bilden sie einen Unterbaum des gesamten Dateisystems. Für den Benutzer ist nicht sichtbar, ob das Dateisystem, in dem er gerade arbeitet, ein solches Teil-Dateisystem ist oder nicht (siehe Bild 5-1).

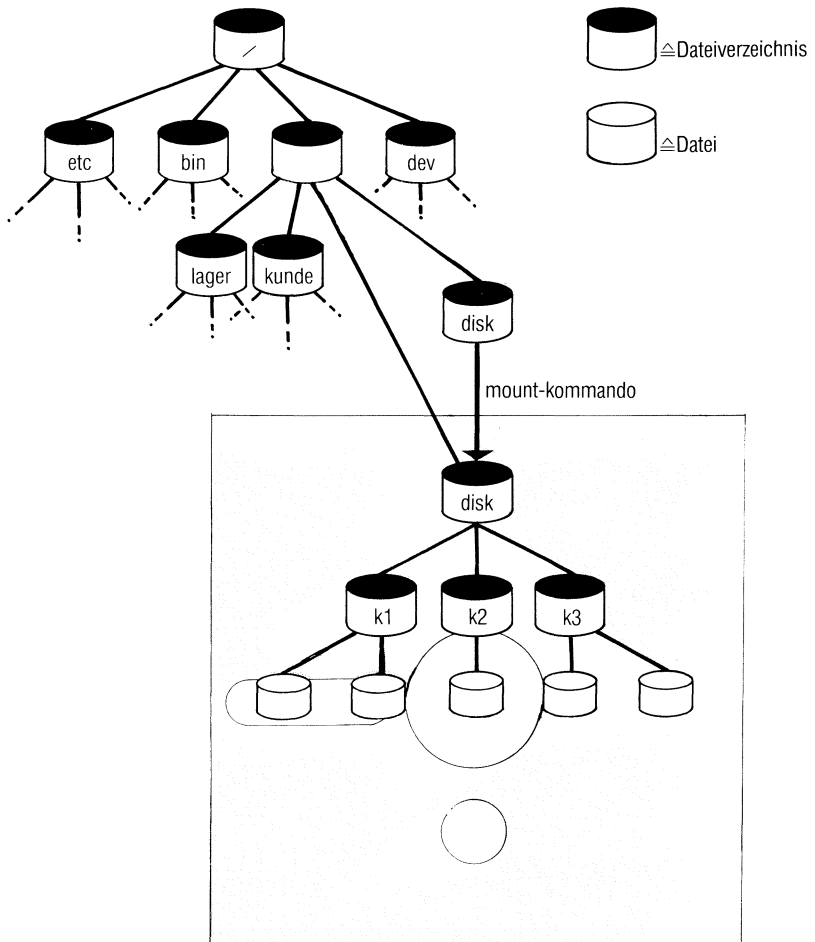


Bild 5-1 Dateisystem auf Diskette

Wenn Sie auf einer Diskette ein eigenes Dateisystem erstellen und an das bestehende Dateisystem anhängen wollen, müssen Sie folgende Kommandos benutzen:

- `/etc/mkfs-`, `/etc/mount-` und `/etc/umount` .

Das `/etc/mkfs`-Kommando

Lesen Sie diese Kommandobeschreibung gut durch, bevor Sie das erste Mal das `mkfs`-Kommando ausprobieren. Ein falsch eingegebenes `mkfs`-Kommando kann viel zerstören.

Das `mkfs`-Kommando erzeugt auf einer formatierten Diskette oder in einer Datei zwei Dinge:

- ein physikalisches Dateisystem
- ein "logisches Dateisystem". Dieses Dateisystem besteht aus mindestens einem Dateiverzeichnis (siehe: Beispiel 1) oder wird nach in einer Datei definierten Angaben aufgebaut.

Für eine Diskette ist ein physikalisches Dateisystem die Voraussetzung, um darauf ein "logisches Dateisystem" aufbauen zu können, d.h. ein Dateisystem das aus Dateien und Dateiverzeichnissen besteht (siehe auch `fsck`-Kommando). Sie können mit dem `mkfs`-Kommando ein physikalisches/logisches Dateisystem entweder direkt auf einer Diskette einrichten oder es zuerst in einer Datei erstellen und anschließend auf eine Diskette kopieren. Ein auf einer Diskette stehendes physikalisches/logisches Dateisystem können Sie durch das `/etc/mount`-Kommando mit dem bestehenden Dateibaum verbinden.

`/etc/mkfs [-n] [-y] name1 name2`

- `m` `mkfs` kopiert die Modifikationszeiten der Dateien, die Sie in der Datei `name2` angegeben haben.
- `n` Das `mkfs` – Kommando wird beendet, sobald festgestellt wird, daß die unter `name1` angegebene Diskette oder Datei bereits Daten enthält.
- `v` Diesen Schalter müssen Sie angeben, wenn Sie bei `name2` einen Dateinamen angeben und in dieser Datei Shell-Variablen substituiert werden sollen.
- `y` `mkfs` überprüft nicht, ob die unter `name1` angegebene Datei oder Diskette bereits Daten enthält. Es wird in jedem Fall ein neues physikalisches/logisches Dateisystem angelegt.

name1 Zwei Angaben sind möglich:

- Name der Datei, in der Sie ein physikalisches/logisches Dateisystem einrichten wollen
- Name der Gerätedatei, über die Sie die Diskette ansprechen, auf der das neue physikalische/logische Dateisystem erzeugt werden soll. Der Name ist: /dev/fl2.

Achtung

Die Angabe eines falschen Dateinamens kann katastrophale Folgen haben. Befindet sich eine Diskette im Laufwerk, die schon ein Dateisystem oder Daten enthält, werden diese zerstört. Das können Sie durch den n-Schalter vermeiden.

name2 Zwei Angaben sind möglich: (siehe auch Schalter: -m und -n)

- Die maximale Anzahl der Blöcke, die auf die Diskette passen (z.B. 580 Blöcke für eine 5 1/4 Diskette) (siehe: Beispiel 1). Diese Angabe müssen Sie machen, wenn Sie unter name1 angegeben haben: /dev/fl2.
- Den Namen einer Datei, die Definitionen enthält, wie das physikalisch/logische Dateisystem auf der Diskette aussehen soll.

Hinweis

Diese Datei hat nichts mit der unter name1 angegebenen Datei zu tun (siehe: Definition eines physikalisch/logischen Dateisystems in einer Datei).

Beispiel 1

Eingabe: /etc/mkfs /dev/fl2 580

Jetzt wird auf der im Diskettenlaufwerk: /dev/fl2 eingelegten, formatierten Diskette ein physikalisches Dateisystem erzeugt. Es besteht aus einem leeren Dateiverzeichnis, das als Wurzel des neu aufzubauenden Dateisystems dient. Wie können Sie auf dieses Dateiverzeichnis zugreifen? Dazu müssen Sie das neue Dateisystem auf der Diskette mit dem bestehenden Dateisystem verbinden. Das machen Sie mit dem /etc/mount-Kommando.

Definition eines physikalischen/logischen Dateisystems in einer Datei

Sie können in einer Definitionsdatei den Aufbau eines physikalischen/logischen Dateisystems festlegen. Den Namen für diese Datei können Sie frei wählen. Schreiben Sie den Namen dieser Definitionsdatei beim `mkfs`-Kommando an die Stelle `name2`, dann wird ein den Definitionen in dieser Datei entsprechendes physikalisches/logisches Dateisystem erstellt. Mit `name1` geben Sie an, wo dieses Dateisystem erstellt werden soll, d.h. direkt auf einer Diskette (`/dev/f12`) oder in einer Datei (beliebiger Name).

Haben Sie in einer Definitionsdatei ein großes physikalisches/logisches Dateisystem definiert, empfiehlt es sich, diese Definitionen nicht direkt auf einer Diskette ausführen zu lassen, sondern sie über eine Datei (beliebiger Name) auszuführen (siehe Angabe: `name1`). Bei einer Ausführung der Definitionsdatei über eine Datei wird das physikalisch/logische Dateisystem schneller aufgebaut; außerdem haben Sie dann das komplett aufgebaute Dateisystem in dieser Datei, von wo aus Sie es leicht und schnell auf beliebig viele Disketten kopieren können.

Eine Datei, in der Sie ein physikalisch/logisches Dateisystem definieren, muß einen bestimmten Aufbau haben, damit das `mkfs`-Kommando sie lesen kann. Die Einträge in dieser Datei müssen durch Leerzeichen oder `<Neue Zeile>` voneinander getrennt sein. Eine Beispieldatei:

```
:
580 110
d-777 3 1
usr d-777 3 1
  sh  --755 3 1 /bin/sh
  ben d-755 8 1
    joe --777 8 1
    jeff --777 8 1
$
paul d-777 9 1 /usr/paul
$
b0   b-544 3 1 0 0
c0   c-544 3 1 0 0
$
$
```

Was bedeuten die einzelnen Einträge?

Damit die einzelnen Einträge in der Datei erkannt werden, muß die Datei in der **ersten Zeile** "nichts" enthalten. Dieses "nichts" wird in den ersten Block auf der Diskette geschrieben. Es empfiehlt sich deshalb ein :-Kommando dorthin zu schreiben.

Die **zweite Zeile** enthält eine Definition der Anzahl der Blöcke (hier: 580) des einzurichtenden Dateisystems sowie eine Angabe, wieviele Indexnummern vergebbar sein sollen (hier: 110).

Die **dritte Zeile** enthält Definitionen über den Dateityp des erstens zu erstellenden Eintrages. Sie besteht aus einer 5 Zeichen langen Zeichenkette gefolgt von zwei Zeichen, die durch ein Leerzeichen voneinander getrennt sein müssen. Als erstes Zeichen der 5 Zeichen langen Zeichenkette (hier: d) kann man eintragen:

– = Datei

d = Dateiverzeichnis

b = Blockorientierte Datei für Geräte

c = Zeichenorientierte Datei für Geräte

Wird ein d eingetragen erzeugt das mkfs-Kommando ein Dateiverzeichnis mit den Einträgen : . und .. Dieses Dateiverzeichnis ist dann das root-Dateiverzeichnis für den folgenden Dateibaum (siehe: vierte Zeile).

Als zweites Zeichen in der dritten Zeile (hier: –) muß man angeben:

u, g oder –

u = set – UID oder g = set-GID oder - = kein Eintrag

Mit dem dritten, vierten und fünften Zeichen in der dritten Zeile (hier: 777) werden Schutzbits definiert (siehe: chmod-Kommando).

Die beiden letzten Zeichen der dritten Zeile sind die Benutzernummer (UID) und die Gruppennummer (GID).

Ab der **vierten Zeile** kommen Einträge, die den am root-Dateiverzeichnis hängenden Dateibaum definieren. In der Beispieldatei hängt am root-Dateiverzeichnis das Dateiverzeichnis usr (Eintrag: d-777 3 1, siehe Erklärung für Zeile drei). In das Dateiverzeichnis usr wird eine Datei eingetragen mit dem Namen: sh. Diese Datei wird mit dem Inhalt aus der Datei gefüllt, deren Name ganz rechts steht (hier: /bin/sh). Außerdem wird in das Dateiverzeichnis usr das Dateiverzeichnis ben eingetragen. Das Dateiverzeichnis ben soll folgende Dateien enthalten: joe und jeff.

Ist der Eintrag für ein Dateiverzeichnis zu Ende, muß er mit einem \$-Zeichen abgeschlossen sein. Deshalb steht hinter dem Eintrag für die Datei jeff ein \$-Zeichen, d.h. der Eintrag für das Dateiverzeichnis ben ist beendet.

In das Dateiverzeichnis usr wird jetzt noch das Dateiverzeichnis paul eingetragen und mit dem Inhalt von /usr/paul gefüllt. Das \$-Zeichen gibt an, daß keine weiteren Einträge für das Dateiverzeichnis paul folgen. Als letztes werden noch die beiden Gerätedateien b0 und c0 in das Dateiverzeichnis usr eingetragen.

Das folgende \$-Zeichen kennzeichnet das Ende der Einträge für das Dateiverzeichnis usr. Das anschließende \$-Zeichen markiert das Ende der Definitionsdatei.

Das /etc/mount-Kommando

Mit den /etc/mount-Kommando verbindet man ein auf einer Diskette vorhandenes Dateisystem mit einem bestehenden Dateisystem. Dazu müssen Sie vorher im bestehenden Dateisystem ein leeres Dateiverzeichnis einrichten, in das das neue Dateisystem eingehängt werden soll. Über dieses Dateiverzeichnis können Sie nach erfolgreicher Durchführung des mount-Kommandos auf die Dateien und Dateiverzeichnisse des neu eingehängten Dateisystems zugreifen.

Hinweis

Bevor Sie eine "gemountete" Diskette aus dem Laufwerk nehmen, sollten Sie ein umount-Kommando eingeben. Es kann sonst zu Komplikationen kommen, wenn das Betriebssystem runter- und wieder aufgefahen wird.

/etc/mount [dateiname dateiverzeichnis [-r]]

dateiname	Name des Diskettenlaufwerks, auf dem sich die Diskette befindet, die das neue Dateisystem enthält. Der Standardname des Diskettenlaufwerks ist: /dev/f12.
dateiverzeichnis	Name des leeren Dateiverzeichnisses, in das das neue Dateisystem eingetragen werden soll. Dieses Dateiverzeichnis ist die Wurzel des neuen Dateisystems.
- r	Das neue Dateiverzeichnis soll nur lesbar sein, d.h. es darf nicht beschrieben werden.

Gibt man das /etc/mount-Kommando ohne Parameter an, bekommt man Informationen über alle angehängten Dateisysteme ausgegeben. Diese Informationen sind in der Datei /etc/mstab gespeichert.

Nach Eingabe des /etc/mount-Kommandos kann man auf die Dateien und Dateiverzeichnisse des neu eingehängten Dateisystems normal zugreifen, d.h. man kann mit ihm wie mit dem bestehenden Dateisystem arbeiten. Es sind lediglich keine Verweise (siehe ln-Kommando, Kapitel. 6) zwischen Dateien des bestehenden und des neuen Dateisystems möglich.

Hinweis

Das neue Dateisystem muß als "in Ordnung" markiert sein. Ein "defektes" neues Dateisystem wird vom bestehenden Dateisystem abgewiesen. Was kann man dann tun? Man bringt das "defekte" neue Dateisystem mit dem fsck-Kommando wieder "in Ordnung".

Beispiel für das /etc/mount-Kommando:

Eingabe: /etc/mount /dev/fl2 disk

Das neue Dateisystem auf der Diskette im Diskettenlaufwerk /dev/fl2, wird an das leere Dateiverzeichnis: disk im bestehenden Dateisystem angehängt.

Das /etc/umount-Kommando

Mit dem /etc/umount-Kommando hängt man ein Dateisystem ab, das man vorher mit dem /etc/mount-Kommando angehängt hatte. Bevor das /etc/umount-Kommando ein Dateisystem abhängt, prüft es, ob dafür noch Ein-/Ausgabeoperationen anstehen. Wenn ja, werden diese Operationen vor dem Abhängen noch ausgeführt. Ein abgehängtes Dateisystem wird als "in Ordnung" markiert.

/etc/umount dateiname

dateiname Name des Diskettenlaufwerks, auf dem sich die Diskette befindet, die das abzuhängende Dateisystem enthält. Der Standardname des Diskettenlaufwerks ist: /dev/fl2.

5.13 Datensichtstationen aktivieren und deaktivieren

/etc/disable
/etc/enable

Das /etc/disable-Kommando

Mit dem /etc/disable-Kommando kann man Datensichtstationen deaktivieren. Benutzen Sie dieses Kommando an einem Einplatzsystem, dann deaktivieren Sie damit Ihre Datensichtstation. Um wieder mit ihr arbeiten zu können, müssten Sie das Betriebssystem runter- und wieder rauffahren.

/etc/disable ttyn

ttyn n ist die Nummer einer Datensichtstation. Sie können die Nummer durch Eingabe des tty-Kommandos (siehe Kapitel 6) an der entsprechenden Datensichtstation abfragen.

Das /etc/enable-Kommando

Mit dem /etc/enable-Kommando kann man Datensichtstationen aktivieren und deaktivieren.

/etc/enable [-schalter] ttyn ...

schalter e Die Datensichtstationen werden aktiviert.
 d Die Datensichtstationen werden deaktiviert.

ttyn n ist die Nummer der Datensichtstation.

5.14 Datei für Geräte anlegen

`/etc/mknod`

Das `/etc/mknod`-Kommando erzeugt eine Datei, über die man ein Gerät ansprechen kann. Allerdings sollten Sie dieses Kommando nur benutzen, wenn Sie sich genau mit den Treibern für die Geräte Ihres SINIX-Systems auskennen. Um neue Geräte anzuschließen, benutzen Sie am besten den entsprechenden Service der Menüs.

`/etc/mknod dateiname [schalter] majornummer minornummer`

<code>dateiname</code>	Name der Datei. Er muß im Dateiverzeichnis <code>/dev</code> eingetragen werden.
<code>schalter</code>	<ul style="list-style-type: none"><code>b</code> Datei für ein blockorientiertes Gerät (z.B. Platten- speicher oder Diskette).<code>c</code> Datei für ein zeichenorientiertes Gerät (z.B. Drucker, Datensichtstation, Diskette im raw-Modus).
<code>majornummer</code>	Die majornummer bezeichnet den Typ des Gerätes.
<code>minornummer</code>	Mit der minornummer zählt man die Geräte des gleichen Typs an der Anlage hoch.

5.15 Periodische Tätigkeiten anstoßen

`/etc/cron`

Das `/etc/cron`-Kommando führt Routinearbeiten zu festgelegten Zeiten aus. Routinearbeiten sind z.B. Aufräumarbeiten, wie das Löschen temporärer Dateien oder Arbeiten zur Datensicherung.

Das `/etc/cron`-Kommando überprüft im Minutenabstand die Datei `/usr/lib/crontab`. In dieser Datei stehen Kommandos und die Zeiten, zu denen sie durch `/etc/cron` ausgeführt werden sollen. Er wird von der Shell-Prozedur `/etc/rc` gestartet, die `/etc/init` beim Hochfahren des Systems ausführt. `/etc/cron` ist während der gesamten Laufzeit des Systems aktiv.

Die Datei `/usr/lib/crontab` ist eine Tabelle, die aus Zeilen zu je 6 Felder besteht. Eine Zeile in der Datei: `/usr/lib/crontab` hat folgendes Format:

Feld	1	2	3	4	5	6
	Minute	Stunde	Tag im Monat	Monat	Tag der Woche	Kommando

·
·

Die Felder 1 bis 5 legen die Zeit fest, zu der cron das im Feld 6 stehende Kommando ausführen soll.

Minute Mögliche Angaben: 0-59

Stunde Mögliche Angaben: 0-23

Tag im Monat Mögliche Angaben: 1-31

Monat Mögliche Angaben: 1-12

Tag der Woche Mögliche Angaben: 1-7, Montag= 1 usw.

Kommando Gesamt-Pfadname des Kommandos, das durch cron zur festgelegten Zeit ausgeführt werden soll.

Die Felder 1 bis 5 können auf folgende Arten angegeben werden:

- n Eine Zahl im gültigen Bereich.
- n-n Eine Bereichsangabe
- * Ein Stern, der alle Werte im zulässigen Bereich umfaßt.

Die Datei: `/usr/lib/crontab` kann beliebig viele Kommentarzeilen enthalten. Eine Kommentarzeile muß in Spalte 1 das Nummernzeichen `#` enthalten.

Beispiel

Eine typische Datei: `/usr/lib/crontab` mit erklärenden Kommentarzeilen könnte so aussehen:

```
# * =alle erlaubten Werte
# n-n=Bereich von Werten
# n,n=Liste von Werten
#
# minute hour day/month month day/week program
#
# * * * * * /usr/lib/atrun
#
# 0 5 * * 5 /usr/bin/mkdump
#
# 0 5 * * 1-5 /usr/bin/rmjunk
$
```

Erklärung der Beispieltabelle:

Das Programm `/usr/lib/atrun` wird jede Minute an allen Tagen der Woche aufgerufen.

Das Programm `/usr/bin/rmjunk` wird von montags bis freitags jeweils um 5 Uhr gestartet.

Das Programm `/usr/bin/mkdump` wird jeden Freitag um 5 Uhr gestartet.

Hinweis

`/etc/cron` greift auf die Datei `/usr/lib/crontab` nur zu, wenn sie seit dem letzten Zugriff geändert wurde (neues Änderungsdatum). Beachten Sie dabei, daß z.B. das Kommando `mv` das Änderungsdatum nicht setzt.

5.16 Dateisystem auf aktuellem Stand halten

`/etc/update`

Das `/etc/update`-Kommando ist während der gesamten Laufzeit des Systems aktiv. Es führt alle 30 Sekunden das `sync`-Kommando aus. `Sync` schreibt alle Informationen, die sich im Hauptspeicher befinden, auf die Platte. Dadurch wird sichergestellt, daß das Dateisystem im Falle eines Systemabsturzes einigermaßen auf dem neuesten Stand ist.

Das `/etc/update`-Kommando wird beim Hochfahren des Systems durch Aufruf der Shell-Prozedur `/etc/rc` vom Prozeß `/etc/init` gestartet. Sie sollten deshalb das `update`-Kommando nicht direkt aufrufen.

5.17 Prozesse die beim Systemstart ablaufen

/etc/init
/etc/inir
/etc/ttys
/etc/ttytype

Das /etc/init- Kommando wird beim Hochfahren des Systems aufgerufen. /etc/init stößt nacheinander folgende Tätigkeiten an:

- Die Shell-Prozedur /etc/rc wird ausgeführt. Dadurch werden Verwaltungsarbeiten durchgeführt wie z.B.:
 - Löschen von temporären Dateien (z.B. die Dateien in den Dateiverzeichnissen /tmp und /usr/tmp).
 - Hinzufügen von Dateisystemen. Das Dateisystem auf der Festplatte /dev/usr wird an das Dateiverzeichnis /usr angehängt (siehe /etc/mount-Kommando).
 - Starten von Prozessen, die während der gesamten Laufzeit des Betriebssystems aktiv sind (z.B. /etc/cron).
- Bei einem Mehrplatzsystem liest /etc/init aus der Datei /etc/ttys, welche Datensichtstationen angeschlossen werden müssen. Für jede dieser Datensichtstationen erzeugt init einen Prozeß zum Eröffnen für Lesen und Schreiben. Standardmäßig werden dabei folgende Dateideskriptoren zugeordnet:

Standardeingabe	– Dateideskriptor 0
Standardausgabe	– Dateideskriptor 1
Standardfehlerausgabe	– Dateideskriptor 2

Die Dateideskriptoren sind den Datensichtstationen zugeordnet und werden beim Umleiten der Standard-Ein-/Ausgabe benutzt (siehe: Kapitel 3.2).

Das Eröffnen einer Datensichtstation kann erst abgeschlossen werden, wenn diese Datensichtstation tatsächlich eingeschaltet ist.

Bei einem Einplatzsystem macht /etc/init das gleiche, wie oben beschrieben, für eine Datensichtstation.

-
- `init` ruft nach erfolgreichem Eröffnen einer Datensichtstation eines Ein- oder Mehrplatzsystems das `getty`-Kommando auf, das am Bildschirm Ihren Benutzernamen anfordert. Das `getty`-Kommando übergibt diesen Benutzernamen dem `login`-Kommando. Dieses Kommando:
 - fordert eventuell ein Kennwort für den Benutzer an
 - meldet den Benutzer an das Betriebssystem an,
 - schreibt einen Benutzereintrag in die Datei `/etc/utmp` und
 - startet die Shell bzw. das Programm, das in der Datei `/etc/passwd` eingetragen ist, als Standardprogramm für den Benutzer.
 - initialisiert für jede Datensichtstation den in `/etc/termcap` definierten Zeichensatz

Der Benutzer kann nun seine Arbeiten an der Datensichtstation durchführen.

Der `/etc/init`-Prozeß wartet, bis der Benutzer die Shell bzw. das Programm beendet. `init` führt nun die folgenden Tätigkeiten aus:

- `init` löscht den Benutzereintrag in der Datei `/etc/utmp`. In dieser Datei sind die gerade aktuellen Benutzer des Betriebssystems notiert.
- `init` eröffnet die betreffende Datensichtstation eines Ein- oder Mehrplatzsystems und ruft das `getty`-Kommando auf, damit sich erneut ein Benutzer anmelden kann.

Hinweis

Der `init`-Prozeß ist immer der Vaterprozeß des Shell-Prozesses, der nach dem Anschließen der Datensichtstation aktiv wird.

/etc/inir-Kommando

Das `inir`-Kommando wird an Stelle des `init` -Kommandos aufgerufen, wenn die Ladeprozedur erkennt, daß das Dateisystem nicht als "in Ordnung" markiert ist. Eine Ursache dafür kann sein, daß z.B. das System nicht durch das `haltsys`-Kommando gestoppt wurde, sondern durch einen Hardwareoder Softwareabsturz.

Das `inir`-Kommando meldet sich mit dem Text:

Das System wurde nicht ordnungsgemaess abgestellt.

Deshalb muß die Konsistenz des Dateisystems ueberprueft werden.

Anschließend überprüft das `fsck`-Kommando automatisch das `root`-Dateisystem und repariert gefundene Schäden. Danach hält das System an. Sie müssen es nun neu hochfahren.

5.18 Systemuhr stellen

`/etc/asktime`
`/etc/mc`

Ihr System besitzt zwei Uhren:

- eine batteriegepufferte Hardwareuhr
- eine Systemuhr, die nur läuft, wenn Ihr System läuft.

Sie können beide Uhren stellen und lesen:

- die Hardwareuhr mit dem `/etc/mc`-Kommando,
- die Systemuhr mit dem `datum`, bzw. `date`-Kommando.

Stellen Sie die Hardwareuhr, wird die Systemzeit automatisch mit der Zeit der Hardwareuhr synchronisiert.

Beim Hochfahren des Systems wird die Shell-Prozedur `/etc/asktime` aufgerufen. Diese versucht, die Zeit der Hardwareuhr zu lesen. Ist das nicht möglich, weil entweder:

- die Datei für die Hardwareuhr (`/dev/mc`) fehlt, oder
- die Hardwareuhr defekt ist, oder
- die Hardwareuhr noch nie gestellt wurde,

fordert `asktime` Sie auf, das Datum und die Zeit einzugeben. `asktime` setzt, wenn möglich, die Zeit der Hardwareuhr mit dem `/etc/mc`-Kommando, sonst nur die Systemzeit mit dem `datum`-Kommando.

Kann `/etc/mc` die Hardwareuhr lesen, synchronisiert `mc` nur die Systemzeit mit der Hardwareuhr.

Sie sollten die Systemzeit nicht ohne Grund mit dem `datum`-Kommando setzen, da dann Systemzeit und Hardwareuhr nicht synchron laufen.

Wie können Sie das `/etc/mc`-Kommando benutzen?

Zum Abfragen und Setzen der Hardwareuhr gibt es das `mc`-Kommando. Das Kommando können Sie nicht benutzen, um Datum und Zeit auszugeben; `/etc/mc` gibt nur Fehlermeldungen aus. Nur der Systemverwalter kann die Zeit setzen.

`/etc/mc [[jj]mm][tt]hhmm[.ss]`

wobei bedeuten:

jj: die beiden letzten Ziffern der Jahreszahl,
mm: die Monatszahl (zweistellig): also 01 für Januar,
tt: der Tag (zweistellig),
hh: die Stunde (zweistellig),
mm: die Minuten (zweistellig),
ss: die Sekunden (zweistellig).

Arbeitsweise

Haben Sie kein Argument angegeben, versucht `mc`, die Hardwareuhr zu lesen. Ist das erfolgreich, synchronisiert `mc` die Systemzeit mit der Zeit der Hardwareuhr. Andernfalls gibt `mc` die Frage aus:

Zeit eingeben: `[jj][mm][tt]hhmm[.ss]:`

`mc` überprüft Ihre Eingabe auf Plausibilität. Fehlende Teile des Datums belegt `mc` mit den Daten der letzten Änderung des `root`-Indexeintrags. Ist ihre Eingabe für `mc` nicht plausibel, wiederholt `mc` die Abfrage, bis Sie ein gültiges Datum eingegeben haben. Kann `mc` Ihre Eingabe erkennen, versucht `mc`, die Hardwareuhr auf diese Zeit zu setzen. Gelingt dies nicht, bricht `mc` mit einer Fehlermeldung ab.

Nach dem Setzen der Zeit versucht `mc` noch einmal, die Zeit zu lesen. Ist das wieder nicht möglich, bricht `mc` ebenfalls mit einer Fehlermeldung ab. (Es liegt dann wahrscheinlich ein Hardwarefehler vor).

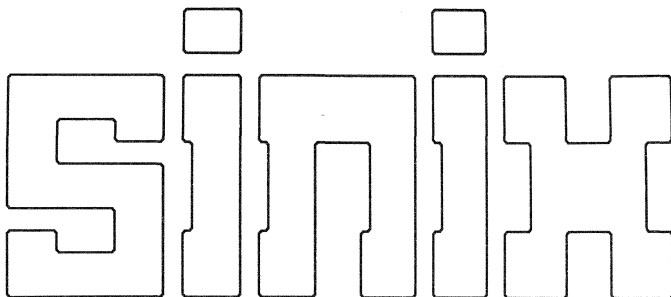
Zuletzt setzt `mc` die Systemzeit auf die Zeit der Hardwareuhr.

Haben Sie eine Zeitangabe als Argument angegeben, überprüft `mc` sie auf Plausibilität, setzt die Hardwareuhr und synchronisiert damit die Systemzeit.

5.19 Begrüßungsbildschirm ändern

/etc/herald/*

Wenn Sie an einem Ein- oder Mehrplatzsystem Ihre Login-Shell beenden, oder bei einem Mehrplatzsystem Ihre Datensichtstation schon eingeschaltet hatten, bevor das System hochgefahren wurde, erscheint der Begrüßungsbildschirm:



Rechnername: sie001
Benutzerkennung: _

Das `/etc/getty`-Kommando hat diese Grafik aus der Datei `/etc/herald/<dss>`, geholt und auf den Bildschirm geschrieben. `<dss>` ist der Name Ihrer Datensichtstation im Dateiverzeichnis: `/dev`. Den Namen können Sie mit dem `tty`-Kommando (siehe Kapitel 6) feststellen. Bei einem Einplatzsystem liest das `/etc/getty`-Kommando den Begrüßungsbildschirm aus der Datei `/etc/herald/console`.

Möchten Sie Ihren eigenen Begrüßungsbildschirm definieren, schreiben Sie einfach Ihre Definition in die zu Ihrer Datensichtstation gehörige Datei im Dateiverzeichnis `/etc/herald`.

Hinweis

Eine Datei, die einen Begrüßungsbildschirm enthält, darf keine Tabulatorzeichen enthalten - 'X'08' (siehe: Anhang ASCII-Tabelle).

5.20 Neue Software installieren

`/etc/superinstall`

Wollen Sie neue Software auf Ihrem System installieren, müssen Sie die Shell-Prozedur `/etc/superinstall` benutzen.

`/etc/superinstall`

Nach dem Aufruf fordert `superinstall` die erste Diskette des neu zu installierenden Softwareprodukts an. Schieben Sie die entsprechende Diskette ins Laufwerk und `superinstall` wird selbsttätig die erforderlichen Tätigkeiten durchführen.

Eine erfolgreiche Installation beendet `superinstall` mit der Meldung: Installation erfolgreich abgeschlossen.

Andernfalls meldet das Kommando: Installation konnte nicht erfolgreich abgeschlossen werden.

5.21 Die Druckerverwaltung

`/etc/qdaemon`

Den Druckerbetrieb steuert ein Programm, das die Drucker und die dazugehörigen Warteschlangen verwaltet. Seine Aufgaben sind:

- Einreihen von Druckaufträgen in Warteschlangen
- Abarbeiten der Warteschlangen.

Mit dem `lpr`-Kommando können Sie Druckaufträge aufgeben und steuern. Aufruf und Parameter sind im Kapitel 6 beschrieben.

Arbeitsweise

`/etc/qdaemon`

Die Shell-Prozedur `/etc/rc`, die beim Hochfahren des Systems aufgerufen wird, startet den Prozeß `/etc/qdaemon`. Er ist während der gesamten Laufzeit des Systems aktiv und steuert den Druckbetrieb.

`/etc/qdaemon` führt nach dem Aufruf folgende Tätigkeiten durch:

- Beenden eines eventuell noch laufenden `qdaemon`-Prozesses,
- Abrechnen laufender Druckaufträge,
- Statusdateien in definierten Zustand bringen,
- Ausführen vorhandener alter Aufträge.

Die Prozeßnummer (PID) (siehe `ps`-Kommando, Kapitel 6) des `/etc/qdaemon`-Prozesses ist in der Datei `/usr/spool/lpd/pid` abgelegt.

`/etc/qdaemon` benutzt zwei Programme zum Bedienen der verschiedenen Drucker, die Sie an Ihr System anschließen können:

- `/usr/lib/lp9001` für den 9001-Drucker,
- `/usr/lib/lp9004` für den 9004-Drucker.

/bin/lpr

/bin/lpr ist die Benutzerschnittstelle des Druckerbetriebes. /bin/lpr reiht Druckaufträge in das Dateiverzeichnis /usr/spool/lpd/qdir ein. Dateien in diesem Dateiverzeichnis enthalten Verweise auf die zu druckenden Dateien. Das Dateiverzeichnis /usr/spool/lpd/stat enthält für Ein- und Mehrplatzsysteme entsprechende Statusdateien für angeschlossene Drucker.

Ist beim lpr-Kommando der Schalter -cp angegeben, kopiert lpr die angegebenen Dateien in das Dateiverzeichnis /usr/tmp/copies.

/bin/lpr kommuniziert mit dem Prozeß /etc/qdaemon durch Senden von Signalen. Empfängt /etc/qdaemon ein Signal, sucht er im Dateiverzeichnis /usr/spool/lpd/qdir nach Druckaufträgen und führt diese nach Durchführen einiger Plausibilitätskontrollen aus.

Die Konfigurationsdatei: /usr/lib/qconfig

Die Druckerverwaltung benötigt für die Verwaltung der Warteschlangen die Konfigurationsdatei /usr/lib/qconfig.bin. Grundlage für diese binäre Konfigurationsdatei ist die Datei /usr/lib/qconfig. Durch das Programm /usr/lib/digest wird die Konfigurationsdatei /usr/lib/qconfig/ umgesetzt in die binäre Konfigurationsdatei /usr/lib/qconfig.bin. Das Programm /usr/lib/digest kann man aufrufen durch:

- das Kommando: lpr – rr
- Starten von /etc/qdaemon

Hinweis

Die Konfigurationsdatei /etc/lib/qconfig sollte nur über das Menüsystem umkonfiguriert werden.

Die Konfigurationsdatei `/usr/lib/qconfig` enthält folgende Informationen:

- welche Geräte (Mehrplatzsystem) bzw. welches Gerät (Einplatzsystem) die Warteschlange bedient,
- welche Geräte (Mehrplatzsystem) bzw. welches Gerät (Einplatzsystem) von welchem Programm bedient wird,
- Abarbeitungsvorschriften für die Warteschlangen.

Die Datei ist aufgeteilt in Einträge für Warteschlangen und Geräte. Teile eines Eintrags in eckigen Klammern sind optional.

Einträge für eine Warteschlange:

Bedeutung:

<code><wsname></code>	Name der Warteschlange
<code>queue = <name></code>	<code><name></code> definiert Drucker
<code>geraet = <gername></code>	Verweis auf den Geräteeintrag
<code>[modus = { $\left. \begin{array}{l} \text{fifo} \\ \text{kurz} \end{array} \right\}$ }]</code>	Bei Angabe von <code>fifo</code> wird der älteste Auftrag, bei <code>kurz</code> der kürzeste zuerst abgearbeitet. Fehlt dieser Eintrag wird <code>fifo</code> angenommen.

Einträge für ein Gerät:

Bedeutung:

<code><gername></code>	Gerätename, wie bei Warteschlangeneintrag
<code>file = <dateiname></code>	Name der Datei, über die das Gerät angesprochen wird
<code>treiber = <prname></code>	Name des Programms, das das Gerät bedient
<code>kopf = { $\left. \begin{array}{l} \text{immer} \\ \text{gruppe} \\ \text{niemals} \end{array} \right\}$ }</code>	Beim Ausdrucken wird am Dateianfang entweder bei jedem Druckauftrag oder bei Gruppenwechsel der Auftraggeber oder nie ein Kopfeintrag mit Informationen über den Druckauftrag eingefügt.
<code>anhang = { $\left. \begin{array}{l} \text{immer} \\ \text{gruppe} \\ \text{niemals} \end{array} \right\}$ }</code>	Nach denselben Bedingungen wie bei <code>kopf</code> wird am Dateiende ein Anhang ausgedruckt
<code>[zugriff = { $\left. \begin{array}{l} \text{schreiben} \\ \text{lesen} \end{array} \right\}$ }]</code>	Wenn der Treiber Rückmeldungen vom Gerät erwartet, muß für die Druckerverwaltung <code>zugriff=lesen</code> gesetzt sein; Standardeinstellung ist <code>zugriff=schreiben</code> .

Prinzipiell kann ein Drucker von mehreren Warteschlangen bedient werden. Es müssen dann alle bis auf eine im Zustand AUS sein.

Beispiel für eine Datei: /usr/lib/qconfig (Mehrplatzsystem)

Diese Konfigurationsdatei beschreibt zwei Warteschlangen, die zwei Drucker bedienen. Die Drucker werden über die Dateien /dev/lp9001-1-D1 und /dev/lp9004-2-D1 angesprochen. Der zweite Drucker wird vom Programm /usr/spool/lp9004 bedient. Beide Warteschlangen werden nach `modus = fifo` (Standard) abgearbeitet.

```
lpa:
    queue = D1
    geraet = lp1
lp1:
    file = /dev/lp9001-1-D1
    treiber = /usr/lib/lp9001
    kopf = gruppe
    anhang = niemals
lpb:
    queue = D2
    geraet = lp2
lp2:
    file = /dev/lp9004-2-D2
    treiber = /usr/lib/lp9004
    kopf = niemals
    anhang = gruppe
```

Mit dem `lpr`-Kommando können Sie sich über den Druckerzustand informieren (siehe Kommandobeschreibung, Kapitel 6).

Die Treiberprogramme

Treiber für den Drucker 9001: /usr/lib/lp9001

Dieses Programm interpretiert die Schalter -pb, -pl, -pb1, -pb2, -pb3, -ab, -bis, -int, -dt und kopiert die Datei (name) auf die Standard-Ausgabe. Die Standard-Ausgabe wird von /etc/qdaemon auf die Datei eröffnet, die in /usr/lib/qconfig mit dem Parameter file= vereinbart wurde. Rückmeldungen vom Drucker werden ausgewertet und zeitlich überwacht.

Unmittelbar vor und nach der Ausgabe der Datei wird auf eine Reaktion vom Drucker gewartet, außerdem wird jede Ausgabe einer Zeile zeitlich überwacht. Wenn innerhalb von 60 Sekunden keine Reaktion erfolgt, wird über die mail-Funktion die Meldung ausgegeben: Drucker 9001 gestört. Tritt dieser Zustand öfter als drei mal auf, werden die Ausgabeversuche eingestellt, der Auftrag abgebrochen und dieser Drucker für Ausgaben gesperrt. Wenn der Fehlerzustand behoben und der Drucker mit "lpr -du" wieder entsperrt ist, wird die Ausgabe von Anfang an wiederholt.

Treiber für den Drucker 9004: /usr/lib/lp9004

Dieses Programm akzeptiert die Schalter: -pb1, -pb2, -pb3, -int, -dt und interpretiert die Schalter: -pb, -pl -ab, -bis und kopiert die Datei (name) auf die Standard-Ausgabe.

Die Standard-Ausgabe wird von /etc/qdaemon auf die Datei eröffnet, die in /usr/lib/qconfig mit dem Parameter "file=" vereinbart wurde. Beim Aufruf des Programms durch die Druckerverwaltung ist die Standard-Ausgabe auf die Datei eröffnet, die in der Konfigurationsdatei vereinbart ist. Rückmeldungen vom Drucker werden ausgewertet und zeitlich überwacht. Unmittelbar vor und nach der Ausgabe der Datei wird auf eine Reaktion vom Drucker gewartet, außerdem wird jede Ausgabe einer Textzeile zeitlich überwacht. Erfolgt innerhalb von 200 Sekunden keine Reaktion, wird über die mail-Funktion die Meldung ausgegeben: Drucker 9004 gestört. Tritt dieser Zustand mehr als drei mal auf, werden die Ausgabeversuche eingestellt, der Auftrag abgebrochen und dieser Drucker für Ausgaben gesperrt. Zusätzlich werden die Statusmeldungen des Druckers 9004 ausgewertet und dem Auftraggeber nach Beendigung des Auftrags über die mail-Funktion zugestellt. Wenn der Fehlerzustand behoben und der Drucker mit "lpr -du" wieder entsperrt ist, wird die Ausgabe von Anfang an wiederholt.

5.22 Datensichtstationsdefinitionen

`/etc/termcap`

In der Datei `/etc/termcap` sind jeweils Eigenschaften beschrieben für:

- die Datensichtstationen eines Mehrplatzsystems
- die Datensichtstation eines Einplatzsystems

Die Datei enthält z.B. die Zeichenfolgen, die von bestimmten Tasten gesendet werden, oder die nötig sind, um bestimmte Funktionen am Bildschirm auszuführen.

Die Datei wird von Programmen benutzt, die bildschirmorientiert arbeiten, wie z.B. das Menüsystem und der CED-Editor. Außerdem benutzt `/etc/getty` die in `/etc/termcap` zur Variablen `:is` gehörende Information, um eine Datensichtstation beim Anschalten zu initialisieren.

Die Datei */etc/termcap* besteht aus Blöcken von Einträgen. Jeder Block beginnt mit einer Kopfzeile, in der in Kurz- und Langform erklärt ist, für welche Datensichtstationen, bzw. Programme die folgende Information bestimmt ist. Alle Zeilen eines Blocks bis auf die letzte enden mit dem Zeichen '\'. Die Einträge eines Blocks sind durch das Zeichen ':' voneinander getrennt. Jede Zeile beginnt und endet mit einem ':'. Jeder Eintrag besteht aus mindestens einer zweibuchstabigen Identifikation, die die Bedeutung des Eintrags festlegt.

Es gibt drei verschiedene Arten von Einträgen:

- Bool'sche, die angeben, ob die Datensichtstation eine bestimmte Fähigkeit hat, z.B. :bs: bedeutet: diese Datensichtstation hat eine Backspace-Funktion.
- Numerische Eigenschaften; diese Einträge beginnen mit der Identifikation. Es folgt auf das '#'-Zeichen eine Dezimalzahl, z.B. :li # 5: diese Datensichtstation hat 25 Zeilen.
- Zeichenfolgen; sowohl von der Tastatur gesendete Zeichen, als auch an die Datensichtstation zu sendende, um bestimmte Funktionen auszulösen. Diese Einträge haben die Form: :Identifikation=zeichenkette: . Bei der Angabe der Zeichenkette gelten folgende Konventionen:
 - x wird als `CTRL` `x` interpretiert;
 - eine mit \$0 beginnende Zahl wird als Oktalzahl interpretiert;
 - \$E wird als Escape-Zeichen interpretiert X'27';
 - % steht für zu berechnende und in die Zeichenkette einzusetzende Zeichen.
z.B. \$E[%i%d;%dH: um die Schreibmarke direkt zu adressieren.

Zur Arbeit mit diesen Einträgen stehen Ihnen im CES besondere Prozeduren zur Verfügung (siehe auch: tcout-Kommando).

5.23 Tips für den Systemverwalter

- **Der Drucker läuft nicht mehr?**

Meistens haben die Schwierigkeiten einfache Ursachen. Stellen Sie deshalb zuerst fest, ob:

- der Drucker angeschaltet ist;
- der Drucker online geschaltet ist.

Hilft das nicht, stellen Sie mit dem Kommando: `lpr -q` fest, ob der Drucker bereit ist. Wenn nicht, aktivieren Sie ihn mit dem Kommando: `lpr -du` (`lpr`-Kommando siehe Kapitel 6).

Ist das nicht erfolgreich, stellen Sie fest, ob der Druckerverwaltungsprozeß `/etc/qdaemon` aktiv ist. Geben Sie dazu das Kommando:

```
ps -a
```

ein. Ist `/etc/qdaemon` nicht in der Ausgabeliste des `ps`-Kommandos enthalten, starten Sie ihn neu. Geben Sie ein:

```
/etc/qdaemon
```

Wenn alles nichts hilft, müssen Sie einige Aufräumarbeiten leisten. Beenden Sie zuerst den Druckerverwaltungsprozeß `/etc/qdaemon` und eventuell aktive Treiberprogramme mit Hilfe des `kill`-Kommandos (siehe Kapitel 6). Löschen Sie dann alle Dateien in den Dateiverzeichnissen:

- `/usr/spool/lpd/qdir` und
- `/usr/spool/lpd/stat`.

Starten Sie dann den Prozeß `/etc/qdaemon` neu.

- **Ein Benutzer hat sein Kennwort vergessen?**

Hat ein Benutzer sein Kennwort vergessen, kann er sich nicht mehr an das System anmelden. Da es auch dem Systemverwalter aus Sicherheitsgründen nicht möglich ist, ein verschlüsseltes Kennwort zu entschlüsseln, ist dieses Kennwort verloren. Was tun? Der Systemverwalter muß mit dem `passwd`-Kommando (Kapitel 6) für den Benutzer ein neues Kennwort definieren.

- **Der Systemverwalter hat sein Kennwort vergessen?**

Vergessen Sie es nicht. Es ist nicht möglich, ein vergessenes root-Kennwort zu restaurieren. Die letzte Rettung ist, das gesamte System von der boot-Diskette aus neu zu installieren, und anschließend mit Ihren Sicherungsdisketten das System wiederherzustellen.

- **Die Platte ist voll?**

Dazu sollte es nie kommen. Eine Platte sollte wenigstens ca. 15 % freien Speicherplatz im /usr-Dateisystem haben. Überwachen Sie deshalb den freien Speicherplatz sorgfältig. Dazu stehen das du-, df-, qout- und find-Kommando zur Verfügung (siehe Kapitel 6). Ist Ihr Plattenplatz doch einmal erschöpft, versuchen Sie mit dem qout- und find- Kommando Dateien zu finden, die Sie auslagern oder löschen können.

- **Es läuft nichts mehr?**

1) Ihre Datensichtstation "versteht" nichts mehr.

Es kann vorkommen, daß Ihr System die Taste `[J]` nicht mehr versteht. Dann können Sie keine Kommandos mehr abschließen. Benutzen Sie stattdessen die Taste `[MENU]`. Bekommen Sie jetzt eine Antwort vom System, befindet sich Ihre Datensichtstation wahrscheinlich im raw-Modus. Sehen Sie zusätzlich nicht einmal mehr auf dem Bildschirm, was Sie eintippen, ist zusätzlich noch das Echo ausgeschaltet worden. Trotzdem kommen die von Ihnen angegebenen Zeichen noch beim Betriebssystem an und werden von diesem interpretiert, allerdings nicht in der gewohnten Art.

Diesen Zustand können Sie mit dem stty-Kommando (Kapitel 6) wieder in den Normalzustand zurücksetzen. Dabei müssen Sie folgendermaßen vorgehen.

- Drücken Sie zuerst die Taste `[MENU]`, um sicher zu sein, daß alle vorher von Ihnen eingegebenen Zeichen verarbeitet sind.
- Geben Sie das Kommando ein: `stty -raw echo -nl` (Taste `[MENU]` drücken)

Achtung

Sie müssen das Kommando mit der Taste `[MENU]` abschließen.

Achtung

Sie können Ihre Eingabe nicht mit der Taste `[X]` korrigieren.
Wenn Sie sich vertippen, müssen Sie Ihre Eingabe wiederholen.

Jetzt sollte Ihre Datensichtstation wieder "normal" arbeiten.

- 2) Ihre Datensichtstation zeigt nur noch seltsame Zeichen an?
Offensichtlich ist ein falscher Zeichensatz in den Bereitstellungsbe-
reich geladen worden. Um das zu beheben, versuchen Sie folgendes:
 - Versuchen Sie zuerst, den alten Zeichensatz wieder zu laden. Drük-
ken Sie die Tasten:
`[ESC] [I] [B]` für den Zeichensatz International oder `[ESC] [I] [K]` für
den deutschen Zeichensatz

Jetzt müßten Sie wieder die richtigen Zeichensatz auf Ihrem Bild-
schirm sehen.
 - Versuchen Sie jetzt, einige Sonderzeichen einzutippen. Erscheinen
stattdessen andere Sonderzeichen auf dem Bildschirm, ist die Bele-
gung Ihrer Tastatur geändert worden:
 - Stellen Sie fest, welche Taste das Zeichen: [liefert.
Tippen Sie folgende Tastenfolge ein:
`[ESC] [I] [6] [u]`
um Ihre internationale Tastatur zu belegen, oder
`[ESC] [I] [7] [u]`
um Ihre deutsche Tastatur zu belegen.
- 3) Ihre Datensichtstation wird durch einen Prozeß blockiert, den Sie nicht
mehr stoppen können. Dieser Prozeß kann z.B. im Hintergrund laufend
alle Ihre Eingaben verschlucken. Sie können dann kein Kommando
mehr eingeben. Haben auch die Unterbrechungstasten keine Wirkung
mehr, können Sie folgendes machen:
 - Falls der Prozeß keinen Schaden anrichtet und Sie begründete
Hoffnung haben, daß er sich von selbst beendet, warten Sie!
 - Arbeiten Sie an einem Mehrplatzsystem, dann gehen Sie zu einer
anderen Datensichtstation und geben das Kommando ein: `ps -a`.
Die Ausgabe dieses Kommandos (PID-Spalte) liefert Ihnen die
Nummer des Prozesses, der die andere Datensichtstation blockiert.

Als Systemverwalter können Sie jetzt versuchen, mit einem der beiden Kommandos:

kill -2 PID

oder:

kill -3 PID

den Prozeß zu abbrechen. Wenn keines der beiden Kommandos wirkt, benutzen Sie das Signal 9 beim kill-Kommando: kill -9 PID. Jetzt wird der Prozeß mit Sicherheit beendet; aber vielleicht temporäre Dateien und die Datensichtstation in einem seltsamen Zustand hinterlassen.

4) Sie können keine Eingabe mehr machen?

- Versuchen Sie es trotzdem. Vielleicht gelingt es Ihnen, noch Daten (z.B. beim CED-Editor) zu retten.
- Haben Sie alles gerettet oder reagiert das System wirklich nicht mehr, dann schalten Sie das System aus und fahren es, nachdem sich die Systemeinheit abgeschaltet hat, wieder hoch.
An einem Mehrplatzsystem erreichen Sie das durch Ein- und Ausschalten der Console; an einem Einplatzsystem durch Ein- und Ausschalten des Systems an der Systemeinheit.
- Schaltet sich das System nicht automatisch aus, drücken Sie den RESET-Taster an der Bedieneinheit. Der RESET-Taster ist bei einem Mehrplatzsystem der obere der beiden Knöpfe, die sich rechts neben der LED-Leuchtkette an der Systemeinheit befinden. Bei einem Einplatzsystem ist der RESET-Taster rechts neben dem Netzschalter. Sie können ihn mit einem kleinen Schraubendreher drücken. Das System wird dadurch automatisch wieder gestartet.

• **Das System ist abgestürzt?**

Schreiben Sie in diesem Fall alle Meldungen auf, die das System vor und beim Absturz entweder auf die Datensichtstation des Einplatzsystems oder auf die Console des Mehrplatzsystems geschrieben hat. Schalten Sie das System aus, und warten Sie, bis sich die Systemeinheit abgeschaltet hat. Anschließend können Sie es wieder anschalten. Da das Dateisystem beim Absturz nicht ordnungsgemäß abgeschlossen wurde, wird es automatisch auf seine Konsistenz geprüft.

Funktioniert das nicht, drücken Sie den RESET-Taster wie im vorgehenden Abschnitt unter 4) beschrieben.

- **Definition neuer Benutzer führte zu Inkonsistenzen?**

Neue Benutzer darf man nur über das Systemverwaltermenü: Login-Administration definieren. Manipulationen an einzelnen Systemtabellen (z.B. an der Datei: /etc/passwd) führen zu Inkonsistenzen, die man nicht mehr automatisch beseitigen kann. Falls solche Inkonsistenzen eintreten, muß man folgende Dateien von Hand auf den gewünschten Stand bringen:

```
/etc/passwd
/etc/group
/usr/admin/.benutzer/*
/usr/admin/gruppen
/usr/admin/stop.dat
/usr/menus/app/develop/login.dat
/usr/menus/app/develop/deauth.dat
```

Um die Änderungen durchführen zu können, muß der Systemverwalter aus dem Menüsystem in die SINIX-Shell wechseln, dort die entsprechenden Änderungen durchführen und anschließend folgende Prozedur aufrufen:

```
/usr/menus/app/control/genacf.scr
```

Hinweis

- 1) Diese Prozedur kann man nur unter der Benutzerkennung: admin direkt zum Ablauf bringen.

Beim Aufruf unter der Benutzerkennung: root müssen vorher folgende Variablen definiert sein:

```
DEV=/usr/menus/app/develop
APP=/usr/menus/app/control
SAEXEC=/usr/menus/sabin
export DEV APP SAEXEC
```

- 2) Gleichzeitiges Arbeiten des Systemverwalters an mehreren Datenstationen kann zu Inkonsistenzen im System führen.

6 Die Kommandos

Beschrieben sind die SINIX-Kommandos, die Sie als Normalbenutzer verwenden können. Eine ausführliche Übersicht, gegliedert nach Funktionen, finden Sie in Abschnitt 6.5.

Nicht enthalten sind hier die Kommandos für den Systemverwalter (Kapitel 5) und die Kommandos für Shell-Prozeduren, die die Shell direkt interpretiert. Diese finden Sie in Abschnitt 3.8.

6.1 Kommandos eingeben, aber richtig

Darstellung

Alle SINIX-Kommandos sind dargestellt, wie im folgenden Beispiel:

```
ls[_-schalter...][_name...]
```

In dieser Darstellung bedeuten:

- ls** ist der Kommandoname. Er ist fett gedruckt. Daran sehen Sie, daß Sie die Angabe "ls" schreiben müssen.
- _** steht für ein Leerzeichen, das zwischen verschiedenen Angaben zu schreiben ist. Sie können auch mehrere Leerzeichen schreiben. Das macht keinen Unterschied.
- []** Angaben in eckigen Klammern können Sie weglassen. Natürlich beeinflußt das die Wirkung des Kommandos. Die eckigen Klammern selbst dürfen Sie nicht schreiben.
- schalter** kennzeichnet Wahlmöglichkeiten für die Wirkung des Kommandos. Für schalter setzen Sie eine aktuelle Angabe ein. Meist sind verschiedene Buchstaben als Schalterangabe möglich. Welche, das finden Sie jeweils in der Beschreibung des Kommandos. Diese Angaben können Sie i.a. kombinieren, z.B.
ls -l ruft ls mit Schalter l auf,
ls -lig ruft ls mit Schaltern l, i und g auf.

name	Für name setzen Sie eine aktuelle Angabe ein, hier z.B. den Namen einer Datei oder eines Dateiverzeichnisses. Die jeweilige Bedeutung der Angabe entnehmen Sie der Kommandobeschreibung.
...	heißt, daß Sie die davorstehende Angabe mehrmals machen dürfen, z.B könnten Sie mehrere Schalter auch so angeben: ls -l -i -g wirkt genauso, wie ls -lig In diesem Beispiel können Sie auch mehrere Namensangaben machen, z.B. ls -l adam eva Abgekürzte Dateinamen gelten auch als mehrere Einträge: ls -li c* (alle Dateien, die mit c beginnen, siehe Abschnitt 2.2.3).

Beachten Sie:

- Manchmal sind die Zeichen "[" und "]" keine Metazeichen, sondern hinzuschreiben. Dann ist dies entsprechend beschrieben.
- Bei manchen Kommandos gibt es Schalter mit vorangestelltem "-" und solche ohne "-" oder auch mit "+". Dann ist dieses Vorzeichen bei jedem Schalter beschrieben.
- Gehört zu einem Schalter ein Argument, dann ist die Syntax dafür jeweils eigens beschrieben, z.B. bedeutet -f[_datei], daß Sie z.B. angeben können: "-f" oder "-f ausgabe".
- Wenn Sie bei einem Kommando einen Schalter angeben der nicht beschrieben ist, bekommen Sie entweder eine Fehlermeldung oder es kann zu undefinierten Ereignissen kommen.

Kurz zusammengefaßt:

Symbol *Bedeutung*

Fettdruck Konstante, die so einzugeben ist.

normaler Druck

Angabe, für die Sie aktuell etwas einsetzen müssen.

[] wahlfreie Angabe. Die Klammern sind nicht zu schreiben.

... Wiederholung der vorhergehenden Angabe ist möglich.

Sonderzeichen außer "[", "]" und "..."

sind Konstanten, also zu übernehmen.

Die Anführungszeichen haben zwar Bedeutung für die Shell, werden aber auch im Text verwendet, um Zeichen oder Zeichenfolgen hervorzuheben, wie z.B. oben. Sie sehen jeweils aus dem Zusammenhang, welche Bedeutung die Anführungszeichen haben.

Wie eingeben?

Kommandos schreiben Sie anschließend an das Bereit-Zeichen (\$) in eine Zeile. Sie schließen die Eingabe mit der Taste `[↵]` ab. `[↵]` ist für die Shell das Zeichen, alles vorhergehende als Kommando zu interpretieren.

Mehrere Kommandos in einer Zeile

kommando1 ; kommando2 ; kommando3

Die Kommandos werden nacheinander angestoßen, ohne Verbindung von Ein- und Ausgaben.

Beachten Sie auch die Möglichkeit, Kommandos mit Pipelines zu ketten (siehe Abschnitt 3.3).

Die Zeile ist zu kurz?

Schreiben Sie am Zeilenende einfach weiter, ohne `[↵]` zu drücken.

Eine Zeile fortsetzen können Sie auch, indem Sie am Ende der Zeile einen Gegenschrägstrich (\) schreiben.

Daten eingeben

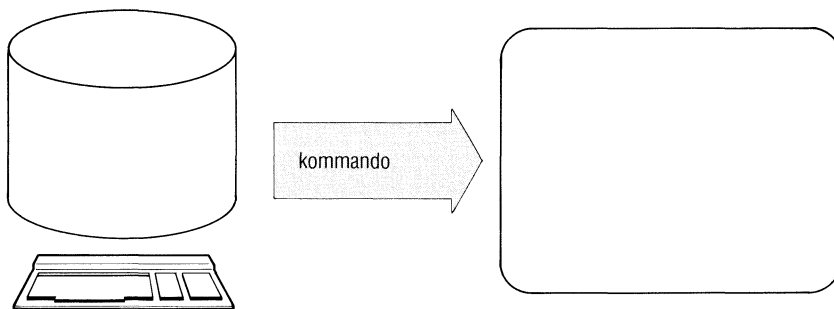
Erwartet ein Kommando Eingaben von der Tastatur, setzt es die Schreibmarke auf den Beginn der nächsten Zeile. Schreiben Sie nun Ihre Eingabe, wobei Sie jede Zeile mit der Taste `[↵]` abschließen. Das Kommando erhält jede Zeile wie die Zeile einer Datei.

Taste `[END]` beendet die Dateneingabe, d.h. für das Kommando ist das Dateiende erreicht und das Kommando wird fertig ausgeführt.

6.2 Was Sie zu jedem Kommando wissen sollten

Ein Kommando, das eine Eingabe erwartet, liest diese von der Standard-Eingabe oder von Dateien, deren Namen Sie angeben müssen, wie beim jeweiligen Kommando beschrieben.

Ein Kommando gibt aus auf die Standard-Ausgabe.



Standard-Eingabe ist die Tastatur, Standard-Ausgabe ist der Bildschirm.

Mit Pipelines und mit Umlenken der Ein- oder Ausgabe schaffen Sie sich mehr Möglichkeiten, z.B.:

- ein Kommando, das normalerweise von der Tastatur liest, kann auch aus einer Datei lesen oder seine Eingabe von einem vorhergehenden Kommando bekommen.
- ein Kommando, das normalerweise auf den Bildschirm ausgibt, kann auch in eine Datei ausgeben oder das Ergebnis an ein weiteres Kommando übergeben.

Lesen Sie dazu Abschnitte 3.2 und 3.3.

Gibt ein Kommando auf den Bildschirm aus und ist die Ausgabe länger als eine Bildschirmseite, dann können Sie:

- die Ausgabe anhalten mit `CTRL S`, Ausgabe fortsetzen mit `CTRL Q`,
- die Kommandos "more" bzw. "page" verwenden.

Dateinamen

Wenn Sie Dateinamen oder Namen von Dateiverzeichnissen angeben, haben Sie immer mehrere Möglichkeiten:

- Sie geben einen einfachen Namen an. Damit beziehen Sie sich auf das aktuelle Dateiverzeichnis.
- Sie geben einen Pfadnamen an. Damit können Sie beliebige Dateien und Dateiverzeichnisse im ganzen Dateisystem benutzen, vorausgesetzt, Sie haben die Zugriffsrechte.
- Sie können die Sonderzeichen für Dateinamen verwenden, besonders um mehrere Dateien gleichzeitig anzugeben (siehe Abschnitt 2.2.3).

Ende-Status:

Jedes Kommando liefert einen Ende-Status, der aussagt, wie das Kommando abgelaufen ist. Der Ende-Status steht als Zahlenwert in der Variablen `?`. Sie können ihn z.B. mit `echo $?` abfragen.

Ist bei der Beschreibung eines Kommandos weiter nichts angegeben, dann ist der Ende-Status:

0	bei fehlerfreiem Ablauf
3	bei Abbruch des Kommandos mit der Taste <input type="text" value="DEL"/>
ungleich 0	bei fehlerhaftem Ablauf

Fehlermeldungen

Zu jedem Kommando gibt es Fehlermeldungen. Sie sind weitgehend selbsterklärend. Fehlermeldungen gehen auf die Standard-Fehlerausgabe. Standard-Fehlerausgabe ist normalerweise der Bildschirm. Sie können Sie aber umlenken mit `2 > & datei`.

Fehlermeldungen des Betriebssystems deuten meist auf einen hardwarebedingten fehlerhaften Ablauf. Als Maßnahme können Sie die Meldung ignorieren und es nochmals versuchen. Wiederholt sich der Fehler mehrmals, verständigen Sie den Systemkundendienst. Solche Fehler haben z.B. die Form:

```
ERR ON DEV 1/23
BN= . . . . .
```

6.3 Welches Kommando für welche Aufgabe?

Kommandos sind sehr vielseitig zu verwenden. Die folgende Übersicht teilt die Kommandos nach ihrer hauptsächlichen Funktion ein. Dabei kommen einige Kommandos mehrmals vor.

Datensichtstations- und Benutzereigenschaften ändern

cd	Dateiverzeichnis wechseln
login	Benutzerkennung wechseln
mesg	Ausgabe von Meldungen verhindern oder erlauben
newgrp	Benutzergruppe wechseln
passwd	Kennwort für Benutzerkennung eintragen oder ändern
script	Sitzung protokollieren
stty	Eigenschaften der Datensichtstation ändern
su	Benutzerkennung vorübergehend wechseln

Dateien verwalten und bearbeiten

sichern und archivieren

far	Archivieren auf Diskette
tar	Archivieren auf Band oder Diskette

ausdrucken

lpr	Dateien ausdrucken und Druckaufträge steuern
print	Dateien ausdrucken am Drucker

ausgeben

cat	Dateien ausgeben
head	Anfangszeilen von Dateien ausgeben
more	Bildschirmausgabe steuern
num	Datei ausgeben mit Zeilennummer
page	Bildschirmausgabe steuern mit Bildschirm löschen
pr	Dateien aufbereiten zum Ausdrucken
split	Datei aufteilen auf mehrere Dateien
tail	Endabschnitt einer Datei ausgeben
xd	Dateiinhalte hexadezimal ausgeben

bearbeiten

awk	Dateien durchsuchen und bearbeiten
cmp	Dateien zeichenweise vergleichen
comm	Sortierte Dateien vergleichen
crypt	Dateien verschlüsseln
diff	Dateien zeilenweise vergleichen und ed-Skript erstellen
diff3	Drei Dateien zeilenweise vergleichen
egrep	Erweiterte Muster suchen
fgrep	Einfache Muster schnell suchen
grep	Muster in Dateien suchen
join	Dateien verbinden nach Vergleichsfeldern
look	Zeilen mit bestimmtem Anfang suchen
prep	Text statistisch aufbereiten
rev	Reihenfolge von Zeichen umkehren
sort	Sortieren und mischen von Dateien
sum	Prüfsumme einer Datei berechnen
tr	Zeichen durch andere ersetzen
uniq	Mehrfache Zeilen suchen
wc	Zeilen, Worte und Zeichen zählen

Dateisystem organisieren und Dateieigenschaften ändern

chgrp	Gruppennummer für eine Datei ändern
chmod	Zugriffsrechte ändern
chown	Eigentümer einer Datei ändern
cp	Datei kopieren
copy	Dateien gruppenweise kopieren
find	Dateiverzeichnisse durchsuchen
ln	Verweis auf eine Datei eintragen
ls	Informationen über Dateiverzeichnisse und Dateien
make	Gruppen von Dateien verwalten
mkdir	Dateiverzeichnis einrichten
mv	Dateien umbenennen oder übertragen
rm	Dateien löschen
rmdir	Dateiverzeichnisse löschen
settime	Zeit des letzten Zugriffs oder der letzten Änderung setzen
sync	Systempuffer zurückschreiben
touch	Zeit der letzten Änderung einer Datei auf aktuelles Datum setzen

Editoren

ced	Bildschirmorientierter Editor
ed	Zeilenorientierter Editor im Dialogbetrieb
sed	Editor im Prozedurbetrieb

Hilfskommandos für Shell-Prozeduren

basename	Dateinamen vom Pfad trennen
echo	Zeichenfolgen ausgeben
expr	Ausdrücke auswerten
false	Leeres Kommando mit Endestatus 1
sleep	Prozesse zeitweise stilllegen
tee	Gleichzeitig auf Standard-Ausgabe und in eine Datei ausgeben
test	Bedingungen prüfen
true	Leeres Kommando mit Endestatus 0

Informationen über Systemdaten

dateityp	Art einer Datei bestimmen, Ausgabe deutsch
file	Art einer Datei bestimmen, Ausgabe englisch
ls	Informationen über Dateiverzeichnisse und Dateien
printenv	Variablenwerte ausgeben
ps	Prozeßdaten abfragen
pstat	Systeminformation ausgeben
pwd	Pfadnamen des aktuellen Dateiverzeichnisses ausgeben
tty	Pfadname Ihrer Datensichtstation ausgeben
what	Versionsnummern ausgeben
who	Aktive Benutzerkennungen anzeigen

Kalenderfunktionen und Termine

at	Prozedurdateien zu einer bestimmten Zeit ausführen, Datum in englischer Schreibweise
cal	Kalender ausgeben
calendar	Erinnerungsdienst, Datum in englischer Schreibweise
date	Datum und Uhrzeit ausgeben, englische Schreibweise
datum	Datum und Uhrzeit ausgeben, deutsche Schreibweise
kalender	Erinnerungsdienst, Datum in deutscher Schreibweise
um	Prozedurdateien zu einer bestimmten Zeit ausführen, Datum in deutscher Schreibweise

Kommunikation mit anderen Benutzern

enroll	Schlüssel für geheime Post festlegen
mail	Post senden und empfangen
write	Dialog mit anderen Benutzern
xget	Geheime Post lesen
xsend	Geheime Post senden

Prozesse steuern

kill	Prozesse beenden, Signale senden
make	Gruppen von Dateien verwalten
nice	Priorität von Kommandos ändern
nohup	Signale ignorieren
time	Laufzeit eines Kommandos messen

Rechenfunktionen

bc	Arithmetische Sprache
dc	Tischrechner
units	Einheiten umrechnen

Speicherplatz-Belegung überprüfen

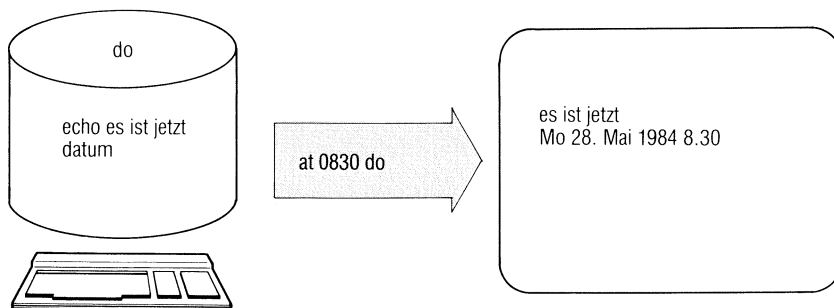
df	Dateisystem auf freien Platz prüfen
du	Belegten Speicherplatz ausgeben
quot	Dateisystem prüfen auf Belegung pro Benutzer

6.4 Vollständige Beschreibung der Kommandos in alphabetischer Reihenfolge

Querverweise zu verwandten Kommandos finden Sie am Ende jeder Kommandobeschreibung, z.B.

>>>> `rmdir, mkdir` verweist auf die genannten Kommandos.

Prozedur-Dateien zu einer bestimmten Zeit ausführen, Datum englisch – execute commands at a later time



Mit dem at-Kommando können ein oder mehrere Kommandos zu einem festgesetzten (späteren) Zeitpunkt ausgeführt werden.

Das Kommando "um" hat dieselbe Funktion wie "at", nimmt aber deutsche Datumsangaben.

at[_zeit][_tag][_datei]

zeit Für den Operanden zeit müssen 1 bis 4 Ziffern angegeben werden.

- Ein- und zweistellige Zahlen interpretiert das Kommando als Stunden.
- Drei- und vierstellige Zahlen interpretiert das Kommando als Stunden und Minuten.

Bündig an die Ziffern können wahlweise folgende Buchstaben angegeben werden:

- a Vormittag, (am auch zulässig)
- p Nachmittag, (pm auch zulässig)
- n Mittag
- m Mitternacht

Wenn nach den Ziffern keine Buchstaben angegeben sind, nimmt das Kommando die 24-Stunden-Uhrzeit an.

tag

Für den Operanden tag können Sie angeben:

- ein Datum in der Form: May 12, oder
- einen Tag der Woche

Ist für den Operanden ein Tag der Woche angegeben und anschließend das Kennwort week, so wird die Shell-Kommandodatei am angegebenen Tag in der folgenden Woche ausgeführt.

Ist für tag der aktuelle Tag angegeben, führt at den Auftrag in der folgenden Woche aus.

Die Namen der Monate und Tage werden in englischer Sprache angegeben und können abgekürzt sein.

Standard (keine Angabe): at führt den Auftrag am selben Tag aus, bzw. am folgenden Tag, falls die angegebene Uhrzeit schon vergangen ist.

datei

Name der Kommandodatei, die die auszuführenden Kommandos enthält. at erstellt daraus eine neue Kommandodatei (im Dateiverzeichnis /usr/spool/at) mit gleicher Benutzer- und Gruppenidentifikation, die zusätzlich einen Verweis auf das aktuelle Dateiverzeichnis und adäquate Änderungen der Umfeldvariablen enthält.

Standard (keine Angabe): at liest die auszuführenden Kommandos von der Standard-Eingabe.

Hinweis

- Um die neue Kommandodatei erstellen zu können (s.o.), muß für at die Kommandodatei des Benutzers lesbar sein, d.h. das r-Bit für Gruppen muß gesetzt sein.
- Die neue Kommandodatei erhält einen Namen, der sich aus dem Datum und einer Identifikationsnummer zusammensetzt.
Mit `ls /usr/spool/at/*` werden alle neu erstellten Kommandodateien aufgelistet, deren Inhalt mit `cat` gelesen werden kann.
- Die Ausführung der von at erstellten Kommandodatei wird über das Kommando `atrun` gesteuert. `atrun` wird in regelmäßigen Abständen von `/etc/cron` aufgerufen, das ständig im Hintergrund läuft.
- Standard- und Fehlerausgaben der Kommandodatei gehen verloren, wenn man sie nicht umleitet.
- Zeitlich sich überholende Aufträge werden parallel ausgeführt, was zu Problemen führen kann.

Beispiele

1. Die Datei `merkedatum` habe folgenden Inhalt:

```
date >> datefile
at 830a merkedatum
```

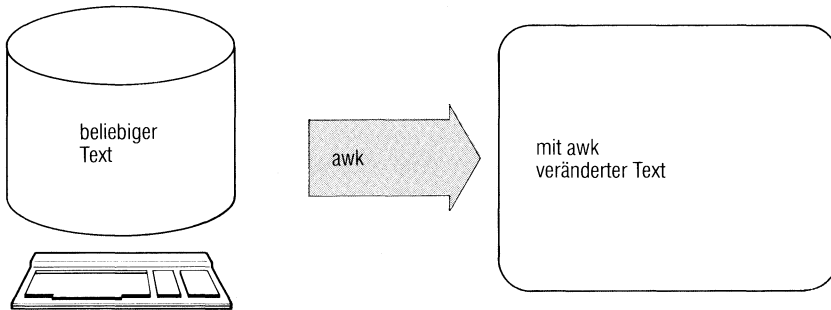
Ein einmaliger Aufruf von `merkedatum` (direkt oder mit `at`) bewirkt, daß täglich um 8 Uhr 30 das aktuelle Datum in die Datei `datefile` geschrieben wird.

2. Mit folgenden Kommandos wird am 1. April um 13 Uhr das Datum auf der Konsole ausgegeben.

```
at 1pm apr 1
/bin/date >> /dev/console
END
```

>>>> um

Dateien durchsuchen und bearbeiten



awk ist eine Programmiersprache, mit der übliche Aufgaben aus der Textverarbeitung angenehm zu formulieren sind. Erforderliche Daten werden mit Hilfe von Mustern aus Dateien gesucht; die gewünschten Aufgaben können dann durch Angabe von Aktionen gelöst werden. awk-Programme werden interpretiert.

awk[**-Fc**][**-f**]**programm**[**datei...**]

schalter

- Fc** Jede Zeile der Eingabedatei ist in Felder unterteilt, die durch das Zeichen "c" getrennt sind.
Standard (keine Angabe): Leer- und Tabulatorzeichen dienen als Trennzeichen zwischen Feldern.
- f** das awk-Programm steht in einer Programmdatei.
Standard (keine Angabe): das awk-Programm steht in Hochkommas eingeschlossen in der Kommandozeile an der Stelle "programm".
- programm** awk-Programm in Hochkommas eingeschlossen oder Name der Datei in der das awk-Programm steht, falls Schalter f gesetzt ist.

datei Datei für Eingabedaten. Bei mehreren Angaben verarbeitet awk die Dateien in der angegebenen Reihenfolge. Steht für einen Dateinamen das Zeichen "-", liest awk von der Standard-Eingabe.

Standard (keine Angabe): awk liest von der Standard-Eingabe.

Programmstruktur

Ein awk-Programm hat folgende Struktur:

```
[ BEGIN {aktion } ]           Beginn-Teil (optional)
  muster {aktion }
  .
  .
  .
  muster { aktion }
[ END { aktion } ]           Ende-Teil (optional)
```

BEGIN und END sind zwei ausgezeichnete Muster (siehe Muster). Jede Aktion muß in geschweifte Klammern eingeschlossen sein.

Arbeitsweise

awk liest die Eingabedateien Zeile für Zeile. Jede Zeile wird nach allen im awk-Programm aufgeführten Mustern durchsucht. Sobald ein Muster paßt, wird die entsprechende Aktion ausgeführt.

Fehlt bei einem Paar (muster {aktion}) die Angabe für muster, so wird die Aktion für jede Zeile ausgeführt; fehlt die Angabe für aktion, dann wird jede passende Zeile ausgegeben. Eine der beiden Angaben muß vorhanden sein.

Zeilen und Felder

Eine Eingabedatei ist in Zeilen unterteilt, die durch den Zeilentrenner RS getrennt sind. Die spezielle Variable RS enthält als Voreinstellung "neue Zeile" (siehe unten, Variablen). An RS kann jedes beliebige Zeichen als Trennsymbol zugewiesen werden.

Die Zeilen werden in der speziellen Variablen NR durchgezählt.

Jede Zeile ist in Felder unterteilt, die durch den Feldtrenner FS getrennt sind. Die spezielle Variable FS ist mit dem Leer- oder Tabulatorzeichen vorbesetzt. Diese Standardeinstellung kann durch Setzen des Schalters Fc (siehe oben) oder durch Zuweisung eines beliebigen anderen Zeichens an FS geändert werden.

Die spezielle Variable NF enthält die Anzahl der Felder der aktuellen Zeile.

Besonderheiten

- Enthält RS einen Leerstring, so gilt eine Leerzeile als Zeilentrenner, Leer- oder Tabulatorzeichen und "neue Zeile" sind dann Feldtrenner.
- Enthält FS das Leerzeichen, werden führende Leer- und Tabulatorzeichen überlesen. Beliebig viele aufeinanderfolgende Leer- bzw. Tabulatorzeichen zählen als ein Zeichen.
- Enthält FS ein anderes Zeichen als das Leerzeichen, dann werden führende Trenner mitgezählt.

Zugriff

Auf Zeilen und Felder können mit folgenden Variablen zugegriffen werden:

\$0	gesamte Zeile
\$1	erstes Feld
\$2	zweites Feld
.	
.	
.	
\$.NF	letztes Feld

Beispiel

Ausgabe des zweiten und ersten Feldes einer Zeile: {print \$2,\$1}

Hinweis

- Für $i > NF$ ist $\$i$ nicht unbedingt der Leerstring.
- Zuweisungen an Felder sind möglich, auch an $\$i$, falls $i > NF$. Dabei wird jedoch NF nicht verändert!

Muster

Als Muster erlaubt awk beliebige logische Verknüpfungen von regulären Ausdrücken und Vergleichen. Im Einzelnen sind folgende Möglichkeiten zulässig:

a) BEGIN und END

Diese beiden speziellen Muster kontrollieren das Programm am Anfang und am Ende.

BEGIN Falls vorhanden, muß BEGIN als erstes Muster angegeben sein. Die zugehörige Aktion wird dann genau einmal vor Bearbeitung der ersten Eingabezeile ausgeführt.

END Falls vorhanden, muß END als letztes Muster angegeben sein. Die zugehörige Aktion wird dann genau einmal zum Schluß des Programms ausgeführt.

b) Reguläre Ausdrücke

Wie bei egrep sind bei awk erweiterte reguläre Ausdrücke als Muster zugelassen. Ein regulärer Ausdruck muß in Schrägstriche eingeschlossen sein: /ausdruck/. Bei der Angabe eines regulären Ausdrucks wird die gesamte Zeile nach einer passenden Zeichenfolge durchsucht (siehe auch ed, egrep und Tabelle im Anhang).

c) Vergleiche

Ein Muster, das einen Vergleich darstellt, hat die allgemeine Form:

ausdruck1 relop ausdruck2

relop ist einer der üblichen Vergleichsoperatoren:

<	kleiner
<=	kleiner oder gleich
==	gleich
!=	ungleich
>=	größer oder gleich
>	größer

Bezüglich Ausdrücke siehe unten, Aktionen.

Ist keiner der Operanden numerisch oder ist aus dem Zusammenhang nicht ersichtlich, ob es sich um einen Textvergleich oder numerischen Vergleich handelt, so wird Textvergleich durchgeführt.

Beispiele

1) Bestimme alle Zeilen mit gerader Feldanzahl:

```
NF%2 == 0
```

2) Suche alle Zeilen, deren zweites Feld um mehr als 100 größer ist als das erste:

```
$2 > $1 + 100
```

d) Mustervergleich

Ein Mustervergleich hat die allgemeine Form:

ausdruck op regulärer Ausdruck,

wobei op sein kann:

~	muß passen zu
!~	darf nicht passen zu

Damit ist es möglich, außer ganzen Zeilen (s.o.) auch einzelne Felder mit Hilfe von regulären Ausdrücken anzusprechen, wie z.B.:

\$1 ~/[Aa]+/	alle Zeilen, deren erstes Feld mit "A" oder "a" beginnt.
\$3 !~/[0-9]/	alle Zeilen, deren drittes Feld keine Dezimalziffer enthält.

e) Bedingungen

Bedingungen sind logische Verknüpfungen von regulären Ausdrücken, Vergleichen oder Mustervergleichen mit Hilfe der Operatoren:

! NICHT
 || ODER
 && UND
 () Klammerung

Die Auswertung einer zusammengesetzten Bedingung erfolgt von links nach rechts.

Beispiel

Suche alle Zeilen, deren erstes Feld mit 'M' beginnt aber nicht 'Müller' ist:

```
$1 >= "M" && $1 < "N" && $1 != "Müller"
```

f) Bereiche

Ein Bereich hat die allgemeine Form:

```
muster1 , muster2
```

und bedeutet, daß die zugehörige Aktion für jede Zeile zwischen dem ersten Auftreten von muster1 und dem ersten Auftreten von muster2 (einschließlich) ausgeführt wird.

Beispiel

Bearbeite den Bereich von Zeile 100 bis Zeile 200 (einschließlich):

```
NR == 100, NR == 200
```

Aktionen

Eine Aktion ist eine Folge von Anweisungen, die abgeschlossen werden durch: "neue Zeile" oder ";" oder "}".

Eine Anweisung kann sein:

- a) Ablaufanweisung
- b) Zuweisung
- c) Ausgabeanweisung
- d) Aufruf einer eingebauten Funktion

Eine Anweisung besteht aus Ausdrücken, in denen Sie auch Variablen verwenden können.

Variablen

Variablen haben einen frei wählbaren Namen (auch Ziffern sind zulässig) und können numerische (Gleitkommazahlen) oder alphanumerische (Zeichenfolgen) Werte annehmen. Konstante Zeichenfolgen müssen Sie beim awk immer in doppelte Hochkommas einschließen. Variablen werden nicht deklariert, die Art des Wertes wird nach dem Kontext festgelegt. Sie werden beim ersten Auftreten mit dem numerischen Wert 0 oder der leeren Zeichenfolge initialisiert. Zeichenreihen werden in numerischem Kontext wenn möglich als numerische Werte interpretiert, falls dies nicht möglich ist, erhalten sie in der Regel den numerischen Wert 0.

Beispiel

x = "Müller" die Variable x hat den Wert 'Müller'
x = "3" + 4 die Variable x hat den Wert 7

awk benutzt eine Reihe vordefinierter Variablen, die bei Bedarf verändert werden können.

vordefinierte Variablen

NR	Nummer der gerade bearbeiteten Zeile
NF	Anzahl der Felder einer Zeile
RS	Zeilentrenner, Vorbesetzung: neue Zeile
FS	Feldtrenner, Vorbesetzung: Leer- und Tabulatorzeichen
ORS	Zeilentrenner für die Ausgabe, Vorbesetzung: neue Zeile
OFS	Feldtrenner für die Ausgabe, Vorbesetzung: Leerzeichen
OFMT	Ausgabeformat für numerische Werte, Vorbesetzung: %.6g
FILENAME	Name der gerade bearbeiteten Eingabedatei ("-" bei Standard-Eingabe)

Achtung

Die Zuweisung einer Datei an FILENAME ändert nicht die aktuelle Eingabedatei!

\$0	gesamte aktuelle Zeile
\$i	i-tes Feld der aktuellen Zeile

Die Feldvariablen $\$i$ können wie normale Variablen benutzt werden und je nach Kontext numerische oder alphanumerische Werte annehmen.

Beispiel

```
{if ($3 > 1000) $3 = "zu groß"}
```

Felder

Als Variable sind außerdem Felder (arrays) mit numerischem Index sowie sogenannte assoziative Felder mit textuellem Index zugelassen. Die Bezeichnung eines Feldelementes lautet allgemein:

```
feldname[index]
```

Felder werden ebenso wie einfache Variable nicht deklariert. Bei der ersten Anwendung wird ein Feld angelegt und nach Bedarf dimensioniert. Eine Besonderheit stellen die assoziativen Felder dar, die eine benutzerfreundliche Textverarbeitung ermöglichen. Die einzelnen Feldelemente können wie üblich durch Angabe ihres Indexnamens angesprochen werden. Zum Durchlaufen aller Elemente eines assoziativen Feldes gibt es die spezielle Laufanweisung:

`for (var in feldname)` Anweisung,

wobei `var` in irgendeiner beliebigen Reihenfolge die bisher vorhandenen Indexwerte annimmt und die Anweisung dann für das entsprechende Feldelement ausgeführt wird.

Achtung

- die Reihenfolge ist rein zufällig
- `var` darf nicht verändert werden!
- Verwenden Sie als assoziativen Feldindex eine konstante Zeichenfolge, so müssen Sie diese in doppelte Hochkommas einschließen wie etwa `häuf ["Student"]` (siehe Beispiel).

Ausdrücke

Zum Rechnen mit Variablen und Konstanten stehen Ihnen folgende C-Operatoren zur Verfügung:

Die Operatoren sind nach steigender Priorität aufgeführt.

<code>=, + =, - =, / =, %=</code>	Zuweisungsoperatoren, wobei die zusammengesetzten Operatoren "op=" wie folgt interpretiert werden:
<code> </code>	E1 op= E2 bedeutet E1 = E1 op E2 logisches ODER
<code>&&</code>	logisches UND
<code>!</code>	logisches NICHT
<code>>, >=, <, <=, !=, ==</code>	Vergleichsoperatoren
<code>~, !~</code>	Mustervergleichsoperatoren
aneinanderschreiben	Zeichenfolgenverknüpfung
<code>+, -</code>	plus, minus
<code>++, --</code>	Inkrement, Dekrement

a) Ablaufanweisungen

Ablaufanweisungen haben die gleiche Syntax wie in C und können im Einzelnen eines der folgenden Konstrukte sein:

<code>if (bedingung) anweisung [else anweisung]</code>	Abfrage
<code>for (ausdruck;bedingung; ausdruck) anweisung</code>	Gezählte Wiederholung
<code>for (var in feld) anweisung</code>	Felddurchlauf
<code>while (bedingung) anweisung</code>	Schleife
<code>break</code>	Springe aus Schleife
<code>continue</code>	Starte nächsten Durchlauf
<code>{ [anweisung]...}</code>	Zusammenfassen
<code>next</code>	Sprung zur nächsten Eingabezeile
<code>exit</code>	Springe zu Muster END, falls vorhanden oder beende

b) Zuweisung

Die Zuweisung hat die Form

```
variable = ausdruck
```

Mit der Zuweisung

```
$2 = ""
```

wird z.B. der Inhalt des zweiten Feldes gelöscht.

Achtung

Die Zuweisung bewirkt eine Wertänderung bei der Variablen auf der linken Seite der Zuweisung. Diese Wertänderung betrifft allerdings nur die Ausgabe. Der Inhalt der awk- Eingabedatei wird dadurch nicht geändert.

c) Ausgabeanweisung

Für die Ausgabe stehen zwei awk-Kommandos zur Verfügung:

```
print [arg,...] [> datei]
```

Standard (keine Angabe): eine Zeile wird auf die Standard-Ausgabe ausgegeben.

```
arg1,arg2,...
```

die Argumente werden in der angegebenen Reihenfolge, getrennt durch den Ausgabefeldtrenner OFS (Voreinstellung Leerzeichen), auf die Standard-Ausgabe ausgegeben. An OFS kann eine beliebige Zeichenfolge zugewiesen werden.

Beispiel

```
print $1,$2
```

Ausdruck des ersten und zweiten Feldes getrennt durch OFS

```
print $1$2
```

Ausdruck der konkatenierten Zeichenfolge '\$1\$2' ohne OFS

> datei Umleiten der Ausgabe.
> > und | sind ebenso zulässig.

datei ist entweder eine Variable, die als Wert einen Dateinamen hat, oder ein Dateiname, den Sie in doppelte Hochkommas einschließen müssen.

Achtung

Geben Sie für datei den Dateinamen der awk- Eingabedatei an, so meldet awk keinen Fehler. Sie können dadurch den ursprünglichen Inhalt Ihrer Datei zerstören.

An die Ausgabe einer print-Anweisung wird der Ausgabezeilentrenner ORS angehängt (Voreinstellung neue Zeile). Die Variable OFMT gibt an, in welchem Format numerische Argumente ausgegeben werden (Voreinstellung %.6g)

`printf format [arg,...] [> datei]`

`printf` ist identisch zu `printf` in C. Es werden keine Ausgabetrener OFS und ORS ausgegeben.

Beispiel `printf "%-20g %5d \n", $3"."$2, $1`

d) Aufruf eingebauter Funktionen

`awk` bietet keine Möglichkeit, Funktionen selbst zu definieren, stellt aber die folgenden Funktionen zur Verfügung:

<code>exp(ausdruck)</code>	Exponentialfunktion
<code>int(ausdruck)</code>	Ganzzahliger Anteil eines Ausdrucks
<code>log(ausdruck)</code>	natürlicher Logarithmus zur Basis e
<code>sqrt(ausdruck)</code>	Quadratwurzel
<code>index(s1, s2)</code>	Position des ersten Vorkommens der Zeichenfolge s2 in s1 0, falls nicht vorhanden
<code>length(s)</code>	Länge der Zeichenfolge s
<code>length</code>	Länge der gesamten Zeile,
<code>split(s, feldname, sep)</code>	Teilt die Zeichenfolge s in ein Feld <code>feldname</code> auf, das beginnend mit 1 aufsteigend indiziert wird. Als Trennsymbol zählt <code>sep</code> , falls angegeben, ansonsten FS.
<code>sprintf(format,e1,e2,...)</code>	Formatiert die Ausdrücke <code>e1, e2,...</code> entsprechend <code>format</code> (wie in C) und gibt die so entstandenen Zeichenfolgen aus.
<code>substr(s, m, n)</code>	Teilzeichenreihe von s der Länge n, beginnend ab Position m. Falls n fehlt, geht die Teilzeichenreihe bis zum Ende von s.

Beispiele

1. Die Datei test habe folgenden Inhalt:

```
1 1
2 2
3 3
4 4
5 5
```

Es sollen alle Zeilen ausgegeben werden, deren Quersumme größer als 6 ist:

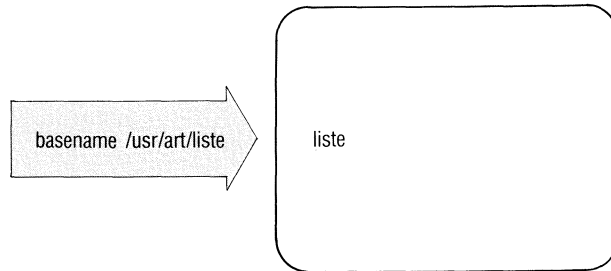
```
awk '$1+$2 > 6 {print $0}' test
```

2. In einer Datei jugend seien Jugendliche geführt, wobei im zweiten Feld die Angabe Schüler, Student, Lehrling oder Sonstige steht. Für eine Statistik sollen Schüler und Studenten gezählt werden:

```
awk '$2 ~ /Schüler/ {häuf["Schüler"]++}
     $2 ~ /Student/ {häuf["Student"]++}
     END {print "Schüler:" häuf["Schüler"];
          print "Student:" häuf["Student"]}' jugend
```

>>>> ed, sed

Dateinamen vom Pfad trennen



`basename` erzeugt aus einer Zeichenfolge mit Zeichen `"/` (einem Pfadnamen) die Zeichenfolge nach dem letzten `"/` (Dateiname ohne Pfad). `basename` verwendet man in Shell-Prozeduren.

basename *zeichenfolge* [*ende*]

zeichenfolge

ein Pfadname. `basename` löscht alle Zeichen bis zum letzten `"/` einschließlich dieses Schrägstrichs, z.B.: `basename /usr/eva/src/prog` ergibt die Ausgabe `"prog"`.

`basename` prüft nicht, ob dieser Pfadname im Dateisystem existiert.

ende

Endbuchstaben, die `basename` ebenfalls aus der Zeichenfolge löschen soll, z.B.: `basename ./src/program ram` liefert die Ausgabe `"prog"`.

basename

Beispiel

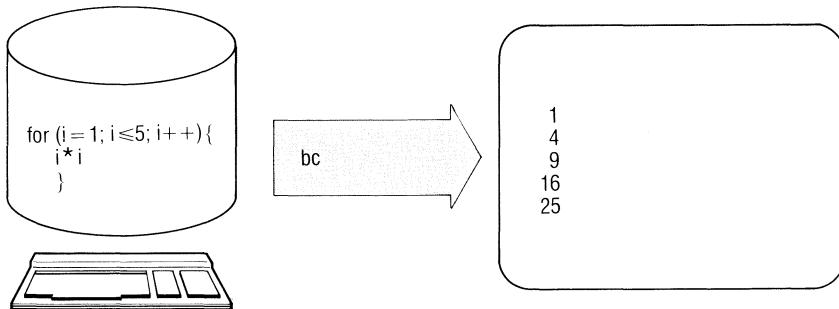
Die folgende Prozedur übersetzt das C-Quellprogramm aus der Datei, deren Name beim Prozeduraufruf in \$1 übergeben wird. Das Übersetzungsergebnis schreibt cc standardmäßig in die Datei a.out. mv benennt a.out um. Den neuen Dateinamen bildet basename aus dem Namen in \$1.

```
cc $1 mv a.out `basename $1 .c`
```

Steht z.B. in \$1 der Name /usr/eva/src/prog.c, dann lautet der neue Name "prog".

>>>> Shell (Abschnitte 3.6 bis 3.8)

Arithmetische Sprache



bc liest C-ähnliche Programme für Rechenfunktionen und macht daraus Anweisungen für den Tischrechner dc. bc ruft dc auf und gibt die Ergebnisse mit beliebiger Genauigkeit aus.

bc[₋schalter...][₋datei...]

schalter

l müssen Sie setzen, wenn Sie eine der folgenden Funktionen verwenden.

s(x)	sin
c(x)	cos
e(x)	Exponentialfunktion
l(x)	log
a(x)	arctan
j(n,x)	BESSEL-Funktion

Alle Funktionsargumente werden als Werte übergeben.

c nur dc-Anweisungen erzeugen.
bc gibt die dc-Anweisungen auf die Standard-Ausgabe aus.

datei Name einer Datei mit einem bc-Programm. Anschließend an die angegebenen Dateien liest bc von der Standard-Eingabe. Sie können dann weitere Anweisungen eingeben oder bc mit quit beenden

Standard (keine Angabe): bc liest von der Standard-Eingabe.

Syntax der bc-Programme:

bc verarbeitet folgende Elemente der Sprache C.

In der folgenden Darstellung bedeutet:

buchstabe	Buchstaben a-z
ausdruck	Ausdruck wie in C
statement	C-Statement

Namen:

- einfache Variablen: buchstabe
- Arrayelemente: buchstabe[ausdruck]
- die Worte : 'ibase', 'obase' und 'scale'

andere Operanden:

- beliebig lange Zahlen, wahlweise mit Vorzeichen und Dezimalpunkt.
- (ausdruck)
- sqrt (ausdruck) Quadratwurzel von ausdruck
- length (ausdruck) Zahl der signifikanten Dezimalziffern
- scale (ausdruck) Zahl der Ziffern rechts vom Dezimalpunkt
- buchstabe (ausdruck,...,ausdruck)

Kommentare:

- werden eingeschlossen in /* und */.

Operatoren:

- + - * / % ^ (% ist der Rest, ^ ist der Exponent)
- ++ -- (Vorstellungen und Anfügungen bei Namen)
- == <= >= != <>
- = += -= /= %= ^=

Statements:

- `ausdruck`
- `{statement;...;statement}`
- `if (ausdruck)statement`
- `while (ausdruck)statement`
- `for (ausdruck;ausdruck;ausdruck)statement`
- `null statement`
- `break`
- `quit`
- Funktionsdefinitionen

```
define buchstabe(buchstabe,...,buchstabe){
    auto buchstabe,...buchstabe
    statement;...statement
    return (ausdruck)
}
```

Hinweis

- Ist ein Statement ein Ausdruck, wird sein Wert ausgedruckt, es sei denn, daß der Hauptoperator eine Zuweisung ist.
- Statements trennen Sie durch Zeichen ";" oder "neue Zeile".
- Mit `scale` können Sie festlegen, mit wievielen Stellen Ausdrücke berechnet werden (ähnlich wie bei `dc`).
- Mit `ibase` oder `obase` können Sie die Ein- und Ausgabe-Nummer-Radices festlegen.
- Sie können denselben Buchstaben gleichzeitig für ein Array, eine Funktion oder eine einfache Variable benutzen. Für ein Programm sind alle Variablen global. "Auto"-Variable werden während Funktionsaufrufen außer Kraft gesetzt. Wenn Sie Arrays als Funktionsargumente verwenden oder als automatische Variable definieren wollen, müssen Sie dem Arraynamen leere eckige Klammern anfügen.

Beispiele

1. Das folgende bc-Programm definiert die Funktion e(x), die angenähert die Werte der e-Funktion berechnet.

```
scale=20
define e(x){
    auto a, b, c, i, s
    a=1
    b=1
    s=1
    for (i=1;1==1;i++){
        a=a*x
        b=b*i
        c=a/b
        if (c==0) return (s)
        s=s+c
    }
}
```

2. `for (i=1;i<10;i++) e(i)`

druckt die angenäherten Werte der Exponentialfunktion für die ersten 10 ganzen Zahlen.

bc ist mit Schalter `l` aufzurufen, da die Funktion `e(x)` enthalten ist:

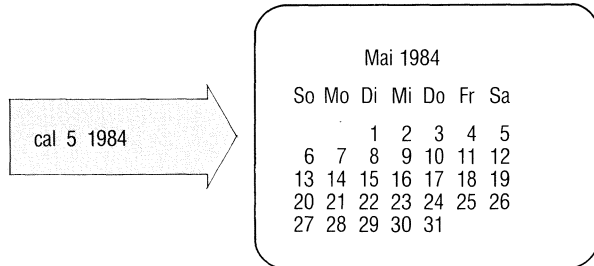
```
$ bc -l
for (i=1;i<=10;i++) e(i)
```

hier kommt das Ergebnis, rechnen Sie nach!

```
quit
$
```

>>>> dc

Kalender ausgeben



Das Kommando `cal` gibt einen Kalender aus.

`cal[_monat]_jahr`

- monat** Der Kalender wird nur für den angegebenen Monat ausgegeben.
Zulässige Werte: 1 bis 12.
- Standard (keine Angabe):* Kalender für das ganze Jahr ausgeben.
- jahr** Der Kalender wird für das angegebene Jahr ausgedruckt.
Zulässige Werte: 1 bis 9999.

Hinweis

Die Angabe `cal_84` bezieht sich auf das Jahr 84 und nicht auf 1984.

Ende-Status: immer 0

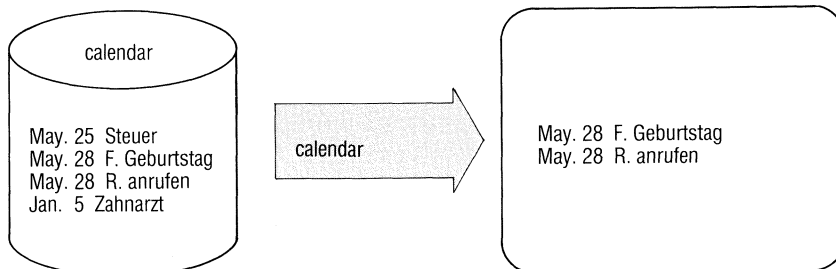
Beispiel

\$ cal 5 1983

 Mai 1983

So	Mo	Di	Mi	Do	Fr	Sa
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

Erinnerungsdienst, Datum in englischer Schreibweise



calendar ist eine Gedächtnisstütze für Termine. Die Termine tragen Sie in einer Datei "calendar" im Login-Dateiverzeichnis ein. Das Kommando calendar gibt aus dieser Datei jeweils die Zeilen aus, die das aktuelle Datum enthalten.

calendar hat diesselbe Funktion wie das Kommando kalender. Die Termine werden lediglich in englischer Schreibweise erfaßt.

calendar[_-]

kein Schalter

calendar druckt aus der Benutzerdatei "calendar" alle Zeilen aus, in denen an beliebiger Stelle das heutige oder morgige Datum steht. Die üblichen Monat-Tag-Angaben wie etwa Dec.7, December 7, 12/7 usw. werden erkannt.

An Wochenenden werden für "morgen" die Tage einschließlich Montag ausgesucht.

- calendar bezieht sich auf alle Benutzer, die eine Datei "calendar" in ihrem Login-Dateiverzeichnis haben. Jedem solchen Benutzer werden die eventuell gefundenen Zeilen aus seiner calendar-Datei mit mail zugesandt.

Hinweis

Das Konzept vom verlängerten "morgen" an Wochenenden gilt nicht an Feiertagen.

Ende-Status: immer 0

Beispiel

Die Datei calendar im Login-Dateiverzeichnis habe folgenden Inhalt:

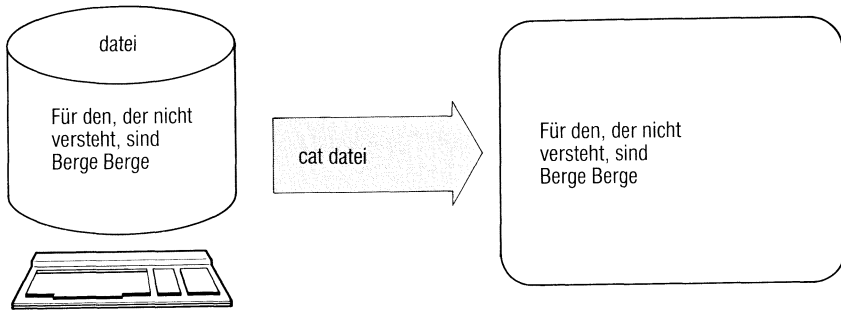
```
May.31 Nicht vergessen: Reinhardt anrufen!  
Jun/8 Bei gutem Wetter Gartenfest bei Stingls  
Heute ist 7/8 !  
August 15 : Termin bei Kantarelis  
Wichtig: Heute Antrag für ALFGM abgeben (Jun.8)  
.  
.
```

Wenn Sie am 8.Juni calendar aufrufen, erhalten Sie die Ausgabe:

```
Jun/8 Bei gutem Wetter Gartenfest bei Stingls  
Wichtig: Heute Antrag für ALFGM abgeben (Jun.8)
```

> > > > at, kalennder, mail

Dateien ausgeben – catenate and print



cat liest Dateien in der Reihenfolge, wie sie genannt sind und gibt deren Inhalte nacheinander aus.

cat[**-u**][**-**datei...]

kein schalter

Ausgabe in Blöcken von 512 Byte Länge (gepuffert).

u byteweise Ausgabe ohne Zwischenspeichern (ungepuffert).

datei Name der auszugebenden Datei.
Das Zeichen "-" anstelle eines Dateinamens bedeutet: cat liest von der Standard-Eingabe.

Standard (keine Angabe): cat liest von der Standard-Eingabe.

Hinweis

Das Kommando `cat a b > a` würde den Inhalt der Datei a zerstören, bevor cat ihn lesen kann - Vorsicht!

Beispiele

1. Welchen Inhalt hat datei1?

```
$ cat datei1
Jeder weiß, was so ein Mai-
käfer für ein Vogel sei.
```

In datei2 schreiben Sie nun zwei Zeilen Text:

```
$ cat >> datei2
in den baeumen hin und her
kriecht und fliegt und krabbelt er
[END]
$
```

Nun soll der Inhalt von datei1 und datei2 in datei3 gebracht werden, ergänzt um zwei weitere neu eingegebene Zeilen:

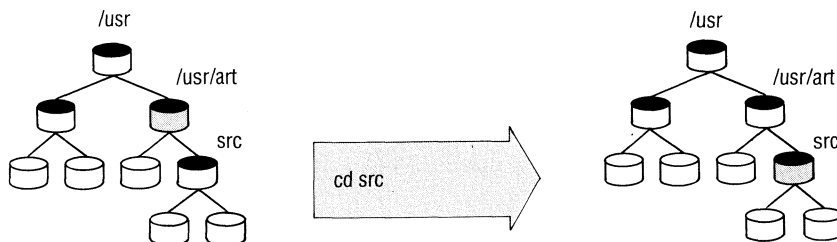
```
$ cat datei1 datei2 - > datei3
max und moritz immer munter
schuettern sie vom baum herunter
[END]
$
```

2. cat ohne Angabe von Dateien wirkt ähnlich wie echo, liest jedoch von der Standard-Eingabe. Das sehen Sie, wenn Sie Schalter u angeben (Ausgabe ungepuffert):

```
$ cat -u
adam
adam
eva
eva
[END]
$
```

> > > > pr, cp

Dateiverzeichnis wechseln – change working directory



cd wechselt vom aktuellen in das angegebene Dateiverzeichnis.

cd[_dateiverzeichnis]

dateiverzeichnis

Name des Dateiverzeichnisses, in das Sie wechseln wollen. Dieses wird zum neuen aktuellen Dateiverzeichnis. Sie müssen die Erlaubnis zum Durchsuchen dieses Dateiverzeichnisses haben (siehe Abschnitt 2.5 und chmod).

Standard (keine Angabe): Wechseln in das Home-Dateiverzeichnis. Das Home-Dateiverzeichnis ist identisch mit dem Login-Dateiverzeichnis, wenn Sie in der Variablen HOME nichts anderes vereinbart haben.

Hinweis

cd erzeugt keinen eigenen Prozeß.

Beispiele

1. Wechseln in das Dateiverzeichnis konten

```
$ cd konten
$ pwd
/usr/blumann/konten
```

Der Pfadname des neuen Dateiverzeichnisses ist `/usr/blumann/konten`.

2. Wechseln in das Dateiverzeichnis adressen, das ebenfalls ein Unterdaterverzeichnis von `/usr/blumann` ist.

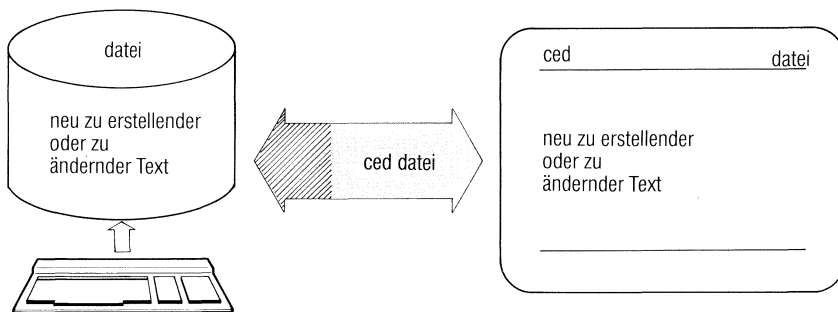
```
cd ../adressen
```

”..” bezeichnet das Dateiverzeichnis, das dem aktuellen (hier `/usr/blumann/konten`) übergeordnet ist. Unabhängig vom aktuellen Dateiverzeichnis könnten Sie eingeben:

```
cd /usr/blumann/adressen
```

>>>> Shell (Abschnitte 3.6 bis 3.8)

Bildschirmorientierter Editor



Der ced ist ein bildschirmorientierter Editor. Bildschirmorientiert heißt: Sie können die Schreibmarke frei über den Bildschirm bewegen, um Text an beliebiger Stelle einzugeben oder zu ändern. Der ced bietet Ihnen ein "Menü" verschiedener Bearbeitungs-Modi. Je nach Modus können Sie:

- direkt Text eingeben oder
- Kommandos eingeben für unterschiedliche Bearbeitungsfunktionen, z.B. Kopieren, Löschen usw.

ced[_-schalter][_-datei]

schalter

f_-zeilennummer

Der ced positioniert die Schreibmarke auf die angegebene Zeile.

s_-suchzeichenfolge

Der ced positioniert die Schreibmarke auf die erste Zeile, die die Suchzeichenfolge enthält. Ist die Suchzeichenfolge nicht in der Datei vorhanden, zeigt der ced den Dateianfang.

t textmarkierung

Der ced öffnet die Datei, die die angegebene Textmarkierung enthält und positioniert die Schreibmarke darauf. Die Funktion setzt voraus, daß Sie mit dem Kommando ctags eine Datei "tags" erstellt haben (siehe C-Entwicklungssystem).

datei

Name der Datei, die Sie bearbeiten wollen.

Gibt es im aktuellen Dateiverzeichnis keine Datei dieses Namens, legt sie ced neu an.

Standard (keine Angabe): ced öffnet die Datei, die Sie zuletzt an dieser Datensichtstation mit dem ced bearbeitet haben und positioniert die Schreibmarke dahin, wo sie zuletzt stand. ced holt diese Informationen aus der Datei /usr/lib/ced/CE.ttyname.benutzerkennung.

Arbeiten mit dem ced

ced arbeitet auf einer Kopie der Datei, die Sie bearbeiten wollen. ced übernimmt die Änderungen erst dann in die Datei, wenn Sie:

- den ced verlassen und dabei "j" angeben (siehe Modus v) oder
- die Änderungen mit Modus a abspeichern. Dabei können Sie auch einen anderen Dateinamen wählen.

Der Bildschirm

** CED Texteditor V1.0 Zeile: 1 Spalte: 1 Name: brief

Sie wollen neuen Text eingeben?
Bitte geben Sie Ihren Text ein!

Kopfzeile:





Die Kopfzeile enthält:

- die ced-Version,
- Zeilen- und Spaltenposition der Schreibmarke,
- den Dateinamen der gerade im Fenster gezeigten Datei.

Fenster:

Das Fenster zeigt einen Ausschnitt aus der bearbeiteten Datei (19 Zeilen). An der Schreibmarkenposition in der Kopfzeile erkennen Sie, welchen Abschnitt der Datei Sie gerade bearbeiten.

Das Fenster können Sie verschieben:

-  nach oben um 9 Zeilen (1/2 Fenster),
-  nach unten um 9 Zeilen (1/2 Fenster),
-  nach unten um 19 Zeilen (1 Fenster),
-  nach oben um 19 Zeilen (1 Fenster).

Mit Modus f können Sie das Fenster beliebig nach oben, unten, rechts und links verschieben.


Das Fenster verschiebt sich automatisch, wenn Sie mit der Schreibmarke an den Rand des gezeigten Bereiches kommen.



Bedienbereich:

Im Bedienbereich zeigt der ced an, welchen Modus Sie gerade gewählt haben, und welche Funktionen Sie darin ausführen können.
































In einigen Modi können Sie im Bedienbereich Text eingeben, z.B. den Dateinamen im Modus d (Datei wechseln) oder die zu suchende Zeichenfolge im Modus s (suchen).

Korrigieren können Sie eine Eingabe im Bedienbereich so:

einzelne Zeichen löschen mit 

die ganze Eingabe löschen mit  

Schreibmarke bewegen:

	nach oben		Tabulator rechts
	nach unten		Tabulator links
	nach rechts		Tabulatorpositionen sind
	nach links		Spalte 1, 9, 17, 25 usw.
	Mit dieser Taste können Sie auf Bereichsgrenzen positionieren:		
 	rechter Rand des Fensters		
 	linker Rand des Fensters		
 	erste Zeile des Fensters		
 	letzte Zeile des Fensters (falls nicht Dateiende)		
 	}	letzte Spalte (127)	
  			
 	}	erste Spalte (1)	
  			
  	erste Zeile der Datei		
  	letzte Zeile der Datei		

Modus auswählen

Drücken Sie nacheinander die Taste `MENU` und eine Buchstabentaste. Sie wählen damit eine der folgenden Funktionen:

<i>Taste</i>	<i>Funktion</i>
<code>MENU</code> <code>a</code>	abspeichern
<code>MENU</code> <code>b</code>	Zeilenbereich bearbeiten
<code>MENU</code> <code>d</code>	Dokument wechseln
<code>MENU</code> <code>e</code>	einfügen
<code>MENU</code> <code>f</code>	Fenster positionieren
<code>MENU</code> <code>k</code>	Kommando ausführen
<code>MENU</code> <code>n</code>	neuen Text eingeben
<code>MENU</code> <code>p</code>	tasten programmieren
<code>MENU</code> <code>r</code>	Rechteck bearbeiten
<code>MENU</code> <code>s</code>	suchen
<code>MENU</code> <code>t</code>	Textmarkierung suchen (nur in Verbindung mit ctags)
<code>MENU</code> <code>v</code>	ced verlassen
<code>MENU</code> <code>x</code>	Tastenbelegung anzeigen
<code>MENU</code> <code>z</code>	Zeile bearbeiten
<code>MENU</code> <code>!</code>	Shell aufrufen

Nach dem Aufruf befindet sich der ced immer im Modus n "neuen Text eingeben".

`MENU` `HELP` zeigt immer die Kurzbeschreibung des ced.

In der Kurzbeschreibung können Sie vorwärts und rückwärts blättern, wie in einer anderen Datei. Wenn Sie vorher im Modus s (suchen) eine Zeichenfolge angeben, können Sie mit F13 bzw. F14 diese Zeichenfolge in der Kurzbeschreibung suchen.

Hinweis

- Der ced verarbeitet Eingaben sofort, ohne daß Sie die Taste `↵` drücken müssen. Eine Ausnahme sind Eingaben in der Bedienzeile.
- Abhängig vom Modus haben die Tasten unterschiedliche Wirkung.

Beschreibung der Modi

a Abspeichern von Text

Im Bedienvereich geben Sie den Namen der Datei an, in der der Text abzuspeichern ist, den Sie gerade bearbeiten. Dann drücken Sie die Taste .

Geben Sie keinen Dateinamen an, schreibt der ced den Text in die aktuelle Datei. Der Dateiname steht in der Kopfzeile. Der ursprüngliche Dateiinhalt wird überschrieben.

b Zeilenbereich bearbeiten

In diesem Modus können Sie Bereiche von aufeinanderfolgenden Zeilen bearbeiten. Markieren Sie die oberste und unterste Zeile des Bereichs (Funktion m).

Folgende Funktionen stehen Ihnen zur Verfügung:

m markieren einer Zeile:

Setzen Sie die Schreibmarke auf die gewünschte Zeile und drücken Sie die Taste "m". ced meldet in der Status-Zeile: MARKE GESETZT. Es gelten immer die beiden zuletzt gesetzten Marken. Löschen eines Bereichs (Funktion l) löscht auch die Marken.

e einfügen von Leerzeilen:

Oberhalb der Schreibmarke werden soviele Leerzeilen eingefügt, wie der zuletzt markierte Bereich umfaßt.

k kopieren des markierten Bereichs:

Oberhalb der Schreibmarke wird eine Kopie des markierten Bereichs eingefügt.

l löschen und speichern des markierten Bereichs:

Der markierte Bereich wird gelöscht und zwischengespeichert. Jedes weitere Löschen und Speichern überschreibt den zuletzt gespeicherten Bereich!

z zurückholen des zwischengespeicherten Bereichs:

Der zuletzt gelöschte und damit zwischengespeicherte Bereich wird oberhalb der Schreibmarke eingefügt.

Wollen Sie nur eine einzige Zeile bearbeiten, können Sie diese durch zweimaliges Drücken der Taste "m" markieren.

d Datei wechseln

Im Bedienbereich geben Sie den Namen einer weiteren Datei an, die Sie bearbeiten möchten. Drücken Sie nun die Taste . Im Fenster sehen Sie den Anfang der neuen Datei. Der ced ist jetzt im Modus n (neuen Text eingeben).

Existiert die angegebene Datei nicht, fragt ced Sie, ob die Datei angelegt werden soll. Antworten Sie mit j, wird diese Datei angelegt.

Mehrere Dateien parallel bearbeiten

Haben Sie mehrere Dateien geöffnet, wie oben beschrieben, dann können Sie:

- von einer Datei auf die nächste umschalten:
Drücken Sie im Modus d nur die Taste . ced schaltet zwischen den beiden zuletzt bearbeiteten um.
- Text von einer Datei in eine andere übertragen:
 - 1) Auf Bereiche, die Sie markiert haben (Modus b, r oder z) können Sie direkt von der zweiten Datei aus zugreifen, wenn Sie in den entsprechenden Modus geschaltet haben.
 - 2) Zwischengespeicherte Bereiche können Sie in der anderen Datei zurückholen (Modus b oder z).

Schalten Sie in den Modus v (ced verlassen), fragt ced für die aktuelle Datei, ob sie gesichert werden soll. Anschließend schaltet ced auf die nächste offene Datei zurück, in der Sie geändert haben. Auf diese Weise können Sie alle während einer ced-Sitzung bearbeiteten Dateien abschließen.

e einfügen

Wollen Sie Text einfügen, schalten Sie in den Modus e (Text einfügen). eingefügt. Der dort stehende Text wird nach rechts verschoben, wenn nötig, auch aus dem Fenster hinaus. Der ced meldet, wenn durch das Einfügen die Zeile zu lang wird (max. 127 Zeichen).

f Fenster positionieren

Das Fenster können Sie in diesem Modus auf zwei Arten verschieben:

- auf eine bestimmte Zeile oder Spalte:
Geben Sie eine Zahl ohne Vorzeichen im Bedienbereich an und drücken Sie eine der folgenden Tasten:

oder oder

Das Fenster beginnt mit der angegebenen Zeile. Die Schreibmarke steht links oben.

oder

Das Fenster beginnt mit der angegebenen Spalte. Die Schreibmarke steht links oben.

- relativ zur aktuellen Position:
Geben Sie eine Dezimalzahl n mit Vorzeichen ”+” oder ”-” ein und drücken Sie eine der folgenden Tasten:

oder

n Zeilen nach unten

n Zeilen nach oben

n Zeilen nach unten

n Spalten nach links

n Spalten nach rechts

Ob Sie $+n$ oder $-n$ angeben, ist völlig gleichgültig!

k Kommando ausführen

In diesem Modus können Sie Shell-Kommandos aufrufen und diesen einen Bereich Ihrer Datei als Eingabe übergeben. Die Ausgabe des ausgeführten Kommandos schreibt ced anstelle des Eingabebereichs in die Datei.

In der Eingabezeile des Menübereichs legen Sie den Eingabebereich und das auszuführende Kommando fest:

[bereich] `␣` kommando

Dabei bedeuten:

bereich Eingabebereich für das Kommando. Dieser Bereich der Datei wird an die Standard-Eingabe des Kommandos übergeben und in der Datei gelöscht.

nl n Zeilen ab der Schreibmarke.

n n Abschnitte ab dem Abschnitt, in dem die Schreibmarke steht. Ein Abschnitt ist ein Bereich ohne Leerzeilen, d.h. mit einer Leerzeile endet ein Abschnitt.

- leerer Eingabebereich, aus der Datei wird nichts gelöscht.

bereich nicht angegeben

Eingabebereich ist der Bereich ab der Schreibmarke bis zum Ende des Abschnitts.

kommando ist das auszuführende Kommando. Eine fehlerhafte Angabe meldet ced und löscht den Eingabebereich nicht. Zugelassen sind alle Shell-Kommandos, außer: login, su und ähnlichen Kommandos, die die Shell-Umgebung verändern.

Die Ausgabe des Kommandos wird an Stelle der gelöschten Eingabe eingefügt. Wollen Sie keine Eingabe löschen, müssen Sie einen leeren Eingabebereich angeben (`-0l` oder `-0` oder einfach `-`).

n **neuen Text eingeben**

In diesem Modus können Sie Text eingeben. Sie überschreiben dabei schon vorhandenen Text.

Folgende Tasten können Sie dabei verwenden:

CHAR

um einzelne Buchstaben einzufügen oder zu löschen

LINE

um Zeilen einzufügen oder zu löschen

p Tasten programmieren

In diesem Modus können Sie beliebige Zeichenfolgen, Schreibmarkenbewegungen, Modusumschaltungen eingeschlossen, (fast) beliebigen Tasten zuweisen. Vor allem sollten Sie dafür die F-Tasten verwenden. Wenn Sie dann eine so definierte Taste drücken, erhält der ced die entsprechenden Zeichenfolgen bzw. Anweisungen.

Wie belegen Sie eine Taste?

Wenn Sie in den Modus "Tasten belegen" umschalten, erscheint in der Kopfzeile der Text `PROTO`. Alles, was Sie von nun an ausführen, wird gespeichert: Text, den Sie eingeben, Modus-Umschaltungen usw.

Um diese Aktionen einer Taste zuzuordnen, drücken Sie erneut die Tasten MENU und p . Der ced meldet:

```
****GEWUENSCHTE TASTE DRUECKEN
```

Nun drücken Sie die Taste, der Sie die gespeicherten Aktionen zuordnen wollen. ced schaltet wieder in den Modus n (neuen Text eingeben).

Welche Tasten können Sie belegen?

Sie können alle Tasten belegen, außer MENU und p .

Verwenden Sie vor allem die Tasten F1, F2 usw.

Die Tasten F9 bis F17 sind vom ced mit Standard-Funktionen vorbelegt (siehe unten).

Neu belegte Tasten verlieren natürlich ihre ursprüngliche Funktion. Die Belegung bleibt nur für die Dauer der ced-Sitzung erhalten!

r Rechteck bearbeiten

Ein Rechteck ist ein Bereich der Datei, den Sie durch Markieren zweier Ecken festlegen können (Funktion m). Die Ecken müssen sich diagonal gegenüberliegen.

m Markieren einer Ecke:

Markiert wird die Position, auf der die Schreibmarke steht, wenn Sie Taste "m" drücken. ced meldet: MARKE GESETZT. Es gelten immer die beiden zuletzt gesetzten Marken. Löschen eines Rechtecks (Funktion l) löscht auch die Marken.

l Löschen eines Rechtecks:

Das markierte Rechteck wird gelöscht. Es wird *nicht* gespeichert! Wenn Sie ein Rechteck verschieben wollen, kopieren Sie es erst (Funktion h, u oder v) und löschen es anschließend.

h horizontal einfügen:

Das markierte Rechteck wird mit seiner linken oberen Ecke an der Schreibmarkenposition eingefügt. Text in den betroffenen Zeilen ab der Schreibmarkenspalte wird nach rechts verschoben.

v vertikal einfügen:

Das markierte Rechteck wird vor der Schreibmarkenzeile eingefügt. Es steht mit dem linken Rand auf der Spalte, in der die Schreibmarke steht.

u überschreibend kopieren:

Das markierte Rechteck wird mit der linken oberen Ecke auf die Schreibmarkenposition kopiert. Vorhandener Text wird dabei überschrieben.

s suchen nach Zeichenfolgen

Im Bedienbereich tragen Sie die Zeichenfolge ein, nach der Sie suchen wollen. Den Suchvorgang starten Sie mit einer der folgenden Tasten:

↓ oder ↘

suchen in Richtung Dateieinde

↑

suchen in Richtung Dateianfang

Tritt die Zeichenfolge bis zum Dateiende bzw. Dateianfang nicht mehr auf, meldet ced TEXT NICHT GEFUNDEN.

Führende und anhängende Leerzeichen sind signifikant. Im Gegensatz zu anderen Kommandos kennt der ced keine regulären Ausdrücke.

t Textmarkierung

Dieses Kommando ist nur dann sinnvoll, wenn Sie das CES (C-Entwicklungssystem) besitzen. Wenn Sie mehrere C-Quelldateien haben, die zu einem größeren Programmsystem gehören, können Sie sich mit Hilfe des Kommandos ctags eine Datei "tags" erzeugen, die die Namen und genauen Fundorte (Datei und Position) aller zu diesem Programmsystem gehörigen Funktionen enthält.

Im Modus "Textmarkierung" geben Sie in der Bedienzeile den Namen der Funktion an, zu der Sie umschalten möchten. Der ced schaltet in die Datei an die Stelle um, an der sich die Definition dieser Funktion befindet. Die Datei "tags" muß im aktuellen Dateiverzeichnis vorhanden sein und den Namen der angegebenen Funktion enthalten.








v ced verlassen


Um Ihre ced-Sitzung zu beenden, drücken Sie und geben ein v ein. Jetzt können Sie wählen, ob Sie die von Ihnen gemachten Änderungen retten wollen oder nicht. Geben Sie j oder n ein:

- j die geänderte Datei wird gerettet, alle Änderungen stehen jetzt in Ihrer Datei.
- n die Änderungen werden vergessen, Ihre Datei hat den ursprünglichen Inhalt. Wurde die Datei neu angelegt, bleibt sie in diesem Fall leer.

x Tastenprogrammierung anzeigen

Wollen Sie die von Ihnen programmierten Tasten und die diesen zugeordneten Texte ansehen, benutzen Sie das Kommando x. Auf dem leeren Schirm wird Ihnen eine vollständige Liste gezeigt. Auf der linken Seite des ersten Doppelpunktes ist die Taste verzeichnet, auf der rechten Seite die dieser Taste zugeordnete Zeichenfolge. Dabei bedeuten:

->		Schreibmarke nach rechts
<-		Schreibmarke nach links
UP		Schreibmarke nach oben
DOWN		Schreibmarke nach unten
HOME		Schreibmarke nach links oben
MENU		gedrückt
RETURN		gedrückt

Die von ced vorbelegten Tasten # F9 bis # F17 werden nicht angezeigt. Zurück in die Menüauswahl gelangen Sie mit .

z eine Zeile bearbeiten

Welche Zeile Sie bearbeiten wollen, wählen Sie mit der Schreibmarke.

- l Löschen und Speichern der Zeile:
Die gewählte Zeile wird gelöscht und zwischengespeichert. Jedes weitere Löschen und Speichern überschreibt die zuletzt gespeicherte Zeile.
- z Zurückholen der zwischengespeicherten Zeile:
Die zuletzt zwischengespeicherte Zeile wird oberhalb der Schreibmarke eingefügt.
- e Einfügen einer Leerzeile:
Eine Leerzeile wird oberhalb der Schreibmarke eingefügt.
- t Teilen der Zeile:
Der Rest der Zeile ab der Schreibmarkenposition wird in eine neue Zeile geschrieben.
- v Verbinden zweier Zeilen:
Die gewählte Zeile und die darauf folgende Zeile werden zu einer Zeile verbunden.

- r Rest der Zeile löschen:
Der Rest der Zeile ab der Schreibmarkenposition wird gelöscht.

! Shell aufrufen

Mit diesem Modus verlassen Sie den ced vorübergehend. Der ced ruft das Programm auf, das Sie in der Shellvariablen SHELL angegeben haben. Diese Variable können Sie z.B. in Ihrer .profile-Datei setzen. Standardmäßig nimmt der ced als Programm /bin/sh an. Sie können so vorübergehend in der Shell arbeiten, ohne daß der Inhalt der editierten Datei verlorengeht. Beenden Sie die Shell mit der Taste **END**, gelangen Sie wieder in den ced zurück.

Tasten mit besonderer Bedeutung

Außer den bereits beschriebenen Tasten haben die folgenden eine besondere Bedeutung:

- XC** verwenden Sie, um einzelne Zeichen zu löschen, auch in der Bedienzeile.
- CTRL R** Damit können Sie den Bildschirminhalt neu ausgeben lassen, wenn er durcheinandergeraten ist, z.B wenn Sie eine Meldung eines Hintergrund-Prozesses erhalten haben.
- CTRL X** Damit können Sie die Eingabe in der Bedienzeile ganz löschen.
- CTRL @** ist die "Notbremse" des ced: wenn er auf keine Eingabe mehr reagiert, können Sie damit den ced beenden. Sollte anschließend immer noch keine normale Eingabe möglich sein, siehe Abschnitt 5.23 unter "Es läuft nichts mehr".

Vordefinierte Standard-Funktionen

- START zurück an den Dateianfang

- F9 Datei sichern

- F10 Dokument wechseln: wechseln in eine zweite Datei, die Sie bereits mit Modus d (Datei wechseln) geöffnet haben und zurück.

- F11 Zeile markieren: markiert wird die Schreibmarkenzeile. Sie kann dann mit F12 kopiert werden.

- F12 Zeile kopieren: die Zeile, die Sie vorher mit F11 markiert haben, wird oberhalb der Schreibmarkenzeile kopiert.

- F13 suchen vorwärts: Sie müssen vorher im Modus s eine Zeichenfolge definiert haben, die gesucht werden soll.

- F14 suchen rückwärts, sonst wie F13.

- F15 eine Seite vorblättern

- F16 eine Seite zurückblättern

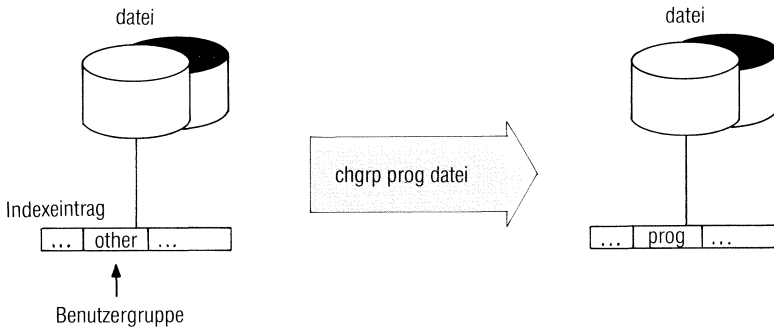
- F17 Zeile zurückholen: die Zeile oder der Zeilenbereich, der zuletzt gelöscht und zwischengespeichert wurde, wird zurückgeholt (abhängig davon, ob Modus z oder b eingestellt war).

Legen Sie den mitgelieferten beschrifteten Streifen auf Ihre Tastatur.

Eine Beispielsitzung mit dem ced finden Sie in Kapitel 4.

> > > > ed, sed

Gruppennummer für eine Datei ändern – change group



chgrp ändert die Gruppennummer für eine Datei. Es können dann Benutzer der neu angegebenen Gruppe auf die Datei zugreifen (Gruppen-Zugriffsrechte vorausgesetzt).

Nur der Systemverwalter kann die Gruppennummer ändern!

chgrp *gruppe* *datei*...

- gruppe** Gruppennummer der Benutzergruppe, die Zugriffsrecht haben soll. Sie können auch den Gruppennamen angeben, falls Sie Namen in der Datei /etc/group definiert haben (siehe Abschnitt 5.10).
- datei** Name der Datei, für die Sie die Gruppennummer ändern wollen.

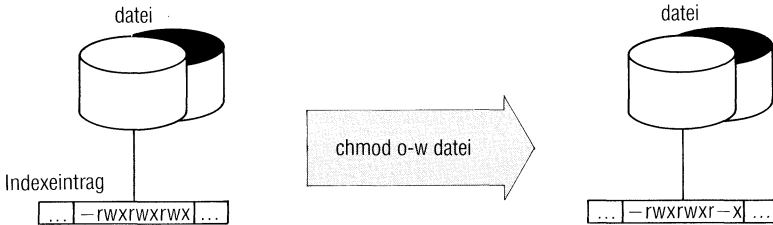
Beispiel

Die Dateien gold und silber sollen der Benutzergruppe mit der Gruppennummer 4 zugeordnet werden:

```
chgrp 4 gold silber
```

> > > > chown, chmod

Zugriffsrechte ändern – change mode



`chmod` ändert die Zugriffsrechte für Dateien und Dateiverzeichnisse (siehe auch Abschnitt 2.5). Zugriffsrechte ändern kann nur der Eigentümer der Datei oder der Systemverwalter.

`chmod` `modus` `datei`...

`modus` Symbolische oder absolute Angabe der Zugriffsrechte.

Symbolische Angabe:

[wer] darf was

`wer` Für wen gelten die angegebenen Rechte. Sie können eine Kombination der Buchstaben `u`, `g` und `o` angeben.

`u` Eigentümer

`g` Gruppe

`o` andere Benutzer

`a` steht für `ugo`, also alle Benutzer

Standard: abhängig von `umask`. Normalerweise alle Benutzer außer `o` mit `w`.

`darf` gibt an, ob die Rechte gegeben oder weggenommen werden sollen.

-
- + vergeben der angegebenen Rechte, alle anderen bleiben unverändert.
 - wegnehmen der angegebenen Rechte, alle anderen bleiben unverändert.
 - = vergeben der angegebenen Rechte, alle anderen werden weggenommen.
- was Art des Zugriffsrechts. Sie können eine Kombination der folgenden Buchstaben angeben:
- r Leseerlaubnis
 - w Schreiberlaubnis
 - x Ausführungserlaubnis, bei einem Dateiverzeichnis die Erlaubnis zum Durchsuchen.
 - s s-Bit setzen (siehe Abschnitt 2.5.2). s wirkt nur, wenn u oder g oder ug angegeben ist. Die Anzeige s beim ls-Kommando überschreibt die Anzeige x.
 - t Sticky-Bit setzen.
t kann nur der Systemverwalter angeben! ls zeigt t an letzter Stelle der Zugriffsrechte an. Die Anzeige t überschreibt die Anzeige x für "andere".

Absolute Angabe:

oktalzahl

Die Zugriffsrechte kann man folgendermaßen durch eine Oktalzahl darstellen:

Eigentümer:	100	ausführen	bzw.	durchlaufen
		(u = x)		
	200	schreiben (u = w)		
	400	lesen (u = r)		
Gruppe:	010	ausführen	bzw.	durchlaufen
		(g = x)		
	020	schreiben (g = w)		
	040	lesen (g = r)		
andere:	001	ausführen	bzw.	durchlaufen
		(o = x)		
	002	schreiben (o = w)		
	004	lesen (o = r)		
	1000	Sticky-Bit (t)		
	2000	s-Bit für Gruppe (g = s)		
	4000	s-Bit für Eigentümer (u = s)		

Diese Zahlenwerte können Sie beliebig addieren, z.B.:
chmod 644 datei setzt die Zugriffsrechte rw- r- - r-
chmod 700 datei setzt die Zugriffsrechte rwx - - - - -

name Datei oder Dateiverzeichnis, für das Sie die Zugriffsrechte ändern wollen.

Ende-Status:

- 0 bei normalem Ablauf
- 1 bei fehlerhaftem Ablauf
- 255 falsche Modus-Angabe

Beispiel

Die Datei motest hat folgende Zugriffsrechte:

```
$ ls -l motest
-rwxrwxr-x 2 blumann    32 May 25 09:40 motest
```

Für alle 3 Benutzergruppen wird die Schreiberlaubnis entzogen.

```
$ chmod -w motest
$ ls -l motest
-r-xr-xr-x 2 blumann    32 May 25 09:40 motest
```

Für "andere" werden alle Zugriffsrechte entzogen:

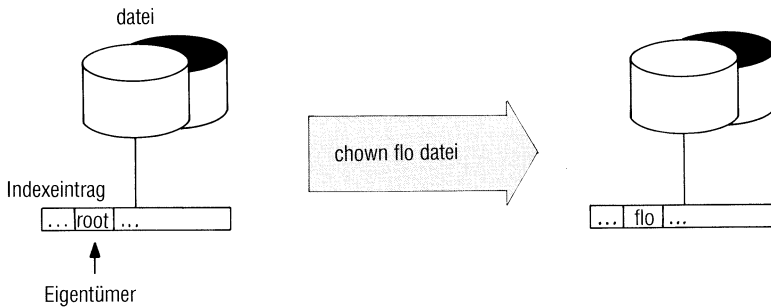
```
$ chmod o-rwx motest
$ ls -l motest
-rwxr-x--- 2 blumann    32 May 25 09:40 motest
```

Alle Benutzer sollen nur lesen dürfen:

```
$ chmod a=r motest
ls -l motest
-r--r--r-- 2 blumann    32 May 25 09:40 motest
```

> > > ls, chown

Eigentümer einer Datei ändern – change owner



chown ändert den Eigentümer einer Datei oder eines Dateiverzeichnisses auf eine andere Benutzerkennung. chown kann nur der Systemverwalter ausführen!

chown *benutzer* *name*...

benutzer Benutzerkennung des zukünftigen Eigentümers. Sie können auch stattdessen die Benutzernummer angeben.

name Name der Datei oder des Dateiverzeichnisses, dessen Eigentümer Sie ändern wollen.

Hinweis

chown verändert nur den Eigentümer, aber *nicht* die Gruppennummer für die Datei.

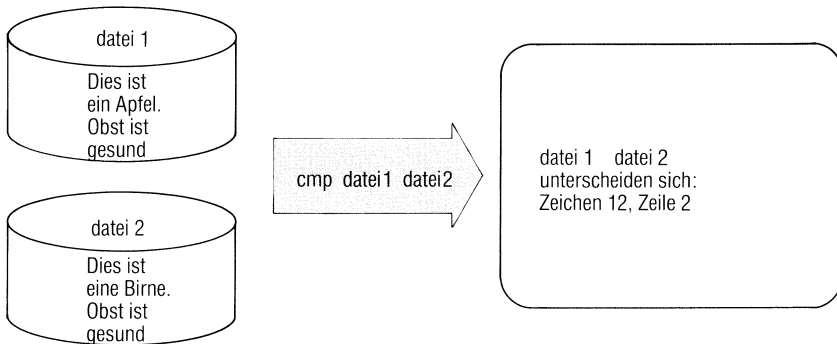
Beispiel

Sie haben als Systemverwalter das Dateiverzeichnis /usr/programme neu eingerichtet. Es soll dem Benutzer "jockl" gehören.

```
# chown jockl /usr/programme  
#
```

>>>> chgrp

Dateien zeichenweise vergleichen – compare two files



cmp vergleicht zwei Dateien Zeichen für Zeichen und gibt die Unterschiede aus.

cmp[₋schalter]₋datei1₋datei2

schalter

- l Auflisten aller Abweichungen in folgender Form:
 zeichennummer zeichen1 zeichen2
 zeichennummer ist die Position des Zeichens ab Dateianfang (dezimal).
 zeichen1 und zeichen2 sind die abweichenden Zeichen in datei1 und datei2. Sie sind oktal dargestellt.
 Eine ASCII-Tabelle finden Sie im Anhang.
- s cmp gibt nur den Ende-Status zurück, wie unten angegeben.

kein Schalter

Dateien sind identisch: cmp gibt nichts aus.

Dateien sind verschieden: cmp gibt Zeichennummer (n) und Zeilennummer (m) des ersten Unterschieds aus:

datei1 datei2 Unterschied: Zeichen n, Zeile m

Beide Zahlenangaben rechnen ab Dateianfang. Bei der Anzahl der Zeichen zählen die Zeichen für "neue Zeile" mit.

datei1 datei2

Namen der beiden Dateien, die verglichen werden sollen.
Wenn für den *ersten* Dateinamen das Zeichen "-" angegeben ist, liest cmp stattdessen von der Standard-Eingabe.

Hinweis

- Endet eine der beiden Dateien, ohne daß cmp einen Unterschied feststellen konnte, meldet cmp: In der Datei name wurde EOF erkannt.
- Ist in einer von zwei sonst gleichen Dateien ein Buchstabe ausgelassen, meldet cmp -l wegen der Verschiebung alle folgenden Zeichen als unterschiedlich.

Ende-Status:

- 0 bei identischen Dateien
- 1 bei unterschiedlichen Dateien
- 2 bei fehlerhaftem Ablauf: z.B. Syntaxfehler, Dateiname fehlt oder Datei kann nicht geöffnet werden.

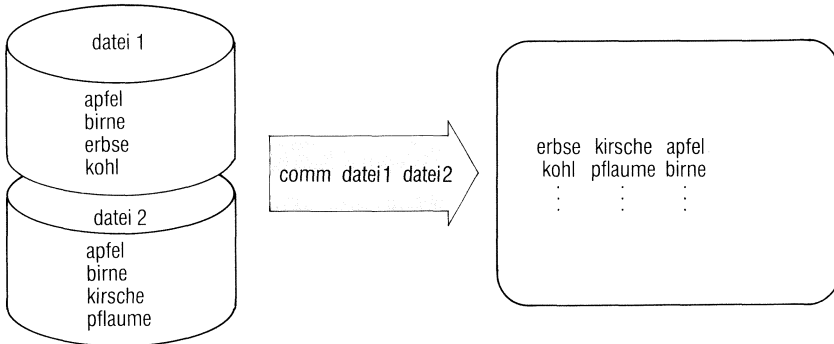
Beispiel

In einer Shell-Prozedur sollen zwei Dateien verglichen und bei Gleichheit eine davon gelöscht werden.

```
if cmp -s $1 $2
then
rm $2
fi
```

> > > > diff, comm

Sortierte Dateien vergleichen – select or reject lines common to two sorted files



comm vergleicht zwei Dateien, deren Zeilen gemäß ASCII sortiert sind.

comm[_schalter...][_datei1 _datei2

schalter

kein Schalter

comm liefert eine 3-spaltige Ausgabe mit folgender Bedeutung:

- Spalte1 : Zeilen, die nur in datei1 vorkommen.
- Spalte2 : Zeilen, die nur in datei2 vorkommen.
- Spalte3 : Zeilen, die in beiden Dateien vorkommen.

1 Unterdrückt die Ausgabe der Spalte 1.

2 Unterdrückt die Ausgabe der Spalte 2.

3 Unterdrückt die Ausgabe der Spalte 3.

Die Schalterkombination -12 gibt also die Zeilen aus, die beiden Dateien gemeinsam sind.

Die Schalterkombination -23 (bzw -13) gibt nur die Zeilen der datei1 (bzw datei2) aus.

Die Schalterkombination -123 erzeugt keine Ausgabe.

datei1 datei2

Namen der beiden sortierten Dateien, die verglichen werden sollen.

Wird für einen Dateinamen das Zeichen "-" angegeben, liest comm von der Standard-Eingabe.

Beispiel

In einer Datei katalog seien Buchtitel mit den zugehörigen Autoren erfaßt, und zwar so, daß in der zweiten Spalte einer Zeile jeweils der Autor aufgeführt ist. Sie wollen jetzt den Katalog nach mehreren Autoren durchsuchen. Diese Aufgabe kann mit comm gelöst werden, wenn die Autoren in sortierter Reihenfolge eingegeben werden.

1. Erstellen einer sortierten Autorenliste aus dem Katalog:

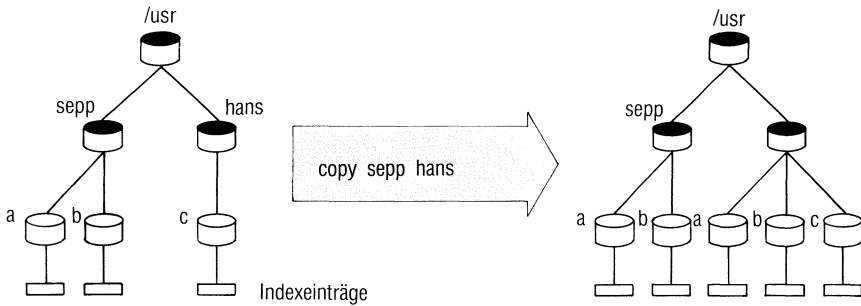
```
awk '{printf #"%s\n", $2}' katalog | sort > autorenliste
```

2. Vergleich der im Katalog erfaßten Autoren mit den gesuchten und Eingabe in sortierter Reihenfolge über Standard-Eingabe:

```
comm -12 autorenliste
```

>>>> cmp, diff, diff3, uniq

Dateien gruppenweise kopieren – copy groups of files.



Das copy-Kommando

- kopiert Inhalte von einem oder mehreren Dateiverzeichnissen in ein anderes Dateiverzeichnis oder
- kopiert Benutzer- und Gerätedateien.

copy[₋schalter...]₋quelle...₋ziel

schalter Mehrere Angaben zum Operanden schalter dürfen in beliebiger Reihenfolge stehen, müssen aber durch mindestens ein Leerzeichen getrennt sein.

kein Schalter

Ist die Quelle kein Dateiverzeichnis, entspricht copy dem Kommando cp.

Ansonsten wird jede Datei aus dem Quell-Dateiverzeichnis kopiert. Untergeordnete Dateiverzeichnisse werden nicht berücksichtigt.

Ein nicht vorhandenes Ziel (Datei oder Dateiverzeichnis) wird neu erstellt und erhält die gleichen Attribute wie die Quelle. Ein bereits vorhandenes Ziel wird beim Kopieren überschrieben.

- a Bevor das Kommando ausgeführt wird, muß der Benutzer dem Kopieren bei jeder einzelnen Datei zustimmen oder widersprechen.
 Beginnt die Antwort des Benutzers mit einem Buchstaben ungleich "j", wird das copy-Kommando nicht ausgeführt.

- l Bei Dateien werden lediglich Verweise gesetzt und keine Kopien erstellt.
 Bei Dateiverzeichnissen und Gerätedateien hat der Schalter keine Bedeutung.

- n Die Zieldatei muß eine neue Datei sein. Wenn die Zieldatei schon vorhanden ist, wird das copy-Kommando auf diese Datei nicht ausgeführt. Bei Gerätedateien ist dieser Schalter Voraussetzung. Ist die Quelle ein Dateiverzeichnis, ist er bedeutungslos; er betrifft aber die im Verzeichnis eingetragenen Dateien.

- o Nur der Systemverwalter darf diesen Schalter angeben. Wenn der Schalter gesetzt ist, werden Eigentümer und Gruppenzugehörigkeit der Quelldatei übernommen. Ist der Schalter nicht gesetzt, wird der Kommandoaufrufende zum Eigentümer der kopierten Datei.

- m
oder Wenn der Schalter gesetzt ist, wird für die kopierte Datei die letzte Änderungs- und Zugriffszeit der Quelldatei übernommen.

- O *Standard (keine Angabe):* Bei Quelle und Ziel wird der Zeitpunkt des Kopierens als letzte Dateiänderung eingetragen.

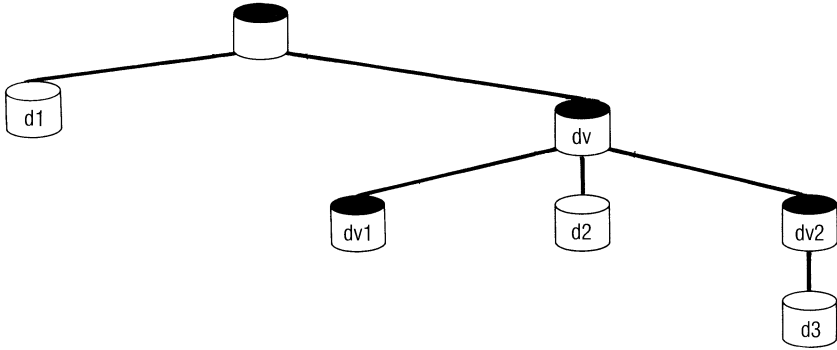
- r Jedes untergeordnete Dateiverzeichnis wird kopiert. Mit diesem Schalter können (rekursiv) ganze Dateisysteme kopiert werden.

- ad Ein untergeordnetes Dateiverzeichnis wird nur kopiert, falls nach Rückfrage der Benutzer eine Antwort eingibt, die mit "j" beginnt.

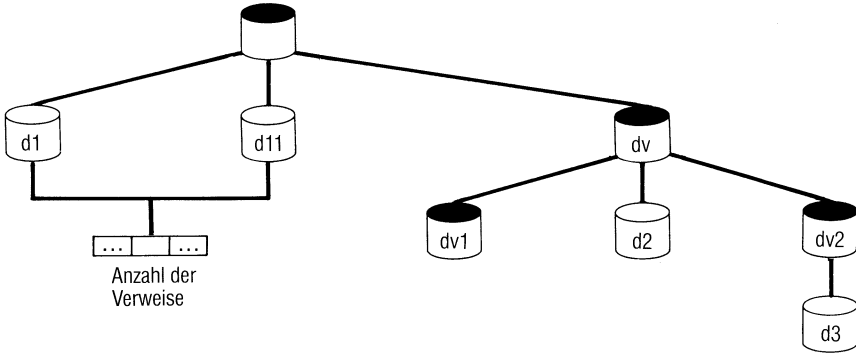
v	Protokollierung des Ablaufs von copy am Eingabegerät.
quelle	Erlaubt sind mehrere Angaben zum Operanden. Quelle kann entweder eine Datei, ein Dateiverzeichnis oder eine Gerätedatei sein. Die Quelle muß vorhanden sein.
ziel	Ziel ist entweder eine Datei oder ein Dateiverzeichnis. Ziel und Quelle dürfen nicht identisch sein. Sollten Sie aus Versehen bei Quelle und Ziel die gleichen Angaben machen, kommt keine Fehlermeldung!

Beispiel siehe Bild nächste Seite.

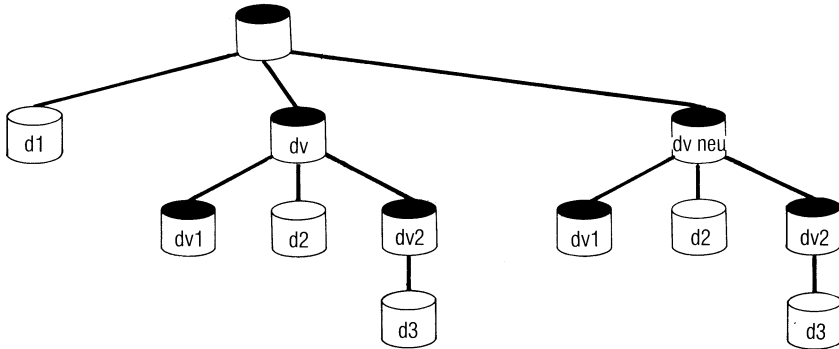
> > > > cp, ln



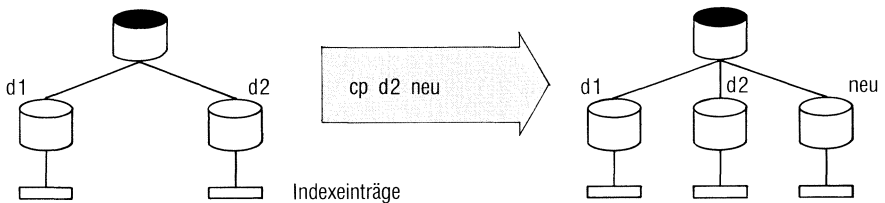
1) copy -l d1 d11



2) copy -r dv dvneu



Datei kopieren – copy files



cp hat zwei Funktionen. Das Kommando

- kopiert die Datei in eine Datei mit anderem Namen im selben Dateiverzeichnis (Format 1), oder
- kopiert eine oder mehrere Dateien in ein anderes Dateiverzeichnis (Format 2). Der alte Dateiname wird beibehalten.

Kopieren heißt: die Datei ist anschließend physikalisch nochmals vorhanden

Format 1

`cp original kopie`

original Dateiname des Originals.

kopie Dateiname der Kopie. Die Kopie muß anders heißen als das Original. Gibt es noch keine Datei mit diesem Namen, wird sie neu angelegt und erhält die Eigenschaften des Originals, wie z.B. Eigentümer und Zugriffsrechte. Die Zeit der letzten Änderung wird für die Kopie aktuell gesetzt. Gibt es im Dateiverzeichnis bereits eine Datei mit Namen kopie, wird sie überschrieben. Die Kopie erhält dann die Eigenschaften der überschriebenen Datei.

Format 2

`cp` `original[...]` `dateiverzeichnis`

`original` Dateiname des Originals.
Sie können mehrere Dateinamen angeben und so mehrere Dateien auf einmal kopieren. Die Kopien erhalten jeweils den Namen des Originals. Gibt es eine der Dateien bereits, wird sie überschrieben (siehe auch Format 1).

`dateiverzeichnis`
Dateiverzeichnis, in das die Kopien einzutragen sind.
Es darf nicht das Dateiverzeichnis sein, in dem die Originale stehen.

Beispiele

1. Die Datei `mist` soll im selben Dateiverzeichnis kopiert werden. Die Kopie soll `zweitmist` heißen:

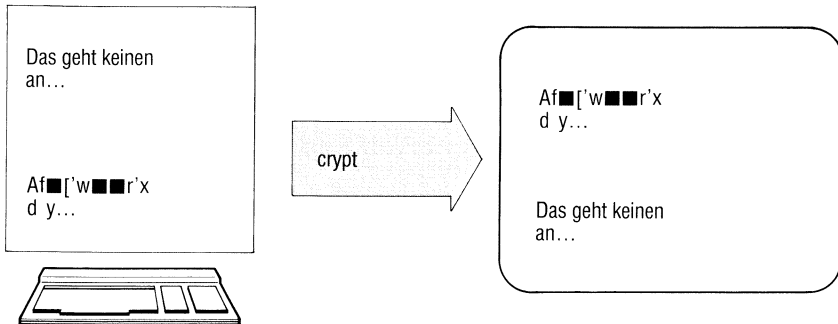
```
cp mist zweitmist
```

2. Alle Dateien, deren Namen mit `"dat"` beginnen, sollen ins Dateiverzeichnis `/usr/fl/sicher` kopiert werden:

```
$ cp dat* /usr/fl/sicher
$ ls -l /usr/fl/sicher
total 6
-rw-rw-r-- 1 blumann      52 May 25 09:58 datei1
-rw-rw-r-- 1 blumann      62 May 25 09:58 datei2
-rw-rw-r-- 1 blumann      61 May 25 09:59 datei3
-rw-rw-r-- 1 blumann      52 May 25 10:19 datum
$
```

>>>> copy, ln, mv

Dateien verschlüsseln



crypt verschlüsselt und entschlüsselt Text. Damit können Sie z.B. Dateien für andere unlesbar machen. crypt liest von der Standard-Eingabe und schreibt verschlüsselt auf die Standard-Ausgabe.

Verschlüsselten Text entschlüsselt script nur richtig, wenn Sie denselben Schlüssel angeben.

crypt[_schlüssel]

schlüssel 1 bis 8 Zeichen. Nur mit demselben Schlüssel können Sie verschlüsselten Text wieder lesen.

Standard (keine Angabe): crypt fordert Sie auf, einen Schlüssel einzugeben. Der eingegebene Schlüssel ist am Bildschirm nicht sichtbar. Sie schließen die Eingabe mit ab.

Hinweis

- Mit crypt verschlüsselte Dateien können Sie auch im Verschlüsselungsmodus des ed lesen und bearbeiten. ed arbeitet dabei auch mit crypt.
- crypt speichert den Schlüssel nicht ab. Wenn Sie den Schlüssel vergessen haben, können Sie verschlüsselte Dateien nicht mehr entschlüsseln.

Beispiele

1. `crypt mythos <klar > geheim`
schreibt den Text der Datei klar verschlüsselt in die Datei geheim.

```
crypt mythos < geheim | lpr  
druckt den Inhalt von geheim im Klartext am Drucker aus.
```

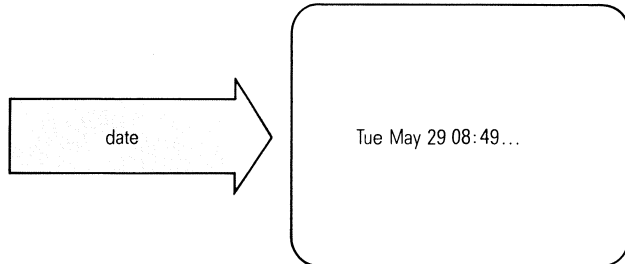
2. Wollen Sie den Schlüssel nicht sichtbar eingeben, schreiben Sie z.B.

```
$ crypt <geheim | lpr  
Schluessel eingeben:  
$
```

Nach der Aufforderung geben Sie den Schlüssel ein.

>>>> ed

Datum und Uhrzeit ausgeben, englische Schreibweise



date zeigt Ihnen das aktuelle Datum und die Uhrzeit in englischer Schreibweise.

Nur für den Systemverwalter:

Mit date können Sie die Systemuhr stellen (siehe Abschnitt 5.18).

date[_[jjmmtt]hhmm[.ss]]

kein Operand angegeben

date gibt das aktuelle Datum und die Uhrzeit aus.

date

Nur für den Systemverwalter

Mit den folgenden Angaben setzt `date` die Systemuhr. Sie sollten nicht ohne Grund die Systemuhr verstellen, da sonst Hardware- und Systemuhr verschieden laufen und erst wieder mit `/etc/mc` synchronisiert werden müssen.

`jjmmtt` Angabe des Datums, z.B. ist 840321 der 21. März 1984
Standard: Das Datum bleibt unverändert.

`hhmm` Angabe der Uhrzeit in Stunden und Minuten, z.B. 1655

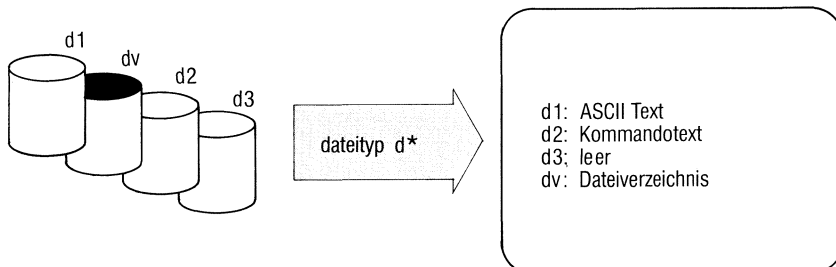
`.ss` gibt wahlweise die Sekunden an, Standard ist 00.

Beispiel

```
$ date
Thu Mar 22 07:59:32 MEZ 1984
$
```

>>>> datum

Art einer Datei bestimmen



dateityp unterscheidet Dateien nach ihrem Inhalt, z.B.

- Text-Dateien (ASCII-Zeichen),
- Shell-Prozeduren,
- C-Programme (Quellprogramme),
- ausführbare C-Programme.

dateityp arbeitet wie das Kommando "file", die Ausgabe ist jedoch deutsch.

dateityp[**-f**]**_**dateiname...

dateiname Name einer oder mehrerer Dateien, deren Art bestimmt wird.

f Bei dateiname ist eine Datei anzugeben, die eine Liste von Dateinamen enthält. Diese Dateien untersucht dateityp.

dateityp gibt aus: "dateiname: dateiart".

Ausführbare C-Programme klassifiziert dateityp weiter nach den Schaltern, die beim cc-Kommando gesetzt waren (siehe C-Entwicklungssystem):

dateityp

Klassifikation	Schalter
Programm und Daten getrennt mit Symboltabelle	i s

Hinweis

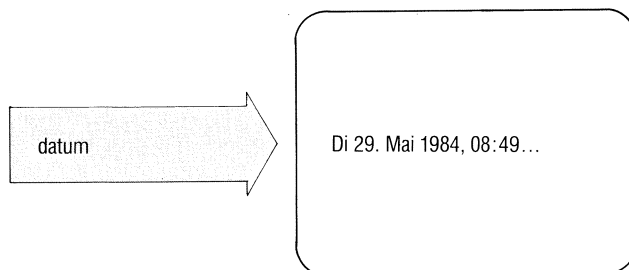
dateityp verwechselt leicht Shell-Prozeduren mit C-Programmen. Ebenso bezeichnet dateityp irrtümlich Programme, die mit Kommentaren beginnen, als Text.

Beispiel

```
$ file *
april: leer
blind: Kommandotext
cobol: Dateiverzeichnis
core: Daten
datei1: ASCII Text
fehler: ASCII Text
neu5: leer
prep.i: Text
$
```

>>>> C-Entwicklungssystem

Datum und Uhrzeit ausgeben, deutsche Schreibweise



datum zeigt Ihnen das aktuelle Datum und die Uhrzeit in deutscher Schreibweise.

Nur für den Systemverwalter:

Mit datum können Sie die Systemuhr stellen (siehe Abschnitt 5.18).

datum[_[jjmmtt]hhmm[.ss]]

kein Operand angeben

datum gibt das aktuelle Datum und die Uhrzeit aus.

datum

Nur für den Systemverwalter

Mit den folgenden Angaben setzt datum die Systemuhr. Sie sollten nicht ohne Grund die Systemuhr verstellen, da sonst Hardware- und Systemuhr verschieden laufen und erst wieder mit /etc/mc synchronisiert werden müssen.

Die folgenden Angaben sind wie beim Kommando date.

jjmmtt Angabe des Datums, z.B. ist 840321 der 21.März 1984

Standard: Das Datum bleibt unverändert.

hhmm Angabe der Uhrzeit in Stunden und Minuten, z.B. 1655

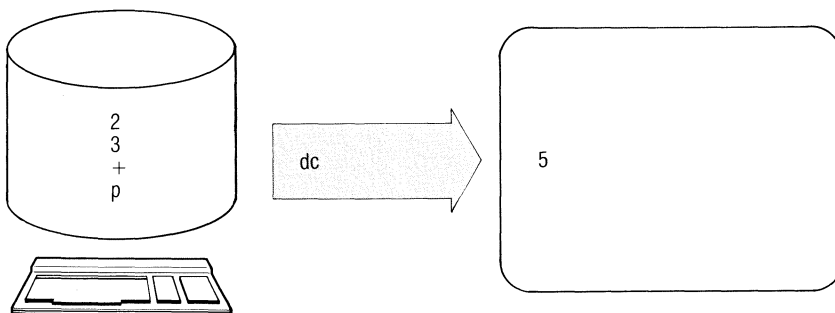
.ss gibt wahlweise die Sekunden an, Standard ist 00.

Beispiel

```
$ datum
Do 22.Mar.1984, 08:00:19 MEZ
$
```

>>>> date

Tischrechner – disc calculator



Mit `dc` können Sie im Dialog mit SINIX wie mit einem Tischrechner rechnen.

`dc[_datei]`

kein Operand

Die Eingabe der Rechenoperation wird von der Standard-Eingabe erwartet.

`datei`

Angabe der Datei, in der die Rechenoperationen stehen, die ausgeführt werden sollen.

Das `dc` Kommando aktiviert den Tischrechner. Anschließend geben Sie an der Datensichtstation oder von Datei mit den folgenden Symbolen Ihre Rechnungen ein:

`zahl`

`zahl` ist eine fortlaufende Folge der Ziffern 0 bis 9. Eine negative Zahl wird mit einem Unterstrich vor der Ziffernfolge gekennzeichnet. Die Ziffernfolge kann einen Dezimalpunkt enthalten. `zahl` wird in den Keller geladen.

+ - / * % Die zwei ersten Werte im Keller werden mit folgenden Rechenoperationen verknüpft:

+ Addition

- Subtraktion

***** Multiplikation

/ Division

% Modulofunktion

^ Exponentialrechnung

Dabei wird ggf. der gebrochene Teil eines Exponenten übergangen. Die Rechenoperanden werden im Keller gelöscht und an ihre Stelle wird das Rechenergebnis abgespeichert.

sx Der Anfang des Kellers wird in ein Register mit dem Namen *x* geladen und anschließend gelöscht. *x* kann ein beliebiges Zeichen sein. Wenn *s* groß geschrieben ist, wird *x* nicht als Register, sondern als Keller behandelt.

lx Der Inhalt des Registers *x* wird in den Keller geladen. Das Register *x* bleibt dabei unverändert. Der Anfangswert aller Register ist Null. Wenn *l* groß geschrieben ist, wird *x* als Keller behandelt und der erste Wert dem Hauptspeicher hinzugefügt.

d Der erste Wert im Keller wird kopiert.

p Der erste Wert im Keller wird ausgedruckt. Dabei bleibt der Wert im Keller unverändert erhalten. Wenn *p* groß geschrieben ist, wird der erste Wert im Keller als Zeichenfolge im ASCII-Code interpretiert und ausgedruckt. Dabei bleibt der Wert im Keller nicht erhalten.

f Der gesamte Inhalt des Kellers und der Register werden ausgedruckt.

-
- q** Beendet das Programm. Wird eine Folge ausgeführt, so wird die Rekursionsstufe um zwei verringert. Ist das q groß geschrieben, so wird der oberste Wert aus dem Keller entfernt und die Ausführungsstufe der Folge um diesen Wert verringert.
- x** Der erste Kellerwert wird als Buchstabenfolge interpretiert, die eine Folge von Anweisungen an das dc Kommando ist. Wenn x groß geschrieben ist, wird der erste Keller mit seinem "Skalenfaktor" ersetzt.
- [...] Der im ASCII-Code geschriebene Wert in eckigen Klammern wird an den Kelleranfang hinzugefügt.
- < x oder > x oder = x
Die beiden obersten Kellerelemente werden aus dem Keller entfernt und miteinander verglichen. Falls die angegebene Relation (<, >, =) erfüllt ist, wird Register x ausgeführt. Ein vorangestelltes "!" negiert die Relation (z.B. !< entspricht größer gleich).
- v** Aus dem ersten Kellerwert wird die Wurzel gezogen und der Wert wird mit dem Ergebnis der Wurzelrechnung überschrieben.
- !** Ab dem Ausrufezeichen wird der Rest der Zeile als SINIX-Kommando interpretiert.
- c** Der gesamte Kellerinhalt wird gelöscht.
- i** Der erste Kellerwert dient als Wurzelexponent für weitere Eingaben und wird anschließend gelöscht. Wenn i groß geschrieben ist, wird die Eingabegrundzahl an den Kelleranfang geladen.
- o** Der erste Kellerwert dient als Wurzelexponent für weitere Angaben und wird anschließend gelöscht. Wenn o groß geschrieben ist, wird die Ausgabegrundzahl an den Kelleranfang geladen.

- k das oberste Kellerelement wird entfernt, der Wert wird als nichtnegativer Skalierungsfaktor aufgefaßt. Die entsprechende Anzahl von Stellen wird bei der Ausgabe gedruckt und während der Multiplikation, Division und Exponentiation beibehalten.
- z Die Kellergröße wird in den Keller geschrieben. Ist Z groß geschrieben, so wird die oberste Zahl im Keller durch ihre Länge ersetzt.
- ? Eine Eingabezeile wird vom Eingabegerät, das normalerweise die Datensichtstation ist, geholt und ausgeführt.
- ; und : verwendet bc für Array-Operationen.

Der Keller ist ein Speicher, der nach dem Prinzip "last in - first out" arbeitet: Der zuletzt gespeicherte Wert steht ganz oben im Keller und kann als erster wieder verarbeitet werden.

Ein Register ist ein Arbeitsspeicher, der über einen Namen angesprochen wird (siehe Schalter s und l). Register verwendet man zum Zwischenspeichern von Werten und von Folgen von dc-Anweisungen

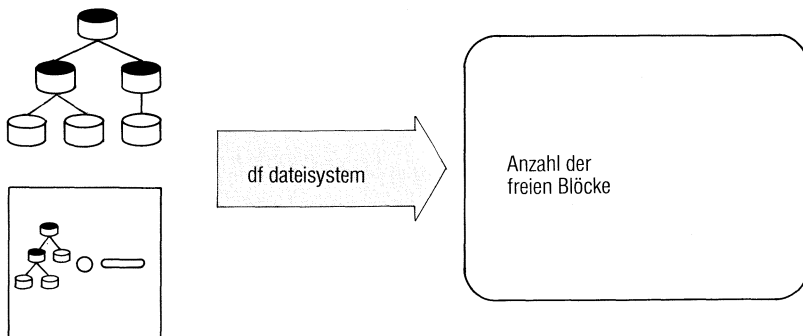
Beispiel

Folgendes Beispiel druckt die Zahlen 0-9

```
[lip1+ si li10>a] sa
0si lax
```

```
> > > > bc
```

Dateisystem auf freien Platz prüfen



df ermittelt den noch freien Speicherplatz in einem Dateisystem in KB (1KB = 1024 Byte). Die Ausgabe ist abgerundet auf volle KB.

df[`[_dateisystem...]`]

dateisystem

Name des Dateisystems, das Sie prüfen wollen, z.B. /dev/f12 für ein Dateisystem auf Diskette (siehe Abschnitt 5.12).

Standard (keine Angabe): df prüft die standardmäßig vorhandenen Dateisysteme /dev/root und /dev/rusr (siehe Abschnitt 5.7)

Beispiele

1. \$ df

```
/dev/rroot 1235
```

```
/dev/rusr 44256
```

```
$
```

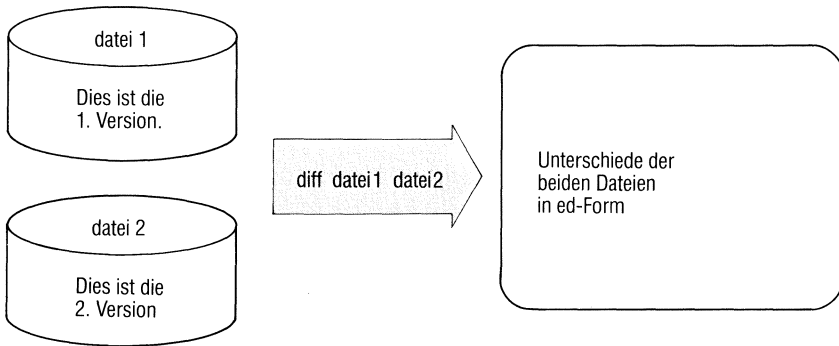
/dev/rroot hat 1235 freie Blöcke, /dev/rusr hat 44256 freie Blöcke.

2. Wieviel Speicherplatz ist noch auf dem Dateisystem der eingelegten Diskette?

```
df /dev/fl2
```

>>>> Abschnitt 5.7

Dateien zeilenweise vergleichen und ed-Skript erstellen differential file comparator



diff vergleicht zwei Dateien und gibt aus:

- die Zeilen, in denen sich die Dateien unterscheiden und
- ed-Kommandos, mit denen man aus datei2 datei1 erzeugen kann.

Mit diff können Sie also die Unterschiede zweier Dateiversionen festhalten.

diff[**-**schalter]**_**datei1**_**datei2

schalter

kein Schalter angegeben

diff gibt Zeilen folgenden Inhalts aus:

```
zeile1  a bereich2
bereich1 d zeile2
bereich1 c bereich2
```

Die linken Angaben bereich1 und zeile1 beziehen sich auf datei1, die rechten Angaben auf datei2 und sind Zeilenbereiche wie bei ed. a, d und c entsprechen ed-Kommandos:

a für anfügen (append)

- d für löschen (delete)
- c für ersetzen (change)

Auf diese Anweisungszeilen folgen Einfügezeilen:

- > bezeichnet Zeilen, die in datei1 einzufügen sind,
- < bezeichnet Zeilen, die in datei2 einzufügen sind.

Diese Angaben lesen Sie folgendermaßen:

Die Anweisungen a, d und c mit den davorstehenden Bereichsangaben zeigen, wie datei1 in datei2 umzuwandeln ist. Dazu gehören die mit ">" markierten Einfügezeilen.

Ersetzen Sie a durch d und d durch a und verwenden die rechts davon stehenden Bereichsangaben, dann zeigen die Anweisungen, wie man datei2 in datei1 überführen kann. Dazu gehören die mit "<" bezeichneten Einfügezeilen.

- e ed-Skript erzeugen. Das ed-Skript enthält a- c- und d-Kommandos für den Editor ed, sowie die zugehörigen Textzeilen. Mit dem Skript als Eingabe wandelt der Editor datei2 in datei1 um.
- f diff erzeugt eine Ausgabe ähnlich einem ed-Skript (nicht für ed zu verwenden). Daraus können Sie entnehmen, mit welchen Änderungen datei1 in datei2 geändert werden kann. Die Ausgabe hat die Form:

anweisung bereich

anschließend evtl. einzufügende Zeilen.

anweisung ist einer der folgenden Buchstaben (wie bei ed):

- a für anfügen
- d für löschen
- c für ändern

bereich ist ein Zeilenbereich, wie bei ed. Eine Zeile mit "." an erster Stelle schließt auf a oder d folgende Einfügezeilen ab (wie bei ed).

-
- b diff berücksichtigt weder Leerzeichen im Text, noch Leerzeichen oder Tabulatorzeichen am Zeilenende.
 - h diff erzeugt nur eine Zeile der Form: 1,\$c1,\$, gefolgt von den entsprechenden Zeilen aus datei1 und datei2. Dabei arbeitet diff schneller und Sie können beliebig lange Dateien bearbeiten. Allerdings sollten die Unterschiede in den Dateien gering und deutlich getrennt sein.

datei1 datei2

Dateien, die diff vergleichen soll. Steht für einen der Dateinamen das Zeichen '-', liest diff stattdessen von der Standard-Eingabe.

Weist einer der beiden Namen auf ein Dateiverzeichnis, sucht diff in diesem Dateiverzeichnis eine Datei mit dem gleichen Namen, wie die angegebene Datei.

Ende-Status:

- 0 Dateien sind gleich
- 1 Dateien sind verschieden
- 2 Eingabefehler

Beispiele

1. Die zwei Dateien datei1 und datei2 werden miteinander verglichen.

```
$ cat datei1
heute ist montag
es ist kalt
$ cat datei2
heute ist dienstag
es ist herbst
es ist kalt
$ diff datei1 datei2
1c 1,2
< heute ist montag
===
> heute ist dienstag
> es ist herbst
```

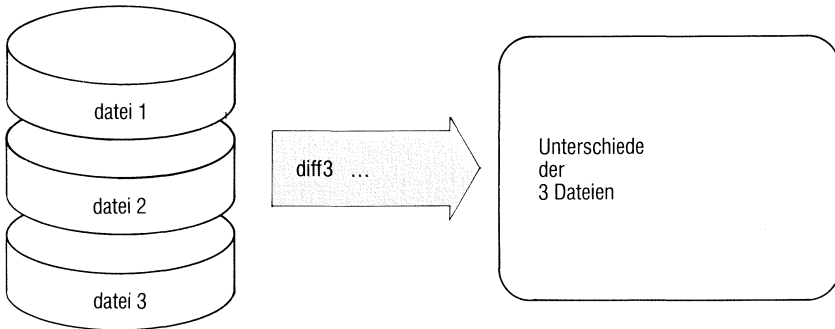
2. Dateien vergleichen und ein ed-Skript erstellen:

```
$ diff -e datei1 datei2
1c
heute ist dienstag
es ist herbst
.
$
```

Die ed-Kommandos, die diff ausgibt, würden datei2 in datei1 umwandeln.

```
>>>>  cmp, comm, diff3, ed
```

Drei Dateien zeilenweise vergleichen three-way differential file comparison



diff3 vergleicht drei Dateien und gibt nicht übereinstimmende Textbereiche aus.

diff3[_schalter]_datei1_datei2_datei3

schalter

kein Schalter

Für nicht übereinstimmende Textbereiche erzeugt diff3 eine Ausgabe der folgenden Art:

marke

eines der vier Zeichen mit der Bedeutung:

- === alle drei Dateien unterscheiden sich
- ==1 datei1 unterscheidet sich, datei2 und datei3 sind identisch
- ==2 datei2 unterscheidet sich, datei1 und datei3 sind identisch
- ==3 datei3 unterscheidet sich, datei1 und datei2 sind identisch.

i: n a

- i** Nummer der Datei entsprechend der Position in der Kommandozeile (1,2 oder 3).
 - n** Zeilennummer in Datei i,
 - a** in Datei i muß nach Zeile n Text angefügt werden.
- oder

i:n1,n2 c
[text]

- i** Nummer der Datei entsprechend der Position in der Kommandozeile (1,2 oder3).
 - n1,n2** Zeilenbereich in der Datei i, falls $n1 = n2$, wird $n2$ weggelassen.
 - c** in datei i muß im Bereich n1,n2 Text geändert werden.
- text** der zu ändernde Text aus datei i, falls der Inhalt von zwei Dateien identisch ist, wird die Textausgabe bei der niedriger indizierten unterdrückt.

.
. .
.

e In Form eines ed-Scriptes wird angezeigt, wie die Unterschiede zwischen datei2 und datei3 in datei1 eingebaut werden können. Eine Ausgabe erfolgt bei diesem Schalter also nur, falls datei3 von datei1 und datei2 verschieden ist. Die Ausgabe hat die Form:

bereich anweisung
eventuell einzufügende Zeilen

.

bereich Zeilenbereich (wie bei ed) in Datei1.
 anweisung einer der beiden Buchstaben (wie bei ed):

a anfügen
 c ändern

- x wie e, jedoch erfolgt eine Ausgabe nur, wenn alle drei Dateien verschieden sind.
- 3 wie e, jedoch erfolgt eine Ausgabe nur, wenn lediglich Datei3 verschieden ist, Datei1 und Datei2 aber identisch sind.

datei1 datei2 datei3

Namen der Dateien, die das Kommando vergleicht.

Beispiele

Sie haben drei Versionen t1, t2, t3 einer Adressendatei mit folgendem Inhalt:

t1:	t2:	t3:
Cooly Carsten	Cooly Carsten	Cooly Carsten
Kuhdamm 14	Kuhdamm 14	Kuhdamm 14
1 Berlin 20	1 Berlin 20	1 Berlin 20
030-123456	030- 987654	030-987654
Freaky Hans	Freaky Hans	Freaky Hans
Hüterweg 3	Hüterweg 3	Hüterweg 3
8906 Hintertupf	8906 Hintertupf	8906 Hinter-
tupf		
09076-345	09076-345	09076-345
		Motz Peter
		Leopoldstr. 6
		8 München 40
		089-777888

- 1) Die Konsistenz der drei Versionen läßt sich überprüfen mit:

```
diff3 t1 t2 t3
```

diff3 liefert die Ausgabe:

```
===1
```

```
1: 4c
```

```
030-123456
```

```
2: 4c
```

```
3: 4c
```

```
030-987654
```

```
===3
```

```
1: 9a
```

```
2: 9a
```

```
3: 10,14
```

```
    Motz Peter  
    Leopoldstr. 6  
    8 München 40  
    089-777888
```

- 2) Wollen Sie jetzt feststellen, wie t1 aktualisiert werden kann, dann rufen Sie auf:

```
diff3 -e t1 t2 t3
```

diff3 liefert die Ausgabe:

```
9a
```

```
Motz Peter
```

```
Leopoldstr. 6
```

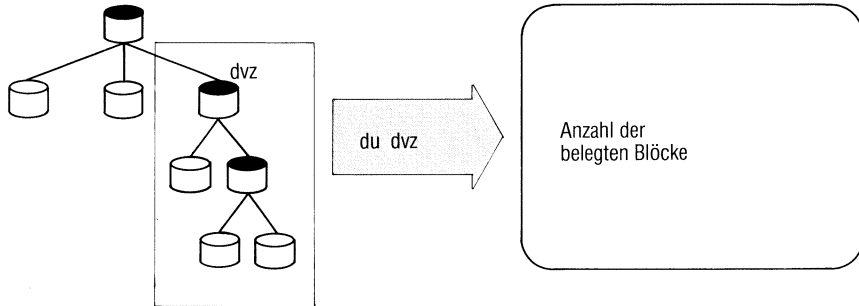
```
8 München 40
```

```
089-777888
```

Beachten Sie bitte, daß die geänderte Telefonnummer von Cooly Carsten nicht ausgegeben wird, da sie in Version t2 und t3 übereinstimmt und der -e Schalter nur Unterschiede zwischen Datei2 und Datei3 berücksichtigt.

```
>>>> comm, cmp, diff
```

Belegten Speicherplatz ausgeben – display used blocks



du zeigt den von Dateien belegten Speicherplatz in KB (1KB = 1024 Byte). Sie können den belegten Speicherplatz ausgeben lassen für die Dateien eines Teilbaumes, d.h. du untersucht ein angegebenes Dateiverzeichnis und alle seine Unter-Dateiverzeichnisse.

du[**-**schalter][**-**name...]

schalter

kein Schalter

du listet den belegten Speicherplatz auf für das angegebene Dateiverzeichnis und alle Unter-Dateiverzeichnisse. Die Anzahl gibt an, wieviel Platz jeweils der Teilbaum belegt, der beim aufgezählten Dateiverzeichnis beginnt.

s du gibt den belegten Speicherplatz des Teilbaumes oder der Datei aus.

a du gibt den belegten Speicherplatz für alle angegebenen Dateien bzw. Dateien des Teilbaumes aus.

name Name einer Datei oder eines Dateiverzeichnisses. Sie können auch mehrere Namen gleichzeitig angeben. Wenn Sie hier einen oder mehrere Dateinamen angeben, müssen Sie gleichzeitig Schalter **a** oder Schalter **s** gesetzt haben. Sonst berücksichtigt du nur Dateiverzeichnisse.

Standard (keine Angabe): das aktuelle Dateiverzeichnis.

Hinweis

Eine Datei, die zwei Verweise auf sich hat, wird nur einmal gezählt. Wenn es zu viele unterschiedliche Dateien mit Verweisen gibt, zählt du diese möglicherweise mehrfach. Die Ausgabe ist auf volle KB abgerundet.

Ende-Status: immer 0

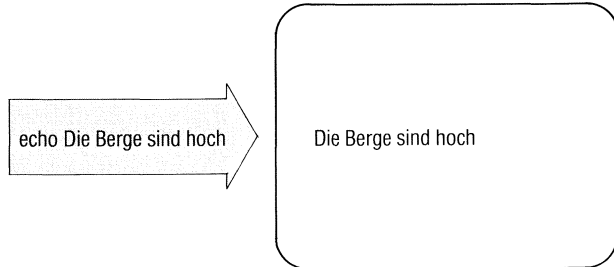
Beispiel

Sie möchten eine Übersicht über alle Dateien in Ihrem aktuellen Dateiverzeichnis haben:

```
$ du -a | sort +1
390  .
1    ./abc
1    ./bild
77   ./cobol
14   ./cobol/demo.cbl
11   ./cobol/ems.cbl
.
.
$
```

> > > > df, quot

Zeichenfolgen ausgeben – echo arguments



echo bildet aus der Angabe argument in der Kommandozeile eine Zeichenfolge und gibt sie aus. Damit können Sie z.B.

- Meldungen in Shell-Prozeduren erzeugen,
- den Inhalt von Variablen abfragen,
- Ausgaben von Kommandos mit Konstanten ergänzen, z.B. in Pipelines,
- Text in Dateien schreiben usw.

echo[_-n][_argument...]

n Die Ausgabe wird nicht mit `\n` abgeschlossen. Bei Shell-Prozeduren können Sie so mehrere Ausgaben in eine Zeile schreiben.

Standard (n nicht angegeben): echo schließt die Ausgabe mit "neue Zeile" ab.

argument Ausdruck, den die Shell als Zeichenfolge interpretieren kann, das ist:

text beliebiger Text, z.B. echo hier bin ich.
Jedes Wort von "hier bin ich" ist *ein* Argument.

'text' Text, der Sonderzeichen enthalten darf, z.B. echo 'gib "\$" aus!'. Der Text in Hochkommas ist *ein* Argument.

"text" oder `text`

Die Shell interpretiert die Zeichenfolge text, wie in Abschnitt 3.4 beschrieben, z.B. ergibt echo `who` gerade die Ausgabe des Kommandos "who".

abgekürzte Dateinamen

wie in Abschnitt 2.2.3 beschrieben, z.B.

echo *	gibt alle Dateinamen des aktuellen Dateiverzeichnisses aus,
echo ?a*	gibt alle Dateinamen des aktuellen Dateiverzeichnisses aus, deren zweiter Buchstabe ein a ist.

Stellungs- und Kennwortparameter

echo gibt deren Inhalt aus, z.B.:

echo \$1 \$2	liefert den Inhalt der ersten beiden Stellungsparameter einer Shell-Prozedur,
echo \$var	liefert den Inhalt des Kennwortparameters "var".

Hinweis

Statt echo können Sie auch das Kommando display verwenden. display ist ein "eingebautes Kommando".

Solche Kommandos führt die Shell direkt aus, sie sind schneller und es ist keine Ein-Ausgabeumlenkung möglich (siehe auch: Kommandos der Shell, Kapitel 3.8).

Beispiele

1. In einer Shell-Prozedur wollen Sie eine Fehlermeldung ausgeben auf die Standard-Fehlerausgabe:

```
.  
.  
echo "Die Datei $1 ist nicht vorhanden" > &2  
.  
.
```

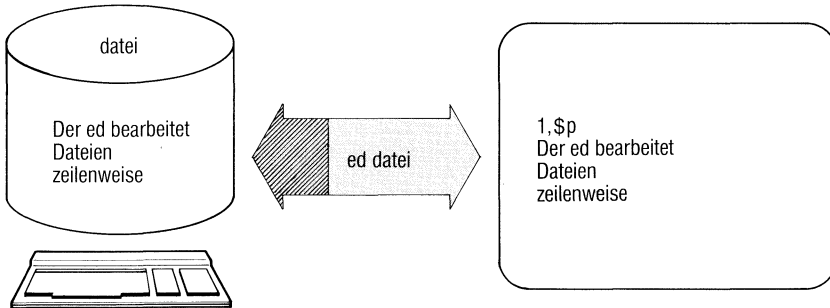
2. Welchen Inhalt hat die Variable \$HOME?

```
$ echo $HOME  
/usr/art  
$
```

3. Vor die Ausgabe des Kommandos "datum" wird eine Zeichenfolge gestellt.

```
$ (echo -n 'Heute ist: ';datum) | cat > d1  
$ cat d1  
Heute ist: Mi 2.Mai.1984, 16:07:53 MEZ  
$
```

Zeilenorientierter Editor im Dialogbetrieb



ed ist ein interaktiver zeilenorientierter Editor. Er eignet sich besonders für kurze Änderungen an Dateien und zur Bearbeitung von Dateien, die nicht druckbare Zeichen enthalten, und deshalb nicht vom bildschirmorientierten Editor ced bearbeitet werden können. Der ed ist ein sehr effizienter Editor; aber seine Benutzerschnittstelle ist für den ungeübten Benutzer kompliziert. Die Benutzung der Schreibmarke ist nicht möglich.

ed[**-**][**-x**][**-**datei]

- unterdrückt die Ausgabe der Anzahl der gelesenen bzw. geschriebenen Bytes beim e(dit), r(ead) und w(rite) Kommando.
- x bewirkt das Umschalten von ed in den Ver—, bzw. Entschlüsselungsmodus. ed fordert die Eingabe eines Schlüssels an, der zur Ver-, bzw. Entschlüsselung der beim e(dit), r(ead) und w(rite) Kommando angegebenen Dateien benutzt wird. Ist eine Datei angegeben, wird ihr Inhalt gelesen und entschlüsselt, falls er verschlüsselt ist.
- datei Geben Sie eine Datei an, führt ed als erstes Kommando ein e(dit) aus, um die Datei in den internen Puffer einzulesen. Sonst arbeiten Sie anfangs auf einem leeren Puffer und bestimmen erst beim Wegschreiben des Pufferinhalts auf eine Datei deren Namen.

Allgemeines

ed arbeitet grundsätzlich auf einer Kopie der angegebenen Datei. Erst wenn Sie ein write-Kommando eingeben, wird der alte Inhalt dieser Datei überschrieben.

Verlassen Sie den Editor, müssen Sie mit write-Kommando die Kopie in die Datei zurückschreiben. Ansonsten sind Ihre Änderungen verloren. Die Kommandos `END`, `q`, `Q` und `e`, mit denen Sie den Inhalt des Puffers löschen können, geben ein `?` aus, wenn der geänderte Inhalt des Puffers vor dessen Löschung nicht gerettet wurde. ed arbeitet in zwei Modi, dem Eingabe- und dem Kommandomodus. Im Eingabemodus, den Sie durch die Befehle `append`, `change` und `insert` einschalten, werden alle folgenden Eingabezeichen, auch verschiedene nicht druckbare Zeichen in den Puffer geschrieben.

Sie verlassen den Eingabemodus durch Drücken der Taste `DEL` oder durch Eingabe eines `.` in der ersten Spalte. Alle Kommandos müssen durch Drücken der Taste `↓` abgeschlossen werden.

I.a. darf nur ein Kommando auf einer Zeile stehen. Für ed existiert zu jedem Zeitpunkt eine aktuelle Zeile. Geben Sie bei einem Kommando keine Adresse an, beziehen sich die Kommandos fast immer auf diese Zeile. Die aktuelle Zeile wird (explizit) durch `'` bezeichnet.

Beschränkungen

Folgende Beschränkungen gelten für die Arbeit mit ed:

- eine Zeile darf höchstens 512 Zeichen lang sein;
- eine globale Kommandoliste darf höchstens 256 Zeichen enthalten;
- ein Dateiname darf höchstens 64 Zeichen lang sein;
- die Datei darf nicht größer als 128 K sein.

Kommandostruktur

ed Kommandos haben i.a. eine sehr einheitliche Struktur: keine, eine oder zwei Adressen gefolgt von einem Befehlsbuchstaben und eventuellen Parametern. Durch die Adressen werden Zeilen im Puffer bezeichnet, auf die das Kommando angewendet wird. Adressenangaben werden wie folgt interpretiert:

- Haben Sie keine Adresse angegeben, nimmt ed die zu jedem Kommando gehörige Standardadresse an; diese ist bei allen Kommandos angegeben.
- Haben Sie eine Adresse angegeben, gilt die durch diese Adresse bezeichnete Zeile als ausgewählt.
- Zwei Adressen kennzeichnen den Bereich zwischen den angegebenen Intervallgrenzen (einschließlich). Ist die zweite Adresse kleiner als die erste, meldet ed einen Fehler. Der Bereich, der sich nur auf die aktuelle Zeile bezieht, wird mit „.,“ bezeichnet.
- Benötigt ed keine Adresse und Sie haben eine angegeben, meldet ed einen Fehler.
- Haben Sie mehr Adressen angegeben als nötig sind, nimmt ed die letzten.

Adressen werden normalerweise durch „,” voneinander getrennt. Sie können auch durch ein „;” getrennt werden. Das hat zur Folge, daß die aktuelle Zeile auf die erste angegebene Adresse gesetzt wird, während bei der „;“-Notation die aktuelle Zeile erst bei der Ausführung von Kommandos verändert wird. Das spielt z. B. eine Rolle, wenn Sie einen Bereich durch die Angabe zweier regulärer Ausdrücke kennzeichnen wollen. Trennen Sie die beiden Ausdrücke durch ein „,”, beginnt die Suche nach beiden Ausdrücken auf der aktuellen Zeile, haben Sie sie durch „;” getrennt, wird die aktuelle Zeile auf die Zeile gesetzt, die eine Zeichenfolge enthielt, zu der der reguläre Ausdruck paßte, bevor nach dem zweiten Ausdruck gesucht wird.

Adressen

Adressen werden wie folgt konstruiert:

1. Das Zeichen "." adressiert die aktuelle Zeile.
2. Das Zeichen "\$" adressiert die letzte Zeile des Puffers.
3. Die Dezimalzahl n adressiert die n -te Zeile im Puffer.
4. 'x (Hochkomma x) adressiert die mit dem Buchstaben x markierte Zeile (s. Kommando k).
5. Ein regulärer Ausdruck in '/' eingeschlossen, adressiert die erste Zeile, die, beginnend mit der aktuellen Zeile, eine Zeichenfolge enthält, die zu dem regulären Ausdruck paßt. Falls notwendig, springt ed vom Ende des Puffers an seinen Anfang, um die Suche fortzusetzen.
6. Ein regulärer Ausdruck in '?' eingeschlossen adressiert die erste Zeile, die, beginnend mit der aktuellen Zeile rückwärts suchend, eine Zeichenfolge enthält, die zu dem regulären Ausdruck paßt. Falls notwendig, springt ed vom Anfang des Puffers an sein Ende, um die Suche fortzusetzen.
7. Eine Adresse gefolgt von einem '+' oder '-' gefolgt von einer Dezimalzahl n , adressiert die Zeile, die n Zeilen hinter bzw. vor der durch die angegebene Adresse bezeichneten Zeile liegt.
8. Beginnt eine Adresse mit einem '+' oder einem '-', so gilt 7. bzgl. der aktuellen Zeile.
9. Endet eine Adresse mit einem '+' oder einem '-', so wird als folgende Dezimalzahl eine 1 angenommen. Die Adresse, die aus einem einfachen '-' besteht, adressiert die Zeile vor der aktuellen Zeile. '+' oder '-'-Zeichen am Ende einer Adresse haben kumulativen Effekt; die Adresse '+ +' adressiert also die zweite auf die aktuelle Zeile folgende Zeile.

Reguläre Ausdrücke

Ein regulärer Ausdruck *r* bezeichnet eine Menge von Zeichenfolgen, die *passenden* Zeichenfolgen (oder *Interpretationen*) von *r*. Z.B. bezeichnet der reguläre Ausdruck `#[abc]#` die drei Zeichenfolgen `#a#`, `#b#` und `#c#`.

ed unterstützt den Gebrauch (eingeschränkter) regulärer Ausdrücke zum Auffinden von Zeilen in einer Datei (s.o. Adressen) sowie in Befehlen für string-Operationen (s. `substitute`-Kommando). Ein regulärer Ausdruck wird immer in `'/'` eingeschlossen.

Für die Suche nach einer passenden Zeichenfolge für einen regulären Ausdruck gelten folgende Regeln:

- Die Zeilen der Datei werden sequentiell von links nach rechts abgesehen.
- Es wird immer die längstmögliche passende Zeichenfolge ausgewählt.
- Für einen regulären Ausdruck der Form `xy` (siehe Punkt 6) wird die längstmögliche Interpretation von `x` gefolgt von einer Interpretation von `y` ausgesucht.
- Der zuletzt in `/r/` angegebene reguläre Ausdruck *r* wird gespeichert und kann mit `//` angesprochen werden.
- Ist die Suche erfolglos, meldet sich ed mit `?` zurück.

Reguläre Ausdrücke

	Syntax	Interpretation	Beispiel
	Ein regulärer Ausdruck ist:	der linksstehende reguläre Ausdruck bezeichnet:	regulärer Ausdruck: passende Zeichenfolge
1	z z jedes Zeichen außer: \, [, -, ., *, \$ /,], ^	das entsprechende Zeichen	a a
2	\z z jedes Zeichen außer: (,), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0	das entsprechende Zeichen (Entwertung von Sonderzeichen)	\a a * *
3	.	ein beliebiges Zeichen (Gesamtzeichenvorrat)	. jedes Zeichen nach ASCII
4a	[s] s Zeichenfolge, in der "]" höchstens als erstes Zeichen	ein Zeichen, das in s vorkommt (Teilmenge)	[af] a oder f [a-c] a, b oder c
4b	[^s] vorkommt oder s Intervall der Form a-b, wobei a < b nach der ASCII-Tabelle	ein Zeichen, das nicht in s vorkommt (Komplement) Hinweis: in s hat \ keine besondere Bedeutung	[^\.&] jedes Zeichen außer \, ., & [^a-z] alle Zeichen außer Kleinbuchstaben
5	r* r regulärer Ausdruck der Form 1-4	eine Folge von null oder mehr passenden Zeichenreihen für r	m* m oder mm, mmm, mmmm, ... usw
6	xy x, y reguläre Ausdrücke	eine passende Zeichenfolge für x gefolgt von einer passenden Zeichenreihe für y (Konkatenation)	.k bel. Zeichenk [afg]z z.B. ok az oder fz oder gz
7a	^r r reg. Ausdruck der Form 1-8	eine Interpretation von r die am Anfang der Zeile vorkommt	^Von Von am Anfang einer Zeile
7b	r\$	eine Interpretation von r die am Zeilenende vorkommt	Rand\$ Rand am Ende einer Zeile

	Syntax	Interpretation	Beispiel
	Ein regulärer Ausdruck ist:	der linksstehende reguläre Ausdruck bezeichnet:	regulärer Ausdruck: passende Zeichenfolge
8a	$\backslash(r\backslash)$ r reg. Ausdruck, in dem höchstens vier " \backslash " vorkommen Bemerkung: die öffnenden \backslash (in einem reg.Ausdruck werden mit 1 beginnend von links nach rechts durchnummeriert	dieselben Zeichenfolgen wie r (Markierung von r)	$\backslash(\text{haus}\backslash)$ haus $\backslash(a[12]\backslash)$ a1 oder a2
8b	$x\backslash n$ $1 \leq n \leq 5$, x reg. Ausdruck, in dem ein mit dem n -ten \backslash (\backslash) Paar geklammerter Teilausdruck r vorkommt	dieselben Zeichenfolgen wie r (Wiederholung eines markierten Ausdruckes)	$\backslash(\text{haus}\backslash(\text{tür}\backslash)\backslash)\backslash 2$ tür

Kommandos

Die folgende Liste enthält in alphabetischer Reihenfolge die Kommandos, die Sie im Kommandomodus eingeben können. Die meisten Kommandos können durch ein angehängtes p- oder l-Kommando dazu veranlasst werden, die aktuelle Zeile - nach Ausführung des Kommandos - auszugeben.

[zeile] a anfügen - append

Ihre Eingabe

.
.
.
.

Standard für zeile: .

Das append Kommando liest den eingegebenen Text und fügt ihn hinter der adressierten Zeile an. Die Adresse 0 ist bei diesem Kommando erlaubt; der Text wird dann vor die erste Zeile des Puffers eingefügt. Die aktuelle Zeile ist nun die zuletzt eingegebene, bzw. falls keine Eingabe erfolgte, die adressierte Zeile.

[bereich] c ändern - change

Ihre Eingabe

.
.
.
.

Standard für bereich: .,

Das change Kommando löscht den angegebenen Bereich und ersetzt ihn durch die eingegebenen Zeilen. Die aktuelle Zeile ist die zuletzt eingegebene, bzw. falls keine Eingabe erfolgte, die Zeile vor den gelöschten Zeilen.

[bereich] d löschen - delete

Standard für bereich: ..

Das delete Kommando löscht den angegebenen Bereich. Die Zeile hinter der letzten gelöschten Zeile wird zur aktuellen Zeile. Ständen die gelöschten Zeilen am Ende des Puffers, wird die neue letzte Zeile die aktuelle.

e [datei] einlesen - edit

Standard für datei: der aktuell gespeicherte Dateiname

Das edit Kommando löscht den gesamten Puffer und liest den Inhalt der Datei ein. Ist der alte Pufferinhalt verändert und nicht gerettet worden, meldet ed ein '?', ohne den Puffer gelöscht zu haben. Geben Sie daraufhin ein weiteres edit Kommando ein, wird es ohne diese Meldung ausgeführt. Die Anzahl der eingelesenen Bytes wird ausgegeben, wenn Sie nicht ed mit dem Schalter "-" aufgerufen haben. Die aktuelle Zeile ist die letzte Zeile des Puffers. Der angegebene Dateiname wird für eventuell folgende edit-, filename-, read- oder write- Kommandos gespeichert.

E [datei] einlesen - Edit

Standard für datei: der aktuell gespeicherte Dateiname

Das Edit Kommando verhält sich wie das edit Kommando, außer daß es, wenn der Inhalt des Puffers verändert und nicht gerettet wurde, ihn ohne Warnung löscht.

f [datei] Dateiname - filename

Standard für datei: der aktuell gespeicherte Dateiname

Ein neuer Dateiname überschreibt den bisher gespeicherten.

[bereich]g/regulärer Ausdruck/kommandoliste

global

Standard für bereich: 1,\$

Das global Kommando markiert im ersten Schritt alle Zeilen, die eine Zeichenfolge enthalten, die zu dem regulären Ausdruck paßt. Dann wird für jede markierte Zeile die Kommandoliste ausgeführt, wobei die aktuelle Zeile jeweils auf die nächste markierte Zeile gesetzt wird. Ein einzelnes Kommando oder das erste einer Liste steht auf derselben Zeile wie das global Kommando. Alle Zeilen einer Kommandoliste außer der letzten müssen mit einem "\ " enden. Die Kommandos append, insert und change mit dazugehöriger Eingabe sind zugelassen. Der Punkt ".", um die Eingabe abzuschließen, kann auf der letzten Zeile der Kommandoliste weggelassen werden. Die Kommandos global, global but (v) und "!" sind in der Kommandoliste nicht zugelassen.

[zeile]i

einfügen - insert

Ihre Eingabe

.
.
.
.

Standard für zeile: .

Das insert Kommando fügt den eingegebenen Text vor der adressierten Zeile ein. Die aktuelle Zeile ist die letzte eingegebene; wurde keine Eingabe gemacht, wird die Zeile vor der adressierten zur aktuellen Zeile. Dieses Kommando unterscheidet sich von dem append-Kommando nur durch die Positionierung des eingegebenen Texts.

[bereich + l]j verbinden - join

Standard für bereich: ..

Das join Kommando verbindet alle im angegebenen Bereich liegenden Zeilen zu einer Zeile, wenn die neue Zeile nicht länger als 512 Zeichen wird. In diesem Bereich liegende Leerzeilen werden gelöscht. Haben Sie nur eine Adresse angegeben, geschieht nichts. Die neue Zeile wird zur aktuellen Zeile.

[zeile] kx markieren - mark

Standard für zeile: .

Das mark Kommando markiert die angegebene Zeile mit dem Buchstaben x. x muß ein Kleinbuchstabe sein. Adressiert wird die markierte Zeile durch 'x (Hochkomma x).

[bereich] l ausgeben - list

Standard für bereich: ..

Das list-Kommando gibt im Gegensatz zum print-Kommando die angegebenen Zeilen wie folgt aus: Nicht druckbare Zeichen werden als zweistellige Oktalzahlen dargestellt; überlange Zeilen werden mehrzeilig ausgegeben, am Ende mit dem Zeilenfortsetzungszeichen "␣" versehen. Das list Kommando darf nach jedem Kommando stehen, das kein Ein-/Ausgabekommando ist.

[bereich] m *adresse* verschieben - move

Standard für bereich: ..

Das move Kommando verschiebt die im angegebenen Bereich liegenden Zeilen hinter die durch *adresse* adressierte Zeile. Die letzte der verschobenen Zeilen wird die aktuelle Zeile.

[bereich] p ausgeben - print

Standard für bereich: ..

Das print Kommando gibt die angegebenen Zeilen aus. Nicht druckbare Zeichen werden nicht dargestellt. Überlange Zeilen werden nicht besonders dargestellt, d.h. man kann sie nicht als solche erkennen. Die aktuelle Zeile ist die zuletzt ausgegebene Zeile. Das print Kommando darf nach jedem Kommando, das nicht E/A-Kommando ist, angegeben werden.

[bereich] P ausgeben - Print

Dieses Kommando ist ein Synonym für print.

q verlassen - quit

Mit dem quit Kommando wird ed beendet. Falls der Inhalt des Puffers nach der letzten Veränderung nicht gerettet wurde, gibt ed ein ? aus und wartet auf weitere Eingabe. Mit einem weiteren quit-Kommando verlassen Sie den ed, ohne den Puffer gerettet zu haben.

Q verlassen - Quit

Das Quit Kommando beendet den ed sofort ohne Meldung, auch wenn der veränderte Inhalt des Puffers nicht gerettet wurde.

[zeile] r [datei] lesen - read

Standard für zeile: \$

Standard für datei: der aktuell gespeicherte Dateiname

Das read Kommando liest die Datei und fügt den Inhalt hinter die angegebene Zeile ein.

Die Adresse 0 ist für dieses Kommando erlaubt.

Sie bewirkt, daß die Datei an den Anfang des Puffers geschrieben wird. Beim Lesen werden ASCII-Null-Zeichen und alle Zeichen nach dem letzten Zeichen "neue Zeile" entfernt. Nach erfolgreichem Lesen wird die Anzahl der gelesenen Bytes ausgegeben, wenn Sie ed nicht mit dem Schalter "-" aufgerufen haben. Die aktuelle Zeile ist die letzte eingelesene Zeile.

ersetzen - substitute

[bereich] s/regulärer Ausdruck/Ersetzungszeichenfolge/ oder
[bereich] s/regulärer Ausdruck/Ersetzungszeichenfolge/ g

Standard für bereich: ..

Das substitute Kommando durchsucht die Zeilen des adressierten Bereichs nach Zeichenfolgen, die zu dem regulären Ausdruck passen. In jeder so gefundenen Zeile wird, falls .../g angegeben wurde, jedes, sonst nur das erste Auftreten der passenden Zeichenfolge durch die Ersetzungszeichenfolge ersetzt. Falls keine passende Zeichenfolge gefunden wurde, meldet ed ein "?".

Um den regulären Ausdruck von der Ersetzungszeichenfolge zu trennen, können auch alle anderen Zeichen außer dem Leerzeichen und dem Zeilenvorschubzeichen benutzt werden. Die aktuelle Zeile ist die Zeile auf der die letzte Ersetzung stattgefunden hat.

Ein "&"-Zeichen in der Ersetzungszeichenfolge wird bei erfolgreicher Suche durch die Zeichenfolge der Zeile ersetzt, die zu dem regulären Ausdruck paßt. Die besondere Bedeutung dieses Zeichens kann durch das Voranstellen eines "\" aufgehoben werden.

Ein "\n" - n ist eine Dezimalzahl - wird ersetzt durch die Zeichenfolge, die zu dem n-ten regulären Unterausdruck paßt, der zwischen "(" und ")" eingeschlossen ist. Wenn geschachtelte, geklammerte Unterausdrücke vorhanden sind, wird n durch Zählen des Auftretens von "(" bestimmt.

\[]

aufteilen - split

Zeilen können geteilt werden, indem ein Zeilenvorschubzeichen in eine Zeile eingefügt wird. Das Zeilenvorschubzeichen "neue Zeile" muß durch Voranstellen eines "\" entwertet werden.

[bereich] ta kopieren - copy

Standard für bereich: ..

Das copy Kommando kopiert den adressierten Bereich hinter die angegebene Zeile a. Die Angabe der Zeile 0 ist zugelassen. Die aktuelle Zeile ist die letzte der kopierten Zeilen.

[bereich] u rückgängig machen - undo

Standard für bereich: ..

Das undo Kommando stellt den alten Inhalt der angegebenen Zeile wieder her, falls die letzte Änderung in dieser Zeile vorgenommen wurde.

[bereich] v/regulärer Ausdruck/Kommando-Liste global but

Standard für bereich: 1,\$

Das v-Kommando unterscheidet sich von dem global-Kommando nur durch die Auswahl der Zeilen. Im Gegensatz zu global werden hier alle Zeilen ausgesucht, die keine Zeichenfolge enthalten, die zu dem regulären Ausdruck paßt.

[bereich] w [datei] schreiben - write

Standard für bereich: 1,\$

Standard für datei: der aktuell gespeicherte Dateiname

Das write Kommando schreibt den angegebenen Bereich in die Datei. Der alte Inhalt der Datei wird dabei überschrieben. Existiert die Datei noch nicht, so wird sie angelegt. Ist ein Name angegeben, überschreibt dieser den bisher gespeicherten. Die Adresse 0 ist nicht zugelassen, d.h. es ist nicht möglich, eine leere Datei zu erzeugen. Die aktuelle Zeile bleibt unverändert. Nach erfolgreichem Schreiben wird die Anzahl der geschriebenen Bytes ausgegeben, wenn Sie nicht den ed mit dem Schalter "- " aufgerufen haben.

[bereich] W [datei]

Write

Standard für bereich: 1,\$

keine Angabe für datei: der in filename gespeicherte Dateiname

Im Gegensatz zum write Kommando schreibt W die angegebenen Zeilen ans Ende der datei; der alte Inhalt wird also nicht zerstört.

x

Schlüssel - key

Das key-Kommando schaltet in den Ent- bzw. Verschlüsselungsmodus um. ed fordert die Angabe eines Schlüssels an. Folgende edit- read- und write-Kommandos ent- bzw. verschlüsseln den betreffenden Text mit Hilfe des Algorithmus des crypt Kommandos. Von crypt und edit verschlüsselte Texte sind also kompatibel. Die Angabe eines leeren Schlüssels schaltet den Ent- bzw. Verschlüsselungsmodus wieder ab.

[zeile] =

Zeilennummer - linenumber

Standard für zeile: \$

Die Nummer der angegebenen Zeile wird ausgegeben. Die aktuelle Zeile wird dadurch nicht verändert.

! <shell kommando >

shell aufrufen

Der Rest der Zeile hinter dem "!" wird an die Shell übergeben, die versucht, die Eingabe zu interpretieren. Nach Beendigung des Shell Kommandos ist ed wieder aktiv. Die aktuelle Zeile wird nicht verändert.

[zeile + zahl]

nächste Zeile - next line

Standard für zeile: .

Standard für zahl: 1

Die Eingabe einer Leerzeile (Sie drücken nur die Taste) bewirkt, daß die nächste Zeile ausgegeben wird. Sind zeile und zahl angegeben, wird die Zeile mit der Nummer (Nummer von zeile + zahl) ausgegeben.

DEL

Kommandoabbruch

Mit der Taste **DEL** können Sie laufende ed Kommandos unterbrechen. ed meldet sich anschließend mit einem '?'.
?

END

Beenden des ed

Die Wirkung ist dieselbe wie beim Kommando q.

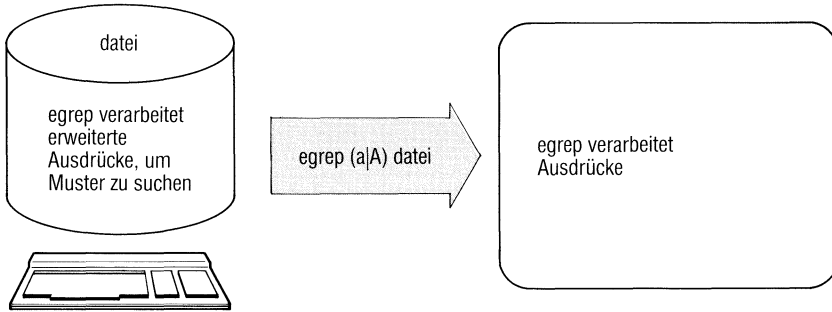
Hinweis

- Liest ed seine Kommandos nicht von der Datensichtstation sondern aus einer Datei, so wird nach dem ersten für ed unverständlichen Kommando abgebrochen.

Fehlermeldungen

- | | | |
|--------|---|---|
| ?datei | – | die Datei ist nicht vorhanden oder kann nicht gelesen werden; |
| ? | – | Syntaxfehler im Kommando; |
| ?TMP | – | Überlauf der temporären Datei. |

Erweiterte Muster suchen – extended grep



`egrep` durchsucht Dateien nach dem angegebenen Muster und gibt jede Zeile aus, die das Muster enthält. `egrep` läßt gegenüber `grep` (bzw. `ed`) erweiterte reguläre Ausdrücke für Muster zu, z.B. "Arbeit(geber|nehmer)".

Benutzen Sie

- `fgrep` um nach konstanten Zeichenfolgen zu suchen, z.B. 'Apfel'. `fgrep` arbeitet schneller als `grep` und `egrep`.
- `grep` um einfachere Muster zu suchen. `grep` läßt als Muster reguläre Ausdrücke zu.

egrep[`_`-schalter...][`_`-muster][`_`-datei...]

schalter

- `v` Ausdrucken aller Zeilen, die das Muster nicht enthalten.
- `c` Anzahl der Zeilen ausdrucken, die das Muster enthalten.
- `l` Namen der Dateien ausdrucken, die das Muster enthalten.
- `n` Zeilennummer vor jeder Zeile ausdrucken.
- `b` Blocknummer vor jeder Zeile ausdrucken.
- `s` Ausgabe unterdrücken. `egrep` liefert nur den Ende-Status, z.B. um in Shell-Prozeduren dem Ergebnis entsprechend zu verzweigen.

- h** Beim Durchsuchen mehrerer Dateien werden die Dateinamen nicht mit ausgedruckt.
- e_muster** ist anzugeben, wenn das Muster mit dem Zeichen "-" beginnt.
z.B. sucht egrep -e -abc alle Zeilen, die das Muster "-abc" enthalten.
- f_datei** egrep entnimmt das Muster der Datei und gibt jede Zeile der durchsuchten Dateien aus, die eines der Muster enthalten. Jede Zeile der Datei gilt als Muster.
- muster** ein regulärer Ausdruck, wie beim Kommando ed beschrieben mit den Erweiterungen 9-11 (Tabelle rechts)
- datei** Datei, die nach dem Muster durchsucht werden soll. Bei mehreren Angaben durchsucht egrep jede Datei und gibt vor den gefundenen Zeilen jeder Datei den Dateinamen aus.
Standard (keine Angabe): egrep liest von der Standard-Eingabe.

Ende-Status:

- 0 Muster gefunden
- 1 Kein Muster gefunden
- 2 Fehlerhafter Ablauf, z.B. Syntaxfehler oder Datei kann nicht geöffnet werden.

	Syntax	Interpretation	Beispiel
9a	r^*	eine Folge von null oder mehr Interpretationen von r	'(ok)*' : , ok, okok, okokok,...
9b	r^+ r beliebiger regulärer Ausdruck	eine Folge von einer oder mehr Interpretationen von r	'(ok)+' : ok, okok, okokok,...
9c	$r?$	eine Folge von null oder einer Interpretation von r	'(ok)?' : , ok
10	$u v$ u, v reg. Ausdrücke u NL v	eine Interpretation von u oder v (Alternative)	'to(day morrow)': today oder tomorrow
11	(r) r regulärer Ausdruck	dieselben Zeichenreihen wie r	'(a b c)' : a oder b oder c

Für die Operatoren gelten folgende Prioritäten:

- Klammern haben höchste Priorität
- auf gleichem Klammerniveau gilt:
 - * , + , ? geht vor Konkatenation geht vor Alternative (| oder NL).

Beispiel

1) Finde alle Zeilen in der Datei test, die eine nichtleere Folge von "." enthalten:

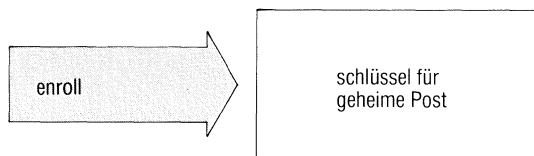
```
egrep '(.)+' test
```

2) Finde alle Zeilen in der Datei test, die weder mit "\$" beginnen noch mit "\\" enden:

```
egrep -v '^\$ | \$' test
```

>>>> ed, fgrep, grep, sed

Schlüssel für geheime Post festlegen



enroll legt einen Schlüssel fest, mit dem geheime Post verschlüsselt oder entschlüsselt werden kann (siehe xsend). Wenn Ihnen ein anderer Benutzer eine verschlüsselte Nachricht senden will (oder Sie sich selbst), müssen Sie mit enroll einen Schlüssel festgelegt haben. Die Nachricht können Sie mit xget lesen, wenn Sie dabei denselben Schlüssel angeben.

enroll

enroll fragt ab "Geben Sie einen Schlüssel an:" Hierauf geben Sie einen Schlüssel ein (1-8 Zeichen) und schließen die Eingabe mit ab. Die Eingabe ist nicht sichtbar.

Jede Nachricht, die Ihnen von jetzt ab mit xget zugeht, wird mit diesem Schlüssel behandelt. Um den Schlüssel zu ändern, rufen Sie enroll erneut auf. Neu eintreffende Nachrichten werden dann mit dem neuen Schlüssel behandelt.

Ende-Status: immer 0

Beispiel

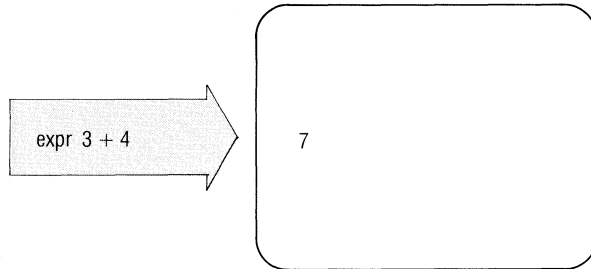
Sie legen als Schlüssel für geheime Post das Wort "eva" fest:

```

$ enroll
Geben Sie einen Schlüssel an: eva
$
  
```

```
>>>> mail, xget, xsend
```

Ausdrücke auswerten



Die in der Kommandozeile angegebenen Argumente interpretiert expr als Ausdrücke und wertet sie nacheinander aus. Die Ergebnisse werden auf die Standard-Ausgabe ausgegeben.

expr argument...

argument Jede Zeichenfolge ohne Zwischenraum wird als ein Argument gewertet. Mehrere Argumente werden durch Leerzeichen voneinander getrennt. Die nachfolgende Liste enthält alle Operatoren zum Verknüpfen von Ausdrücken. Die Liste ist nach steigender Priorität geordnet; Operatoren mit gleicher Priorität sind in einer Gruppe zusammengefaßt.

a1 | a2

Das Ergebnis ist der erste Ausdruck (a1) wenn dieser weder leer noch 0 ist; ansonsten ist der zweite Ausdruck (a2) das Ergebnis.

a1 & a2

Das Ergebnis ist der erste Ausdruck (a1), wenn keiner der Ausdrücke leer oder 0 ist; ansonsten ist das Ergebnis 0.

`a1 relop a2`

relop ist einer der üblichen Vergleichsoperatoren:

`<`, `<=`, `=`, `!=`, `>=`, `>`.

Wenn die angegebene Bedingung erfüllt ist, ist das Ergebnis 1.

Wenn die angegebene Bedingung nicht erfüllt ist, ist das Ergebnis 0.

Wenn beide Ausdrücke vom Typ Integer sind, vergleicht das Kommando numerisch; ansonsten wird alphabetisch verglichen.

`a1 + a2`

Addition

`a1 - a2`

Subtraktion

`a1 '*' a2`

Multiplikation. Das Zeichen "*" ist in Hochkommas einzuschließen, weil es sonst von der Shell interpretiert wird.

`a1 / a2`

Division

`a1 % a2`

Modulofunktion

`a1 : a2`

Das Kommando vergleicht die Zeichenfolge des ersten Arguments mit dem regulären zweiten Ausdruck. Die Syntax des regulären Ausdrucks ist die gleiche, wie die des **ed**-Kommandos. Die Mustersymbole `\(...\)` können verwendet werden, um einen Teil des ersten Arguments auszuwählen. Ansonsten liefert das Ergebnis die Anzahl der Zeichen, die übereinstimmen. Wenn keine Übereinstimmung vorliegt, ist das Ergebnis 0.

(ausdruck)

Runde Klammern fassen Ausdrücke in Gruppen zusammen.

Ende-Status:

- 0 wenn der Ausdruck weder leer noch 0 ist
- 1 wenn der Ausdruck leer oder 0 ist
- 2 bei fehlerhaften Ausdrücken
- 132 bei Division durch 0

Beispiele

1. Um zur Shell-Variablen a 1 dazuzuzählen:

```
a = `expr $a + 1`
```

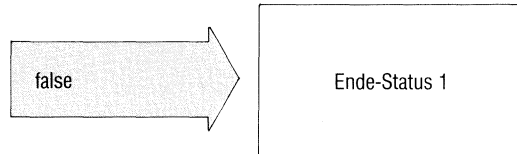
2. Um aus dem in der Variablen a abgelegten Pfadnamen den Datei-Namen-Teil (den am weitesten rechts stehenden Teil), der ein '/' enthalten kann, oder nicht, zu finden:

```
expr $a : '.*\/(.*)' '/' $a
```

Beachten Sie die zitierten Shell-Metazeichen.

>>>> ed, Shell (Abschnitte 3.6 bis 3.8), test

Leeres Kommando mit Endestatus 1



`false` kehrt mit Ende-Status 1 zurück und tut sonst nichts. `false` verwendet man in Shell-Prozeduren, um die Bedingung "falsch" zu erzeugen. Die Bedingung "wahr" (Endestatus 0) erzeugen Sie mit dem Kommando "true".

false

Endestatus: immer 1.

Beispiele

1. Der Ende-Status von false ist 1.

```
$ false
$ echo $?
1
$
```

2. Die folgende Prozedur erzeugt eine Endlos-Schleife. Sie lässt sich z.B. mit der Taste unterbrechen.

```
until false
do
.
.
done
```

> > > > Shell (Abschnitte 3.6 bis 3.8), true

Archivieren auf Diskette – floppy archiver

far archiviert Dateien und ganze Unterbäume des Dateisystems in einem Diskettenarchiv.

Das Kommando far (floppy archiver) ist identisch mit dem Kommando tar mit folgendem Unterschied:

far kennt die Größe einer Diskette und verwendet standardmäßig das Attribut k. Dieses bewirkt, daß far eine Folgediskette anfordert, um eine Datei aufzuteilen, wenn Sie nicht mehr auf die Diskette paßt.

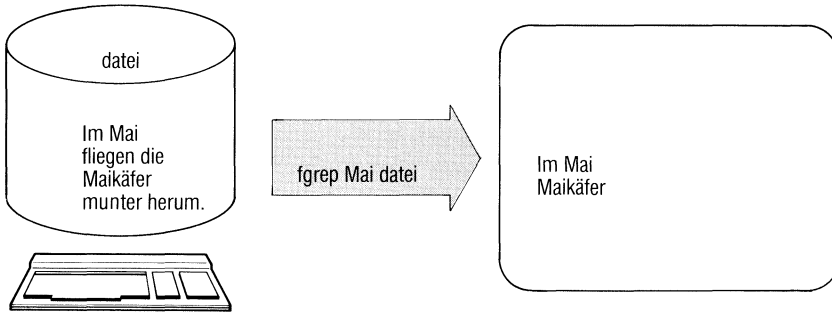
Außerdem verwendet far immer das Attribut n, das den Ablauf beschleunigt.

far `funktion`[attribut...][`argument...`][`datei...`]

Die Beschreibung ist identisch mit der Beschreibung von tar. Sehen Sie bitte dort nach.

> > > > tar

Einfache Muster schnell suchen – fast grep



fgrep durchsucht Dateien nach dem angegebenen Muster und gibt jede Zeile aus, die das Muster enthält. Damit lassen sich z.B. bestimmte Textstellen finden. fgrep läßt als Muster zu: Zeichenfolgen ohne ersetzbare Sonderzeichen, z.B. Wörter wie "Maikäfer".

Benutzen Sie

- `grep` um Muster zu suchen, die Sie durch reguläre Ausdrücke bilden können. `grep` arbeitet langsamer als `fgrep`.
- `egrep` um kompliziertere Muster zu suchen. `egrep` läßt als Muster volle reguläre Ausdrücke zu.

fgrep[`[-schalter...]`][`[-muster]`][`[-datei...]`]

schalter

kein Schalter angegeben

fgrep gibt alle Zeilen aus, die das Muster enthalten.

`v` Ausgeben aller Zeilen, die das Muster nicht enthalten.

`c` Anzahl der Zeilen ausgeben, die das Muster enthalten.

`l` Namen der Dateien ausgeben, die das Muster enthalten.

`n` Zeilennummer vor jeder Zeile ausgeben.

`b` Blocknummer vor jeder Zeile ausgeben.

- s Ausgabe unterdrücken. fgrep liefert nur den Ende-Status, z.B. um in Shell-Prozeduren dem Ergebnis entsprechend zu verzweigen.
- h Beim Durchsuchen mehrerer Dateien werden die Dateinamen nicht mit ausgegeben.
- y Bei Buchstaben im Muster unterscheidet fgrep nicht zwischen Groß- und Kleinschreibung.
- e_muster
 ist anzugeben, wenn das Muster mit dem Zeichen "-" beginnt, z.B. sucht fgrep -e -abc alle Zeilen, die das Muster "-abc" enthalten.
- f_file
 fgrep entnimmt das Muster der Datei file und gibt jede Zeile der durchsuchten Dateien aus, die eines der Muster enthalten. Jede Zeile von file gilt als Muster.
- x alle Zeilen ausdrucken, die nur das Muster enthalten.
- muster eine Zeichenfolge, nach der fgrep jede Zeile durchsucht.
- datei Datei, die nach dem Muster zu durchsuchen ist. Wenn Sie mehrere Dateien angeben, durchsucht sie fgrep alle und gibt vor den gefundenen Zeilen jeder Datei den Dateinamen aus.
- Standard (keine Angabe):* fgrep liest von der Standard-Eingabe.

Ende-Status:

- 0 Muster gefunden
- 1 Kein Muster gefunden
- 2 Fehlerhafter Ablauf, z.B. Syntaxfehler oder Datei kann nicht geöffnet werden.

Beispiele

1. Aus den Dateien kunden1 bis kunden3 sind alle Zeilen zu suchen, die den Namen "Wanninger" enthalten.

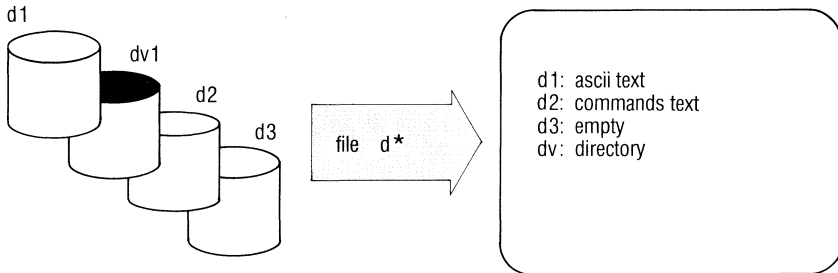
```
$ fgrep Wanninger kunden1 kunden2 kunden3
kunden1: Buchbinderei Wanninger Muenchen
kunden2: Wanninger Herbert, Muenchen 5, Kirschstr.3
kunden3: 120383 1240.25 3 LE Art. 023 Fa. Wanninger
kunden3: 180584 330.87 1 LE Art. 332 Fa. Wanninger
kunden3: 080684 999.98 20 LE Art. 038 Fa. Wanninger
```

2. Eine Datei soll nur ausgedruckt werden, wenn sie die Zeichenfolge "Buch 2" enthält. Das leistet folgende Prozedur

```
if fgrep -s 'Buch 2' $1
print $1
fi
```

>>>> ed, egrep, grep, sed, Shell (Abschnitte 3.6 bis 3.8)

Art einer Datei bestimmen



file unterscheidet Dateien nach ihrem Inhalt, z.B.

- Text-Dateien (ASCII-Zeichen),
- Shell-Prozeduren,
- C-Programme (Quellprogramme),
- ausführbare C-Programme.

file arbeitet wie das Kommando "dateityp", die Ausgabe ist jedoch englisch.

file[_-f]_dateiname...

dateiname Name einer oder mehrerer Dateien, deren Art bestimmt wird.

f Bei dateiname ist eine Datei anzugeben, die eine Liste von Dateinamen enthält. Diese Dateien untersucht file.

file

file gibt aus: "dateiname: dateiart". Für dateiart kann stehen

ascii text	Textdatei
directory	Dateiverzeichnis
commands text	Shell-Prozeduren
c-program text	C-Quellprogramm
executable	ausführbares C-Programm
... input text	Eingabedatei z.B. für make

Ausführbare C-Programme klassifiziert file weiter nach den Schaltern, die beim cc-Kommando gesetzt waren (siehe C-Entwicklungssystem):

Klassifikation	Schalter
separate	i
not stripped	s

Hinweis

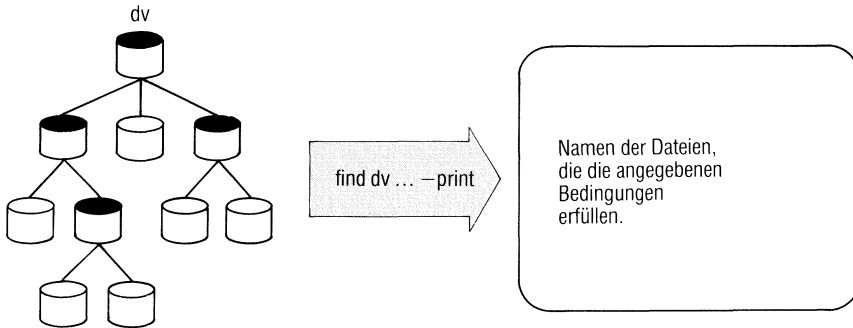
file verwechselt leicht Shell-Prozeduren mit C-Programmen. Ebenso bezeichnet file irrtümlich Programme, die mit Kommentaren beginnen, als Text.

Beispiel

```
$ file *
april: empty
blind: commands text
cobol: directory
core: data
datei1: ascii text
fehler: ascii text
neu5: empty
prep.i: English text
$
```

>>>> C-Entwicklungssystem

Dateiverzeichnisse durchsuchen – find files



find durchsucht Dateiverzeichnisse nach Dateien, die vorgegebene Bedingungen erfüllen. Für jede so gefundene Datei können Sie find folgendes ausführen lassen:

- Dateinamen ausdrucken (-print oder -space)
- ein Kommando ausführen (-exec oder -ok)

find *pfadname*...*bedingung*...*aktion*

pfadname find durchsucht alle Dateiverzeichnisse, die über diesen Pfadnamen erreichbar sind.

bedingung Bedingung, auf die find die Dateien prüft. Sie können mehrere Bedingungen auch verknüpfen, wie unten gezeigt.

name *dateiname*

Die Datei hat den Namen *dateiname*. Für *dateiname* können Sie auch die üblichen Abkürzungen verwenden, z.B. find /usr/adam -name 'gruppe.*' ... ; der Dateiname ist dann in Hochkommas einzuschließen.

Pfadnamen können Sie bei name nicht angeben!

perm *oktalzahl*

Die Zugriffsrechte der Datei entsprechen der Angabe *oktalzahl* (siehe chmod).

type_c

Die Datei ist vom Typ c. Für c können Sie angeben:

- b für Block-Geräte-datei
- c für Byte-Geräte-datei
- d für Dateiverzeichnis
- f für normale Datei

links_n

Auf die Datei bestehen n Verweise. n ist eine Dezimalzahl.

- +n bedeutet mehr als n,
- n bedeutet weniger als n,
- n bedeutet genau n Verweise.

user_benutzername

Die Datei gehört dem Benutzer benutzername.

Statt des Benutzernamens können Sie auch die Benutzer-nummer angeben.

group_gruppenname

Die Datei gehört zur Gruppe gruppenname.

Statt des Gruppennamens können Sie auch die Gruppen-nummer angeben.

size_n

Die Datei belegt n Blöcke (je 512 Bytes).

+n, -n wie bei der Bedingung 'links' beschrieben.

inum_n

n ist die Nummer des Indexeintrags der Datei

(Indexnummer). +n, -n wie bei der Bedingung 'links' beschrieben.

atime_n

Letzter Dateizugriff war vor n Tagen.

+n, -n wie bei der Bedingung 'links' beschrieben.

mtime_n

Letzte Dateiänderung war vor n Tagen.

+n, -n wie bei der Bedingung 'links' beschrieben.

newer_datei

Die geprüfte Datei wurde zu einem neueren Zeitpunkt geändert als die hier genannte Datei datei.

aktion

Angegebene Aktionen führt find aus, wenn alle Bedingungen davor erfüllt sind. Sie können mehrere Aktionen angeben. exec und ok wirken ebenfalls als Bedingung für weitere Aktionen.

find arbeitet also alle Angaben von links nach rechts ab und führt die Aktionen aus, solange die Bedingungen erfüllt sind. Sobald eine Bedingung nicht erfüllt ist, bricht find ab und prüft die nächste Datei.

Wenn Sie nicht wenigstens eine Aktion angeben, z.B. print, liefert find kein Ergebnis.

exec_kommando

kommando wird ausgeführt. Die Bedingung ist erfüllt, wenn das Kommando den Ende-Status 0 hat.

{ } als Kommandoargument steht für den aktuellen Dateinamen, z.B. ... -exec ls -l { } \; ; ... : volle Information über die gefundenen Dateien ausgeben.

Das Kommando müssen Sie mit der Zeichenfolge " \; " abschließen.

ok_kommando

ok wirkt wie exec mit folgendem Unterschied:

find fragt vorher ab: <kommando ... dateiname> ?

Antworten Sie mit:

y für 'Kommando ausführen',

n für 'Kommando nicht ausführen'. Die Bedingung ist dann nicht erfüllt und find unterdrückt alle folgenden Aktionen (-print, -exec, -ok).

print

Ausdrucken des Pfadnamens für jede gefundene Datei.

print setzt keine Bedingung.

space

Wie print (s.o.) plus zusätzlicher Ausgabe der Dateigröße in Byte.

Bedingungen verknüpfen

Bedingungen können Sie wie folgt miteinander verknüpfen:

`\(␣bed ...␣\)`

Klammern fassen Bedingungen zu Gruppen zusammen. Die Klammern sind mit "`\`" für die Shell zu entwerten.

`!␣bed`

Verneinung, z.B. `!-user berta`.

`bed␣bed␣...`

logisches UND. Jede der aneinandergereihten Bedingungen muß erfüllt sein.

`bed␣-o␣bed`

logisches ODER. Eine der Bedingungen muß erfüllt sein.

find verarbeitet die Verknüpfungen in der Reihenfolge:

Klammern, Verneinung, UND, ODER, z.B. ergibt

`-type f -links 2 -o -type d` die Bedingung:

die Datei ist entweder eine Benutzerdatei mit 2 Verweisen oder ein Dateiverzeichnis.

Beispiele

1. Alle Dateien im aktuellen Dateiverzeichnis löschen, auf die länger als 20 Tage nicht zugegriffen wurde. Vor dem Löschen soll abgefragt werden.

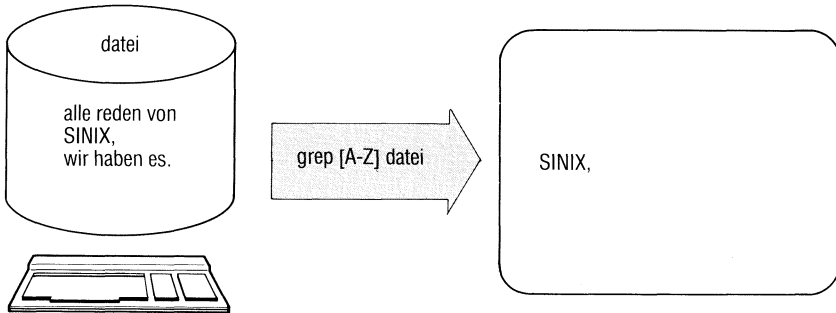
```
find . -atime +20 -ok rm {} \;
```

2. Alle Einträge im Dateiverzeichnis `/usr/florian` ausgeben, deren Eigentümer nicht "florian" ist.

```
find /usr/florian !-user florian -print
```

> > > > Shell (Abschnitte 3.6 bis 3.8), test

Muster in Dateien suchen – global regular expression print



grep durchsucht Dateien nach dem angegebenen Muster und gibt jede Zeile aus, die das Muster enthält. Damit lassen sich z.B. bestimmte Textstellen finden. grep läßt als Muster zu: Zeichenfolgen, die als reguläre Ausdrücke geschrieben werden, z.B. [A-Z].* für Zeichenfolgen, die mit einem Großbuchstaben beginnen.

Benutzen Sie

- `fgrep` um nach konstanten Zeichenfolgen zu suchen, z.B. 'Apfel'. `fgrep` arbeitet schneller als `grep`,
- `egrep` um kompliziertere Muster zu suchen. `egrep` läßt als Muster volle reguläre Ausdrücke zu.

grep[`_-schalter...`]`_muster`[`_datei...`]

schalter

- v Ausdrucken aller Zeilen, die das Muster nicht enthalten.
- c Anzahl der Zeilen ausdrucken, die das Muster enthalten.
- l Namen der Dateien ausdrucken, die das Muster enthalten.
- n Zeilennummer vor jeder Zeile ausdrucken.
- b Blocknummer vor jeder Zeile ausdrucken.

- s Ausgabe unterdrücken. grep liefert nur den Ende Status, z.B. um in Shell-Prozeduren dem Ergebnis entsprechend zu verzweigen.
- h Beim Durchsuchen mehrerer Dateien werden die Dateinamen nicht mit ausgedruckt.
- y Bei Buchstaben im Muster unterscheidet grep nicht zwischen Groß- und Kleinschreibung.

Achtung

Der Schalter funktioniert nicht, falls das Muster ein regulärer Ausdruck der Form "[a-b]" ist, wobei a und b Buchstaben sind.

e_muster

ist anzugeben, wenn das Muster mit dem Zeichen "-" beginnt.
Z.B. sucht grep -e -abc alle Zeilen, die das Muster "-abc" enthalten.

muster ein regulärer Ausdruck, wie beim Kommando ed beschrieben, z.B. '*.strasse', siehe auch Tabelle im Anhang.

datei Datei, die nach dem Muster zu durchsuchen ist. Wenn Sie mehrere Dateien angeben, durchsucht sie grep alle und gibt vor den gefundenen Zeilen jeder Datei den Dateinamen aus.

Standard (keine Angabe): grep liest von der Standard-Eingabe.

Ende-Status:

- 0 Muster gefunden
- 1 Kein Muster gefunden
- 2 Fehlerhafter Ablauf, z.B. Syntaxfehler oder Datei kann nicht geöffnet werden.

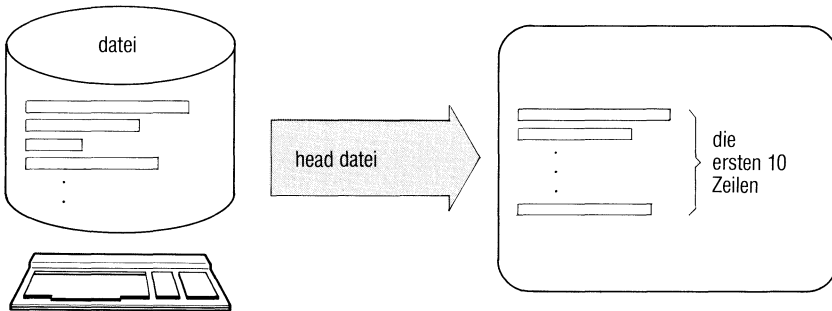
Beispiel

Suche alle Zeilen in der Datei test, die auf "n" oder "N" enden:

```
grep -y 'n$' test
```

>>>> ed, egrep, fgrep, sed, Shell (Abschnitte 3.6 bis 3.8)

Anfangszeilen von Dateien ausgeben



head gibt die Anfangszeilen der angegebenen Dateien aus.

head[-anzahl][-datei...]

anzahl Anzahl der Zeilen, die head ab Dateianfang ausgibt (1-9999).

Standard (keine Angabe): 10 Zeilen.

datei Datei, deren Anfangszeilen auszugeben sind. Bei mehreren Dateien gibt head die Dateinamen mit aus.

Standard (keine Angabe): head liest von der Standard-Eingabe.

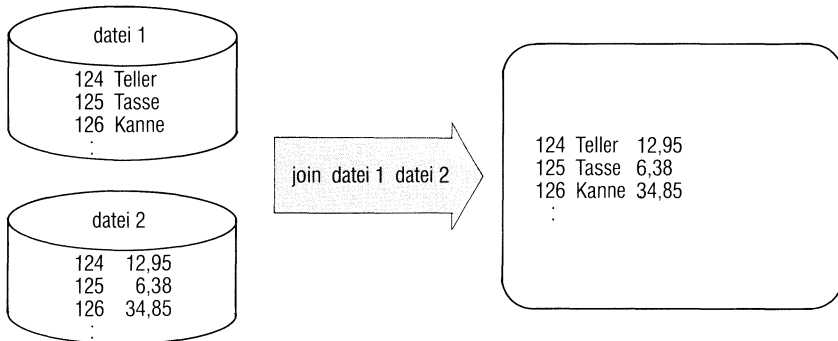
Beispiel

Sie möchten sich einen Überblick verschaffen, welchen Inhalt die Dateien des aktuellen Dateiverzeichnisses haben.

```
find . -type f -print -exec head -5 {} \; | pr | lpr
```

>>>> tail

Dateien verbinden nach Vergleichsfeldern



join vergleicht zwei Dateien nach Vergleichsfeldern, die Sie festlegen können. Für jede Übereinstimmung gibt join eine Zeile aus. Die Ausgabezeile enthält das Vergleichsfeld und auswählbare Felder der ersten Datei und der zweiten Datei. Die Dateien müssen jeweils nach dem Vergleichsfeld aufsteigend sortiert sein.

join[_schalter...]_datei1_datei2

schalter

an

zusätzlich die Zeilen aus datei n (1 oder 2) ausgeben, die keine Übereinstimmung in der anderen Datei haben.

j[n]_m

legt das Vergleichsfeld in datei n fest (1 oder 2).

join vergleicht das m-te Feld. Fehlt die Angabe n, vergleicht join das m-te Feld beider Dateien.

Standard (keine Angabe): join vergleicht das erste Feld beider Dateien.

o_m.n[_m.n ...]

wählt Felder für die Ausgabezeile aus. Ein Eintrag m.n definiert als Ausgabefeld das m-te Feld der n-ten Datei (1 oder 2).

join

- es leere Ausgabefelder durch die Zeichenfolge s ersetzen.
- tc legt ein Zeichen c als Trennzeichen für Felder fest. Zwei aufeinanderfolgende Trennzeichen kennzeichnen ein leeres Feld (siehe auch Kommando sort).

datei1 datei2

Dateinamen der beiden Eingabedateien. Steht "-" für einen der Dateinamen, liest join dafür von der Standard-Eingabe.

Standard (keine Angabe): join liest von der Standard-Eingabe.

Hinweis

- Gilt für die Feldtrennung das Leerzeichen (Standard), muß die Datei mit "sort -b" sortiert sein, d.h. führende Leerzeichen werden nicht berücksichtigt. Ist dagegen bei join Schalter t angegeben, ist normal mit sort zu sortieren.
- Bei der Ausgabe richtet join die Felder nicht spaltenweise bündig aus. Dies können Sie z.B. mit dem awk machen.

Beispiel

In der Datei `ort` ist einem Namen ein Ort zugeordnet, in der Datei `betrag` sind denselben Namen Beträge zugeordnet. Beide Dateien sind nach dem Namen sortiert. `join` soll beide Dateien nach den Namen verbinden.

Inhalt der Datei `ort`:

```
Albert Muenchen
Hugo   Stuttgart
Ilse   Hamburg
```

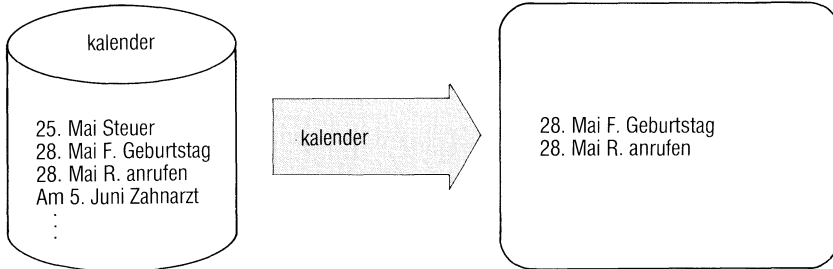
Inhalt der Datei `betrag`:

```
Albert 287.56 20.03.84
Hugo   23.15 25.06.84
Hugo   167.87 16.12.83
Ilse   1212.12 12.12.82
Ilse    1.98 01.01.83
```

```
$ join ort betrag | awk '{printf "%-10s %-15s %-10s %-10s\n", $1, $2, $3, $4}'
Albert Muenchen      287.56 20.03.84
Hugo   Stuttgart     23.15 25.06.84
Hugo   Stuttgart     167.87 16.12.83
Ilse   Hamburg       1212.12 12.12.82
Ilse   Hamburg        1.98 01.01.83
```

> > > > `awk, sort, comm`

Erinnerungsdienst, Datum in deutscher Schreibweise



kalender ist eine Gedächtnisstütze für Termine. Die Termine tragen Sie in einer Datei "kalender" im Login-Dateiverzeichnis ein. Das Kommando kalender gibt aus dieser Datei jeweils die Zeilen aus, die das aktuelle Datum enthalten.

kalender hat dieselbe Funktion wie das Kommando calendar. Die Termine werden lediglich in deutscher Schreibweise erfaßt.

kalender[_-]

kein Schalter

kalender druckt aus der Benutzerdatei 'kalender' alle Zeilen aus, in denen an beliebiger Stelle das heutige oder morgige Datum steht. Die üblichen Tag-Monat-Angaben wie etwa 7.Dezember, 7.Dez, 7.12, 7/12 werden erkannt.

An Wochenenden werden für 'morgen' die Tage einschließlich Montag ausgesucht.

- kalender bezieht sich auf alle Benutzer, die eine Datei 'kalender' in ihrem Login-Dateiverzeichnis haben. Jedem solchen Benutzer werden die eventuell gefundenen Zeilen aus seiner kalender-Datei mittels mail zugesandt.

Hinweis

Das Konzept vom verlängerten "morgen" an Wochenenden gilt nicht an Feiertagen.

Ende-Status: immer 0

Beispiel

Die Datei kalender im Login-Dateiverzeichnis habe folgenden Inhalt:

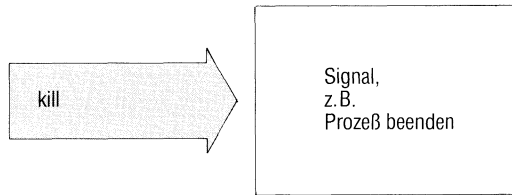
```
31.Mai Nicht vergessen: Reinhardt anrufen!  
8/Jun Bei gutem Wetter Gartenfest bei Stingls  
Heute ist der 7.8. !!  
15 August : Termin bei Kantarelis  
Wichtig: Heute Antrag für ALFGM abgeben (8.Juni)  
.  
.
```

Wenn Sie am 8.Juni kalender aufrufen, erhalten Sie die Ausgabe:

```
8/Jun Bei gutem Wetter Gartenfest bei Stingls  
Wichtig: Heute Antrag für ALFGM abgeben (8.Juni)
```

> > > > at, calendar, mail

Prozesse beenden, Signale senden – kill a process



kill beendet Prozesse der eigenen Benutzerkennung mit Signal 15 oder sendet andere Signale.

kill[_-signal]_prozeßnummer...

signal Signal, das kill senden soll (1-15).

Standard (keine Angabe): Signal 15, d.h. Prozeß beenden.

Hinweis

Prozesse können Signale abfangen und damit unwirksam machen. Signal 9 kann nicht abgefangen werden, d.h. "kill -9 prozeßnummer" beendet einen Prozeß in jedem Fall.

prozeßnummer

Nummer des Prozesses, den Sie beenden wollen. Die aktuellen Prozeßnummern Ihrer Benutzerkennung gibt das Kommando ps aus. Nur solche können Sie angeben.

0 als Prozeßnummer bedeutet: alle Prozesse der eigenen Benutzerkennung.

Mögliche Signale

Folgende Signale sind auf der Kommando-Ebene von Bedeutung. Weitere Signale siehe C-Entwicklungssystem.

2	SIGINT	interrupt	(Taste DEL)
3	SIGQUIT	quit	(CTRL Y)
9	SIGKILL	kill, Prozeß unbedingt abbrechen	
15	SIGTERM	software termination, Prozeß abbrechen	

Hinweis

Der Systemverwalter kann alle Prozesse beenden.

Ende-Status:

0	bei normalem Ablauf
1	bei fehlerhaftem Ablauf
143	bei erfolgreichem Abbruch aller Prozesse mit kill 0

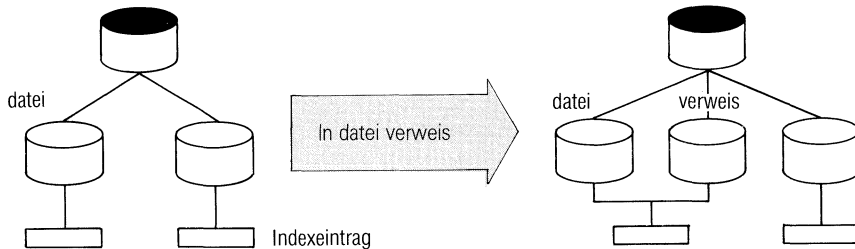
Beispiel

```
$ kill -9 312
$
```

Der Prozeß mit der Nummer 312 wird durch das Signal 9 beendet.

> > > ps, trap (Abschnitt 3.8.14)

Verweis auf eine Datei eintragen – make a link



In trägt in einem Dateiverzeichnis einen Verweis auf eine vorhandene Datei ein (siehe auch Abschnitt 2.3). Sie können dann z.B. von verschiedenen Dateiverzeichnissen aus mit derselben Datei arbeiten, ohne lange Pfadnamen anzugeben.

In_dateiname[_verweisname]

dateiname Name der Datei, auf die Sie einen Verweis eintragen wollen. Diese Datei muß bereits vorhanden sein. Der Verweis wird im aktuellen Dateiverzeichnis eingetragen, falls Sie keinen Pfadnamen angeben.
Nicht verweisen dürfen Sie auf Dateiverzeichnisse und auf andere Dateisysteme.

verweisname Unter diesem Namen können Sie die Datei dann ansprechen.

Standard (keine Angabe):

In nimmt den Namen "dateiname" ohne Pfadbestandteile und trägt den Verweis im aktuellen Dateiverzeichnis ein, z.B. In /usr/wolf/sieben/geisslein trägt im aktuellen Dateiverzeichnis den Namen "geisslein" ein.

Hinweis

Einen Verweis auf eine Datei können Sie nicht mehr vom ursprünglichen Eintrag unterscheiden. Da es sich ja tatsächlich um eine einzige Datei handelt, gibt es nur einen Indexeintrag dazu, auf den jeder Verweis zeigt. Das heißt, die Datei hat auch für alle Verweise dieselben Eigenschaften.

Beim Löschen einer Datei mit `rm` wird nur der Verweis aus dem Dateiverzeichnis entfernt. Erst mit dem letzten Verweis löscht `rm` die Datei wirklich.

Beispiel

Die zwei Benutzer "max" und "moritz" wollen mit derselben Datei ".profile" arbeiten. Sie ist im Dateiverzeichnis `/usr/max` bereits eingerichtet:

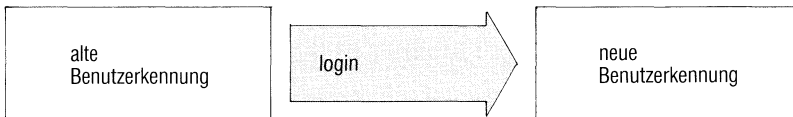
```
In /usr/max/.profile /usr/moritz/.profile
```

Ist `/usr/moritz` das aktuelle Dateiverzeichnis, genügt die Angabe:

```
In /usr/max/.profile
```

> > > > copy, mv, rm

Benutzerkennung wechseln



Mit `login` wechseln Sie direkt in eine andere Benutzerkennung.

login[_benutzer]

`benutzer` Benutzerkennung, in die Sie wechseln wollen.

Standard (keine Angabe): `login` fordert Sie auf, eine Benutzerkennung einzugeben, wie bei Beginn der Sitzung.

Hinweis

- Ist ein Kennwort für die Benutzerkennung vereinbart, fragt `login` dieses im Dialog ab.
- `login` wird von der Shell direkt ausgeführt. Der laufende Prozeß wird überschrieben (siehe auch `newgrp`).
- Die Shell-Umgebung entspricht der neuen Benutzerkennung (siehe Beispiel).

Ende-Status: immer 0.

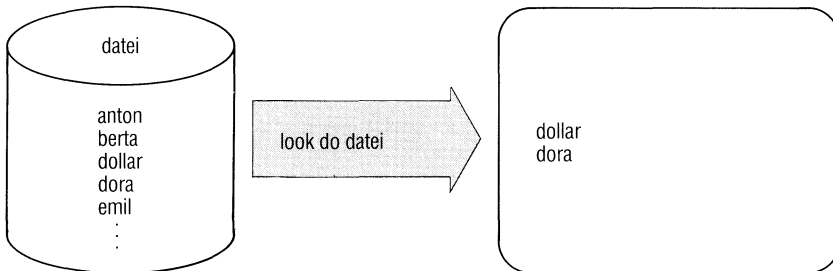
Beispiel

Wechseln in die Benutzerkennung santi. Das Kommando printenv zeigt die neue Shell-Umgebung (vergl. Kommando su).

```
$ login santi
Kennwort:
$ printenv
HOME=/usr/santi
PATH=/bin:/usr/bin
SHELL=/bin/shell
TERM=97801
USER=santi
```

```
> > > > newgrp, su
```

Zeilen mit bestimmtem Anfang suchen



look sucht in einer sortierten Datei alle Zeilen, die mit einer bestimmten Zeichenfolge beginnen und gibt sie aus (siehe auch nächste Seite: Wichtiger Hinweis).

look[-schalter]_zeichenfolge[_datei]

schalter

d look berücksichtigt beim Suchen nur Buchstaben, Ziffern, Leerzeichen und Tabulatorzeichen, z.B.: look -d ab datei findet in datei auch Zeilen, die mit "a,b" oder "a=b" usw. beginnen.

f look unterscheidet nicht zwischen Groß- und Kleinbuchstaben.

zeichenfolge

Zeichenfolge, mit der look die Zeilenanfänge vergleicht. Die Zeichenfolge können Sie so angeben, wie beim Kommando "echo" beschrieben.

datei

zu durchsuchende Datei. Die Datei muß nach ASCII sortiert sein.

Standard (keine Angabe): /usr/dict/words. Diese Datei wird standardmäßig mit der Schalterangabe d und f durchsucht.

Wichtiger Hinweis

Mit dem look-Kommando können Sie nur Dateien durchsuchen, die mit dem sort-Kommando sortiert wurden. Beim look-Kommando wirken die Schalter d und f nur dann, wenn Sie die gleichen Schalter auch beim sort-Kommando angegeben hatten (siehe: sort-Kommando).

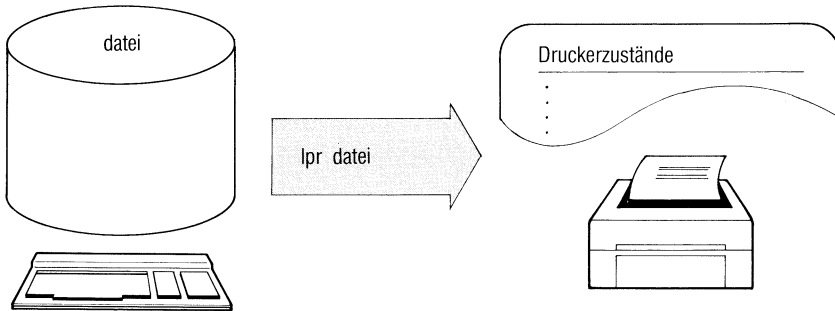
Beispiel

Aus der Datei namen alle Einträge ausgeben, die mit d beginnen:

```
$ look d namen
daniel
david
dora
.
.
$
```

> > > > sort, grep

Dateien ausdrucken und Druckaufträge steuern



lpr steuert Druckaufträge für Dateien. Mit lpr können Sie:

- Dateien am Drucker ausdrucken lassen (Druckauftrag),
- die Form des Ausdrucks beeinflussen, z.B. Zeilen pro Seite,
- die Ausgabe auf einen bestimmten Drucker lenken, wenn mehrere angeschlossen sind,
- Druckaufträge löschen,
- die Priorität von Druckaufträgen ändern,
- den Zustand der Aufträge und die Betriebsbereitschaft der Drucker abfragen.

Als Systemverwalter können Sie außerdem:

- Drucker sperren und wieder freigeben, mit oder ohne Abbruch von laufenden Aufträgen,
- den Druckerbetrieb abschalten,
- die Druckerkonfiguration ändern.

Zur Druckerverwaltung siehe Abschnitt 5.21.

Es kann sein, daß an Ihr System nur ein Drucker angeschlossen ist bzw. nur ein Drucker angeschlossen sein kann. Funktionen oder Angaben, die sich auf mehrere Drucker beziehen, sind dann bedeutungslos.

lpr[**-schalter...**][**-datei...**]

schalter

ws = ger

lpr reiht Ihren Druckauftrag in die Warteschlange des Druckers ein. Eine Liste der an Ihrem System angeschlossenen Drucker erhalten Sie durch Angabe des Schalters q.

Standard (keine Angabe): die erste in der Konfigurationsdatei angegebenen Warteschlange. Der Schalter ist nur anzugeben, wenn Sie mehrere Drucker an Ihr System angeschlossen haben.

pr = n

lpr druckt die angegebenen Dateien mit der Priorität n. Für n sind Werte zwischen 0 (niedrigste Priorität) und 30 zugelassen.

Standard (keine Angabe): 15.

lpr verarbeitet Druckaufträge zuerst nach Priorität, dann wie in der Konfigurationsdatei angegeben (FIFO bzw. KURZ, siehe Abschnitt 5.21).

ap = n

ändert die Priorität bereits abgegebener eigener Druckaufträge nach n.

Fremde Druckaufträge kann nur der Systemverwalter angeben.

ca

löscht den eigenen Druckauftrag für die genannten Dateien und bricht den Auftrag, falls er schon ausgeführt wird, ab. Fremde Druckaufträge kann nur der Systemverwalter angeben.

Gibt es mehrere Druckaufträge mit gleichem Namen, wird nur der älteste gelöscht.

cp

lpr kopiert die angegebenen Dateien in das Dateiverzeichnis /usr/tmp/copies. Sie können dadurch sofort mit der Datei arbeiten, ohne auf die Beendigung des Druckauftrags zu warten.

nc = n

n ist die Anzahl der gewünschten Ausdrücke jeder angegebenen Datei.

Standard (keine Angabe): 1.

no benachrichtigt Sie, wenn fertig ausgedruckt ist, mit "mail".**to = benutzerkennung**

Empfänger des Ausdrucks ist der angegebene Benutzer, das heißt, dieser Name wird im Kopf und im Anhang des Ausdrucks anstelle Ihrer Benutzerkennung eingetragen.

Mit "mail" benachrichtigt werden sowohl Auftraggeber als auch Empfänger. Löschen kann den Auftrag nur der Auftraggeber.

tl = titel

titel erscheint in der Titelzeile im Kopf des Ausdrucks.

Standard: der Name der ausgedruckten Datei.

ab = n

Jede angegebene Datei wird erst ab Seite n ausgedruckt.

bis = n

Jede angegebene Datei wird nur bis Seite n ausgedruckt (einschließlich).

rm Jede angegebene Datei wird nach dem Ausdrucken gelöscht.**q** Informationen ausgeben über den Zustand von Druckaufträgen und die Betriebsbereitschaft von Druckern.

Die Einträge in der Tabelle "Druckerzustände" bedeuten:

WS Name der Warteschlange, wie er in der Konfigurationsdatei /usr/lib/qconfig angegeben ist.

DRUCKER Name des der Warteschlange zugeordneten Druckers wie er in der Konfigurationsdatei /usr/lib/qconfig angegeben ist (Eintrag queue =). Diesen Namen erwartet lpr beim Schalter ws = .

STATUS	Betriebszustand des Druckers
BEREIT	Für diesen Drucker ist kein Druckauftrag vorhanden.
LAEUFT	An diesem Drucker wird gerade ein Druckauftrag abgearbeitet. Der Zustand dieses Auftrags wird in den folgenden Zeilen beschrieben.
WARTET	Die Ausgabe auf dem Drucker ist z.Zt. nicht möglich. Ursachen können sein: Papierende, Papierstau, Farbbandende, Lampe ON-LINE am Gerät brennt nicht.
GESPERRT	lpr nimmt zwar Aufträge für dieses Gerät an. Die Ausführung dieser Aufträge ist aber gesperrt. Wird der Drucker während des Betriebs abgeschaltet, (Netz aus), wird die Druckausgabe gesperrt.
Unbekannt	Unbekannter Betriebszustand, z.B. wegen eines Fehlers in der Konfigurationsdatei. Mit Menüsystem neu konfigurieren.
AUFTRAG	Name der gerade ausgedruckten Datei
SEITEN	Anzahl der bereits gedruckten Seiten
BLK	Größe der auszudruckenden Datei in Blöcken (1 Block = 1024 Zeichen)
% erledigt	Bereits ausgegebener Teil der Datei in Prozent

Die Einträge in der Tabelle "Auftragslage" bedeuten:

WS	Name der Warteschlange
BENUTZER	Kennung des Benutzers, der den Auftrag abgegeben hat
AUFTRAG	Name der auszudruckenden Datei, bzw. Angabe beim Schalter "tl"
BLK	Größe der auszudruckenden Datei in Blöcken
KOP	Anzahl der Kopien
PRI	Priorität des Druckauftrags
ZEIT	Uhrzeit, zu der der Druckauftrag gegeben wurde
AN	Empfänger des Auftrags

pb = nlpr druckt die Zeilen nur bis Spalte n.

Standardwerte:

Drucker 9001: 80 Zeichen bei pb1
96 Zeichen bei pb2
136 Zeichen bei pb3

Drucker 9004: 133 Zeichen

Bei Zeilen, die länger sind, geht der Zeilenrest verloren.

pl = nlpr druckt n Zeilen pro Seite.

Standard: 72

dt	deutscher Zeichensatz (nur für Drucker 9001)
int	ASCII-Zeichensatz (nur für Drucker 9001), Standard
pb1	Zeichenbreite 10 Zeichen / Zoll (nur für Drucker 9001)
pb2	Zeichenbreite 12 Zeichen / Zoll (nur für Drucker 9001)
pb3	Zeichenbreite 17 Zeichen / Zoll (nur für Drucker 9001)

fi lpr druckt die angegebenen Dateien aus, liest anschließend von der Standard-Eingabe und druckt diese Zeilen ebenfalls.

Benutzen Sie lpr als Ende einer Pipeline, und geben Schalter für lpr an, müssen Sie diesen Schalter angeben.

Funktionen für den Systemverwalter

Die folgenden Schalter wirken nur, wenn sie der Systemverwalter angibt.

dd Drucker sperren. Welchen Drucker Sie sperren wollen, geben Sie mit Schalter ws an (Standard ist der erste in der Konfigurationsdatei angegebene Drucker).

Weitere Aufträge für den Drucker nimmt lpr zwar an, führt sie aber nicht aus. Ein eventuell laufender Druckauftrag wird noch zu Ende geführt. Bei der Statusabfrage mit "q" wird der Drucker als GESPERRT gekennzeichnet.

Ist der Drucker nicht betriebsbereit (Status WARTET), dann kann man ihn nur mit Schalter "dk" sperren.

dk Drucker sperren (wie oben), aber:
Ein eventuell laufender Druckauftrag wird abgebrochen. Wird der Drucker später wieder freigegeben, wird der abgebrochene Druckauftrag von Anfang an wiederholt.

du Drucker freigeben. Welchen Drucker Sie freigeben wollen, geben Sie mit Schalter ws an (Standard ist der erste in der Konfigurationsdatei angegebene).
Ein eventuell mit dk abgebrochener Auftrag wird wiederholt.

rr lpr überprüft, ob die binäre Konfigurationsdatei älteren Datums ist als die Datei: /usr/lib/qconfig. Haben Sie sie während des laufenden Betriebs geändert, übersetzt lpr die Konfigurationsdatei. Die Änderungen wirken ab dem nächsten Auftrag, der ausgedruckt wird.

dg Druckerbetrieb abschalten, d.h. das Programm zur Druckerverwaltung wird beendet (siehe Abschnitt 5.21).

su = benutzerkennung

ist anzugeben, um für einen anderen Benutzer die Priorität zu ändern (-pr) oder einen Druckauftrag zu löschen (-ca).

Schalter, die lpr nicht interpretieren kann, werden unverändert an das Treiber-Programm durchgereicht. Dadurch können Sie für ein selbstgeschriebenes Treiberprogramm eigene Schalter definieren.

datei Name der auszudruckenden Datei.

Standard (keine Angabe): lpr liest von der Standard-Eingabe. Beachten Sie dabei Schalter fi.

Hinweis

- In welcher Form lpr die Druckausgabe aufbereitet, hängt davon ab, wie der Drucker in der Konfigurationsdatei beschrieben ist. Lesen Sie dazu 5.21.
- Wurde ein Druckauftrag fehlerhaft beendet, erhalten Sie eine Fehlermeldung mit "mail".

Beispiele

- Information ausgeben über den Druckerzustand:

```
$ lpr -q
      D R U C K E R Z U S T A E N D E

WS  DRUCKER  STATUS      AUFTRAG              SEITEN  % erledigt
---  -
D1Q  D1      GESPERRT
D2Q  D2      LAEUFT      betrag                2        5

      A U F T R A G S L A G E

WS  BENUTZER  AUFTRAG              BLK  KOP  PRI  ZEIT  AN
---  -
D1Q  richter  charts              5    2   15  14.28 richter
D2Q  artmann  betrag              8    1   15  15.16 artmann
D2Q  artmann  ort                 1    1   15  15.18 artmann
```

- Die Datei liste am Drucker D2 ausdrucken:

```
lpr -ws = D2 liste
```

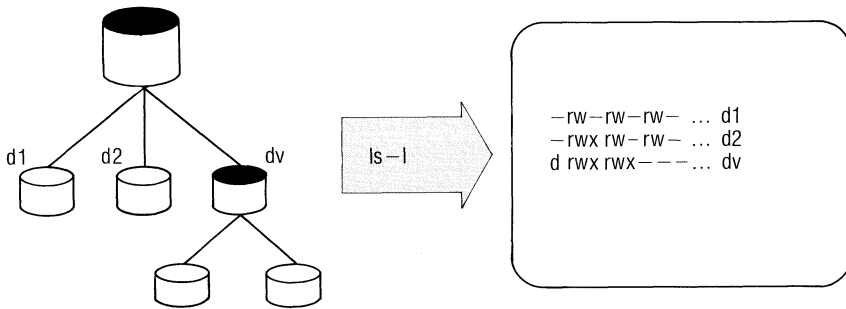
- Den gesperrten Drucker D1 freigeben:

```
lpr -du
```

Die Angabe `-ws = D1` kann entfallen, da D1 der erste Drucker in der Konfigurationsdatei ist.

> > > pr, print, Abschnitt 5.21

Informationen über Dateiverzeichnisse und Dateien list contents of directory



ls gibt aus:

- Informationen über Dateien und Dateiverzeichnisse:
Zugriffsrechte, Eigentümer, Größe, Zeit der letzten Änderung usw.
- welche Dateien in einem Dateiverzeichnis enthalten sind.

ls[₋-schalter...][₋name...]

l (wirkt wie: ls -m)

lf (wirkt wie: ls -F)

ll (wirkt wie: ls -l)

lr (wirkt wie: ls -R)

schalter

kein Schalter

Für Dateiverzeichnisse gibt ls aus:
die Namen aller Dateiverzeichnisse und Dateien, die darin
eingetragen sind in alphabetischer Reihenfolge.

Für Dateien gibt ls nur den Dateinamen aus.

- f Einträge im Dateiverzeichnis ausdrucken in der Reihenfolge, in der sie erfaßt sind.
Der Schalter f setzt den Schalter a und setzt die Schalter l, t, s und r zurück.
- ls interpretiert jede Angabe für name als Dateiverzeichnis. ist eine Datei angegeben, gibt ls -f nichts Sinnvolles aus.
- g Gruppenname statt Benutzerkennung des Eigentümers ausgeben. g wirkt nur zusammen mit l. Ist kein Gruppenname in der Datei /etc/group eingetragen, gibt ls die Gruppennummer aus.
- m Mehrere Namen in einer Zeile ausgeben, durch Kommas getrennt (nicht mit Schalter l).
- l Jeden Namen in einer Zeile ausgeben.
- C Mehrspaltig ausgeben (nicht mit Schalter l).
- q Nicht abdruckbare Zeichen von Dateinamen als "?" ausgeben. Das ist Standard, wenn ls direkt auf den Bildschirm ausgibt.
- b Nicht abdruckbare Zeichen von Dateinamen oktal ausgeben in der Form: \nnn.
- x Zeilenweise statt spaltenweise sortieren bei mehrspaltiger Ausgabe.
- n Benutzernummer statt Name des Eigentümers ausgeben (nur mit Schalter l). Ist auch Schalter g angegeben, zeigt ls die Gruppennummer anstelle des Gruppennamens.
- A Wirkung für Normalbenutzer:
Namen, die mit '.' beginnen, auch ausgeben, nicht aber die Namen '.' und '..', im Unterschied zu Schalter a.
- Wirkung für Systemverwalter:
Namen, die mit '.' beginnen, werden *nicht* ausgegeben. Diese gibt ls sonst standardmäßig mit aus, wenn Sie unter root arbeiten.
- F Dateiverzeichnisse markiert ls mit einem angehängten "/", ausführbare Dateien markiert ls mit einem angehängten "*".
- R Alle Unter-Dateiverzeichnisse und Dateien ab dem angegebenen Dateiverzeichnis ausgeben.

name Name einer Datei oder eines Dateiverzeichnisses.
Mehrere Namensangaben sortiert ls. Sind Dateiverzeichnisse darunter, gibt ls erst alle Dateinamen aus, dann die Dateiverzeichnisse mit ihrem Inhalt.

Standard (keine Angabe): das aktuelle Dateiverzeichnis.

Beispiele

1. Welche Dateinamen gibt ls aus?

ls alle Dateien und Dateiverzeichnisse (ohne Inhalt) des aktuellen Dateiverzeichnisses.

ls * alle Dateien und Dateiverzeichnisse (mit Inhalt) des aktuellen Dateiverzeichnisses.

ls -R alle Dateien und Dateiverzeichnisse des Teilbaums, der beim aktuellen Dateiverzeichnis beginnt. Mit anderen Worten: rekursiv den Inhalt aller Unter-Dateiverzeichnisse.

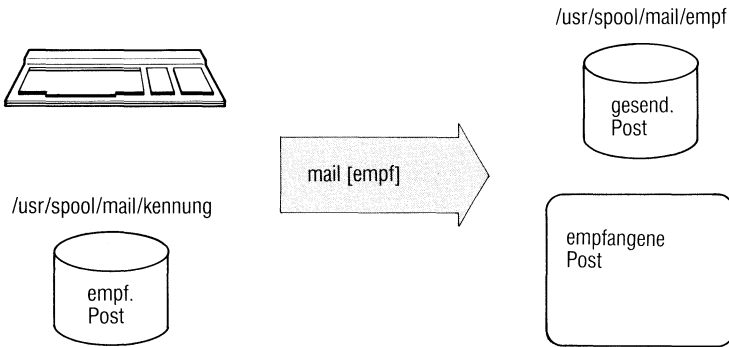
2. Indexnummern der Dateien datei1, datei2 und datei3 ausgeben:

```
$ ls -i datei1 datei2 datei3
1450 datei1
3171 datei2
4176 datei3
```

3. Sie wollen die Namen und Eigenschaften aller Dateiverzeichnisse und Unterdateiverzeichnisse auflisten, die im aktuellen Dateiverzeichnis liegen. Benutzen Sie "find", um die Namen der Dateiverzeichnisse zu finden.

```
find . -type d -exec ls -ld {} \;
```

Post senden und empfangen – send or receive mail



mail hat 2 Funktionen:

- mail steckt Nachrichten (Post) in den eigenen Briefkasten oder in den Briefkasten anderer Benutzer (Format 1). Der Briefkasten ist eine Datei.
- mail holt die Nachrichten aus dem Briefkasten (Format 2).
mail liest die neueste Post zuerst (last in - first out).

Format 1: Nachrichten senden

mail_empfänger...

empfänger Benutzerkennung des Empfängers der Nachricht.
mail liest die Nachricht von der Standard-Eingabe und bringt sie in den Briefkasten des Empfängers.

Die Eingabe der Nachricht beenden Sie wie üblich mit der Taste **END** oder durch die Eingabe von "." am Zeilenbeginn.

Sie können auch Post an sich selbst senden, wenn Sie die eigene Benutzerkennung angeben. Damit können Sie sich z.B. Notizen machen oder Meldungen aus Prozeduren senden.

Format 2: Nachrichten empfangen**mail**[**_**-schalter...]

schalter

- r mail liest die Nachrichten in der Reihenfolge, in der sie gesendet wurden (first in - first out).
- p ganzen Briefkasteninhalt ausgeben ohne Abfrage und mail beenden.
- q mail reagiert auf die Tasten **DEL** bzw. **CTRL** **Y**. mail wird beendet, der Briefkasteninhalt bleibt unverändert.

f**_**datei

mail liest die Nachrichten aus der angegebenen Datei.

keiner der Schalter r, p, q angegeben:

mail liest eine Zeile aus dem Briefkasten, gibt ein "?>" aus und erwartet eine Antwort, was mit der Nachricht geschehen soll:

Taste **↓**

nächste Nachricht ausgeben. Nach der letzten Nachricht beendet sich mail.

+ wie **↓**.

d Nachricht löschen und nächste Nachricht ausgeben. Nach der letzten Nachricht beendet sich mail.

p die zuletzt ausgegebene Nachricht wiederholen.

- die vorhergehende Nachricht wiederholen.

s**_**[datei ...]

Nachricht mit Kopfzeile in die angegebene Datei schreiben. Standardname ist mbox.

Im Briefkasten wird die Nachricht gelöscht.

w_[datei ...]

Nachricht ohne Kopfzeile in die angegebene Datei schreiben. Standardname ist mbox.
Im Briefkasten wird die Datei gelöscht.

m_[benutzerkennung]

Nachricht an den angegebenen Benutzer senden.
Standard ist die eigene Benutzerkennung.

mail beenden:

mail beendet sich, wenn die letzte Nachricht ausgegeben ist, oder Sie geben ein:

Taste `END`

Unbearbeitete Nachrichten aufheben und mail beenden.
Eine Nachricht ist "bearbeitet", wenn Sie mit d, s, w oder m geantwortet haben.

q wie Taste `END`

x Alle Nachrichten aufheben und mail beenden. Auch die mit d gelöschten Nachrichten bleiben erhalten.

Weitere Funktionen:

!kommando

kommando wird ausgeführt, anschließend können Sie mit mail weiterarbeiten.

? Liste der mail-Anweisungen ausgeben.

Hinweis

Jeder Benutzer besitzt einen eigenen Briefkasten. Die Datei hat den Namen der Benutzerkennung und steht im Dateiverzeichnis /usr/spool/mail. Die Zugriffsrechte sind: rw - r - - r - - (oktal 644). Sie können die Lese - erlaubnis wegnehmen. mail löscht leere Briefkästen nicht.

Bei Beginn einer Sitzung meldet SINIX, ob Sie Post haben, d.h. ob Nachrichten in Ihrem Briefkasten sind. root und admin haben den selben Briefkasten. Beachten Sie das z.B. beim Löschen von Post.

Beispiel

1. Senden einer Nachricht an den Benutzer richter:

```
$ mail richter
Die neuen Platten sind da.
Kannste abholen.
.
$
```

2. Sie wollen den Inhalt einer Datei als Nachricht senden:

```
$ cat datei | mail richter
```

3. Sie wollen sich selbst eine Nachricht senden (als Notiz), anschließend sehen Sie sich die eingegangene Post an.

```
$ mail artmann
Die mail im Kasten ist besser als der Knoten im Taschentuch.
```

```
.
$ mail
Von artmann am Mi 13.Jun.1984, 08:46:25
Die mail im Kasten ist besser als der Knoten im Taschentuch.
```

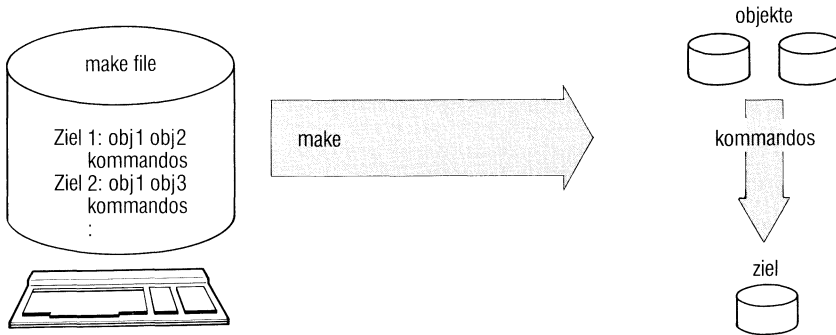
```
? +
Von richter am Di 12.Jun.1984, 14:12:58
Wenn die neuen Platten da sind, sag mir Bescheid.
```

```
? d
$
```

Die erste ausgegebene Nachricht soll noch aufgehoben, die zweite gelöscht werden. Da dies die letzte vorhandene Nachricht war beendet sich mail.

> > > > xget, xsend, write

Gruppen von Dateien verwalten



make dient zur Pflege einer Gruppe von Dateien, die voneinander abhängen. Das sind im allgemeinen Programme eines größeren Programmsystems, können aber auch beliebige Dateien sein. In einer Kommandodatei, der "makefile", legen Sie fest, welche Kommandos jeweils ablaufen sollen, wenn sich bestimmte Dateien geändert haben.

make[_f_name][_schalter...][_ziel...]

-f_name

Angabe einer Datei, die make als Eingabe benutzt. -f_datei können Sie mehrmals angeben.

Standard: make erwartet die Eingabe in einer Datei mit Namen makefile.

Ist "-" angegeben, liest make von der Standard-Eingabe.

schalter

- i** wirkt wie der Eintrag ".IGNORE" in makefile.
- k** Endet ein Kommando in makefile mit Ende-Status ! 0, bricht make nur die Bearbeitung des davon abhängigen Ziels ab. Weitere, von diesem Kommando nicht abhängige Ziele bearbeitet make.
- n** make listet alle Kommandos auf, die ausgeführt würden, führt sie aber nicht aus.

t	make listet nur die Ziele auf, die make bearbeiten würde, führt aber die Kommandos nicht aus. make führt für jedes bearbeitete Ziel das Kommando touch aus.
r	wirkt wie der Eintrag des Zieles .SUFFIXES: mit einer leeren Liste am Anfang der "makefile". r setzt also diese Standardeinstellung außer Kraft.
s	Protokollierung auf Standard-Ausgabe unterdrücken. Normalerweise meldet make jedes ausgeführte Kommando.
q	make überprüft nur, ob die Zieldatei auf dem neuesten Stand ist. Wenn ja, liefert make den Ende-Status 0, sonst $\neq 0$.
d	Namen der betroffenen Dateien und deren Zeitpunkt der letzten Änderung ausgeben.
p	make gibt alle Makrodefinitionen und Zielbeschreibungen aus: die Voreinstellungen und die in "makefile" definierten.
ziel	in "makefile" definiertes Ziel, das make bearbeiten soll. make bearbeitet auch alle Ziele, von denen das angegebene abhängt. <i>Standard (keine Angabe):</i> make bearbeitet das erste Ziel.

Definieren der "makefile"

In der "makefile" legen Sie fest:

- Ziele: Das sind Dateien, die Sie aktualisieren wollen, wenn sich eine der Dateien geändert hat, von denen das Ziel abhängt.
- Objekte: Das sind die Dateien, von denen das Ziel abhängt.
- Kommandos: Diese legen fest, wie aus den Objekten das Ziel zu erzeugen ist.

Dazu ein einfaches Beispiel:

Ein Programm prog wird aus zwei Modulen teil1.o und teil2.o gebunden. Zu diesen gehören die Quellprogramme teil1.c und teil2.c. Ändern Sie nun

eines der beiden Quellprogramme, dann muß dieses neu übersetzt und prog neu gebunden werden. Das formulieren Sie in der "makefile" so:

```
prog: teil1.o teil2.o
    cc teil1.o teil2.o -lm -o prog
teil1.o: teil1.c
    cc -c teil1.c
teil2.o: teil2.c
    cc -c teil2.c
```

Drei Ziele sind definiert: prog, teil1.o und teil2.o. Das Ziel prog hängt ab von teil1.o und teil2.o, diese sind wiederum Ziele, die abhängen von teil1.c und teil2.c.

Rufen Sie nun make auf, dann prüft make für Ziele und Objekte den Zeitpunkt der letzten Änderung. Ist eines der Objekte neueren Datums als das Ziel, führt make die darunter definierten Kommandos aus. In unserem Beispiel bedeutet das: wurde teil2.c geändert, muß teil2.c übersetzt werden, um teil2.o zu erzeugen. Anschließend muß prog aus teil1.o und teil2.o gebunden werden.

Syntax der makefile

ziel1: objekt ...

☞ kommando

·
·
·

ziel2:

·
·
·

Jede Kommandozeile muß mit einem Tabulatorzeichen beginnen! Das ist die Taste ☞ (Tabulator rechts). Der ced akzeptiert diese Taste nicht, wenn Sie jedoch mindestens 7 Leerzeichen am Zeilenanfang stehen haben, macht ced automatisch immer ein TAB daraus.

Reihenfolge der Ziele

make beginnt bei dem im Aufruf angegebenen Ziel, bzw. beim ersten Ziel, wenn nichts angegeben ist. Sind unter den Objekten dieses Zieles weitere Ziele, sucht make diese auf usw.

Die Reihenfolge ist somit durch die Abhängigkeiten von Zielen und Objekten bestimmt, gleichgültig, in welcher Reihenfolge Sie die Ziele definiert haben. make bearbeitet alle Ziele, von denen das angegebene Ziel abhängt.

Makro-Definitionen

Wiederkehrende Zeichenfolgen in der "makefile" können Sie ähnlich wie Variablen definieren (z.B. eine Menge von Objekten).

makro₁ = ₂zeichenfolge

Auf die Zeichenfolge beziehen Sie sich mit der Angabe:

\$(makro)

Steht für "makro" nur ein Buchstabe, können die Klammern entfallen.

Dazu ein Beispiel:

```
objs = teil1.o teil2.o
prog: $(objs)
      cc $(objs) -o prog
.
.
```

Für \$(objs) wird jedesmal die definierte Zeichenfolge eingesetzt. Damit läßt sich Schreibaarbeit sparen.

Kommentare

Zeichenfolgen, die mit "#" beginnen, werden als Kommentar gelesen. Ein Kommentar endet mit der Zeile.

Beispiel

```
prog: $(objs)
      cc $(objs) -o prog      # prog binden
.
.
```

Protokollierung unterdrücken:

Normalerweise protokolliert make jedes ausgeführte Kommando auf der Standard-Ausgabe. Dies können Sie unterdrücken:

- für einzelne Kommandos durch das Zeichen "@" vor dem Kommando: @kommando,

- für alle Kommandos durch Angabe des Zieles ".SILENT" irgendwo in makefile.

```
.SILENT:  
:  
:
```

Schalter s hat dieselbe Wirkung wie .SILENT.

Kommandos mit Ende-Status $\neq 0$

Liefert ein Kommando einen Ende-Status $\neq 0$, dann bricht make normalerweise die Verarbeitung ab. Dies können Sie verhindern:

- für ein Kommando durch das Zeichen "-" vor dem Kommando: -kommando,
- für das jeweils betroffene Ziel durch den Schalter k,
- für alle Kommandos durch Angabe des speziellen Zieles ".IGNORE" irgendwo in der "makefile". Dieselbe Wirkung hat Schalter i.

Vordefinierte Regeln und Makros

make kennt eine Reihe von Regeln und Makros, die gelten, wenn Sie nichts anderes definieren. Zum Beispiel nimmt make an, daß ein Ziel datei.o aus Objekten datei.c durch Übersetzen mit cc zu erzeugen ist. Das anfangs zitierte Beispiel könnten Sie abgekürzt so schreiben:

```
prog: teil1.o teil2.o  
    cc teil1.o teil2.o -lm -o prog
```

Welche Makros und Regeln definiert sind, erfahren Sie durch "make -p".

Nachsilben (Suffixes)

Unter dem speziellen Ziel ".SUFFIXES" ist eine Folge von Nachsilben definiert, nach denen make standardmäßig Abhängigkeiten bildet. Vordefiniert ist:

```
.SUFFIXES: .out.o.c.e.r.f.y.l.s
```

Ist z.B. ein Ziel wie folgt definiert

```
prog: teil1.o teil2.o
```

dann genügt es, zu schreiben

```
.o.c: ;kommando ...
```

make sucht dann nach Objekten teil1.c und teil2.c, um die Ziele teil1.o und teil2.o zu erzeugen.

In den Kommandoaufrufen können Sie sich auf gefundene Objekte bzw. Ziele wie folgt beziehen:

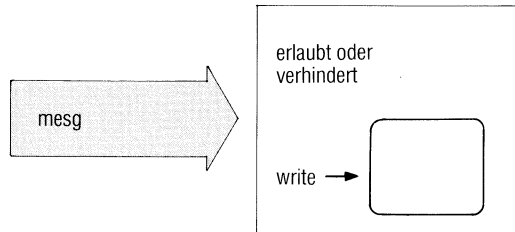
- \$* ist der Name des Zieles ohne Suffix,
- \$@ ist der Name des Zieles einschließlich Suffix,
- \$< ist die Liste aller Objekte,
- \$? ist die Liste der Objekte, die nicht aktuell sind.

Hinweis

- Pro Kommandozeile wird ein eigener Prozeß erzeugt. Eingebaute Kommandos, z.B. `cd`, wirken daher nur für die Zeile, in der sie stehen. Mehrere Kommandos in einer Zeile können Sie mit ";" trennen.
- Eine Fortsetzungszeile erwartet make, wenn eine Zeile mit "\" endet.

> > > > Shell (Abschnitte 3.6 bis 3.8), touch

Ausgabe von Meldungen verhindern oder erlauben



`mesg` verhindert, daß Meldungen anderer Benutzer auf Ihrem Bildschirm ausgegeben werden (siehe auch `write`).

`mesg[_n][_j]`

`n` verhindert Meldungen, die Zugriffsrechte auf `/dev/tty*` sind

`j` erlaubt Meldungen, die Zugriffsrechte auf `/dev/tty*` sind

kein Operand
`mesg` gibt den aktuellen Stand aus.

Hinweis

- Meldungen des Systemverwalters (mit `/etc/wall`) kann `mesg` nicht verhindern.
- Bei Einplatzsystemen ist das Kommando sinnlos, da kein anderer Benutzer gleichzeitig arbeiten kann.

Ende-Status:

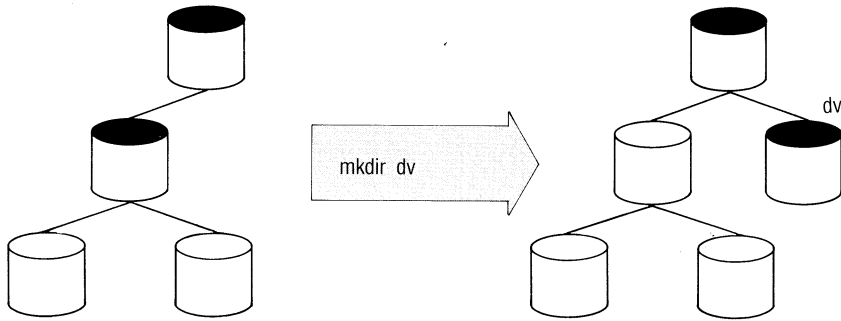
0 Meldungen sind erlaubt
1 Meldungen werden verhindert
255 bei Fehler

Beispiel

```
$ mesg n  
$ mesg  
Meldungen an Ihr Datensichtgerät sind nicht erlaubt.  
$
```

```
>>>> write
```

Dateiverzeichnis einrichten – make a directory



`mkdir` richtet ein neues Dateiverzeichnis ein.

`mkdir` `dateiverzeichnis...`

`dateiverzeichnis`

Name des neuen Dateiverzeichnisses.

Wenn Sie keinen Pfadnamen angeben, trägt `mkdir` den Namen im aktuellen Dateiverzeichnis ein, z.B.:

`mkdir subdir` richtet das Dateiverzeichnis `subdir` als Unterverzeichnis des aktuellen Dateiverzeichnisses ein.

Sie können auch einen vollständigen Pfadnamen angeben, z.B.: `mkdir /usr/adam/dateien` oder `mkdir listen/beispiele`

`/user/adam` bzw. `listen` müssen vorhandene Dateiverzeichnisse sein. Darin trägt `mkdir` den Namen ein.

`mkdir` trägt im neuen Dateiverzeichnis folgende Verweise ein:

”.” für das Dateiverzeichnis selbst und

”..” für das übergeordnete Dateiverzeichnis

Hinweis

- Ein neues Dateiverzeichnis hat die Zugriffsrechte rwx rwx r-x (oktal 775).
- mkdir setzt Schreiberlaubnis für das Dateiverzeichnis voraus, in das das neue eingetragen werden soll.
- Ein Dateiverzeichnis löschen Sie mit dem Kommando rmdir.

Endestatus:

0 wenn alle Dateiverzeichnisse erzeugt wurden
1 bei Fehler

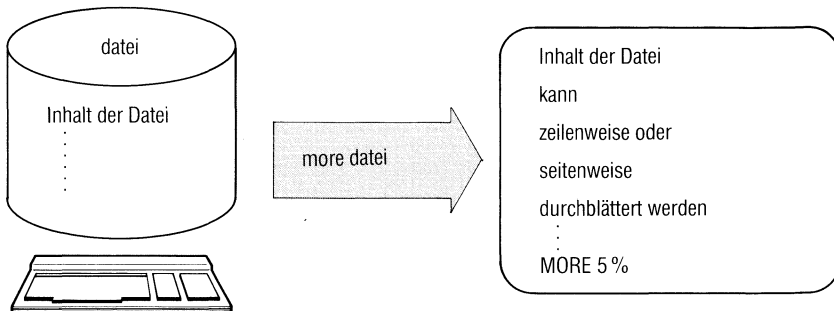
Beispiel

Im Dateiverzeichnis /usr/art/programme sind die neuen Dateiverzeichnisse test und demo anzulegen.

```
$ pwd
/usr/art
$ mkdir test demo
$ ls -l
drwxrwxr-x 2 art          32 May 25 09:40 demo
drwxrwxr-x 2 art          32 May 25 09:40 test
```

> > > > rmdir

Bildschirmausgabe steuern



Die Kommandos `more` und `page` erlauben das interaktiv gesteuerte Durchblättern von Dateien am Datensichtgerät. Während `more` das Blättern durch Hochschieben der Zeilen realisiert, löscht `page` jeweils den Bildschirm bevor eine neue Seite ausgegeben wird. Die Beschreibung von `more` ist identisch mit der Beschreibung von `page`.

more[₋schalter][₋ + zeilennr][₋ + /muster]₋dateiname...

schalter

kein Schalter

Wird `more` ohne Schalter aufgerufen, unterbricht das Kommando die Ausgabe nach jedem Bildschirm. Mit den untenstehenden Kommandos kann dann die weitere Ausgabe gesteuert werden.

d `more` gibt nach der Ausgabe eines Schirms die Meldung aus: "Leertaste, um weiterzublättern, DEL, um abzubrechen!"

f `more` zählt eine überlange Zeile (> 80), die auf dem Bildschirm mehrere Zeilen benötigt, als eine Zeile.

Standard (keine Angabe): `more` zählt die Zeilen auf dem Bildschirm.

l `more` ignoriert Formularvorschubzeichen (`CTRL` `L`).

Standard (keine Angabe): `more` unterbricht bei Formularvorschub (`CTRL` `L`) die Ausgabe der Datei.

-
- n more interpretiert n als die Anzahl der Zeilen auf dem Bildschirm.
Standard (keine Angabe): more liest die Bildschirmzeilenzahl aus der Datei /etc/termcap.
- + zeilenr more beginnt die Ausgabe der Datei mit der Zeile zeilenr.
- + /muster more beginnt die Ausgabe der Datei zwei Zeilen vor der ersten Zeile, die eine zu dem regulären Ausdruck "muster" passende Zeichenfolge enthält.
Ansonsten wird die Meldung "Muster nicht gefunden" ausgegeben. (bzgl. regulärer Ausdrücke siehe ed oder Tabelle im Anhang).
- dateiname Name der Datei, die ausgegeben werden soll.
Geben Sie mehrere Dateien an, so wird vor jeder Datei eine Kopfzeile ausgegeben, die den Namen der Datei enthält.
Liest more von einer Datei und nicht von einer Pipeline, zeigt more am Ende eines Bildschirms den schon angezeigten Teil der Datei in Prozent der Gesamtlänge an.

Kommandos

Wenn more die Ausgabe am Ende eines Bildschirms unterbrochen hat, können Sie die weitere Ausgabe von more durch folgende Kommandos steuern:

- i Leertaste
more gibt i weitere Zeilen aus.
Standard (keine Angabe): more gibt den nächsten Bildschirm aus.
- i more gibt i weitere Zeilen aus.
Standard (keine Angabe): more gibt die nächste Zeile aus.
- id oder i
more gibt i weitere Zeilen aus und setzt die Blätter-Größe auf i.
Standard (keine Angabe): more gibt die nächsten 11 Zeilen aus.
- iz
more gibt den nächsten Bildschirm aus und setzt die Bildschirmgröße auf i.
- is
more überspringt i Zeilen und gibt den dann folgenden Bildschirm aus.
- if
more überspringt i Bildschirme und gibt den dann folgenden Bildschirm aus.
- q oder Q oder :q oder :Q
beendet more.
- =
more gibt die aktuelle Zeilennummer aus.
- h
HELP-Taste: Anzeige aller more-Kommandos mit kurzer Funktionsbeschreibung in deutscher Sprache.

c Aufruf des Editors `ced` mit der aktuellen Datei und der aktuellen Zeile als Parameter. Aktuelle Zeile ist die letzte auf dem Bildschirm.

i/muster

`more` sucht nach dem *i*-ten Auftreten des regulären Ausdrucks 'muster'. Tritt das Muster weniger als *i* mal auf und liest `more` von einer Datei und nicht von einer Pipeline, bleibt die Position in der Datei unverändert. Sonst wird ein neuer Bildschirm ausgegeben, der zwei Zeilen vor der Zeile beginnt, in der der reguläre Ausdruck das *i*-te Mal auftritt.

in

`more` sucht nach dem *i*-ten Auftreten des zuletzt eingegebenen regulären Ausdrucks.

'

`more` geht zu dem Punkt, an dem die letzte Suche begann. Alle folgenden ' - Eingaben werden ignoriert. Falls noch keine Suchfunktion aufgerufen wurde, geht `more` an den Anfang der Datei.

!kommando

`more` startet eine neue Shell, die das Kommando ausführen soll. Bei der Formulierung des Kommandos stehen Ihnen zwei Variablen zur Verfügung:

`%` wird zum Namen der aktuellen Datei expandiert

`!` wird zum zuletzt eingegebenen Shellkommando expandiert;

Wollen Sie ein `%` oder ein `!` explizit ins Kommando einfügen, müssen Sie das Zeichen durch ein vorangestelltes `\` entwerten.

i:n

`more` springt zur *i*-ten Datei der in der Kommandozeile angegebenen Dateien. Sind weniger als *i* Dateien angegeben, springt `more` zur letzten. Ist kein *i* angegeben, springt `more` zur nächsten; befinden Sie sich in der letzten, wird `more` beendet.

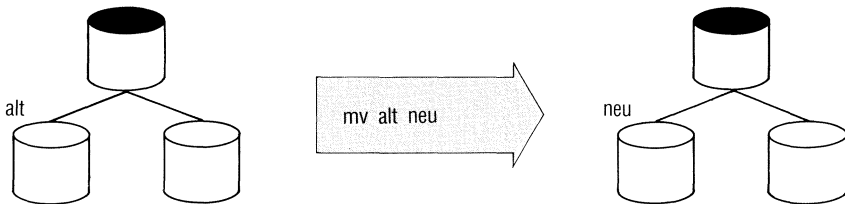
- i:p** more springt zur i-ten vorhergehenden Datei der in der Kommandozeile angegebenen. Sind weniger als i Dateien angegeben worden, springt more zur ersten angegebenen Datei. Das Kommando funktioniert nur, wenn nicht von einer Pipeline gelesen wird.
- :f** more gibt den Namen der aktuellen Datei und die aktuelle Zeilennummer aus.
- .** more wiederholt das vorhergegangene Kommando.

Hinweis

- more und page arbeiten im cbreak-Modus d.h., sobald eines der oben genannten Kommandos vollständig angegeben ist, erfolgt eine Reaktion, ohne daß die Eingabe mit abgeschlossen wird.
- more und page arbeiten im noecho-Modus d.h., die Kommandoeingaben - ausgenommen Shellkommandos und reguläre Ausdrücke - erscheinen nicht am Bildschirm.

> > > > page

Dateien umbenennen oder übertragen – move or rename files



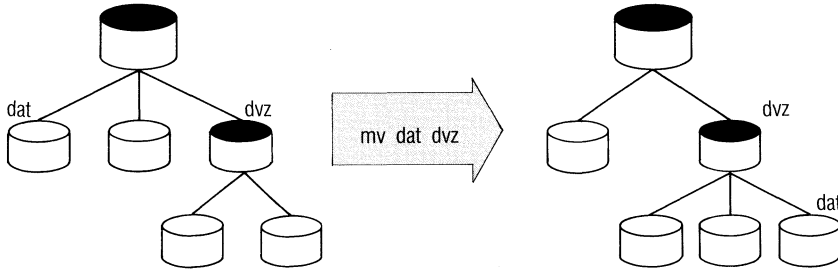
mv hat 2 Funktionen:

- mv gibt vorhandenen Dateien oder Dateiverzeichnissen einen neuen Namen (Format1) oder
- mv überträgt die Namen von Dateien in ein anderes Dateiverzeichnis (Format2).

Format1: Dateien oder Dateiverzeichnisse umbenennen

mv_namealt_nameneu

namealt	Name einer Datei oder eines Dateiverzeichnisses, das Sie umbenennen wollen.
nameneu	neuer Name für die Datei oder das Dateiverzeichnis. Achtung! Ist dieser Name bereits vorhanden, löscht mv den Eintrag! Falls Sie jedoch keine Schreiberlaubnis besitzen, meldet mv: Die Datei name existiert.



Format2: Dateien in ein anderes Dateiverzeichnis übertragen

`mv name... dateiverzeichnis`

name Name einer Datei. mv überträgt alle angegebenen Namen nach dateiverzeichnis. Im ursprünglichen Dateiverzeichnis löscht mv die Namen. Dateiverzeichnisse kann man nicht übertragen.

dateiverzeichnis Dateiverzeichnis, in das die Namen einzutragen sind. Sie müssen Schreiberlaubnis für dieses Dateiverzeichnis haben.

Sind in diesem Dateiverzeichnis bereits gleichnamige Einträge, löscht mv diese (siehe Format 1).

Hinweis

mv sollten Sie nicht mit cp bzw. copy verwechseln. cp kopiert Dateien, d.h. die Datei ist dann physisch zweimal vorhanden und für jede gibt es einen Indexeintrag mit Zugriffsrechten usw.

mv erzeugt keine Kopie, sondern überträgt nur einen Eintrag aus einem Dateiverzeichnis in ein anderes. Die Datei bleibt dieselbe, der alte Eintrag wird gelöscht. Indexeintrag und Indexnummer der Datei ändern sich nicht.

Beispiel

1. Die Datei "w" soll umbenannt werden in "cassiopeia" und gleichzeitig vom aktuellen Dateiverzeichnis ins Dateiverzeichnis "sterne" übertragen werden.

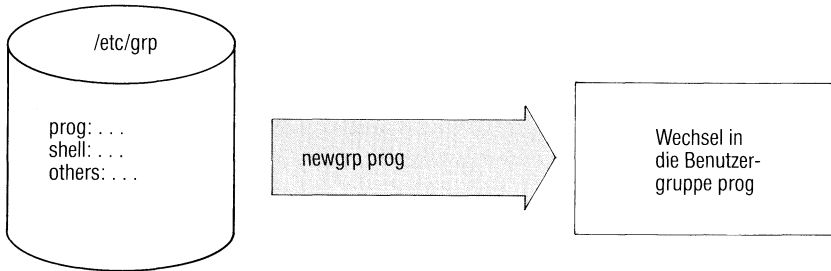
mv w sterne/cassiopeia

2. Die Dateien "loewe", "steinbock" und "skorpion" sind ins Dateiverzeichnis "tierkreis" zu übertragen. Sie sollen ihre Namen behalten.

mv loewe steinbock skorpion tierkreis

> > > > cp, chmod, copy, ln

Benutzergruppe wechseln



Mit `newgrp` wechseln Sie in eine andere Benutzergruppe. Dies wirkt sich aus auf:

- Ihre Zugriffsrechte für bestehende Dateien und
- die Gruppenidentifikation von Dateien, die Sie neu anlegen.

Sie können nur in eine Benutzergruppe wechseln, wenn Sie in der Datei `/etc/group` als berechtigt eingetragen sind (siehe Abschnitt 5.10).

`newgrp`_gruppe

gruppe Ein Gruppenname, der in der Datei `/etc/group` festgelegt ist (nicht die Gruppennummer).

Hinweis

- Wenn zu dem neuen Gruppennamen ein Kennwort gehört, fordert `newgrp` dieses an.
- Die Shell führt das Kommando `newgrp` direkt aus, überschreibt aber den alten Prozeß, d.h. `newgrp` wirkt bis zum Ende der Sitzung oder bis Sie erneut ein `newgrp`-Kommando geben.

Ende-Status: immer 0.

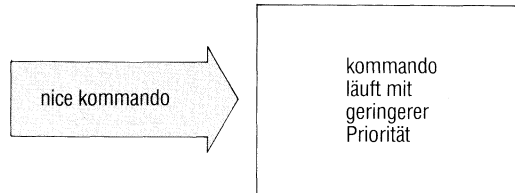
Beispiel

```
$ newgrp consul  
$
```

Der Benutzer wechselt in die Gruppe mit der Gruppenidentifikation consul.

> > > > login, su

Priorität von Kommandos ändern



nice ändert die Priorität, mit der ein Kommando in SINIX ablaufen soll. Als Normalbenutzer können Sie die Priorität nur verringern. Der Systemverwalter kann die Priorität auch erhöhen.

nice[₋zahl]₋kommando

zahl Zahl von 1 bis 19. Um diesen Zahlenwert erhöhen Sie die Prioritätsangabe und verringern damit die Priorität, mit der das Kommando abläuft (siehe NICE beim Kommando ps).

Hoher Zahlenwert = niedrige Priorität,
niedriger Zahlenwert = hohe Priorität.

Eine negative Zahlenangabe erhöht die Priorität, z.B. nice -- 10 ... (nur Systemverwalter).

Standard (keine Angabe): 10

kommando kommando läuft mit geänderter Priorität ab. Das gilt auch für alle weiteren Prozesse, die dieses Kommando erzeugt.

Hinweis

Normalerweise läuft ein Kommando mit der Priorität 20 ab. Die niedrigste Priorität ist 39, die höchste Priorität ist 0.

Ende-Status: der Ende-Status des ausgeführten Kommandos.

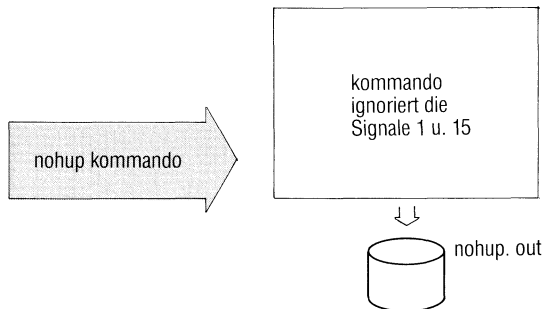
Beispiel

Die Prozedur "auftrag" soll im Hintergrund und mit Priorität 35 ablaufen:

nice -15 auftrag &

> > > > nohup, ps

Signale ignorieren



`nohup` führt ein anderes beliebiges Kommando so aus, daß es auf die Signale `HANGUP` (1) und `TERMINATE` (15) nicht reagiert. Zugleich läuft das Kommando mit niederer Priorität ab (Wert um 5 erhöht, siehe `nice`).

`nohup`_kommando

`kommando` `kommando` wird ausgeführt. Es reagiert nicht auf die Signale 1 und 15 und läuft mit niederer Priorität ab.

Hinweis

`HANGUP` (Signal 1, Abschalten der Datensichtstation) wird bei `SINIX` nicht verwendet. Das Signal `TERMINATE` (15) beendet den laufenden Prozeß. Zu Signalen siehe auch `kill`, `trap` (Abschnitt 3.8.14) und C-Entwicklungssystem.

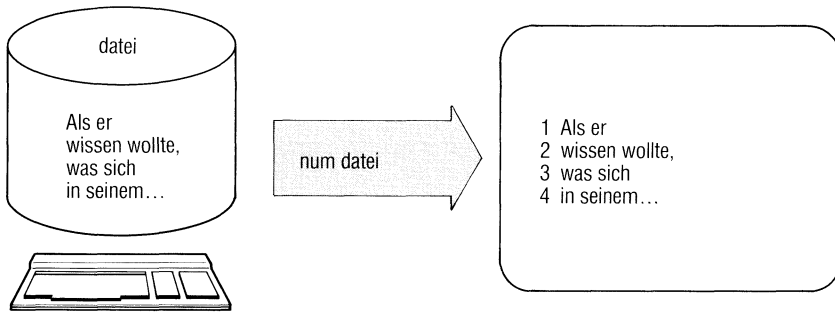
`nohup` speichert die Ausgabe des Kommandos standardmäßig in der Datei `nohup.out` ab, sie kann jedoch z.B. mit `>` umgeleitet werden.

Beispiel

```
$nohup date  
sending output to 'nohup.out'  
$cat nohup.out  
Tue Oct 4 11:18:31 MEZ 1983  
$
```

> > > kill, nice, C-Entwicklungssystem

Datei ausgeben mit Zeilennummern



num gibt eine Datei aus (wie cat) und schreibt dabei vor jede Ausgabezeile eine laufende Nummer.

num[_datei...]

datei Name der auszugebenden Datei.

Standard (keine Angabe): num liest von der Standard-Eingabe.

Die Numerierung beginnt mit 1. Die Zeilennummern bestehen aus höchstens 6 Ziffern und werden rechtsbündig zusammen mit zwei folgenden Leerzeichen vor den eigentlichen Text eingefügt, d.h. der Text beginnt auf Spalte 9.

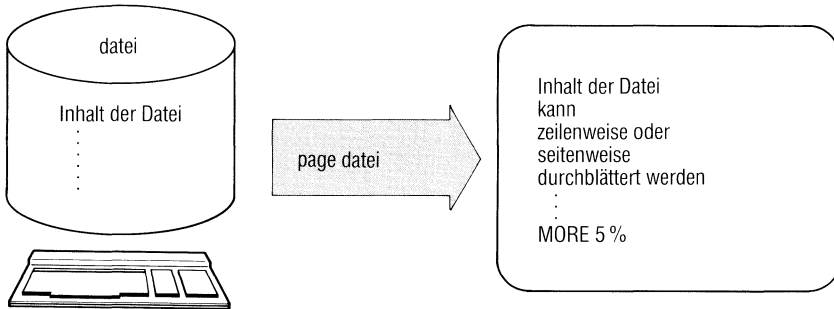
Beispiel

1. Dateien kunden.a und kunden.b durchnummeriert ausgeben:

```
$ num kunden.a kunden.b
   1 Abzwirz
   2 Angnirz
   3 Auflorx
   .
   .
 348 Badlitz
 349 Baefmurz
   .
   .
$
```

Haben Sie mehrere Dateien angegeben, zählt num fortlaufend durch.

Bildschirmausgabe steuern



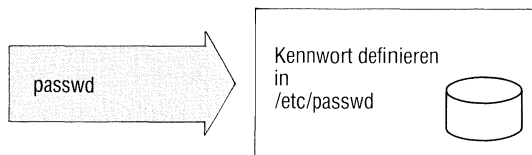
Die Kommandos `page` und `more` erlauben das interaktiv gesteuerte Durchblättern von Dateien am Datensichtgerät. Während `more` das Blättern durch Hochschieben der Zeilen realisiert, löscht `page` jeweils den Bildschirm bevor eine neue Seite ausgegeben wird.

page[`_-schalter`][`+ zeilennr`][`+ /muster`]`_dateiname...`

Die Beschreibung von `page` ist identisch mit der Beschreibung von `more`.

>>>> more

Kennwort für Benutzerkennung eintragen oder ändern



Das passwd-Kommando richtet ein neues Kennwort ein oder ändert ein bereits bestehendes.

passwd[₋benutzerkennung]

benutzerkennung

Angabe der Benutzerkennung, deren Kennwort geändert werden soll. Der Operand benutzername kann entfallen. Er ist nur für den Systemverwalter sinnvoll. Nach der Kommandoeingabe erfragt das Kommando das alte Kennwort und das neue Kennwort. Um Schreibfehler zu vermeiden, wird das neue Kennwort zweimal angefordert. Aus Gründen des Datenschutzes werden das alte und das neue Kennwort nicht am Bildschirm gezeigt.

Hinweis

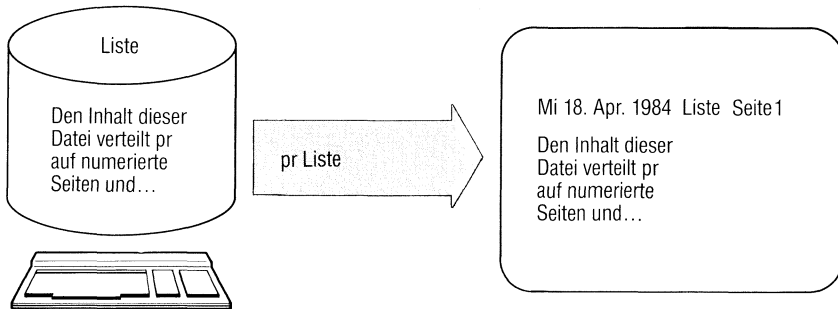
- Nur der Benutzer oder der Systemverwalter darf ein Kennwort einrichten oder ändern.
- Damit Ihr Kennwort nicht so leicht bekannt wird, sollten Sie es nicht zu kurz wählen (z.B. 5-8 Zeichen). Sie können auch Groß- und Kleinschreibweise mischen, sowie Ziffern und Sonderzeichen verwenden.
- Meldet passwd: 'Bitte geben Sie ein längeres Kennwort ein', dann geben Sie nicht auf. Beim dritten oder vierten Versuch, Ihr (kurzes) Kennwort einzugeben, werden Sie Erfolg haben.
- Ein vergessenes Kennwort muß der Systemverwalter aus der Datei /etc/passwd löschen. Sie können dann ein neues einrichten.

Beispiel

```
$ passwd
Änderung des Kennworts für anna
Altes Kennwort:
Neues Kennwort:
Neues Kennwort wiederholen:
$
```

> > > > crypt, login

Dateien aufbereiten zum Ausdrucken



pr bereitet Dateien zum Ausdrucken auf. Das Kommando

- verteilt die Ausgabe auf Seiten,
- gibt auf jeder Seite einen Seitenkopf mit Seitennummer aus,
- kann eine Datei mehrspaltig ausgeben,
- kann mehrere Dateien spaltenweise nebeneinander ausgeben.

pr[_Schalter...][_datei...]...

schalter

Schalter können Sie

- einmal für alle angegebenen Dateien setzen, z.B. pr -2 liste oder
- vor jeder Dateiangabe neu setzen, z.B. pr -3 liste -l30 tab1 tab2
liste wird 3-spaltig ausgegeben, tab1 und tab2 werden einspaltig, aber mit 30 Zeilen pro Seite ausgegeben.

kein Schalter angegeben

pr gibt pro Seite 72 Zeilen aus:

2 Leerzeilen

datum dateiname seite

2 Leerzeilen

60 Zeilen Text

7 Leerzeilen

Dabei bedeuten:

datum das Datum der letzten Änderung der
Datei

dateiname der Dateiname

seite eine Seitennummer in der Form: Seite n.

+ n ausgeben erst ab Seite n.

Standard: Seite 1

-l[n]

Anzahl der Zeilen pro Seite, einschließlich Kopfzeilen.

Standard: 72, wie oben beschrieben.

Ist n zu klein angegeben, nimmt pr den Standardwert von 72 Zeilen an.

Seitenkopf, Seitenende

Der Seitenkopf besteht aus 5 Zeilen (2 Leerzeilen, Kopfzeile 2 Leerzeilen), das Seitenende bilden 7 Leerzeilen.

-h zeichenfolge

zeichenfolge schreibt pr anstelle der Angabe dateiname in der Kopfzeile (maximal 48 Zeichen).

-t Den 5 Zeilen langen Seitenkopf, sowie die 7 Leerzeilen am Seitenende nicht ausgeben.

-T 5 Leerzeilen (Zeichen "neue Zeile") anstelle des 5-zeiligen Seitenkopfes ausgeben.

mehrspaltig ausgeben

- n** Datei in n Spalten ausgeben.
pr gibt Zeile für Zeile in die erste Spalte einer Seite aus, füllt dann die nächste Spalte usw., z.B.:

```

xxxxx      xxxx      xxxxxxxxxx  xxxxxxxx
xxxxx      xxxxxxxx  xxxxxxxxxx  xxxxxxxxxx
.
.

```

Die Spaltenbreite berechnet pr aus der Zeilenbreite (Standard 72) dividiert durch n (siehe Schalter -w). Ist eine Zeile länger als die Spalte Zeichen besitzt, schneidet pr den Rest der Zeile ab. In der letzten Spalte gibt pr immer die ganze Zeile aus.

- w[n]** n ist die gesamte Zeilenbreite in Zeichen bei mehrspaltiger Ausgabe. Die Angabe wirkt sich nur auf die Spaltenbreite aus.

- s[trennzeichen]** pr trennt die Spalten mit dem angegebenen Trennzeichen pr -3 -s! liste gibt aus:

```

Adam! Emil! Lore
Berta! Emilio! Lorenz
.
.

```

Der Schalter -w wirkt nicht zusammen mit -s.

Standard (trennzeichen nicht angegeben): ein Leerzeichen.

- m** pr gibt die angegebenen Dateien nebeneinander aus: jede Datei bildet eine Spalte.
- datei** Dateiname der auszugebenden Datei.
Standard (keine Angabe): pr liest von der Standard-Eingabe.

Beispiele

1. Die Datei wortliste enthält ein Wort pro Zeile.
Sie soll in 4 Spalten ausgegeben werden, jede Spalte mit 20 Wörtern.
Es ist keine Überschrift auszugeben.

```
pr -l20 -t -4 wortliste
```

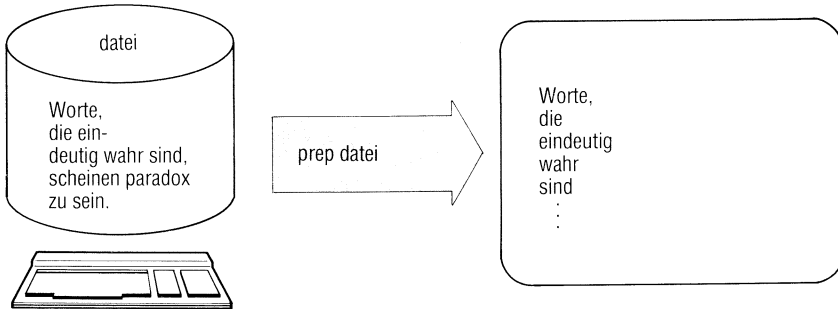
2. Sie wollen die Datei neudruck am Drucker ausgeben. Anstelle des Dateinamens in der Überschrift soll jedoch der Text "Neue Kunden" stehen.

```
pr -h 'Neue Kunden' neudruck | lpr
```

Die Pipeline `pr | lpr`, also `pr` ohne Schalter, wird durch das Kommando `print` realisiert.

```
> > > > cat, lpr, print
```


Text statistisch aufbereiten – prepare for statistics



prep bereitet Text auf für die statistische Verarbeitung. prep liest eine Datei und gibt jedes Wort in einer eigenen Zeile aus. Ein Wort ist dabei eine Folge von Buchstaben, die begrenzt wird von Leerzeichen oder von Sonderzeichen.

prep[_L-schalter][_Ldatei...]

schalter

- d zählt alle gelesenen Worte. prep schreibt die Wortnummer vor jedes Wort (max. 6-stellig).
- i Bestimmte Wörter nicht berücksichtigen. Die erste der angegebenen Dateien muß eine Liste von Wörtern enthalten: ein Wort pro Zeile. Kommt ein Wort dieser Liste in den aufzubereitenden Dateien vor, wird es nicht ausgegeben, zählt jedoch für die Wortnummer des Schalters d mit.
- o Nur bestimmte Wörter ausgeben. Die erste angegebene Datei muß eine Liste von Wörtern enthalten: ein Wort pro Zeile. Kommt ein Wort dieser Liste in den aufzubereitenden Dateien vor, wird es ausgegeben, sonst nicht. Der Wortzähler zählt aber alle Worte (Schalter d).
- p Interpunktionszeichen gibt prep als getrennte Ausgabezeilen aus. Die Interpunktionszeichen zählt der Wortzähler nicht mit (Schalter d).

datei Name der aufzubereitenden Datei. Haben Sie Schalter i oder o angegeben, muß die erste Datei die entsprechende Liste enthalten.

Standard: prep liest von der Standard-Eingabe.

Hinweis

- Wörter mit Bindestrich teilt prep in zwei Worte.
- Am Zeilenende mit "-" getrennte Worte zieht prep zu einem Wort zusammen.
- Hochkommas innerhalb und am Ende eines Wortes gehören zum Wort.
- Ziffernfolgen und Sonderzeichen berücksichtigt prep nicht.
- Sie können Schalter i und Schalter o nicht gleichzeitig angeben.
- Zeilen werden übersprungen, wenn sie mit . oder ' beginnen.
- Großbuchstaben werden generell in Kleinbuchstaben umgewandelt.

Beispiel

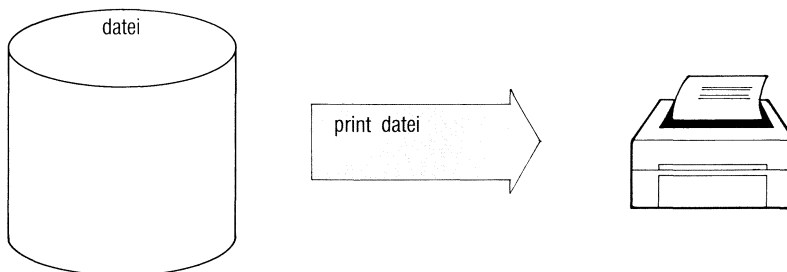
```
$ cat worte
Worte, die eindeutig wahr sind,
scheinen paradox zu sein.
```

```
          Lao Tse
```

```
$ prep -d worte
 1 worte
 2 die
 3 eindeutig
 4 wahr
 5 sind
 6 scheinen
 7 paradox
 8 zu
 9 sein
10 Lao
11 Tse
```

```
$
```

Dateien ausdrucken am Drucker – pr to the line printer



print druckt Dateien in einem Standardformat am Drucker aus. print prüft, ob die Datei existiert, nicht leer und kein Dateiverzeichnis ist und führt dann pr datei ... | lpr aus.

print_datei...

datei Name der auszudruckenden Datei.

print gibt die Dateien so aus, wie es der normalen Ausgabe von pr und lpr entspricht:

Jede Seite enthält folgende Kopfzeile:

datum dateiname seite

Dabei bedeuten:

datum das Datum der letzten Änderung der Datei

dateiname der Dateiname

seite eine Seitennummer in der Form: Seite n.

Hinweis

- print kann nicht am Ende einer Pipe stehen.
- Wird die Datei geändert oder gelöscht, bevor sie fertig ausgedruckt ist, führt das zu Fehlern.

Beispiel

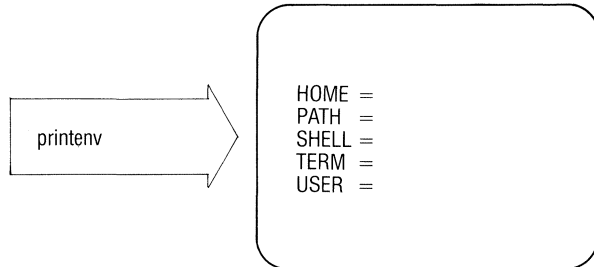
Alle Dateien des aktuellen Dateiverzeichnisses ausdrucken:

```
print *
```

Dateiverzeichnisse und leere Dateien übergeht print.

```
>>>> lpr, pr
```

Variablenwerte ausgeben – print environment



`printenv` gibt die Werte von Variablen aus, die in Sub-Shells zur Verfügung stehen (Shell-Umgebung).

`printenv[_variable]`

variable Name einer Variablen.

Standard (keine Angabe): alle Variablen mit Name und Wert ausdrucken, die in Sub-Shells verfügbar sind.

Ende-Status:

- 0 Die angegebene Variable ist in Sub-Shells verfügbar.
- 1 Die angegebene Variable ist nicht in Sub-Shells verfügbar.

Beispiele

1. Alle Variablen ausdrucken, die in Sub-Shells zur Verfügung stehen:

```
$printenv  
HOME=/usr/santi  
PATH=: /bin: /usr/bin  
SHELL=/bin/shell  
TERM=97801  
USER=santi
```

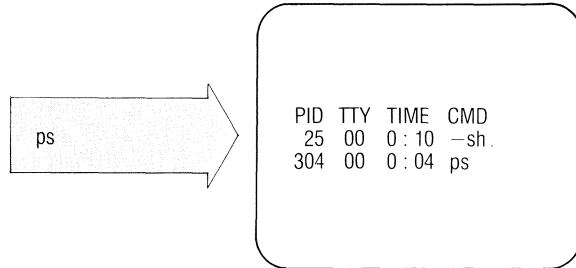
2. Exportierte Variable stehen in Sub-Shells zur Verfügung.

```
$ var1=loewe  
$ var2=tiger  
$ export var1  
$ printenv var1  
loewe  
$ printenv var2  
$ echo $?  
1  
$
```

Die Variable var2 wurde nicht exportiert, der Ende-Status ist 1.

>>>> Abschnitt 3.6.1

Prozeßdaten abfragen



ps informiert Sie über Prozesse:

- welche Prozesse sind mit welchen Datensichtstationen verbunden ?
- welche Prozesse sind nicht mit Datensichtstationen verbunden ?
- Informationen über diese Prozesse, wie Prozeßnummer, verbrauchte Rechenzeit usw.

ps[₋schalter][₋prozeßnummer...]

schalter

kein Schalter angegeben

ps gibt Information aus über Prozesse, die von Ihrer Datensichtstation aus gestartet wurden. Die Ausgaben bedeuten:

PID Prozeßnummer. Unter dieser Nummer kann der Prozeß mit kill beendet werden.

TTY Datensichtstation, die den Prozeß kontrolliert.

TIME Die aufsummierte Ausführungszeit des Prozesses in Minuten und Sekunden.

COMMAND

ist der Name des Kommandos.

e alle dem System bekannten Prozesse ausgeben.

- d alle Prozesse ausgeben, außer dem Prozeßgruppenführer.
- a nur Prozesse ausgeben, die von einer Datensichtstation kontrolliert werden, außer Prozeßgruppenführer.

l ps gibt aus:

F Prozeßart (Flags des Prozesses)

- 01 Prozeß im Hauptspeicher
- 02 Systemprozeß
- 04 im Hauptspeicher blockiert, z.B. für physikalische Ein-/Ausgabe
- 10 ausgelagert (swapped)
- 20 von einem anderen Prozeß kontrolliert (traced)
- 40 kontrolliert einen anderen Prozeß (tracer)

Diese Angaben sind oktal und können miteinander kombiniert sein (durch addieren).

S Prozeßstatus

- O Prozeß ist nicht vorhanden
- S Prozeß schläft
- W Prozeß wartet
- R Prozeß läuft
- I Prozeß läuft in einer internen Verwaltungsroutine (Zwischenzustand)
- Z Prozeß ist beendet
- T Prozeß ist angehalten

UID Benutzernummer des Prozeßeigentümers.

PID Prozeßnummer. Unter dieser Nummer kann z.B. der Prozeß mit kill beendet werden.

PPID Prozeßnummer des Vaterprozesses.

C Prozessorzeit für das Scheduling.

PRI Prozeßpriorität. Hohe Werte bedeuten eine niedrigere Ablaufpriorität.

NI Mit diesem Wert wird die Priorität berechnet.

ADDR	Hauptspeicheradresse des Prozesses in Vielfachen von 2K. Ist der Prozeß nicht im Hauptspeicher, enthält diese Spalte die Plattenadresse (swap-device) in Vielfachen von 1K.
SZ	Größe des Prozesses in Vielfachen von 2K. Das ist der Platz, der beim Beenden des Prozesses mindestens wieder freigegeben wird.
WCHAN	Hexadezimal verschlüsselte Kanalnummer, über die ein schlafender oder wartender Prozeß ein Ereignis erwartet. Wenn die Spalte leer ist, läuft der Prozeß. Die Angabe ist für Normalbenutzer belanglos.
TTY	Datensichtstation, die den Prozeß kontrolliert. Wenn der Prozeß von keiner bestimmten Datensichtstation kontrolliert wird, enthält diese Spalte ein Fragezeichen.
TIME	Die aufsummierte Ausführungszeit des Prozesses in Minuten und Sekunden.
CMD	Der Name des Kommandos.

f ps gibt aus:

UID	Benutzerkennung des Prozeßeigentümers.
PID	Prozeßnummer, wie bei Schalter l.
PPID	Prozeßnummer des Vaterprozesses, wie bei Schalter l.
C	Prozessorzeit für das Scheduling, wie bei Schalter l.
STIME	Startzeit des Prozesses
TTY	wie bei Schalter l.
TIME	wie bei Schalter l.
COMMAND	ist der vollständige Kommandoaufruf mit allen Argumenten. Kann ps diese Angaben nicht liefern, steht der Name des Kommandos in eckigen Klammern.

n_**l**namelist

Falls mehr als ein SINIX-Kern verwendet wird, kann hier die momentan aktive SINIX-Version anderen Namens angegeben werden.

s_**l**swapgerät

Anstelle von /dev/swap wird das angegebene Gerät als swapgerät angenommen. Dies ist dann nützlich, wenn man einen Speicherabzug des Systems untersuchen will (post mortem dump). Wird als swapgerät /dev/null angegeben, so wird der user-block auf null gesetzt.

c_**l**coredatei

ps entnimmt das Speicherabbild der Datei coredatei und nicht /dev/mem.

t_**l**datensichtstationen

nur Prozesse angeben, die von den angegebenen Datensichtstationen kontrolliert werden. Datensichtstationen können Sie in einer Liste wie folgt angeben:

tty01,tty02,... oder
'tty01_ltty02_l...'
'tty01,tty02,...'

p_**l**prozeßnummern

nur Prozesse mit der angegebenen Prozeßnummer ausgeben. Prozeßnummern können Sie in einer Liste angeben, wie bei Schalter t.

g_**l**prozeßgruppen

nur Prozesse der angegebenen Prozeßgruppen ausgeben. Prozeßgruppen können Sie in einer Liste angeben, wie bei Schalter t.

u_**l**benutzerkennungen

nur Prozesse der ausgegebenen Benutzerkennung ausgeben. Benutzerkennungen können Sie in einer Liste angeben, wie bei Schalter t. Anstelle der Benutzerkennung können Sie auch die Benutzernummer angeben.

Hinweis

- Da ps relativ langsam ist, können während der Laufzeit des Kommandos Veränderungen im Prozeßbild stattfinden. ps ist daher nur als Annäherung an die reale Situation zu verstehen.
- Angaben für "defunct"-Prozesse sind meistens irrelevant.
- Die Optionen n, s und c sind für den "normalen" SINIX-Benutzer ohne großen Wert; sie sind hier hauptsächlich der Vollständigkeit halber beschrieben.
- Die maximale Anzahl der Prozesse ist 80. Bei Prozessen mit großen Code- bzw. Datenbereichen können es auch weniger sein; ebenso wenn viele Prozesse das "sticky-bit" gesetzt haben.
- Die maximale Anzahl der Prozesse pro Benutzer ist 15 (ausser root und admin).
- Bei Einplatzsystemen gibt es nur eine Datensichtstation mit dem Namen console

Beispiele

1. Welche Prozesse laufen zur Zeit?

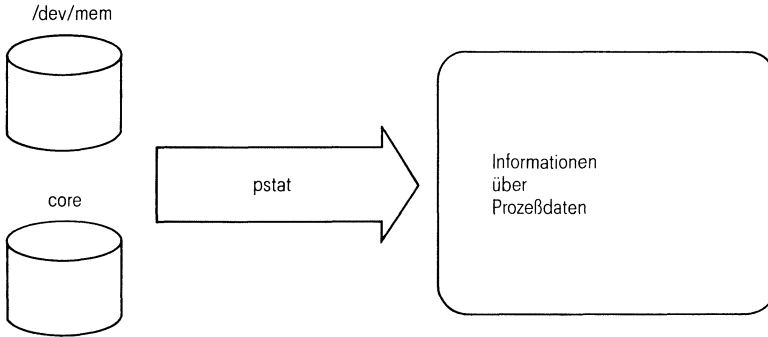
```
$ps -e
PID TTY  TIME  COMMAND
  0  ?  1460:47  swapper
  1  ?  0:04  init
531 00  0:07  sh
 15  ?  0:39  update
 17  ?  1:32  cron
492 02  0:03  getty
1140 00  0:04  server
1142 00  1:06  em
1214 00  0:00  sh
 89 02  0:01  qdaemon
1215 00  0:07  ps
1216 00  0:00  tee
```

2. Ausführliche Informationen ausgeben über Prozesse, die von der Datensichtstation tty02 kontrolliert werden.

```
$ps -l -t tty02
F S  UID  PID  PPID  C  PRI  NI  ADDR  SZ  WCHAN  TTY  TIME  CMD
1 S   0  492    1  0   28  20   55  14   cae2  02  0:03  getty
1 S   0   89    1  0   40  20   68  12   f800  02  0:01  qdaemon
```

> > > > kill, nice, pstat

Systeminformation ausgeben – print system facts



pstat liest System-Tabellen und gibt darin enthaltene Informationen aus:

- Indexeintrag-Tabelle
- Text-Tabelle
- Prozeß-Tabelle

pstat[₋schalter][₋datei]

schalter Mindestens ein Schalter ist anzugeben.

i Indexeintrag-Tabelle ausgeben. Die Einträge bedeuten:

LOC Die Hauptspeicheradresse dieses Tabelleneintrages

FLAGS Verschiedene Status-Variablen, die auf folgende Weise codiert sind:

L gesperrt (lock).

U die Aktualisierungszeit filsys muß korrigiert werden.

A die Zugriffszeitangabe muß korrigiert werden.

M hier ist ein Datei-System hinzugefügt.

W von einem anderen Prozeß gewünscht (L Schalter ist gesetzt).

	T	enthält eine Text-Datei.
	C	die geänderte Zeit muß korrigiert werden.
	CNT	Anzahl der für diesen Indexeintrag eröffneten Datei-Tabelleneinträge.
	DEV	Geräteklasse und Gerätenummer des Dateisystems, in dem sich dieser Indexeintrag befindet.
	INO	Indexnummer innerhalb des Gerätes.
	MODE	Zugriffsrechte (siehe chmod).
	NLK	Zahl der Verweise (link) auf diesen Indexeintrag.
	UID	Benutzer-Identifikation des Eigentümers.
	SIZ/ DEV	Zahl der Bytes in einer Benutzerdatei oder Geräteklasse und Gerätenummer bei einer Gerätedatei.
x		Texttabelle ausgeben. Die Einträge bedeuten:
	LOC	Die Hauptspeicheradresse dieses Tabelleneintrages.
	FLAGS	Verschiedene Status-Variablen, die auf folgende Weise codiert sind:
	T	ptrace ist in Kraft.
	W	der Text ist noch nicht auf das Auslagerungs-Gerät geschrieben.
	L	der Ladevorgang dauert an.
	K	gesperrt (lock).
	W	angefordert (L Anzeige ist gesetzt).
	DADDR	Plattenadresse innerhalb des Auslagerungsbereiches, in Vielfachen von 512 Bytes angegeben.
	CADDR	Hauptspeicheradresse, in Vielfachen von 64 Bytes angegeben.
	SIZE	Größe des Text-Segmentes, in Vielfachen von 64 Bytes angegeben.

	IPTR	Hauptspeicheradresse des zugehörigen Indexeintrags.
	CNT	Zahl der Prozesse, die dieses Text-Segment benutzen.
	CCNT	Anzahl der im Hauptspeicher befindlichen Prozesse, die dieses Text-Segment benutzen.
p		Prozeßtablette der aktiven Prozesse ausgeben. Die Einträge bedeuten:
	LOC	Die Hauptspeicheradresse dieses Tabelleneintrages
	S	Ablauf-Status, der in folgender Weise codiert ist: 0 kein Prozeß 1 wartet auf ein Ereignis 3 ablauffähig 4 wird gerade erstellt 5 wird beendet 6 stop unter trace (Ablaufverfolger)
	F	Status des Prozesses: 01 geladen 02 der Scheduler-Prozeß 04 gesperrt (lock) 010 ausgelagert 020 von einem Prozeß kontrolliert (traced) 040 kontrolliert einen Prozeß (tracer) 01000 durch lock gesperrt Die Angaben können kombiniert sein (durch Addieren).
	PRI	Priorität des Scheduling, siehe nice.
	SIGNAL	Empfangene Signale (die Signale 1-16 als bits 0-15 codiert).
	UID	Wirkliche Benutzer-Identifikation.
	TIM	Die residente Verweilzeit in Sekunden; Zeiten größer als 127 sind als 127 codiert.

CPU	Gewichtetes Integral der CPU-Zeit, für den Scheduler.
NI	Nice-Stufe, siehe nice.
PGRP	Prozeß-Nummer des Ursprunges einer Prozeß-Gruppe (dem Eröffner der steuernden Daten-sichtstation).
PID	Die Prozeßnummer
PPID	Die Prozeß-Identifikation des Vater-Prozesses.
ADDR	Wenn sich der Prozeß im Hauptspeicher befindet, so ist ADDR die physikalische Adresse der 'u-area' des Prozesses, in Vielfachen von 64 Bytes gemessen. Ist er ausgelagert, so ist ADDR die Position innerhalb des Auslagerungs-Bereiches, in Vielfachen von 512 Bytes gemessen.
SIZE	Größe des Prozeß-Image in Vielfachen von 64 Bytes.
WCHAN	Kanalnummer des "wait" eines wartenden Prozesses.
LINK	Verweis-Zeiger in die Tabelle der ablauffähigen Prozesse.
TEXTP	Wenn nur Text vorkommt, Zeiger auf die Stelle des Text-Tabelleneintrags.
CLKT	Verbleibende Restzeit für alarm, gemessen in Sekunden.
a	Prozeßtabelle aller Prozesse ausgeben, sonst wie Schalter p. Der Schalter a wirkt nur zusammen mit Schalter p.
u _L addr	Informationen über den Prozeß ausgeben, dessen Adresse Sie hier angeben (siehe oben: ADDR). Geben Sie 0 an, entnimmt pstat die Prozeßdaten der bei datei angegebenen Datei. Diese muß ein Prozeßabbild enthalten, wie es z.B. bei manchen Fehlern abgespeichert wird (Dateiname core).

- f Tabelle der geöffneten Dateien ausgeben. Die Einträge bedeuten:
- LOC Die Hauptspeicheradresse des Tabelleneintrages.
 - FLG Status-Variablen, die wie folgt abgekürzt sind:
 - R für Lesen eröffnet
 - W für Schreiben eröffnet
 - P Pipe
 - CNT Zahl der Prozesse, die diese eröffnete Datei kennen.
 - INO Die Stelle des Indexeintrag-Tabelleneintrages dieser Datei.
 - OFFS Die Distanz innerhalb der Datei, siehe lseek.
- datei Das Kommando erwartet die Tabelle, die es interpretieren soll, in der angegebenen Datei. Wenn der Operand `datei` nicht angegeben ist, sucht das Kommando die Tabellen in der Datei `/dev/mem`. Die erforderliche Symboltabelle entnimmt `pstat` der Datei `/xenix`.

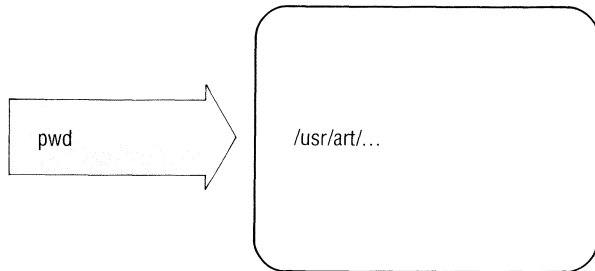
Beispiel

Sie möchten die Indexeintrag-Tabelle und die Prozeßtable ausgeben für einen Prozeß, dessen Abbild in der Datei `core` steht:

`pstat -ip -u 0 core`

>>>> ps

Pfadnamen des aktuellen Dateiverzeichnisses ausgeben path of working directory



`pwd` gibt den Pfadnamen des aktuellen Dateiverzeichnisses aus.

pwd

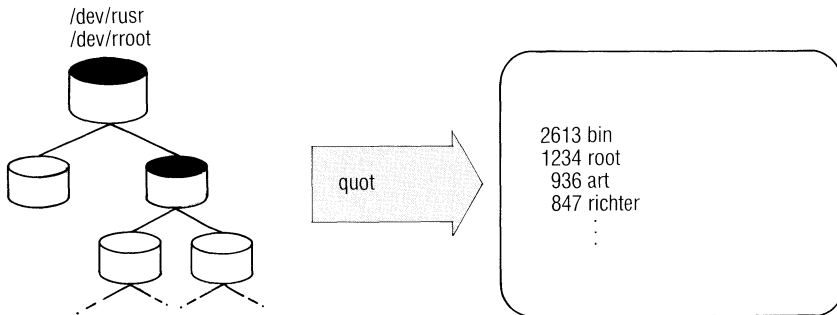
Beispiele

Welches ist das aktuelle Dateiverzeichnis? Dieses Dateiverzeichnis wollen Sie anschließend als Home-Dateiverzeichnis definieren.

```
$pwd
/usr/art/cobol/prg
HOME=`pwd`
$
```

```
>>>> cd
```

Dateisystem prüfen auf Belegung pro Benutzer



`quot` prüft für ein Dateisystem,

- wieviele Blöcke pro Benutzer belegt sind,
- wieviele Dateien bestimmter Größe existieren,
- die Anzahl der Dateien pro Benutzer.

quot_[_schalter]_[_dateisystem]

schalter Sie können nur einen der folgenden Schalter angeben.

kein Schalter angeben

Blockanzahl pro Benutzer (Eigentümer) ausgeben.

c `quot` gibt in 3 Spalten aus:

Spalte 1: Anzahl der belegten Blöcke
 Spalte 2: Anzahl der Dateien dieser Größe
 Spalte 3: Anzahl der Blöcke aller Dateien mit einer Blockanzahl wie in Spalte 1 oder kleiner.

f `quot` gibt in 3 Spalten aus:

Spalte 1: Anzahl der belegten Blöcke
 Spalte 2: Anzahl der Dateien
 Spalte 3: Eigentümer dieser Dateien

n quot erwartet von der Standard-Eingabe eine sortierte Liste von Indexnummern: eine Indexnummer pro Zeile, der Rest wird als Kommentar gelesen.
 quot gibt anstelle der Indexnummer jeweils den Eigentümer aus.

dateisystem

Name des Dateisystems, das Sie prüfen wollen, z.B. /dev/fl2 für ein Dateisystem auf Diskette (siehe Abschnitt 5.12).

Standard (keine Angabe): quot prüft das standardmäßig vorhandene Dateisystem /dev/rusr (siehe Abschnitt 5.7).

Beispiele

1. Blockanzahl pro Benutzer:

```
$ quot /dev/rroot
/dev/rroot:
2613 bin
1234 root
  169 richter
   23 blumann
```

2. Blockanzahl und Anzahl der Dateien pro Benutzer

```
$ quot -f /dev/rroot
/dev/rroot:
2613 145 bin
1234 166 root
  169   6 richter
   23   2 blumann
```

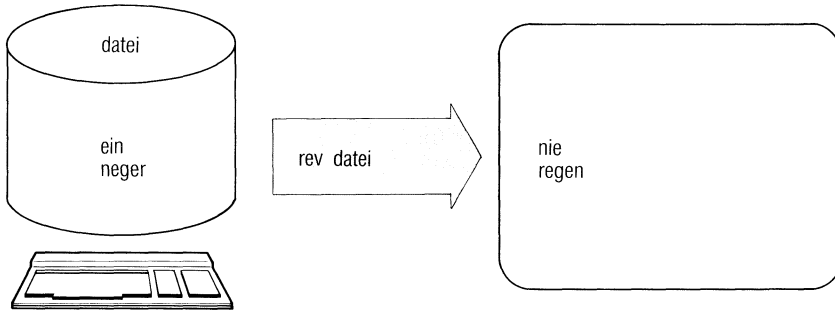
3. Sie wollen eine Liste erstellen, die für alle Dateien eines Dateisystems auf Diskette den Eigentümer ausgibt:

```
/etc/ncheck /dev/fl2 | sort +0n | quot -n /dev/fl2
```

/etc/ncheck ist ein Kommando des Systemverwalters!

> > > > du, ls

Reihenfolge von Zeichen umkehren



rev kehrt die Anordnung der Zeichen jeder Dateizeile um und gibt das Ergebnis an der Standard-Ausgabe aus. rev ist nur für Textdateien sinnvoll.

rev[_datei...]

datei Dateiname der zu lesenden Datei.

Standard (keine Angabe): rev liest von der Standard-Eingabe.

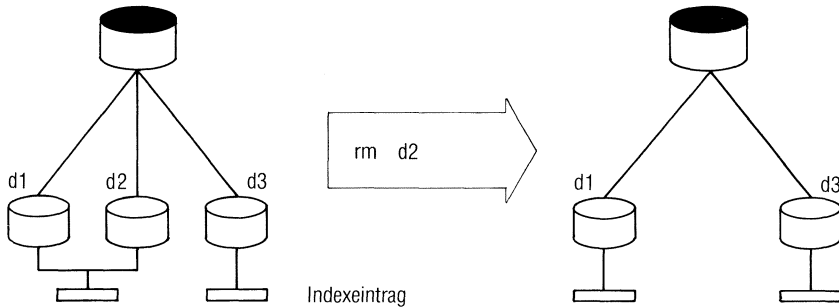
Hinweis

rev bearbeitet Zeilen mit maximal 255 Zeichen.

Beispiel

```
$ rev
einnegermitgazel
lezagtimregennie
END
$
```

Dateien löschen – remove files



rm löscht für eine oder mehrere Dateien den Eintrag im Dateiverzeichnis (Verweis). Sind noch weitere Verweise auf die Datei vorhanden, bleibt die Datei physisch vorhanden. Mit dem letzten Verweis löscht rm auch die Datei selbst.

rm[`_-`schalter...]`_`name...

schalter

kein Schalter angegeben

rm löscht Einträge nur, wenn Sie Schreiberlaubnis für das Dateiverzeichnis haben, in dem der Eintrag steht.

Haben Sie keine Schreiberlaubnis für die Datei, dann gibt rm die Zugriffsrechte aus (als Oktalzahl) und fragt ab, ob der Eintrag gelöscht werden soll.

Antwort: j Eintrag löschen,
 n Eintrag nicht löschen.

f rm löscht Einträge ohne Rückfrage, auch wenn Sie keine Schreiberlaubnis für die Datei haben (für das Dateiverzeichnis müssen Sie immer Schreiberlaubnis haben!).

- r** Ist bei name ein Dateiverzeichnis angegeben, löscht rm alle Einträge dieses Dateiverzeichnisses und das Dateiverzeichnis selbst. rm löscht dabei auch alle Unterverzeichnisse samt Inhalt. Kann rm einen Eintrag nicht löschen (Zugriffsrechte), bleiben alle übergeordneten Dateiverzeichnisse erhalten.
- i** rm fragt für jede angegebene Datei ab, ob der Eintrag gelöscht werden darf (siehe oben). i wirkt auch zusammen mit r.
- name** Name der Datei, für die der Eintrag zu löschen ist. Zusammen mit Schalter r ist der Name eines Dateiverzeichnisses anzugeben.

Beispiel

1. Löschen aller Dateien, die auf ".prog" enden mit Abfrage:

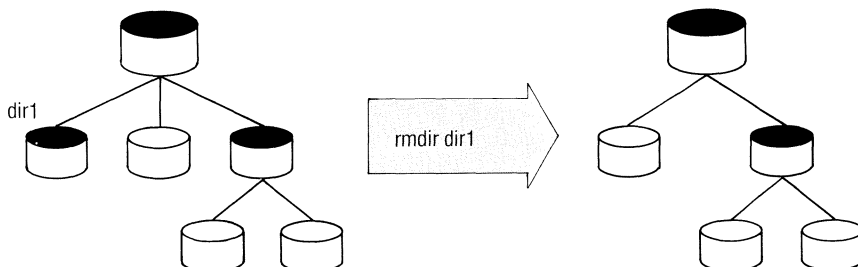
```
$ rm -i *.prog
ablauf.prog: j
code.prog: j
eingabe.prog: n
zufall.prog: n
$
```

2. Löschen des Dateiverzeichnisses "norm" mit allen Dateien und Unterverzeichnissen:

```
rm -r norm
```

> > > > rmdir

Dateiverzeichnisse löschen – remove directories



rmdir löscht ein oder mehrere leere Dateiverzeichnisse.
Dateiverzeichnisse mit Inhalt löschen Sie mit rm (Schalter r).

rmdir *dateiverzeichnis*...

dateiverzeichnis

Name des Dateiverzeichnisses, das Sie löschen wollen. Das Dateiverzeichnis muß leer sein. Sie brauchen Schreiberelaubnis für das übergeordnete Dateiverzeichnis.

Beispiel

Löschen der Dateiverzeichnisse dir1 und dir2

```
$ rmdir dir1 dir2
```

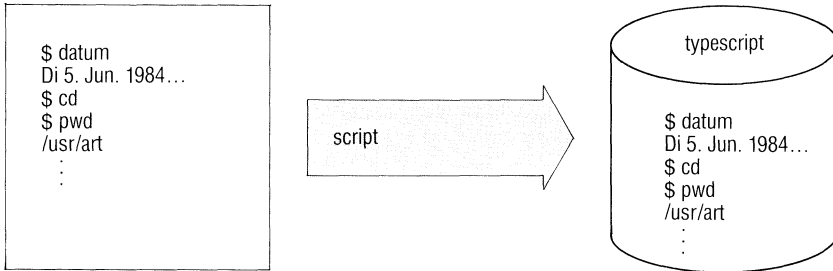
```
rmdir: Das Dateiverzeichnis dir2 ist nicht leer.
```

```
$
```

Dateiverzeichnis dir2 wurde nicht gelöscht, da es noch Dateien enthält. Sie können es einschließlich Dateien löschen mit "rm -r dir2".

>>>> rm

Sitzung protokollieren



`script` schreibt ein Protokoll Ihrer Sitzung in eine Datei.

`script`[`_-`schalter][`_-`datei]

schalter

a `script` fügt das Protokoll an die Protokolldatei an.

Standard: `script` löscht die Protokolldatei oder legt sie neu an, falls sie nicht vorhanden ist.

q `script` unterdrückt die Meldungen zu Beginn und am Ende der `script`-Sitzung.

S`_-`shell

`script` ruft die angegebene Shell auf.

Standard: `/bin/sh`, falls Sie in der Variablen `SHELL` nichts anderes definiert haben.

datei `script` schreibt das Protokoll auf "`datei`".

Standard: `typescript`

Arbeitsweise

Nachdem Sie das script-Kommando eingegeben haben, meldet script: "Script gestartet, Protokolldatei ist datei". Die in der Variablen SHELL oder durch Angabe des Schalters -S vereinbarte Shell meldet sich. Alle nun von Ihnen gemachten Eingaben und sämtliche Ausgaben schreibt script in die Protokolldatei. Diese können Sie sich später z.B. mit dem lpr-Kommando ausdrucken lassen.

script kommuniziert mit der neuen Shell über eine Pipe. Kommandos, deren Arbeitsweise davon abhängt, ob Sie über eine Pipe lesen oder schreiben, arbeiten meist nicht korrekt. Dies sind u.a. Kommandos, die im cbreak- oder raw-Modus arbeiten (siehe stty-Kommando), z.B. more, page, ced, stty. Auch die Kommandos newgrp, su und login arbeiten unter script nicht korrekt. Das Kommando who am I meldet für die Datensichtstation nur "tty??" . Was direkt auf /der/... ausgegeben wird, protokolliert script nicht; z.B. Fehlermeldungen für die Console.

Um script zu beenden, drücken Sie die Taste **END**. script beendet dann alle von Ihnen gestarteten und noch laufenden Prozesse, indem es die Pipes zu diesen schließt.

script verwenden Sie vor allem, wenn Sie Fehler protokollieren wollen.

Beispiel

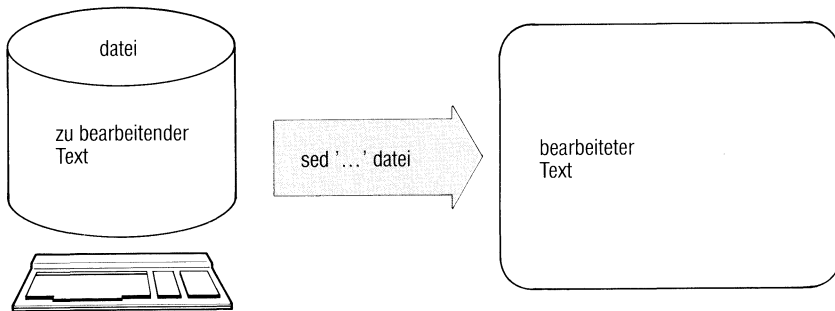
```
$ script protokoll
Script gestartet, Protokolldatei ist protokoll.
$ cat > admin.c
main() {}
```

END

```
Script beendet, Protokolldatei war protokoll.
$
```

script schreibt das Protokoll auf die Datei "protokoll". Die neue Shell meldet sich mit dem Zeichen "\$". Sie schreiben dann Zeilen in die Datei "admin.c". Durch Drücken der Taste **END** beenden Sie script. script schließt die Pipe zu der neuen Shell, und beendet diese und das cat-Kommando dadurch.

Editor im Prozedurbetrieb



Der sed (stream editor) ist ein nicht interaktiver zeilenorientierter Editor, der besonders dann benutzt wird, wenn

- mehrfach globale Editierfunktionen effizient in einem Schritt durchgeführt werden sollen,
- die Ausgabe eines Kommandos über eine Pipeline auf einfache Weise editiert werden soll,
- die Folge von Editierkommandos zu kompliziert ist, um sie komfortabel interaktiv einzugeben.

sed liest Dateien sequentiell, bearbeitet die eingelesenen Zeilen entsprechend einer vom Benutzer spezifizierten Bearbeitungsanweisung, dem Skript, und gibt die editierte Form zeilenweise auf die Standard-Ausgabe aus, wenn nicht andere Ausgabeanweisungen im Skript angegeben wurden. Beachten Sie, daß der sed den ursprünglichen Inhalt der Eingabedatei nicht verändert! sed arbeitet auf einer Kopie und gibt auf die Standard-Ausgabe aus, schreibt also nicht zurück in die Eingabedatei!

sed[**-n**][**-e** skript][**-f** skriptdatei][**datei...**]

schalter

n Unterdrückt die Ausgabe der eingelesenen Zeilen.

e skript

Die angegebenen sed Kommandos - das Skript - werden vom sed zur Bearbeitung der Eingabe verwendet. Enthält das Skript mehr als ein Wort oder Sonderzeichen, muß es in Hochkommas eingeschlossen werden. Mehrere -e und -f Angaben sind erlaubt; sed sammelt alle Angaben auf und faßt sie zu einem Skript zusammen.

f skriptdatei

sed liest das Skript aus der Datei skriptdatei. Mehrere -e und -f Angaben sind erlaubt; sed sammelt alle Angaben auf und faßt sie zu einem Skript zusammen.

datei

sed bearbeitet den Inhalt dieser Datei; ist keine Datei angegeben, liest sed von der Standard-Eingabe.

Allgemeines

sed arbeitet grundsätzlich auf einer Kopie der Eingabezeilen, dem Musterspeicher. Als Zeilennummer des Musterspeichers gilt die Zeilennummer der zuletzt in den Speicher gelesenen Zeile. sed liest zyklisch eine Eingabezeile in den Musterspeicher, führt alle Skript-Kommandos aus, die diesen Musterspeicher adressieren, gibt den nun eventuell veränderten Inhalt des Musterspeichers auf die Standard-Ausgabe aus und leert ihn. Zur Zwischenspeicherung von Eingabezeilen steht noch ein Bereich, genannt Haltespeicher, zur Verfügung. Muster- und Haltespeicher fassen je 4000 Zeichen.

Kommandoformat

sed Kommandos haben i.a. eine sehr einheitliche Struktur:

[adresse [,adresse]] funktion [parameter...]

keine, eine oder zwei Adressen gefolgt von einem Befehlsbuchstaben und eventuellen Parametern.

Adressen

Adressen werden nach den folgenden vier Regeln gebildet:

- 1) Eine Dezimalzahl bezeichnet eine Eingabezeile.
Die Zeilen aller Eingabedateien werden fortlaufend durchnummeriert.
- 2) "\$" bezeichnet die letzte Eingabezeile.
- 3) Ein regulärer Ausdruck in "/" eingeschlossen bezeichnet wie bei dem Editor ed beschrieben, eine Zeile, die eine zu dem regulären Ausdruck passende Zeichenfolge enthält.
- 4) "/\n/" bezeichnet das Zeichen "neue Zeile" im Musterspeicher.

Durch die Adressen werden wie beim Editor ed Eingabezeilen ausgewählt, die bei sed jedoch im Musterspeicher stehen. Adressenangaben werden wie folgt interpretiert:

- Haben Sie bei einem Kommando keine Adresse angegeben, gilt jede Zeile im aktuellen Musterspeicher als ausgewählt.
- Haben Sie eine Adresse angegeben, gelten die Zeilen des aktuellen Musterspeichers nur dann als ausgewählt, wenn die aktuelle Zeilennummer des Musterspeichers (siehe oben, "Allgemeines") mit der angegebenen übereinstimmt.
- Zwei Adressen, durch "," voneinander getrennt, kennzeichnen einen Bereich. Es gelten alle Zeilen als ausgewählt, die in dem Bereich liegen zwischen dem Musterspeicher, dessen Adresse mit der ersten angegeben übereinstimmt und dem Musterspeicher, dessen Adresse mit der zweiten angegebenen übereinstimmt.
Ist die zweite Adresse eine Zeilennummer, die kleiner oder gleich der ersten ist, wird nur die erste Zeile ausgewählt.
Ist der ganze Bereich abgearbeitet, wird der Prozeß wiederholt. D.h. sed sucht den nächsten Bereich, der durch die beiden Adressen ausgewählt wird.

Skript

Ein Skript ist eine Menge von sed Kommandos der oben angegebenen Form. Dabei darf nur jeweils ein Kommando auf einer Zeile stehen. Sollen auf einen ausgewählten Musterspeicher mehrere Kommandos angewendet werden, muß diese Kommandoliste in geschweifte Klammern "{ " "}" eingeschlossen werden.

Kommandos

Die folgende Liste enthält in alphabetischer Reihenfolge die sed Kommandos. Vor dem Kommandonamen ist in Klammern die maximale Anzahl zulässiger Adressen angegeben.

(1) **a** anfügen - append
text

text ein Text, der aus einer oder mehr Zeilen besteht, von denen alle bis auf die letzte mit einem "\ enden müssen, um das folgende Zeichen "neue Zeile" zu entwerten, da es sonst als Kommandotrennzeichen interpretiert wird. Führende Tabulator- oder Leerzeichen in text werden von sed entfernt. Um das zu verhindern, können sie durch ein vorangestelltes "\

sed fügt an seine Ausgabe des Musterspeichers auf die Standard-Ausgabe die Ausgabe von text an. Danach liest er die nächste Eingabezeile ein.

(2) **b** marke verzweigen - branch

Standard für marke (keine Angabe): Verzweigen zum Ende des Skriptes.

Das branch Kommando verzweigt zu dem ":" Kommando, das mit der Marke marke markiert ist.

(2) **c** verändern - change
text

text ein Text aus einer oder mehr Zeilen wie beim Kommando append beschrieben.

sed löscht den Musterspeicher. Ist keine oder nur eine Adresse angegeben, gibt sed den Text aus. Sind zwei Adressen angegeben, löscht sed den gesamten adressierten Bereich, wenn er sich im Musterspeicher befindet und gibt erst dann den Text aus. Anschließend sucht sed wieder nach einer Übereinstimmung des Musterspeichers mit der ersten Adresse.

(2) **d** löschen - delete

sed löscht den Musterspeicher und startet den nächsten Zyklus.

(2) **D** löschen bis "neue Zeile"

sed löscht den Anfang des Musterspeichers bis zum ersten Zeichen "Neue-Zeile" und startet den nächsten Zyklus.

(2) **g** ersetzen im Musterspeicher
sed ersetzt den Inhalt des Musterspeichers durch den Inhalt des Haltespeichers.

(2) **G** anfügen an Musterspeicher
sed fügt den Inhalt des Haltespeichers an den Musterspeicher an.

(2) **h** ersetzen Haltespeicher
sed ersetzt den Inhalt des Haltespeichers durch den Inhalt des Musterspeichers.

(2) **H** anhängen an Haltespeicher
sed fügt den Inhalt des Musterspeichers an den Haltespeicher an.

(1) **i** einfügen - insert
text

text ein Text aus ein oder mehr Zeilen wie beim Kommando
append beschrieben.

sed gibt den Text aus, bevor er den Musterspeicher ausgibt.

(2) **l** auflisten - list
sed gibt den Inhalt des Musterspeichers auf die Standard-Ausgabe aus. Nicht druckbare Zeichen werden durch ihren ASCII-Code als zweistellige Oktalzahlen dargestellt. Überlange Zeilen werden eindeutig dadurch gekennzeichnet, daß als letztes Zeichen jedes ausgegebenen Teils dieser Zeile ein "}" ausgegeben wird.

(2) **n** nächste Eingabe - next

sed gibt den Inhalt des Musterspeichers aus, wenn der -n Schalter nicht gesetzt ist, und ersetzt den alten Inhalt durch die nächste Eingabezeile.

(2) **N** nächste Eingabe

sed fügt die nächste Eingabezeile an den Musterspeicher an. Dadurch wird die aktuelle Zeilennummer des Musterspeichers auf die Zeilennummer der angefügten Zeile gesetzt (siehe oben, "Allgemeines"). Sind zur Ausführung eines N-Kommandos nicht mehr genügend Textzeilen vorhanden, geht der Rest des Textes verloren.

(2) **p** ausgeben - print

sed gibt den Inhalt des Musterspeichers auf die Standard-Ausgabe aus. Nicht druckbare Zeichen werden nicht dargestellt.

(2) **P** ausgeben bis "neue Zeile"

Der Inhalt des Musterspeichers bis einschließlich dem ersten Zeichen "neue Zeile" wird auf die Standard-Ausgabe ausgegeben.

(1) **q** beenden - quit

sed wird beendet.

(1) **r rdatei** lesen - read

rdatei Eingabedatei; das Argument rdatei muß, durch genau ein Leerzeichen vom Befehlsbuchstaben "r" getrennt, am Ende der Kommandozeile stehen.

sed liest den Inhalt der rdatei und gibt ihn aus, bevor die nächste Eingabezeile gelesen wird.

ersetzen - substitute

(2) **s/regulärer Ausdruck/Ersetzungszeichenfolge/schalter**

schalter

kein schalter

Die erste Zeichenfolge im adressierten Musterspeicher, zu der der reguläre Ausdruck paßt, wird durch die Ersetzungszeichenfolge ersetzt. Statt des Trennzeichens "/" kann jedes andere Zeichen benutzt werden. Bzgl. regulärer Ausdrücke siehe ed oder Tabelle über reguläre Ausdrücke im Anhang.

g global, alle Zeichenfolgen im Musterspeicher, zu der der reguläre Ausdruck paßt, werden ersetzt.

p print, der Musterspeicher wird ausgegeben, falls eine Ersetzung durchgeführt wurde.

(2) **t** marke test

Standard für marke (keine Angabe): Verzweigen zum Ende des Skriptes.

sed springt zu der angegebenen Marke, falls irgendwelche Ersetzungen durchgeführt wurden, seit die letzte Eingabezeile gelesen oder ein t Kommando ausgeführt wurde.

(2) **w** wdatei schreiben - write

wdatei Ausgabedatei, muß am Ende der Kommandozeile stehen und durch genau ein Leerzeichen vom Kommandobuchstaben "w" getrennt sein.

sed fügt den Inhalt des Musterspeichers an die Datei wdatei an.

(2) **x** austauschen - exchange

sed tauscht die Inhalte des Muster- und des Haltespeichers aus.

(2) **y**/zeichenfolge1/zeichenfolge2/ ersetzen Zeichen - transform

sed ersetzt jedes Auftreten eines Zeichens aus zeichenfolge1 durch das entsprechende Zeichen aus zeichenfolge2. Die beiden Zeichenfolgen müssen gleich lang sein.

(2) **!**sed-kommando Negation - don't
oder

(2) **!**{kommandoliste}

sed wendet das sed-Kommando (oder Kommandoliste, falls "{}" angegeben ist) nur auf Zeilen an, die nicht durch die Adressen ausgewählt sind.

(0) `:marke` markieren - mark

Dieses Kommando setzt die Marke `marke`, die von `b` und `t` Kommandos angesprochen werden kann. Ansonsten tut das Kommando nichts.

(1) `=` Zeilennummer - linenummer

`sed` gibt die aktuelle Zeilennummer in einer eigenen Zeile aus.

(2) `{` zusammenfassen - group

`sed` führt die folgenden Kommandos bis zu einem schließenden `}`, das allerdings am Anfang einer Zeile stehen muß, nacheinander aus, falls die angegebenen Adressen den aktuellen Musterspeicher betreffen.

(0) leer - empty

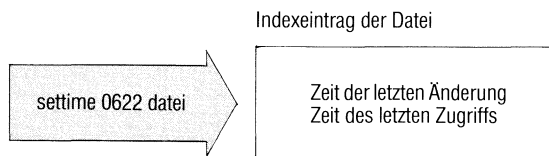
Ein leeres Kommando wird ignoriert.

Beispiel

Alle Zeilen einer Datei, die keine Leerzeilen sind, sollen um ein `"tab"` eingerückt werden:

```
sed -e '/^$/!s/^/"tab"/' dateiname
```

Zeit des letzten Zugriffs oder der letzten Änderung einer Datei setzen



`settime` setzt die Zeit des letzten Zugriffs und die Zeit der letzten Änderung einer Datei oder eines Dateiverzeichnisses auf einen gewünschten Wert. Beide Angaben ändert `settime` im Indexeintrag der Datei.

Format 1: Setzen mit eigener Angabe

`settime_ [jjmmtt]hhmm[.ss]_datei...`

- `jjmmtt` Jahr, Monat und Tag (je zwei Ziffern)
- `hhmm` Stunden und Minuten (je zwei Ziffern). Stunden sind im 24 Stunden-System anzugeben.
- `ss` Sekunden (wahlweise, Standard: 00)
- `datei` Name der Datei, für die Sie die Zeitangaben setzen wollen.

Mit diesen Angaben setzen Sie die Zeit der letzten Änderung und des letzten Zugriffs auf den angegebenen Zeitpunkt. Jahr, Monat und Tag können Sie weglassen. `settime` ergänzt die Angabe mit den aktuellen Werten. Dabei füllt `settime` nach vorne auf: die Angabe 0622 bedeutet z.B. 6 Uhr 22 und wird ergänzt mit dem aktuellen Datum.

Format 2: Setzen auf die Werte einer anderen Datei

```
settime[_f_name1]_name2...
```

`f_name1`

settime setzt die Zeit der letzten Änderung und die Zeit des letzten Zugriffs auf die Werte der Datei `name1`.

`name2`

Name der Datei, für die Sie die Zeitangaben ändern wollen.

Beispiele

1. Für die Datei `beispiel` wollen Sie die Zeit der letzten Änderung auf den 10.3. 14 Uhr 12 setzen:

```
settime 03101412 beispiel
```

settime ergänzt die Jahreszahl 84 und setzt auch die Zeit des letzten Zugriffs auf denselben Wert.

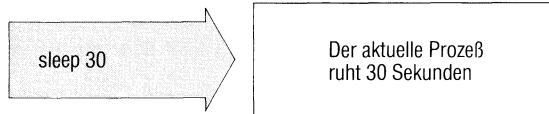
2. Für die Datei `beispiel` wollen Sie die Werte der Datei `richter` übernehmen:

```
$ settime richter beispiel
$ ls -l richter beispiel
drwxrwxr-x 3 blumann      48 Oct 11  15:08 beispiel
drwxrwxrwx 3 blumann     208 Oct 14  15:08 richter
$ ls -lu richter beispiel
drwxrwxr-x 3 blumann      48 Dec 12  11:31 beispiel
drwxrwxrwx 3 blumann     208 Dec 12  11:31 richter
$
```

settime setzt auch die Zeit des letzten Zugriffs auf den Wert der Datei `richter`.

```
> > > >  ls, touch
```

Prozesse zeitweise stilllegen



Mit dem sleep-Kommando verzögern Sie die Ausführung des nächsten Kommandos um eine frei wählbare Zeitspanne. sleep benützt man vor allem in Shell-Prozeduren. Für die angegebene Zeit ruht der Prozeß, in dem sleep aufgerufen wird, weil er durch sleep blockiert ist.

sleep_zeit

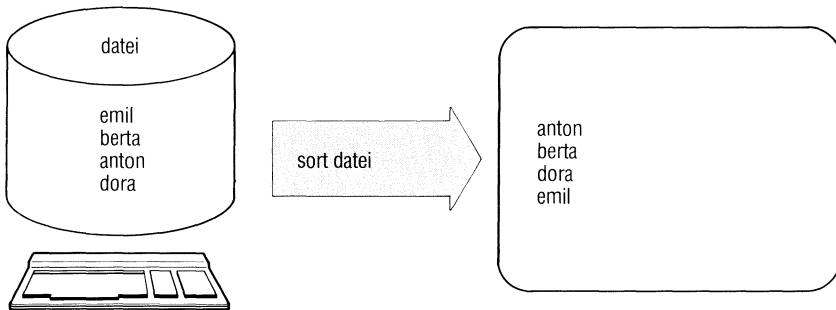
zeit Zeit in Sekunden, die der Prozeß ruhen soll.
Maximalwert: 65535 Sekunden.

Beispiel

Nach einer Verzögerungszeit von 30 Sekunden wird das Kommando date ausgeführt.

```
$ sleep 30;date
Tue Oct 4 11:22 MEZ 1983
$
```

Sortieren und mischen von Dateien



sort sortiert die Eingabedatei in alphabetischer Reihenfolge (Reihenfolge des ASCII). Das Ergebnis schreibt sort in eine Ausgabedatei, die Eingabedatei bleibt unverändert.

Geben Sie mehrere Eingabedateien an, sortiert und mischt sort die Dateien in einem Arbeitsgang, d.h. in der Ausgabedatei steht sortiert der Inhalt aller Eingabedateien.

Sortierfeld ist standardmäßig die ganze Zeile.

```
sort      [-s-schalter...][_pos...][_T_dateiverzeichnis]
          [_-o_ausgabe][_eingabe...]
```

schalter

- c nur prüfen, ob die Eingabedatei bereits sortiert ist.
Datei sortiert: Ende-Status 0;
Datei nicht sortiert: Meldung und Ende-Status 1.
- m mischen bereits sortierter Dateien.
Die Funktion ist schneller als mischen und sortieren. Man kann z.B. eine längere sortierte Liste ergänzen, indem man die neu dazukommenden Zeilen erst allein sortiert und dann dazumischt.
- u gleichlautende Zeilen nur einmal ausgeben.
sort erkennt in diesem Fall Zeilen als gleich, wenn sie in den

Sortierfeldern übereinstimmen (siehe Angaben + pos1 und – pos2).

tx definiert x als Trennzeichen für die Felder einer Zeile. Jedes Trennzeichen begrenzt ein Feld, z.B. feld1:feld2::feld4 mit Trennzeichen ":". Dabei dürfen Felder fehlen, im Beispiel feld3.

Standard (t nicht angegeben): Felder sind Zeichenfolgen, getrennt von einem oder mehreren Leerzeichen, z.B.

feld1 feld2 feld3 feld4

Die folgenden Schalter können Sie auch pro Sortierfeld angeben (siehe Angaben + pos1 und – pos2).

b führende Leerzeichen und Tabulatorzeichen nicht berücksichtigen.

d lexikalisch sortieren, d.h. sort berücksichtigt *nur*: Buchstaben, Ziffern und Leerzeichen.

f Groß- und Kleinbuchstaben nicht unterscheiden.

i nur ASCII-Zeichen von 040 bis 0176 berücksichtigen (siehe Anhang), d.h. nicht druckbare Zeichen werden ignoriert.

n nach Zahlenwerten sortieren. Ein Zahlenwert besteht aus Ziffern 0–9, Vorzeichen "+" und "-", Dezimalpunkt, z.B. +200, 7, 3.5, .23, –65. Der Zahlenwert muß am Beginn des Sortierfeldes stehen. Führende Leerzeichen werden nicht berücksichtigt (wie bei Schalter b).

r sortieren in umgekehrter Reihenfolge.

+ pos1 legt die Anfangsposition des Sortierfeldes fest.

– pos2 legt die Position des ersten Zeichens fest, das nicht mehr zum Sortierfeld gehört.

Ist pos2 nicht angegeben, nimmt sort Zeilenende an und ignoriert weitere Positionsangaben und deren Schalter.

pos1 und pos2 geben Sie wie folgt an:

+ m[.n] [– m[.n]]

m und n sind Zahlenwerte, die bedeuten:

m Felder der Zeile überspringen,

n Zeichen (des Feldes m + 1) überspringen. Ist Schalter b angegeben, zählen Leerzeichen am Feldanfang nicht mit. Ist .n nicht angegeben, bedeutet das .0, d.h. Feldanfang. Feld: siehe Schalter t.

Beispiel: Der Sortierbegriff beginnt im zweiten Feld beim dritten Zeichen und endet mit diesem Feld. Dann schreiben Sie: sort +1.2 -2

```
030537 A.Meierlein München 40
      |         |
      Sortierfeld
```

Definieren Sie mehrere Sortierfelder, dann sortiert sort erst nach dem zuerst definierten, bei Gleichheit nach dem nächsten usw. (siehe Beispiel).

Zeilen, die in allen Sortierfeldern gleich sind, vergleicht sort in der ganzen Länge.

Für jedes Sortierfeld können Sie folgende Schalter neu setzen, indem Sie sie hinter die Angabe + pos schreiben: b, d, f, i, n, r.

Beachten Sie: Vor den Positionsangaben gesetzte Schalter gelten für alle Sortierfelder, wenn für kein Sortierfeld eigene Schalter gesetzt sind, z.B.: sort -r +1 -2 +3 sortiert nach dem zweiten und vierten Feld in umgekehrter Reihenfolge.

Haben Sie für mindestens ein Sortierfeld Schalter gesetzt, gelten die vorher gesetzten Schalter ab hier nicht mehr, z.B.: sort -r +0 -1 +2 -3 +1n -2 sortiert erst umgekehrt nach dem ersten Feld, dann umgekehrt nach dem dritten Feld, dann numerisch nach dem zweiten Feld in normaler Reihenfolge.

T_dateiverzeichnis

Temporäre Hilfsdateien legt sort im angegebenen Dateiverzeichnis an.

Standard: aktuelles Dateiverzeichnis.

oausgabe

Dateiname für die Ausgabedatei mit sortiertem Inhalt.

Standard (keine Angabe): sort schreibt auf die Standard-Ausgabe.

eingabe

Dateiname der zu sortierenden Datei. Geben Sie für einen der Dateinamen das Zeichen "-" an, liest sort dafür von der Standard-Eingabe.

Standard: sort liest von der Standard-Eingabe.

Ende-Status:

0 bei normalem Ablauf

1 bei Fehler oder wenn Schalter c gesetzt und die Eingabe nicht sortiert war

Dateien:

/usr/temp/stm*

/tmp/*

Dies sind temporäre Hilfsdateien, d.h. sort löscht sie wieder.

Beispiele

1. Eine Liste von Worten ist alphabetisch zu sortieren, wobei gleichlautende Wörter nur einmal vorkommen sollen. Groß- und Kleinschreibung ist nicht zu berücksichtigen:

```
sort -uf liste
```

2. Die Paßwort-Datei ist nach Benutzernummern zu sortieren. In der Paßwortdatei sind die Felder durch ":" getrennt. Die Benutzernummer steht im dritten Feld:

```
sort -t: +2n /etc/passwd
```

3. Aus einer sortierten Datei (Name: monate) mit Einträgen "Monat Tag" soll die erste Zeile jedes Monats ausgegeben werden:

```
sort -um +0 -1 monate
```

u gibt nur die erste Zeile pro "Monat" aus, da sort nur das erste Feld vergleicht.

m beschleunigt den Ablauf, da die Datei sortiert ist.

4. Sortieren nach mehreren Sortierfeldern.

Die folgende Datei ist nach dem ersten und nach dem vierten Feld zu sortieren (Name und Straße):

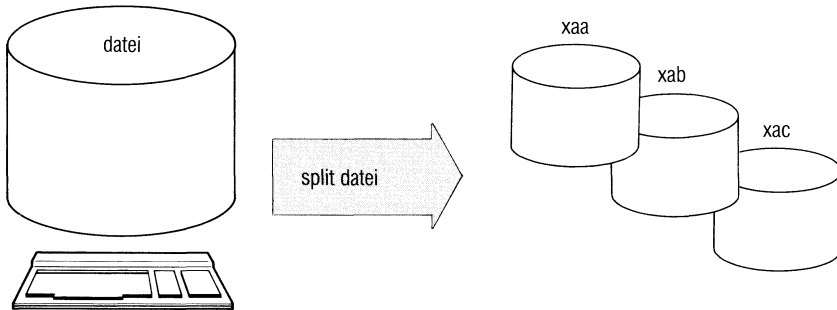
```
maier  albert  muenchen  bachingerweg 2
huber  franz    muenchen  gartenstrasse 5
maier  hans     muenchen  hauptstrasse 34
maier  irene    muenchen  fuerstenweg 37
```

```
sort +0 -1 +3 -4 datei
```

```
huber  franz    muenchen  gartenstrasse 5
maier  albert  muenchen  bachingerweg 2
maier  irene    muenchen  fuerstenweg 37
maier  hans     muenchen  hauptstrasse 34
```

>>>> uniq, comm, rev, join

Datei aufteilen auf mehrere Dateien



Mit dem `split` Kommando teilen Sie eine große Datei in kleinere Stücke zu je `n` Zeilen. Das Kommando speichert jedes Teilstück in einer Ausgabedatei ab. Eine Zeile ist eine beliebige Zeichenfolge, die durch das Zeichen "neue Zeile" abgeschlossen ist. Am Bildschirm ist dieses Zeichen nicht sichtbar. Im Normalfall entspricht eine Dateizeile einer Bildschirmzeile.

`split[_n][_datei[_name]]`

- n** Anzahl der Zeilen jedes Teilstücks.
Standard: 1000 Zeilen je Teilstück.
- datei** Name der Datei, die in Stücke geteilt wird. Ist für eine Datei das Zeichen "-" angegeben, liest `split` von der Standard-Eingabe.
Standard (keine Angabe): `split` liest von der Standard-Eingabe.
- name** Name der Ausgabedatei. `split` fügt an den Namen der ersten Ausgabedatei die Nachsilbe "aa" an und führt diese Nachsilbe für die weiteren Ausgabedateien in alphabetischer Reihenfolge weiter.
Erzeugt `split` mehr als 676 Dateien, hängt `split` ein drittes Zeichen an den Dateinamen an, z.B. `xzz`, `xaab` usw.
Standard: `x`

Beispiele

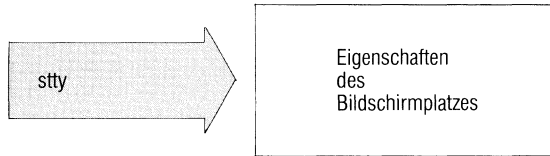
1. Der Inhalt der Datei "beispiel" ist zu je 20 Zeilen auf verschiedene Dateien zu verteilen.

```
$ split -20 beispiel
$ ls -l
-rw-rw-r-- 1 blumann      917 Oct 20 11:42 beispiel
-rw-rw-r-- 1 blumann      335 Oct 20 11:43 xaa
-rw-rw-r-- 1 blumann      261 Oct 20 11:43 xab
-rw-rw-r-- 1 blumann      210 Oct 20 11:43 xac
-rw-rw-r-- 1 blumann      111 Oct 20 11:43 xad
```

2. Je zwei Zeilen der Standard-Eingabe sind in eine eigene Datei "out.." zu schreiben.

```
$ split -2 - out
Was wahr ist, war immer wahr
und wird immer wahr bleiben.
Was aber nicht wahr ist, war nie wirklich
und wird nie wirklich werden.
END
$ ls -l out*
-rw-rw-r-- 1 art          58 Oct 20 11:51 outaa
-rw-rw-r-- 1 art          71 Oct 20 11:51 outab
```

Eigenschaften der Datensichtstation ändern



stty paßt die Ein- und Ausgabe softwaremäßig an die technischen Bedingungen der Datensichtstation an.

stty[_Schalter...]

schalter

kein Schalter angegeben

stty gibt die eingestellten Werte aus.

even die Parität ist gerade

– even die gerade Parität ist verboten

odd die Parität ist ungerade

– odd die ungerade Parität ist verboten

raw Für die Eingabe gelten *nicht*:

– Zeichenlöscher

– Zeilenlöscher

– Taste **DEL** (Signal 2, Interrupt)

– **CTRL Y** (Signal 3, Quit)

– Taste **END** (**CTRL D**)

”raw” setzt gleichzeitig ”nl”.

-
- raw rüćksetzen des raw – Modus
 - cooked gleichbedeutend mit -raw
 - cbreak Jedes eingegebene Einzelzeichen wird gelesen (Systemaufruf read, siehe C-Entwicklungssystem). Es gelten nicht:
 - Zeilenlöscher
 - Zeichenlöscher
 - cbreak
 - Nur vollständige Zeilen, abgeschlossen durch `␣`, werden gelesen.
 - nl Zeilenabschluß (Zeichen "neue Zeile") wird nur erzeugt durch die Taste `␣`.
 - Taste `␣` liefert den Code 0D.
 - Taste `␣` liefert den Code 0A.
 - nl Zeilenabschluß (Zeichen "neue Zeile") wird erzeugt durch die Taste `␣` oder die Taste `␣`.
 - Beide Tasten wirken wie die Taste `␣`, d.h. es wird der Code 0A gelesen, das ist das Zeichen "neue Zeile".
 - echo Jedes eingetippte Zeichen wird sofort an der Datensichtstation angezeigt.
 - echo Eingetippte Zeichen werden nicht angezeigt.
 - lcase Kleinbuchstaben werden als Großbuchstaben angezeigt.
 - lcase Großbuchstaben werden als Großbuchstaben angezeigt, Kleinbuchstaben als Kleinbuchstaben.
 - tabs ersetzt bei der Ausgabe das Tabulatorzeichen durch Leerzeichen.
 - tabs Die Tabulatorzeichen (X'09' `␣` `␣`) werden ausgegeben.
 - ek setzt als Standardwerte für den Zeilenlöscher "@" und für den Zeichenlöscher "#".
 - erase**_{LC}
 - Zeichenlöscher ist das Zeichen "c". Voreinstellung ist `␣` `␣` (Korrekturtaste `␣`).

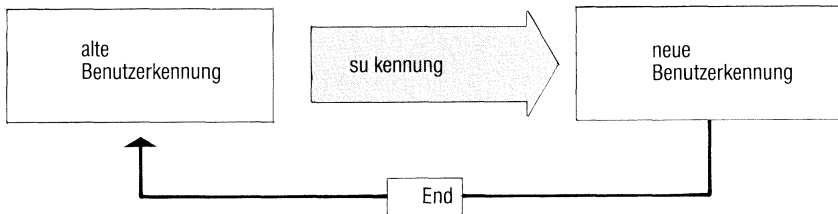
kill_c	Zeilenlöscher ist das Zeichen "c". Voreinstellung ist <input type="checkbox"/> CTRL <input type="checkbox"/> X.
crn	Verzögerung für den Wagenrücklauf (n = 0,1,2,3).
nln	Verzögerung für <input type="checkbox"/> (n = 0,1,2,3).
tabn	Verzögerung für Tabulator (n = 0,1,2,3).
ffn	Verzögerung für Formularvorschub (n = 0,1)
bsn	Verzögerung für die Korrekturtaste (Backspace, n = 0,1)
hup	Datentelefon beim letzten Schließen auflegen
0	Datentelefon sofort auflegen.
n	Auswahl einer Datenübertragungsgeschwindigkeit (in Baud). Für n können Sie folgende Werte einsetzen:
	50 134 300 1800 9600
	75 150 600 2400 exta
	110 200 1200 4800 extb

Beispiel

In einer Prozedur soll eine Eingabe von der Tastatur nicht sichtbar eingetippt werden:

```
.  
stty -echo  
echo -n 'Bitte Kennzahl eingeben: '  
read $zahl  
stty echo  
.
```


Benutzerkennung vorübergehend wechseln – substitute user id



Mit `su` wechseln Sie vorübergehend in eine andere Benutzerkennung. Dabei ändern sich weder das aktuelle Dateiverzeichnis noch die Shell-Umgebung. Das Home-Dateiverzeichnis wird auf das Login-Dateiverzeichnis der neuen Benutzerkennung gesetzt (Variable `HOME`).

`su` erzeugt eine Sub-Shell unter dieser Kennung entsprechend dem Eintrag in der Variablen `SHELL`.

Wenn Sie die Taste `END` drücken, beenden Sie diese Sub-Shell und können anschließend sofort wieder unter der ursprünglichen Kennung arbeiten.

`su[_benutzerkennung]`

`benutzerkennung`

Benutzerkennung, unter der Sie arbeiten wollen. Ist die Kennung mit einem Kennwort geschützt, fragt `su` dieses ab.

Standard (keine Angabe): root (Systemverwalter)

Hinweis

- Sie können mit su z.B. Systemverwalterfunktionen ausführen, ohne Ihre aktuelle Shell zu beenden.
- Sie können auf Dateien der neu angegebenen Kennung wie der Eigentümer zugreifen. Bei neu angelegten Dateien wird diese Kennung als Eigentümer eingetragen.
- Das Kommando "who" meldet immer die ursprüngliche Benutzerkennung!

Ende-Status:

- 0 bei normalem Ablauf
- 1 bei fehlerhaftem Ablauf
- 2 Kennwort falsch

Beispiele

1. Sie möchten die Zugriffsrechte einer Datei ändern, die "gast" gehört.

```
$ su gast
$ chmod 757 datei
$ pwd
/usr/art
$ cd
$ pwd
/usr/gast
$ printenv
HOME=/usr/gast
PATH=:/bin:/usr/bin
SHELL=/bin/shell
TERM=97801
USER=art
END
```

Das Kommando pwd zeigt, daß sich das aktuelle Dateiverzeichnis nicht ändert. In der Shell-Umgebung wird die Variable HOME für das Home-Dateiverzeichnis neu gesetzt.

Obwohl USER noch auf "art" gesetzt ist, gilt für Dateizugriffe die neue Benutzerkennung "gast".

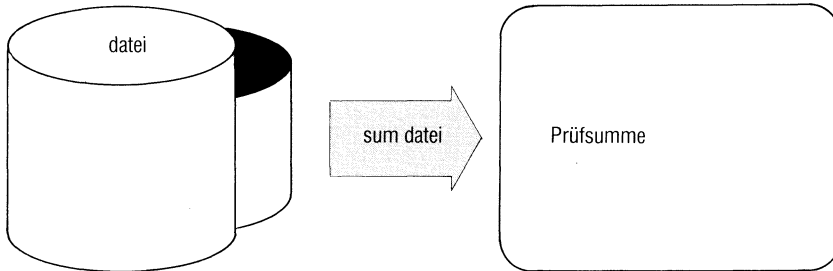
-
2. Sie möchten Systemverwalter-Funktionen ausführen:

```
$ su
Password:          Kennwort eingeben
# chown ...
# /etc/wall ...

END
$
```

>>>> login, newgrp

Prüfsumme einer Datei berechnen



sum berechnet eine 5-stellige Prüfsumme für eine Datei. Mit der Prüfsumme können Sie feststellen, ob eine Datei verändert wurde, z.B. durch einen Fehler bei einer Dateiübertragung. Die Datei kann auch nicht abdruckbare Zeichen enthalten.

Außerdem gibt sum den belegten Speicherplatz in KB aus.

sum[_name...]

name Name einer Datei oder eines Dateiverzeichnisses. Geben Sie mehrere Namen an, gibt sum die Dateinamen mit aus.

Standard (keine Angabe): sum liest von der Standard-Eingabe.

Ende-Status:

- 0 normaler Ablauf
- 10 Datei ist nicht vorhanden

Beispiel

1. Prüfsumme der Datei "prot":

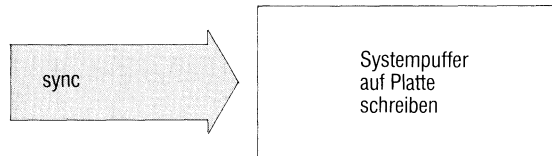
```
$ sum prot
30207 5
$
```

2. Sind die Dateien datei1 und datei2 identisch?

```
$ sum datei*
51997 1 datei1
51997 1 datei2
$
```

>>>> du, wc

Systempuffer zurückschreiben



sync bewirkt, daß noch nicht ausgeführte Ausgabeoperationen des Dateisystems durchgeführt werden.

sync

Arbeitsweise

Alle Information im Hauptspeicher, die sich auf der Festplatte oder Diskette befinden soll, wird herausgeschrieben. Dazu gehören

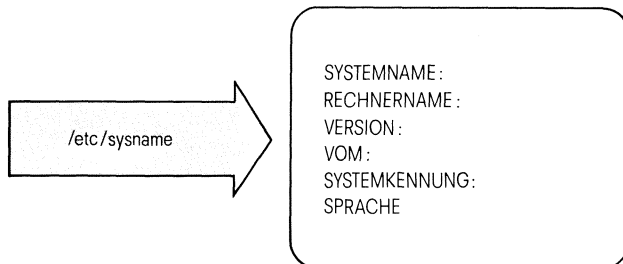
- modifizierte Superblöcke,
- modifizierte Indexeinträge und
- noch nicht durchgeführte blockweise Ein – oder Ausgabe.

Sie sollten sync benutzen, bevor Sie ein Kommando aufrufen, das das Dateisystem überprüft.

Hinweis

Da sync das Zurückschreiben nur veranlaßt, muß es noch nicht abgeschlossen sein, wenn sync beendet ist.

Das `/etc/sysname`-Kommando gibt allgemeine Systeminformationen aus.

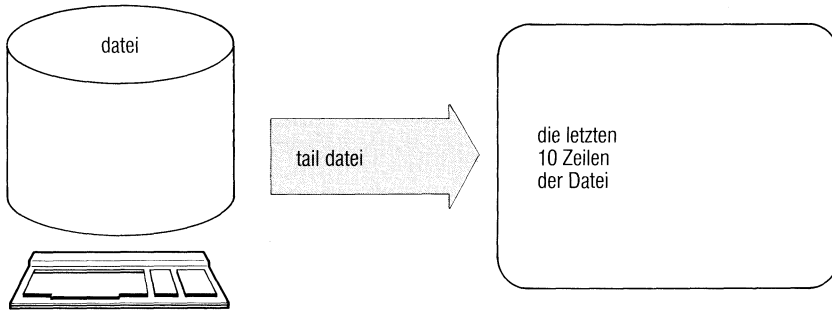


Beispiel

Eingabe: `/etc/sysname`

Ausgabe: `SYSTEMNAME: SINIX-M-A`
`RECHNERNAME: sie001`
`VERSION: 1.0C`
`VOM: 28. Mar. 1985`
`SYSTEMKENNUNG: Sie000001`
`SPRACHE: deutsch`

Endabschnitt einer Datei ausgeben



tail gibt eine Datei ab einer festgelegten Stelle aus.

tail[`[-anzahl`][`[-schalter...`]][`[-dateiname]`]

anzahl

+ n ab Zeile n ausgeben.

- n die letzten n Zeilen ausgeben.

Standard: -10, d.h. tail gibt die letzten 10 Zeilen aus.

schalter legt fest, ob n in Zeilen, Blöcken oder Zeichen gezählt wird.

l Zeilen (Standard).

b Blöcke zu 512 Byte. tail gibt ab dem n-ten Block, bzw. die letzten n Blöcke aus.

c Zeichen. Tail gibt ab dem n-ten Zeichen, bzw. die letzten n Zeichen aus.

-rn

die letzten n Zeilen in umgekehrter Reihenfolge ausgeben.

Hinweis

Schalter l, b, c ohne Zahlenangabe erkennt tail nicht.

dateiname Name der Eingabedatei.

Standard (keine Angabe): tail liest von der Standard-Eingabe.

Hinweis

Dateiabscnitte, die vom Dateiende aus gezählt werden ($-n$), muß tail in einem Puffer zwischenspeichern. Ihre Länge ist daher begrenzt auf 4096 Bytes.

Beispiel

1. Die ersten 40 Zeilen der Datei presto sollen bei der Ausgabe übersprungen werden:

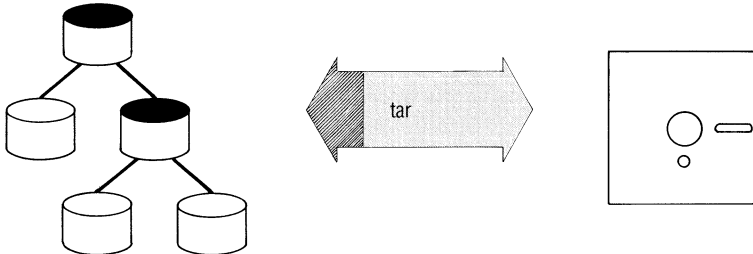
```
tail +41 presto
```

2. Den letzten Block der Datei andante ausgeben:

```
tail -1b andante
```

```
>>>> head
```

Archivieren auf Band oder Diskette – tape archiver



tar archiviert Dateien und ganze Unterbäume des Dateisystems in einem Diskettenarchiv oder Bandarchiv.

Das Kommando `far` (floppy archiver) ist identisch mit `tar`, setzt aber standardmäßig einige Angaben für Disketten (siehe Kommando `far`).

Bevor Sie `tar` oder `far tar` aufrufen, müssen Sie die Diskette, auf der sich Ihr Archiv befindet, in das Diskettenlaufwerk einlegen. Die Diskette muß formatiert sein.

Nach Beendigung des Archivvorgangs sollten Sie Ihre Diskette sofort aus dem Laufwerk entfernen, damit nicht ein anderer Benutzer Ihr Archiv beschädigt.

tar `funktion`[`attribut...`][`argument...`][`datei...`]

funktion Die Funktion wählen Sie durch Angabe genau eines der folgenden Buchstaben aus:

Archiv anlegen und beschreiben:

- c tar legt ein neues Archiv auf der Diskette an und schreibt die angegebenen Dateien hinein. Ein vorhandenes Archiv, bzw. ein Dateisystem, das sich auf der Diskette befand, wird überschrieben.
Ist eine der angegebenen Dateien ein Dateiverzeichnis, so werden rekursiv alle zu diesem Dateiverzeichnis gehörigen Dateien kopiert. Die Struktur des archivierten Unterbaums bleibt also erhalten.
Haben Sie keine Dateien angegeben, so wird ein leeres Archiv angelegt.
- r tar hängt die angegebenen Dateien an das Ende eines bestehenden Archivs an. Sind diese bereits im Archiv vorhanden, wird beim Wiedereinlesen nur die neuere Version, nämlich die angehängte Datei eingelesen (replace). Sie können mit dieser Funktion also Ihr Archiv aktualisieren, machen es dabei aber immer größer.
- u tar hängt die angegebenen Dateien an das Ende eines bestehenden Archivs an. Im Gegensatz zu Funktion r geschieht das nur, wenn die Dateien entweder noch nicht im Archiv vorhanden oder seit der letzten Archivierung verändert worden sind (update). Die Funktion kann sehr langsam sein.
- C wie c, aber mit Kontrolle.
- R wie r, aber mit Kontrolle.
- U wie u, aber mit Kontrolle.

Bei C, R und U liest tar nach dem Schreiben die Daten nochmals ein und vergleicht die gelesenen Daten mit den geschriebenen. Bei Fehler bricht tar mit einer Meldung ab. Die Funktion ist nur anwendbar bei Block-Geräte-dateien mit Blockungsfaktor 1.

Beachten Sie: Diese Funktionen sind nur sinnvoll, wenn Sie mit Disketten arbeiten. Bandgeräte arbeiten sowieso mit Kontrolle.

Archiv lesen:

- x Die angegebenen Dateien werden aus dem Archiv kopiert (extract).
Steht eine Datei mit vollem Pfadnamen im Archiv, wird sie in das entsprechende Dateiverzeichnis kopiert (Zugriffsrechte vorausgesetzt). Sonst kopiert tar die Datei ins aktuelle Dateiverzeichnis.
Ist ein Dateiverzeichnis angegeben, so werden alle zu diesem Dateiverzeichnis gehörigen Dateien rekursiv kopiert. Ist keine Datei angegeben, so wird der Inhalt des gesamten Archivs kopiert. Befindet sich eine Datei mehrfach im Archiv, d.h. es existieren Dateien mit gleichen Namen, so wird beim Kopieren der letzten der Inhalt der zuvor kopierten überschrieben. Beim Kopieren werden, falls möglich, Eigentümer, Zeit der letzten Änderung und Zugriffsrechte von der archivierten Datei übernommen.
- t Inhaltsverzeichnis des Archivs ausgeben (table). Jedes Vorhandensein der angegebenen Dateien im Archiv wird aufgelistet. Sind keine Dateien angegeben, so wird der gesamte Inhalt des Archivs aufgelistet.
- attribut Die ausgewählte Funktion kann durch Angabe beliebig vieler Funktionsattribute gesteuert werden. Die Attribute werden ohne Leerzeichen an die Funktion angefügt und bilden so zusammen mit der Funktionsauswahl eine Zeichenfolge. Argumente für verschiedene Attribute werden in der Reihenfolge der Attribute auf diese Zeichenfolge folgend angegeben. Als Attribute stehen Ihnen zur Verfügung:
- v alle durchgeführten Aktionen auf Standard-Ausgabe auflisten (verbose). tar gibt für jede bearbeitete Datei den Dateinamen und die Aktion aus. Dabei bedeuten:
- a Die Datei wird ins Archiv geschrieben (Funktionen a und r).

- x Die Datei wird aus dem Archiv kopiert.
Zusätzlich wird für jede Datei die Anzahl der belegten Blöcke auf der Diskette ausgegeben.

Mit Funktion t zusammen gibt v Information über die Datei aus, ähnlich wie beim Kommando ls -l.

Ist v nicht angegeben, protokolliert tar nichts.

- w tar verlangt für jede Datei eine Antwort, bevor die Aktion ausgeführt wird (warten). Geben Sie "j" oder "n" ein:

- j die Aktion wird ausgeführt,
- n die Aktion wird nicht ausgeführt.

Jede andere Eingabe bedeutet ebenfalls, daß die Aktion nicht ausgeführt wird.

- f Dateiname des Gerätes für das Archiv (file). tar interpretiert das zugehörige Argument als Name der Gerätedatei für das Archiv. Geben Sie als Argument '-' an, so liest tar von der Standard-Eingabe, bzw. schreibt auf die Standard-Ausgabe, abhängig von der ausgewählten Funktion. tar kann also auch in Verbindung mit Pipelines benutzt werden, um z.B. Hierarchien innerhalb des Dateibaums zu verschieben wie in:

```
cd dv1; tar cf - . | (cd dv2; tar xf -)
```

Damit kopieren Sie den Unterbaum des Dateisystems, der am Dateiverzeichnis dv1 hängt, an das Dateiverzeichnis dv2.

Standard (f nicht angegeben): /dev/fd2
(Diskettenlaufwerk).

- b** Blockungsfaktor im Archiv (nur bei Magnetbändern sinnvoll).
tar interpretiert das zugehörige Argument als Blockungsfaktor. Die Blockgröße der Dateien des Archivs ist:
Blockungsfaktor x 512 Bytes.
Beim Lesen eines Archivs wird die Blockgröße automatisch festgestellt.
- Standard: 1*
Minimalwert: 1
Maximalwert: 20

Hinweis

Geben Sie **b** nicht an bei Archiven, die Sie später nachführen wollen, da die Funktionen **r** und **u** nur mit dem Blockungsfaktor 1 funktionieren.

- l** Verweise überprüfen.
tar meldet, wenn er beim Schreiben in ein Archiv Verweise auf andere Dateien nicht auflösen kann. Nur Verweise auf Dateien, die mit ins Archiv geschrieben werden, bleiben erhalten und können beim Lesen aus dem Archiv wiederhergestellt werden.
- m** tar setzt beim Lesen aus dem Archiv die Zeit der letzten Änderung auf das aktuelle Datum.
Ist das **m**-Attribut nicht angegeben, wird die im Archiv gespeicherte Angabe eingesetzt.
- k** Angabe der maximalen Größe des Speichermediums in KB (1 KB 1024 Byte).
tar interpretiert das zugehörige Argument.
Damit können Sie Archive bearbeiten, die auf mehreren Disketten stehen. Ist beim Schreiben die angegebene Größe erreicht, fordert tar auf:
Bitte neue Diskette einlegen und mit **'j'** bestätigen:
- Bei **far** ist dieses Attribut Standard. Wenn Sie also Disketten beschreiben und dazu **far** verwenden, brauchen Sie **k** nicht anzugeben.

Der Aufruf `tar ck 580 ...` entspricht dem Aufruf `far c ...`

Eine 5.25 Zoll Diskette faßt maximal 584 KB (1168 Blöcke).

Minimalangabe: 250

Hinweis

- Beim Lesen eines solchen Archivs fordert tar die Folgediskette an, falls beim Schreiben die letzte Datei geteilt wurde.
 - Attribut `k` setzt gleichzeitig Attribut `n`.
- n** Beschleunigt den Lesevorgang, falls das verwendete Gerät frei positionieren kann (z.B. Diskettenlaufwerke). `n` darf für Geräte im raw-Modus nicht verwendet werden.
- o** Überschreiben von Dateien mit neuerem Änderungsdatum (overwrite). tar überschreibt beim Einlesen (Funktion `x`) Dateien auch, wenn Sie neueren Datums sind (Zeit der letzten Änderung).
- Standard (o nicht angegeben):* Findet tar beim Einlesen eine Datei vor, die neueren Datums ist, verlangt tar eine Antwort, ob die Datei überschrieben werden darf (`j/n`). Mit Schalter `o` entfällt diese Rückfrage.
- p** Zugriffsrechte übernehmen (protection). tar setzt die Zugriffsrechte nach der Angabe im Archiv (bei Funktion `x`).
- Standard (p nicht angegeben):* Die Dateien erhalten die standardmäßigen Zugriffsrechte (nach `umask`).
- argument** Zu den Attributen `f`, `b` und `k` gehören Argumente. Diese geben Sie anschließend an Funktion und Attribute an, getrennt durch Leerzeichen. tar interpretiert die Argumente in der Reihenfolge, in der die Attribute angegeben sind.
- Beispiel:** `tar xvfk /dev/fl2 580. /dev/fl2` gehört zur Angabe `f`, `580` gehört zur Angabe `k`.

datei Name einer Datei oder eines Dateiverzeichnisses. Es gelten die üblichen Konventionen für Sonderzeichen. Der Name wird ins Archiv übernommen, wie er angegeben wurde: Die beiden Dateinamen ./abcd und abcd bezeichnen z.B. im Archiv unterschiedliche Dateien, im Dateisystem jedoch dieselbe Datei (siehe auch Funktion x).

Mit einem Dateiverzeichnis werden auch immer alle darin enthaltenen Dateien und Dateiverzeichnisse bearbeitet.

Sonderzeichen für Dateinamen können Sie wie folgt verwenden:

Beim Schreiben: wie üblich,

tar r d* fügt z.B. alle Dateien an das Archiv an, deren Name mit d beginnt.

Beim Lesen: wollen Sie alle Dateien

aus dem Archiv einlesen, die dort vorhanden sind und mit .c enden, schreiben Sie: tar x '*.c'

Die Hochkommas sind nötig, damit tar sich auf die im Archiv vorhandenen Dateien bezieht und nicht auf die im aktuellen Dateiverzeichnis.

Standard (keine Angabe): Beim Schreiben muß eine Datei angegeben sein, beim Lesen bearbeitet tar alle Dateien des Archivs.

Beispiele

1. Sie wollen ein neues Archiv auf Diskette anlegen und den Unterbaum beginnend mit dem aktuellen Dateiverzeichnis archivieren. Dabei wollen Sie jedoch einige Dateien nicht kopieren. Deshalb soll tar der Reihe nach die Namen aller zu verarbeitenden Dateien und die auf sie anzuwendende Funktion auflisten:

```
tar cw .
```

Geben Sie nun nach jeder von tar ausgegebenen Zeile eine mit j beginnende Zeichenfolge an, so wird die entsprechende Datei kopiert. Jede andere Eingabe verhindert das Kopieren.

2. Sie wollen aus Ihrem Archiv auf Diskette die Dateien xx1, xx2 und xx3 lesen. tar soll Meldungen über die bearbeiteten Dateien ausgeben:

```
tar xv xx1 xx2 xx3
```

3. Sie wollen die Benutzerdaten der Benutzer florian, fridolin und nikolai sichern. Das Archiv könnte länger werden, als 1168 Blöcke.

```
# cd /usr
```

```
# tar cvk 584 florian fridolin nikolai
```

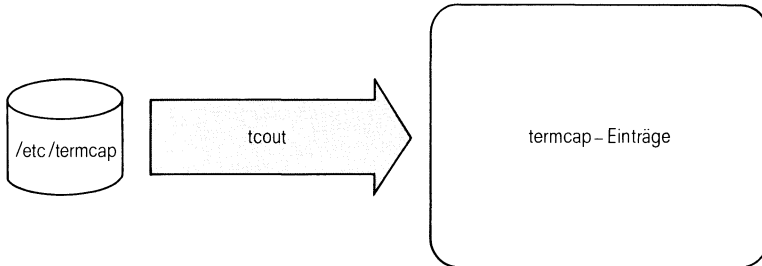
```
tar meldet die geschriebenen Dateien
```

```
Bitte neue Diskette einlegen und mit 'j' bestätigen: j
```

```
#
```

```
> > > > far
```

Einträge aus der Datei /etc/termcap lesen



Das `tcout`-Kommando liest Einträge aus der `termcap`-Datei und schreibt sie auf die Standard-Ausgabe.

`tcout id [zeile] [spalte]`

`id` Name des zu lesenden `termcap`-Eintrages.
Die folgende Tabelle enthält einen Auszug aus den in der `termcap`-Datei enthaltenen Einträge:

Name	Beschreibung
ae	Ausschalten des alternativen Zeichensatzes
al	Einfügen Zeile
am	Schreibmarke springt beim Erreichen des Zeilenendes auf nächste Zeile
as	Einschalten des alternativen Zeichensatzes
bs	Datensichtstation kennt Backspace-Zeichen '\b'
bt	Rückwärts-Tabulator-Zeichen
cd	Löschen ab Schreibmarken-Position bis Bildschirmende
ce	Löschen des Inhaltes der nächsten Zeile
cl	Löschen Bildschirminhalt
cm	Schreibmarke bewegen
co	Anzahl Spalten pro Zeile
cs	Einstellen Scroll-Bereich
dc	Löschen des ersten Zeichens in der nächsten Zeile
dl	Löschen der folgenden Zeile
do	Schreibmarke eine Zeile nach unten
ho	Schreibmarke an Bildschirmanfang
ic	Einfügen Zeichen
is	Datensichtstations-Initialisierungs-Sequenz
kd	Von der Taste 'Schreibmarke nach unten' gesendete Zeichenfolge
kh	Vor der Taste 'Schreibmarke an Bildschirm-anfang' gesendete Zeichenfolge
kl	Von der Taste 'Schreibmarke nach links' gesendete Zeichenfolge
kr	Von der Taste 'Schreibmarke nach rechts' gesendete Zeichenfolge
ku	Von der Taste 'Schreibmarke nach oben' gesendete Zeichenfolge
li	Anzahl Zeilen am Bildschirm
nd	Schreibmarke um eine Spalte nach rechts
se	Ausschalten Invers-Modus
sf	Scrollen nach oben
so	Einschalten Invers-Modus
sr	Scrollen nach unten
ta	Tabulator-Zeichen
ue	Ausschalten Unterstreichen-Modus
up	Schreibmarke nach oben
us	Einschalten Unterstreichen-Modus

zeile Mit diesen beiden Angaben kann man Dezimalzahlen angeben, die von
spalte bestimmten id-Einträgen als Koordinaten verwendet werden (z.B. cm oder cs).
Standardwert: 1

Hinweis

tcout zählt Koordinaten auf dem Bildschirm relativ zu 1, d.h. der Bildschirmfang hat damit die Koordinaten: 1,1.

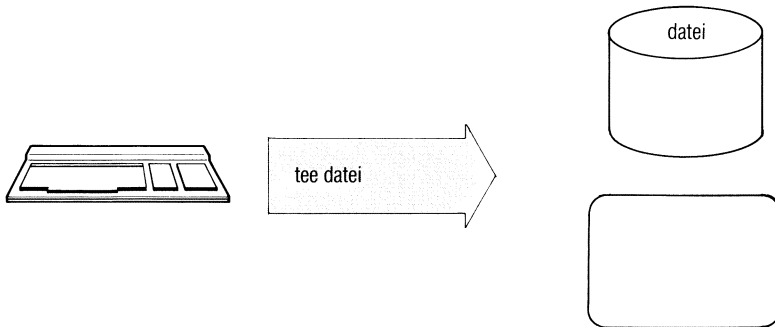
Das tcout-Kommando liefert folgenden Exit-Code:

- 0 Kommando erfolgreich ausgeführt.
- 1 Zuwenig oder zuviele Parameter eingegeben.
- 2 Die Datei /etc/termcap oder der entsprechende Eintrag ist nicht vorhanden.
- 3 Die angegebene id wurde nicht gefunden.

Beispiele

tcout cl (Bildschirm wird gelöscht)
tcout cm4,co15 (Schreibmarke wird auf Zeile 4, Spalte 15 positioniert)
tcout so (Inverse Darstellung wird eingeschaltet)
tcout se (Inverse Darstellung wird ausgeschaltet)
tcout cs 12 24 (Scroll-Bereich einstellen)

Gleichzeitig auf Standard-Ausgabe und in eine Datei ausgeben



Das tee-Kommando überträgt Daten von der Standard-Eingabe zur Standard-Ausgabe und speichert zusätzlich eine Kopie der Daten in einer Datei ab.

Mit dem tee-Kommando erhalten Sie Ausgabedaten zum Anschauen sofort am Bildschirm und gleichzeitig hinterlegen Sie die Ausgabedaten zur Dokumentation in einer Datei. Wenn das tee-Kommando in einer Pipeline angegeben ist, kann es Zwischenstufen dieser Pipeline dokumentieren, die ansonsten verloren wären.

tee[₋schalter]₋dateiname

schalter

- i Das Kommando ignoriert Unterbrechungssignale.
- a Wenn die angegebene Datei schon vorhanden ist, überschreibt tee nicht den Inhalt dieser Datei, sondern fügt die Kopie an das Dateieende an.

dateiname Dateiname für die Kopie.

Beispiel

1. \$ tee -a beispiel2

es ist kalt

es ist kalt

\$ cat beispiel2

heute ist dienstag

es ist herbst

es ist kalt

\$

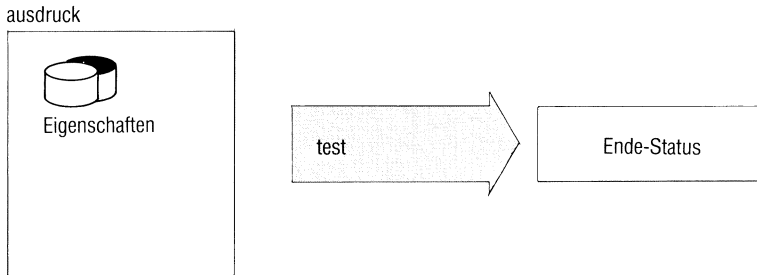
2. Die Ausgabe des Kommandos who erscheint auf dem Bildschirm und wird gleichzeitig in der Datei "anwender" gespeichert.

\$ who | tee anwender

blumann tty00 4 Oct 19 11:54

\$

Bedingungen prüfen



test prüft Bedingungen und setzt den Ende-Status 0, wenn die Bedingungen erfüllt sind oder den Ende-Status 1, wenn die Bedingungen nicht erfüllt sind. Bedingungen sind:

- Eigenschaften von Dateien und Dateiverzeichnissen prüfen,
- Vergleiche von Zeichenfolgen,
- algebraische Vergleiche ganzer Zahlen.

Bedingungen können Sie miteinander verknüpfen. test verwendet man in Shellprozeduren. Abhängig vom Ende-Status können Sie Kommandos ausführen oder Schleifen abbrechen usw.

test_┌ausdruck oder _└[┌ausdruck└]

_┌ausdruck└]

Steht ausdruck in eckigen Klammern, führt die Shell das test-Kommando als "eingebautes Kommando" aus, d.h. es läuft schneller.

ausdruck eine der folgenden Bedingungen oder mehrere Bedingungen, die miteinander verknüpft sind, wie unten beschrieben. Die Bedingungen geben Sie wie Schalter an.

Eigenschaften von Dateien und Dateiverzeichnissen

Für name geben Sie jeweils den Namen einer Datei oder eines Dateiverzeichnisses an. Geben Sie für name folgendes an: " (d.h. "nichts"), wird dafür der Name des aktuellen Dateiverzeichnisses eingesetzt.

- r_name
name existiert und Sie haben Leseerlaubnis.
- w_name
name existiert und Sie haben Schreiberlaubnis.
- f_name
name existiert und ist kein Dateiverzeichnis.
- d_name
name existiert und ist ein Dateiverzeichnis.
- c_name
name existiert und ist eine zeichenorientierte Datei für Geräte.
- b_name
name existiert und ist eine blockorientierte Datei für Geräte.
- u_name
name existiert und für den Eigentümer ist das s-Bit gesetzt.
- g_name
name existiert und für die Gruppe ist das s-Bit gesetzt.
- k_name
name existiert und das sticky-Bit ist gesetzt.
- s_name
name existiert und ist nicht leer, belegt also mindestens einen Block.

-t_l[dateideskriptor]

dateideskriptor kann sein:

- 0 für Standard-Eingabe,
- 1 für Standard-Ausgabe (Standard, wenn Sie nichts angeben),
- 2 für Standard-Fehlerausgabe.

test prüft, ob die angegebene Ein- bzw. Ausgabe mit einer Datensichtstation verbunden ist (Ende-Status 0) oder auf eine Datei umgewiesen ist (Ende-Status 1).

Vergleiche von Zeichenfolgen

Zeichenfolgen können Sie angeben, wie z.B. bei "echo" beschrieben (direkt oder in Variablen).

-z_lzeichenfolge

Die Zeichenfolge ist leer (Länge gleich 0).

-n_lzeichenfolge

Die Zeichenfolge ist nicht leer (Länge größer 0).

zeichenfolge1_l = _lzeichenfolge2

Die beiden Zeichenfolgen sind gleich.

zeichenfolge1_l != _lzeichenfolge2

Die Zeichenfolgen sind verschieden.

Algebraische Vergleiche ganzer Zahlen

Zahlenwerte können Sie direkt oder in Variablen angeben.

wert1_{op}wert2

test vergleicht wert1 und wert2 algebraisch. op kann sein:

eq	gleich
ne	ungleich
gt	größer als
ge	größer gleich
lt	kleiner als
le	kleiner gleich

Bedingungen verknüpfen

Bedingungen können Sie wie folgt miteinander verknüpfen:

\(_{bed} ... \)

Klammern fassen Bedingungen zu Gruppen zusammen.
Die Klammern sind mit "\ " für die Shell zu entwerten.

!_{bed} Verneinung, z.B. "!-r datei".

{bed}-a{bed}

logisches UND. Jede der aneinandergereihten Bedingungen muß erfüllt sein.

{bed}-o{bed}

logisches ODER. Eine der Bedingungen muß erfüllt sein.

test verarbeitet die Verknüpfungen in der Reihenfolge:
Klammern, Verneinung, UND, ODER.

Ende-Status:

- 0 Die Bedingungen sind erfüllt
- 1 Die Bedingungen sind nicht erfüllt

Beispiele

1. Die folgende Shell-Prozedur prüft, ob der angegebene Parameter der Name einer Datei oder eines Dateiverzeichnisses ist.

```
if test -f "$1"
then
echo $1 ist eine Datei
elif test -d "$1"
then
echo $1 ist ein Dateiverzeichnis
fi
```

\$1 ist in Anführungszeichen eingeschlossen, damit test keine Fehlermeldung ausgibt, wenn kein Parameter angegeben ist.

2. Diese Prozedur vergleicht die Anzahl der Zeilen in zwei Dateien.

```
if test `cat "$1" | wc -l` -gt `cat "$2" | wc -l`
then
echo $1 enthält mehr Zeilen als $2
fi
```

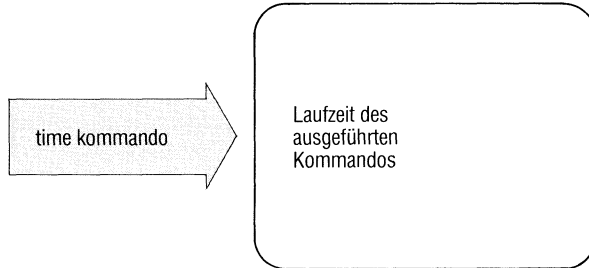
test vergleicht die beiden vom wc-Kommando gelieferten Werte mit dem Operator -gt.

3. In Prozeduren können Sie folgende Schreibweise verwenden:

[ausdruck]

Anstelle von `if test -f "$1"`
schreiben Sie z.B. `if [-f "$1"]`

Laufzeit eines Kommandos messen



Mit dem time-Kommando messen Sie die Laufzeit eines beliebigen Kommandos. time gibt aus:

- die reale Laufzeit (real) in Sekunden, das ist die Zeit zwischen Kommandoaufruf und Kommandoabschluß,
- die CPU-Zeit der Benutzerphase (user) in 1/60 Sekunden,
- die CPU-Zeit der Systemphase (sys) in 1/60 Sekunden.

time_kommando

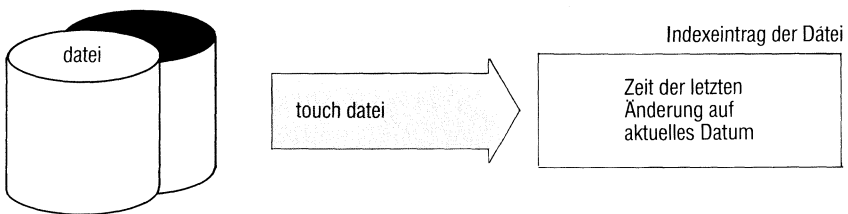
kommando kommando wird ausgeführt. Anschließend gibt time die Zeiten aus:

Beispiel

```
$ time sort liste > liste.sort
```

```
real      10.0
user       0.4
sys        5.9
$
```

Zeit der letzten Änderung einer Datei auf aktuelles Datum setzen



touch setzt die Zeit der letzten Änderung einer Datei auf die aktuelle Zeit und das aktuelle Datum. touch liest dazu ein Zeichen der Datei und schreibt es zurück, so daß die Datei nicht verändert wird.

touch[_{-c}]₋datei...

c Die Datei muß bereits vorhanden sein, sonst meldet touch einen Fehler.

Standard (c nicht angegeben): touch erzeugt die Datei, falls sie nicht vorhanden ist.

datei Name einer Datei.

Ende-Status: immer 0.

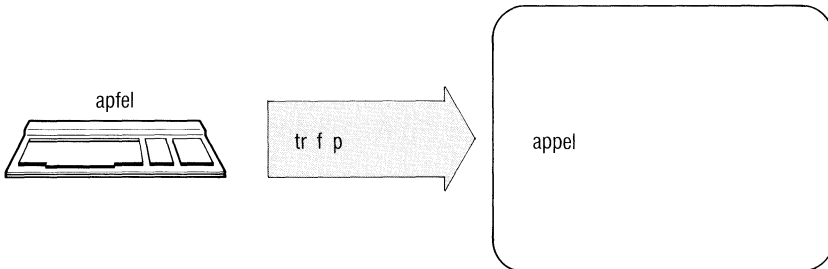
Beispiel

Sie möchten alle Dateien löschen, die länger als 8 Wochen nicht geändert wurden. Die Dateien, die mit "liste." beginnen, sollen aber erhalten bleiben, auch wenn sie älter sind. Das erreichen Sie mit:

```
$ touch liste.*
$ find . -mtime +56 -exec rm {} ;;
$
```

```
>>>> settime
```

Zeichen durch andere ersetzen



Das Kommando `tr` überträgt Zeichen aus dem Eingabetext der Standard-Eingabe zum Ausgabebetext der Standard-Ausgabe. Dabei ersetzt `tr` Zeichen in einer Eingabezeichenfolge durch "korrespondierende" Zeichen, die in der Ausgabezeichenfolge den gleichen Platz einnehmen.

`tr[_schalter][_zeichenfolge1][_zeichenfolge2]`

schalter

- c Alle Zeichen, außer den in `zeichenfolge1` aufgeführten, werden durch die korrespondierenden Zeichen aus `zeichenfolge2` ersetzt.
- d Alle Zeichen, die in `zeichenfolge1` aufgeführt sind, werden gelöscht.
- s Mehrfach auftretende Ausgabezeichen, die in der `zeichenfolge2` vorkommen (siehe Beispiel), werden zu Einzelzeichen zusammengezogen.

zeichenfolge1 zeichenfolge2

Zeichenfolgen können folgende Formen haben:

- einfach aneinandergereihte Zeichen z.B. "abc" oder "012345" oder "ABCDEF".
- Zeichenbereiche, z.B. a–n. Sie bezeichnen alle Zeichen, die in der lt. ASCII aufsteigenden Reihenfolge zwischen a und n liegen.
- Sonderzeichen, dargestellt durch \n[n[n]], wobei n Oktal– ziffern sind, die der ASCII-Verschlüsselung entsprechen (siehe Tabelle im Anhang). Beispiel "\012" bedeutet Zeilenvorschub. Bei allen anderen Zeichen hat das Voranstellen von "\" keine Bedeutung.

Ist zeichenfolge2 kürzer als zeichenfolge1, wird das letzte Zeichen aus zeichenfolge2 so oft wiederholt, bis beide Zeichenfolgen gleich lang sind.

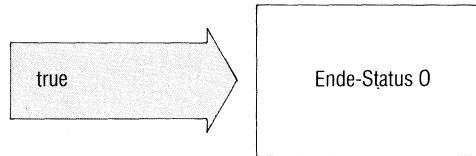
Beispiel

Sie wollen eine Liste aller Worte (Folge von Buchstaben) in Ihrer Datei erstellen, und zwar so, daß jedes Wort auf einer neuen Zeile steht. Dies erreichen Sie mit der Eingabe:

```
tr -sc A-Za-z '\012' liste
```

```
> > > > ed
```

Leeres Kommando mit Endestatus 0



true kehrt mit Ende-Status 0 zurück und tut sonst nichts. true verwendet man in Shell-Prozeduren, um die Bedingung "wahr" zu erzeugen. Die Bedingung "falsch" (Endestatus 1) erzeugen Sie mit dem Kommando "false".

true

Ende-Status: immer 0.

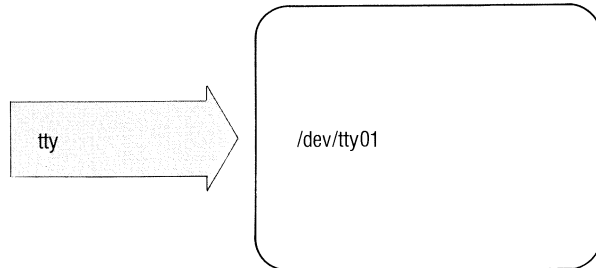
Beispiel

Folgende Prozedur erzeugt eine Endlos-Schleife. Sie kann z.B. mit der Taste `[DEL]` abgebrochen werden.

```
while true
do
    Kommandofolge
done
```

> > > > false, Shell (Abschnitte 3.6 bis 3.8)

Pfadname Ihrer Datensichtstation ausgeben



tty gibt den Pfadnamen Ihrer Datensichtstation aus. Der Ende-Status sagt aus, ob die Standard-Eingabe eine Datensichtstation ist.

tty[_-s]

s tty liefert nur den Ende Status und gibt sonst nichts aus.

Standard (s nicht angegeben): Ist die Standard-Eingabe keine Datensichtstation, meldet tty: "tty liest nicht von einer Datensichtstation".

Ende-Status:

- 0 Standard-Eingabe ist eine Datensichtstation.
- 1 Standard-Eingabe ist keine Datensichtstation.

Hinweis

Bei Einplatzsystemen gibt es nur eine Datensichtstation mit dem Pfadnamen /dev/console.

Beispiele

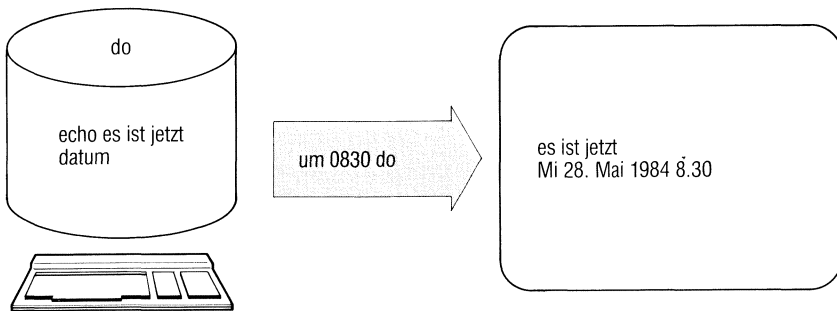
1. In einer Prozedur soll eine Ausgabe auf den Bildschirm gelenkt werden, auch wenn die Standard-Ausgabe eine Datei ist.

```
.  
echo 'Diese Ausgabe geht auf die Datensichtstation' > `tty`  
.
```

2. Falls die Standard-Eingabe nicht die Datensichtstation ist, soll in der folgenden Prozedur eine Fehlermeldung erzeugt werden.

```
if tty -s  
then  
read eingabe  
.  
else  
echo 'Standard-Eingabe ist keine Datensichtstation' >&2  
fi
```

Prozedur-Dateien zu einer bestimmten Zeit ausführen, Datum deutsch



Mit dem um-Kommando können ein oder mehrere Kommandos zu einem festgesetzten (späteren) Zeitpunkt ausgeführt werden.

um hat dieselbe Funktion, wie das Kommando "at". Sie ist ausführlich bei at beschrieben. "um" nimmt deutsche Datumsangaben an.

um *zeit* [*tag*] [*datei*]

tag Für tag können Sie angeben:

- eine Datumsangabe in der Form: 12 mai oder
- einen Wochentag, z.B: mittwoch

Ist für Tag ein Wochentag angegeben und anschließend das Kennwort "woche", so wird der Auftrag am angegebenen Tag in der folgenden Woche ausgeführt.

Ist für tag der aktuelle Tag angegeben, führt um den Auftrag in der folgenden Woche aus.

Die Namen der Tage und Monate werden in deutscher Sprache angegeben und können abgekürzt sein.

Standard (keine Angabe): um führt den Auftrag am selben Tag aus, bzw. am folgenden Tag, falls die angegebene Uhrzeit schon vergangen ist.

Alle anderen Angaben siehe Kommando at.

Beispiele

1. Die Datei merkedatum habe folgenden Inhalt:

```
datum >> datefile  
um 830a merkedatum
```

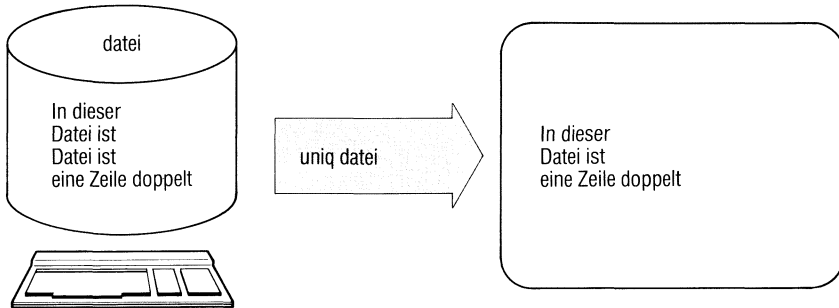
Ein einmaliger Aufruf von merkedatum (direkt oder mit um) bewirkt, daß täglich um 8Uhr 30 das aktuelle Datum in die Datei datefile geschrieben wird.

2. Mit folgenden Kommandos wird am 1.April um 13Uhr das Datum auf der Konsole ausgegeben.

```
um 1pm 1 apr  
/bin/datum >> /dev/console  
END
```

> > > > at

Mehrfache Zeilen suchen – unique lines



uniq durchsucht eine Datei nach aufeinanderfolgenden gleichen Zeilen, gibt die Datei aus und läßt dabei die Wiederholungen weg.

uniq[₋schalter[_{-n}][_{-m}]][₋eingabe[₋ausgabe]]

schalter Sie können nur einen der Schalter angeben.

kein Schalter

Datei ausgeben und Wiederholungen weglassen.

u Nur nicht wiederholte Zeilen ausgeben, d.h. uniq läßt alle mehrfachen Zeilen ganz weg.

d Nur die mehrfachen Zeilen auflisten, d.h. uniq gibt von den mehrfachen Zeilen jeweils eine aus.

c Anzahl der Wiederholungen ausgeben. uniq löscht mehrfache Zeilen bis auf eine (wie Standard) und schreibt vor jede Zeile, wie oft sie in der Eingabe nacheinander vorkommt. Die Anzahl steht rechtsbündig bis Spalte 4, der Zeileninhalt steht ab Spalte 6.

-n Die ersten n Felder jeder Zeile *nicht* berücksichtigen beim Vergleichen. Ein Feld ist eine nichtleere Zeichenfolge, die vom Nachbarfeld durch ein Leerzeichen oder ein Tabulatorzeichen getrennt ist. Das Leerzeichen gehört zum folgenden Feld.

uniq

+ m	Die ersten m Zeichen ab Zeilenanfang bzw. ab Feld n + 1 nicht berücksichtigen beim Vergleichen.
eingabe	Name der Eingabedatei. <i>Standard (keine Angabe):</i> uniq liest von der Standard-Eingabe.
ausgabe	Name der Ausgabedatei. <i>Standard (keine Angabe):</i> Standard-Ausgabe.

Beispiele

1. Durchsuchen einer Datei nach gleichen Zeilen unabhängig davon, wo sie in der Datei stehen. Für jede dieser Zeilen ist auszugeben, wie oft sie vorkommt.

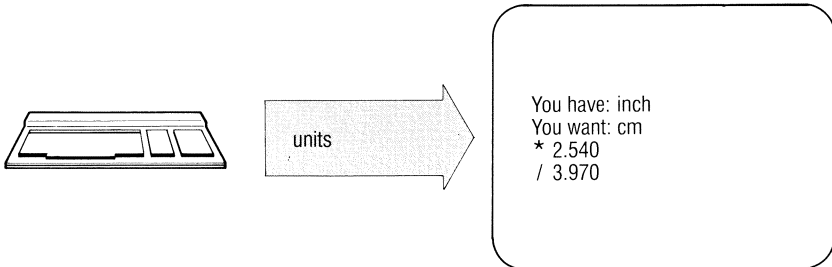
```
sort datei | uniq -c
```

2. Auszugeben sind die 10 häufigsten Wörter eines Textes. Das leistet folgende Prozedur:

```
prep $* |      # Dateien ausgeben, ein Wort je Zeile
sort |         # sortieren
uniq -c |      # mehrfache Zeilen suchen mit Anzahl
sort +0nr |    # sortieren nach Anzahl, größte Zahl zuerst
head          # die ersten 10 ausgeben
```

> > > > comm, sort

Einheiten umrechnen



units berechnet Umrechnungsfaktoren zwischen von Ihnen angegebenen Einheiten, z.B. inch in cm.

units

Arbeitsweise

Nach Eingabe des Kommandos fordert units Sie auf, die erste der beiden Einheiten anzugeben. Geben Sie also z.B. inch ein.

You have: inch

Nun fordert units Sie auf, die Einheit, in die Sie die erste umrechnen wollen, anzugeben. Das sei hier cm.

You want: cm

Als Umrechnungsfaktoren gibt units aus:

```
* 2.54000e+00
/ 3.93701e-01
```

d.h. Sie müssen 'inch' mit 2.54 multiplizieren, um 'cm' zu erhalten, und 'cm' mit 0.393701 multiplizieren, um 'inch' zu erhalten.

Als Einheiten können Sie alle units bekannten Zeichenfolgen angeben. units versteht die meisten internationalen und angelsächsischen Einheiten, allerdings keine deutschen Spezialeinheiten wie z.B. Pfund.

Auch einige Konstante erkennt units, z.B:

pi die Konstante Pi
c Lichtgeschwindigkeit
e Elektronenladung
g Gravitationskonstante

Bei der Eingabe einer Einheit können Sie diese durch Vorstellen einer ganzzahligen Zahl multiplizieren oder durch Anfügen einer positiven ganzen Zahl potenzieren. Ebenso können Sie zwei Einheiten durch den Operator '/' dividieren, z.B. m/sec oder km/hour

Zusammengesetzte Einheiten werden auch zusammengesrieben, z.B. 'lightyear'. Britische Einheiten, die nicht mit den entsprechenden amerikanischen übereinstimmen, müssen Sie durch Vorstellen von 'br' kennzeichnen, also z.B. 'brgallon'.

Währungen werden benannt:

belgiumfranc
germanymark oder auch nur mark usw.

Währungen berechnet units nach den \$-Kursen, die in der Datei /usr/lib/units angegeben sind (eingetragen ist der Stand von 1978!). Diese Angaben können Sie in dieser Datei ändern (nur Systemverwalter).

units führt nur multiplikative Umrechnungen durch. Sie können also z.B. nicht Grad Celsius in Grad Fahrenheit umrechnen.

Eine Liste aller units bekannten Einheiten mit Umrechnungsfaktoren finden Sie in der Datei: /usr/lib/units.

Beispiel

Wieviele km/h sind 240 m/sec?

\$ units

437 units; 3191 bytes

you have: 240 m/sec

you want: km/hour

* 8.64000e+02

/ 1.15741e-03

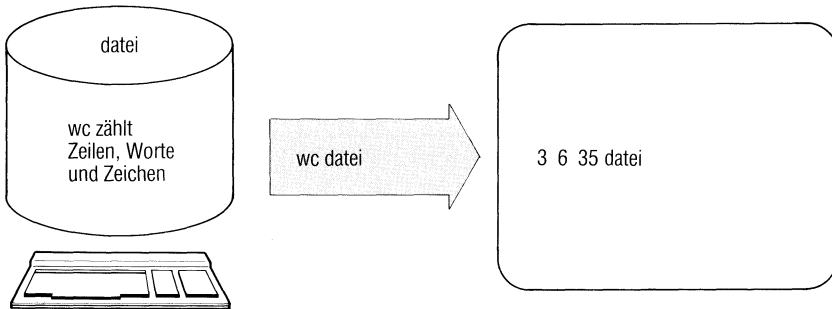
you want:

END

\$

Ergebnis: 864 km/h

Zeilen, Worte und Zeichen zählen – word counter



Das wc-Kommando zählt die Anzahl der Zeilen, Worte und Zeichen von Dateien.

`wc[_-schalter][_datei...]`

schalter

kein Schalter angegeben

wc gibt drei Zahlenwerte aus für die Anzahl der
Zeilen Worte Zeichen.

Bei mehreren Dateien summiert wc die Werte.

- l wc zählt Zeilen (lines). Die Anzahl ermittelt wc aus der Anzahl der Zeichen "neue Zeile".
- w wc zählt Worte (words), das sind nichtleere Zeichenmen-
gen, getrennt durch Zwischenräume.
- c wc zählt Zeichen. Dabei zählt wc Leerzeichen und Zeichen
"neue Zeile" mit.

datei Name der zu bearbeitenden Datei.

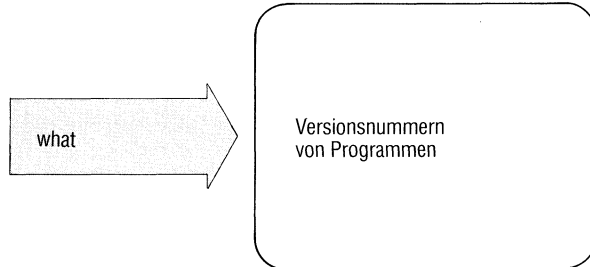
Standard (keine Angabe): wc liest von der Standard-Eingabe.

Beispiel

Für die Dateien logik, plan und rest ist die Anzahl der Zeilen, Worte und Zeichen auszugeben.

```
$ wc logik plan rest
   27   139   1077 logik
    5    15    140 plan
    3     6     51 rest
   35   160   1268 summe
$
```

Versionsnummern ausgeben



what gibt die Versionsnummern von Kommandos oder anderen Programmen aus. Diese Information ist wichtig für den System-Kundendienst, falls ein Software-Fehler in SINIX auftritt.

what_datei

datei Name des Kommandos.

Hinweis

what kann nur der Systemverwalter anwenden wegen der Zugriffsrechte für die Kommandodateien.

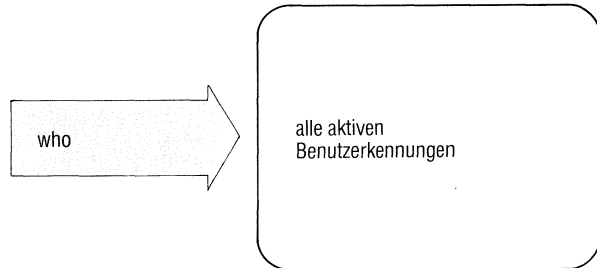
what ist ein Kommando des SCCS im C-Entwicklungssystem.

Beispiel

Version des Kommandos far ausgeben:

```
#what /bin/far
far:
    tar.c    1.4 84/05/18
#
```

Aktive Benutzerkennungen anzeigen – who is in the system



who gibt alle Benutzerkennungen aus, die zur Zeit aktiv sind. Das sind die, die mit einer Datensichtstation verbunden sind.

who[am]

nichts angegeben

who listet die aktiven Benutzerkennungen auf(siehe Beispiel).

am i

who gibt die Benutzerkennung aus, unter der Sie eingeschlossen sind (siehe Beispiel).

Ende-Status: immer 0.

Hinweis

Bei Einplatzsystemen kann zu einem Zeitpunkt nur eine Benutzerkennung aktiv sein (an der Datensichtstation console).

who

Beispiel

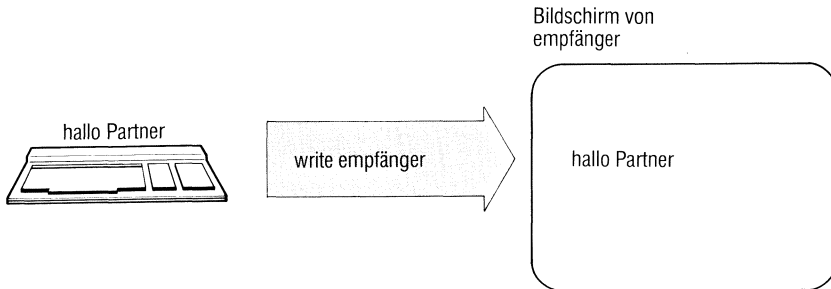
Sie möchten wissen, welchen Benutzern Sie mit dem Kommando `write` eine Nachricht senden können:

```
$ who
art      tty01   May  4 15:07
richter  tty03   May  4 12:56
```

Sie möchten wissen, unter welcher Kennung Sie arbeiten. Dann geben Sie ein:

```
$ who am i
art      tty01   May  4 15:07
```

Dialog mit anderen Benutzern



write sendet Nachrichten direkt an einen anderen Benutzer. Der Empfänger muß aktuell angeschlossen sein. Bei Einplatzsystemen ist das Kommando daher nicht sinnvoll.

write[_empfänger[_station]]

empfänger Benutzererkennung des Empfängers.

station Name der Datensichtstation des Empfängers (siehe tty).
Der Name ist anzugeben, wenn mehrere Datensichtstationen gleichzeitig unter derselben Benutzererkennung arbeiten.

write überträgt Zeilen von Ihrer Datensichtstation zur Datensichtstation des Empfängers. Vor der ersten Nachrichtenzeile gibt write beim Empfänger aus:

Nachricht von absender datenstation

Der Empfänger kann nun seinerseits mit write antworten.

write überträgt jede geschriebene Zeile an den Partner. Sie können natürlich auch mehrere Zeilen nacheinander absenden.

Sobald beide Partner ein write-Kommando gegeben haben, können sie einen Dialog in beiden Richtungen führen.

Damit der Dialog in geordneten Bahnen verläuft, können Sie ein Zeichen für das Ende einer Nachricht vereinbaren, ebenso ein Zeichen für das Ende des Dialogs.

Dialog beenden:

Taste drücken beendet write. Beim anderen Partner zeigt write dann *** ENDE *** an. Jeder Partner muß sein write-Kommando selbst beenden.

Kommandos ausführen, ohne den Dialog abubrechen:

!kommando Zeilen, die mit einem "!" beginnen interpretiert write als Kommandozeilen. kommando wird ausgeführt.

write-Nachrichten unterbinden:

Mit dem Kommando mesg können Sie verhindern, daß Sie Nachrichten erhalten, z.B. wenn Sie nicht bei Ihrer Arbeit unterbrochen werden wollen.

Beispiel

Datensichtstation 1

```
$ write peter
Guten Morgen, du Schlafmuetze
wenn du essen gehst, hol mich ab
in Ordnung ?
Nachricht von peter tty02...
geht in Ordnung
*** ENDE ***

```

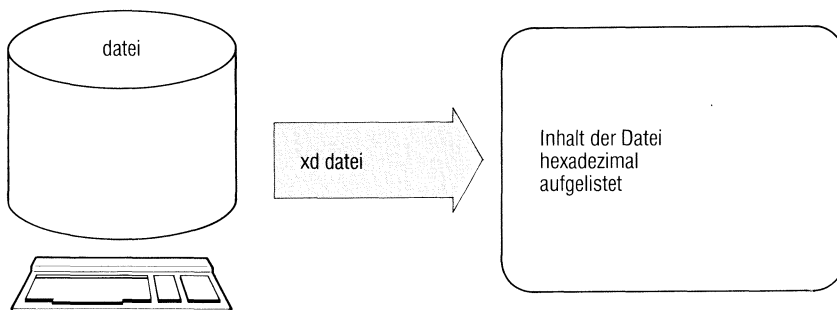
Datensichtstation 2

```
Nachricht von hans tty00...
Guten Morgen, Du Schlafmuetze
wenn du essen gehst, hol mich ab
in Ordnung ?
$ write hans
geht in Ordnung

*** ENDE ***
```

> > > > mesg, who, mail

Dateiinhalte hexadezimal ausgeben



Hilft Ihnen weder das `cat`-Kommando noch ein Editor weiter, um den Inhalt einer Datei zu identifizieren, bleibt Ihnen nur noch das `xd` Kommando. Es listet den Inhalt der angegebenen Dateien hexadezimal und als Buchstabenfolge auf.

```
xd[_dateiname...][_ [+ ]offset1[.[b]]][_offset2[.[b]]]
```

dateiname Name der auszugebenden Datei (auch Dateiverzeichnisse möglich).

Standard (keine Angabe): `xd` liest von der Standard-Eingabe.

+ Geben Sie keinen Dateinamen, aber einen offset an, müssen Sie vor diesem das Zeichen ”+” angeben.

offset1 bewirkt, daß `xd` erst an der Position `offset1` mit der Ausgabe beginnt. `xd` interpretiert `offset1` als Hexadezimalzahl.

Standard: Ausgabe ab Dateianfang.

. Fügen Sie direkt an `offset` einen Punkt an, so interpretiert `xd` `offset` als Dezimalzahl.

b Fügen Sie an den ”.” den Buchstaben `b` an, so interpretiert `xd` `offset` dezimal in Einheiten von 512 Byte Blöcken.

offset2 bezeichnet das Ende der Ausgabe. Ist offset2 kleiner oder gleich der Angabe offset1, gibt xd bis zum Dateiende aus. offset2 wird angegeben wie offset1 (auch ”.” und b).

Standard: Dateiende

Arbeitsweise

xd listet den Inhalt der angegebenen Dateien Zeichen für Zeichen hexadezimal auf. Die Darstellung entspricht dem Wert des jeweiligen Zeichens in der ASCII-Tabelle (siehe Anhang).

In der ersten Spalte steht hexadezimal die Anzahl der bis zu dieser Zeile ausgegebenen Bytes. Darauf folgen 4 Blöcke zu je 8 Buchstaben, die je vier Bytes hexadezimal darstellen. In der letzten Spalte finden Sie die 16 Bytes als Buchstaben dargestellt, nicht druckbare Zeichen als Leerzeichen. xd ist z.B. dann hilfreich, wenn Sie den Inhalt einer Datei lesen wollen, die Sie mit anderen Kommandos nicht vernünftig lesen können, oder wenn Sie die Ausgabe eines Programms nicht interpretieren können und vermuten, daß sie nicht druckbare Zeichen enthält.

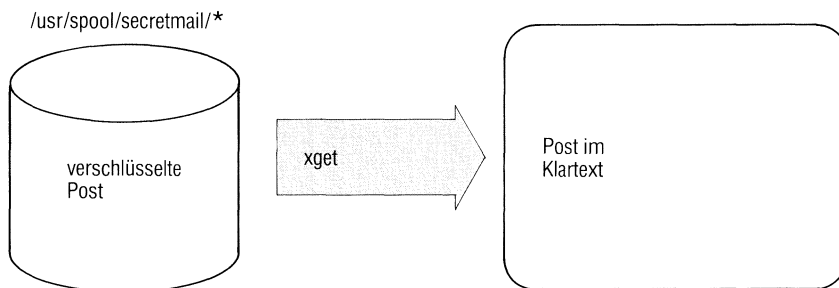
Beispiel

Gibt man z.B. xd als Eingabe den ersten Abschnitt dieser Seite, erhalten Sie die Ausgabe:

0	78640a09	48696c66	74204968	6e656e20	xd Hilft Ihnen
10	77656465	72206461	73206361	74204b6f	weder das cat Ko
20	6d6d616e	646f206e	6f636820	65696e20	mmando noch ein
30	45646974	6f722077	65697465	722c2075	Editor weiter, u
40	6d206465	6e0a0949	6e68616c	74206569	m den Inhalt ei
50	6e657220	44617465	69207a75	20696465	ner Datei zu ide
60	6e746966	697a6965	72656e2c	20626c65	ntifizieren, ble
70	69627420	49686e65	6e206e75	72206e6f	ibt Ihnen nur no
80	63680a09	64617320	7864204b	6f6d6d61	ch das xd Komma
90	6e646f2e	20457320	6c697374	65742064	ndo. Es listet d
a0	656e2049	6e68616c	74206465	7220616e	en Inhalt der an
b0	67656765	62656e65	6e204461	74656965	gegebenen Dateie
c0	6e0a0968	65786164	657a696d	616c2075	n hexadezimal u
d0	6e642061	6c732042	75636873	74616265	nd als Buchstabe
e0	6e666f6c	67652061	75662e0a		nfolge auf.

Das Ende einer Zeile erkennen Sie am Zeichen ’0a’ in den vier hexadezimalen Spalten. ’0a’ (dezimal 10) ist der Wert des Zeichens ”neue Zeile”.

Geheime Post lesen



xget liest verschlüsselte Nachrichten, die mit xsend gesendet wurden. xget arbeitet wie mail, fordert aber einen Schlüssel zum Entschlüsseln der Nachricht an. Diesen Schlüssel müssen Sie mit enroll festgelegt haben, bevor Ihnen mit xsend eine Nachricht zugestellt werden kann.

xget

xget meldet: "Geben Sie Ihren Schlüssel an". Sie geben den Schlüssel ein, den Sie mit enroll festgelegt haben und schließen die Eingabe mit ab. Die Eingabe ist nicht sichtbar.

Anschließend gibt xget die eingetroffenen Nachrichten der Reihe nach aus, ähnlich wie mail.

Nach jeder Nachricht gibt xget ein Fragezeichen aus. Sie haben folgende Antwortmöglichkeiten:

ein beliebiges Zeichen, außer die unten beschriebenen nächste Nachricht ausgeben.

oder d oder n

Nachricht löschen und nächste Nachricht ausgeben.

q oder END

xget beenden

s_[datei] oder w_[datei]
schreibt die Nachricht unverschlüsselt in die angegebene Datei.

Standard für datei: mbox

!kommando

Das angegebene Kommando wird ausgeführt.

Nach jeder Eingabe müssen Sie noch drücken.

Beispiel

Lesen geheimer Post:

\$ xget

Geben Sie Ihren Schlüssel an: eva

Von sissi Thu Jun 14 07:40:42 1984

Weißt du schon, daß der H. dem R. gesagt hat, daß die S.
auch schon Bescheid weiß, das weiß ich von A.

?

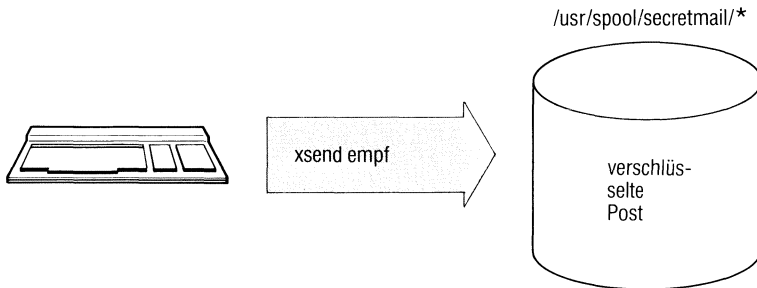
END

\$

Nach Ausgabe dieser Nachricht hat der Benutzer xget beendet. Der Schlüssel (eva) ist beim Eintippen nicht sichtbar.

> > > > enroll, mail, xsend

Geheime Post senden



xsend sendet Nachrichten an einen anderen Benutzer, ebenso wie mail. Die Nachrichten werden jedoch verschlüsselt abgelegt. Dazu muß der Empfänger irgendwann vorher mit `enroll` einen Schlüssel festgelegt haben. Mit `xget` kann er die Nachricht lesen, wenn er dabei diesen Schlüssel angibt.

xsend_Lempfänger

empfänger Benutzerkennung des Empfängers. Sie können nur einen Empfänger angeben.

xsend gibt die Meldung aus: "Warnung: Die Schlüsseldatei des Adressaten kann verändert werden". Dann liest xsend die Nachricht von der Standard-Eingabe. Beenden Sie die Eingabe der Nachricht mit der Taste `END`.

Hinweis

- Falls der Empfänger keinen Schlüssel festgelegt hat, können Sie ihm keine geheime Post senden. xsend meldet dann: "Der Adressat ist nicht eingetragen".
- Der Empfänger wird mit einer "normalen Post" benachrichtigt, daß er geheime Post erhalten hat.

Beispiel

1. Sie möchten eine geheime Post an den Benutzer peter senden:

```
$ xsend peter
```

Warnung: Die Schlusseldatei des Adressaten kann veraendert werden. Weißt du schon, daß der H. dem R. gesagt hat, daß die S. auch schon Bescheid weiß, das weiß ich von A.

```
END
```

```
$
```

2. Die zu versendende Nachricht steht in der Datei g.post:

```
$ cat g.post | xsend peter
```

Warnung: Die Schlusseldatei des Adressaten kann veraendert werden

```
$
```

> > > > enroll, mail, xget

Anhang

Inhalt

Ausdrücke	A-2
Reguläre Ausdrücke	A-3
Erweiterte reguläre Ausdrücke	A-4
Die ASCII-Zeichen	A-5

Ausdrücke

Folgende Kommandos verarbeiten Ausdrücke:

reguläre Ausdrücke

- ed
- expr
- grep
- more
- page
- sed

erweiterte reguläre Ausdrücke

- awk
- egrep

Bitte herausklappen

Reguläre Ausdrücke

	Syntax	Interpretation	Beispiel
	Ein regulärer Ausdruck ist:	der linksstehende reguläre Ausdruck bezeichnet:	regulärer Ausdruck: passende Zeichenfolge
1	z z jedes Zeichen außer: \, [,], ., *, -, \$, /, ^	das entsprechende Zeichen	a a
2	\z z jedes Zeichen außer: (,), 1, 2, 3, 4, 5, 6, 7, 8, 9, 0	das entsprechende Zeichen (Entwertung von Sonderzeichen)	\a a * *
3	.	ein beliebiges Zeichen (Gesamtzeichenvorrat)	. jedes Zeichen nach ASCII
4a	[s] s Zeichenfolge, in der "]" höchstens als erstes Zeichen vorkommt oder s Intervall der Form a-b, wobei a < b nach der ASCII-Tabelle	ein Zeichen, das in s vorkommt (Teilmenge)	[af] a oder f [a-c] a, b oder c
4b	[^s]	ein Zeichen, das nicht in s vorkommt (Komplement) Hinweis: in s hat \ keine besondere Bedeutung	[^\.&] jedes Zeichen außer \, ., & [^a-z] alle Zeichen außer Kleinbuchstaben
5	r* r regulärer Ausdruck der Form 1-4	eine Folge von null oder mehr passenden Zeichenreihen für r	m* ┐ oder m, mm, mmm, mmmm mmmm, ... usw
6	xy x, y reguläre Ausdrücke	eine passende Zeichenreihe für x gefolgt von einer passenden Zeichenreihe für y (Konkatenation)	.k bel. Zeichenk z.B. ok [afg]z az oder fz oder gz
7a	^r r reg. Ausdruck der Form 1-8	eine Interpretation von r die am Anfang der Zeile vorkommt	^Von Von am Anfang einer Zeile
7b	r\$	eine Interpretation von r die am Zeilenende vorkommt	Rand\$ Rand am Ende einer Zeile

	Syntax	Interpretation	Beispiel
	Ein regulärer Ausdruck ist:	der linksstehende reguläre Ausdruck bezeichnet:	regulärer Ausdruck: passende Zeichenfolge
8a	\(r\) r reg. Ausdruck, in dem höchstens vier "\" vorkommen	dieselben Zeichenfolgen wie r (Markierung von r)	\(haus\) haus \(a[12]\) a1 oder a2
		Bemerkung: Die öffnenden \(in einem reg. Ausdruck werden mit 1 beginnend von links nach rechts durchnummeriert	
8b	x\n 1 ≤ n ≤ 5, x reg. Ausdruck, in dem ein mit dem n-ten \(\) Paar geklammerter Teilausdruck r vorkommt	diesselben Zeichenfolgen wie r (Wiederholung eines markierten Ausdruckes)	\(haus\(tür\)\)\2 tür

Erweiterte reguläre Ausdrücke

Zusätzlich zu den acht Regeln für reguläre Ausdrücke gelten die folgenden Regeln.

	Syntax	Interpretation	Beispiel
9a	r*	eine Folge von null oder mehr Interpretationen von r	'(ok)*': ┐, ok, okok, okokok, ...
9b	r+	r beliebiger regulärer Ausdruck	'(ok)+': ok, okok, okokok, ...
9c	r?	eine Folge von null oder einer Interpretation von r	'(ok)?': ┐, ok
10	u v u NL v	u, v reg. Ausdrücke (Alternative)	'to(day morrow)': today oder tomorrow
11	(r)	r regulärer Ausdruck	'(a b c)': a oder b oder c

Die ASCII-Zeichen

dezi- mal	oktal	hexa- dez.		Bedeutung	Control
0	00	00	NUL	Null, keine Operation	Ⓢ
1	01	01	SOH	Start of Heading Vorspannanfang	A
2	02	02	STX	Start of Text Textanfang	B
3	03	03	ETX	End of Text Textende	C
4	04	04	EOT	End of Transmission Übertragungsende	D
5	05	05	ENQ	Enquiry	E
6	06	06	ACK	Stationsanruf Acknowledge Bestätigung	F
7	07	07	BEL	Bell Klingel	G
8	10	08	BS	Backspace Korrekturtaste	H
9	11	09	HT	Horizontal Tabulation Tabulatorzeichen	I
10	12	0A	LF	Line Feed Zeilenvorschub, neue Zeile	J
11	13	0B	VT	Vertical Tabulation	K
12	14	0C	FF	Form Feed Formularvorschub	L
13	15	0D	CR	Carriage Return Wagenrücklauf	M
14	16	0E	SO	Shift Out Umschalten Zeichensatz	N
15	17	0F	SI	Shift In Zurückschalten Zeichensatz	O
16	20	10	DLE	Data Link Escape Austritt aus der Datenverbindung	P
17	21	11	DC1	Device Control 1 Gerätesteuerung 1, Ausgabe fortsetzen	Q
18	22	12	DC2	Device Control 2	R
19	23	13	DC3	Device Control 3 Ausgabe anhalten	S
20	24	14	DC4	Device Control 4	T
21	25	15	NAK	Negative Acknowledge Fehlermeldung	U
22	26	16	SYN	Synchronous Idle Synchronisierung	V
23	27	17	ETB	End of Transm. Block Datenblockende	W
24	30	18	CAN	Cancel ungültig, Zeilenlöscher	X
25	31	19	EM	End of Medium Datenträgerende, quit (Signal3)	Y
26	32	1A	SUB	Substitute Character Zeichen ersetzen	Z

ASCII

dezi- mal	oktal	hexa- dez.		Bedeutung	Control	
27	33	1B	ESC	Escape	Rücksprung	
28	34	1C	FS	File Separator	Dateitrennung	\
29	35	1D	GS	Group Separator	Gruppentrennung]
30	36	1E	RS	Record Separator	Satztrennung	^
31	37	1F	US	Unit Separator	Einheitentrennung	␣ oder DEL
32	40	20	SP	SPACE	Leerzeichen	
33	41	21	!			
34	42	22	"			
35	43	23	#	Nummernzeichen		
36	44	24	\$	oder nationales Währungssymbol		
37	45	25	%			
38	46	26	&			
39	47	27	'			
40	50	28	(
41	51	29)			
42	52	2A	*	(Stern gilt oft als Multiplikations- zeichen)		
43	53	2B	+			
44	54	2C	,			
45	55	2D	-			
46	56	2E	.			
47	57	2F	/	(dient als Divisionszeichen)		
48	60	30	0			
49	61	31	1			
50	62	32	2			
51	63	33	3			
52	64	34	4			
53	65	35	5			
54	66	36	6			
55	67	37	7			
56	70	38	8			
57	71	39	9			
58	72	3A	:			
59	73	3B	;			
60	74	3C	<			
61	75	3D	=			
62	76	3E	>			
63	77	3F	?			
64	100	40	@	(kaufmännisches "at" oder \$)		
65	101	41	A			
66	102	42	B			
67	103	43	C			
68	104	44	D			
69	105	45	E			
70	106	46	F			
71	107	47	G			
72	110	48	H			
73	111	49	I			
74	112	4A	J			
75	113	4B	K			
76	114	4C	L			
77	115	4D	M			

dezi- mal	oktal	hexa- dez .		Bedeutung	Control
78	116	4E	N		
79	117	4F	O		
80	120	50	P		
81	121	51	Q		
82	122	52	R		
83	123	53	S		
84	124	54	T		
85	125	55	U		
86	126	56	V		
87	127	57	W		
88	130	58	X		
89	131	59	Y		
90	132	5A	Z		
91	133	5B	[oder Ä	
92	134	5C	\	Gegenschragstrich oder Ö	
93	135	5D]	oder Ü	
94	136	5E	^	oder ↑	
95	137	5F	-	Unterstrich oder →	
96	140	60	`		
97	141	61	a		
98	142	62	b		
99	143	63	c		
100	144	64	d		
101	145	65	e		
102	146	66	f		
103	147	67	g		
104	150	68	h		
105	151	69	i		
106	152	6A	j		
107	153	6B	k		
108	154	6C	l		
109	155	6D	m		
110	156	6E	n		
111	157	6F	o		
112	160	70	p		
113	161	71	q		
114	162	72	r		
115	163	73	s		
116	164	74	t		
117	165	75	u		
118	166	76	v		
119	167	77	w		
120	170	78	x		
121	171	79	y		
122	172	7A	z		
123	173	7B	{	oder ä	
124	174	7C		oder ö	
125	175	7D	}	oder ü	
126	176	7E	~	oder ß	
127	177	7F	DEL	Delete Löschenzeichen, Interrupt (Signal2)	

Fachwörter deutsch - englisch

Administrator	super user
aktuelles Dateiverzeichnis	working directory
aktuelles Umfeld	current environment
Anfangszeile	head line
Apostrophen-Mechanismus	quoting mechanism
Ausführungserlaubnis	execute permission
Benutzer	user
Benutzererkennung	user identification
Benutzernummer	user id (UID)
Bereit-Zeichen	prompt
Bildschirm	screen
Datei	file
Dateideskriptor	file descriptor
Dateisystem	file system
Dateiverzeichnis	directory
Datensichtstation	terminal
Diskette	floppy
Eigentümer	owner
Eingabeaufforderungszeichen	prompt
eingebaute Kommandos	built-in commands
Ende-Status	exit status
Feld	array, field
Feldtrenner	field separator
Festplatte	disk
Freispeicherliste	free list
Gedächtnisstütze	reminder
Gegenschrägstrich	backslash
Geräte-datei	special file
geschweifte Klammern	braces/curly braces
Gruppenname	group name
Gruppennummer	group id (GID)
Hintergrundprozess	background-process
Home-Dateiverzeichnis	home-Directory
Indexeintrag	INODE
Indexnummer	inumber
Kennwort	password
Kennwortparameter	user-defined variable, keyword parameter
Konsole	console
Korrekturtaste	back space key
Laufzeit	elapsed time
Leeres Kommando	null command
Leerzeichen	blank
Leseerlaubnis	read permission
Login-Dateiverzeichnis	login-Directory
Metazeichen	meta character
Muster	pattern

neue Zeile	new line
Pfad	path
Pfadname	pathname
Prozeß	process
Pipeline	pipeline
Prozedur	procedure,script
Punkt-Kommando	dot command
Quellcode	source code
regulärer Ausdruck	regular expression
Root-Dateiverzeichnis	root directory
runde Klammern	parentheses
Schalter	option
Schreiberlaubnis	write permission
Schreibmarke	cursor
Schutzbit	protection bit
Shell	shell
Shell-Umgebung	environment
Sonderzeichen	special character
Standard-Ausgabe	standard output
Standard-Eingabe	standard input
Stellungsparameter	positional parameter
Suchzeichenfolge	search string
Systemkern	kernel
Systemverwalter	super user/administrator
Tastatur	keyboard
umleiten	redirection
Vergleichsausdruck	relational expression
Verweis	link
Wagenrücklauf	carriage return
Zeichenfolge	string
Zeilentrenner	record separator
Zugriffsrecht	mode

Fachwörter englisch - deutsch

array/field	Feld
back space key	Korrekturtaste
background-process	Hintergrundprozeß
backslash	Gegenschrägstrich
blank	Leerzeichen
braces/curly braces	geschweifte Klammern
built-in commands	eingebaute Kommandos
carriage return	Wagenrücklauf
console	Konsole
current environment	aktuelles Umfeld
cursor	Schreibmarke
directory	Dateiverzeichnis
disk	Festplatte
dot command	Punkt-Kommando
elapsed time	Laufzeit
environment	Shell-Umgebung
execute permission	Ausführungserlaubnis
exit status	Ende-Status
field separator	Feldtrenner
file	Datei
file descriptor	Dateideskriptor
file system	Dateisystem
floppy	Diskette
free list	Freispeicherliste
group identification (GID)	Gruppennummer
groupname	Gruppenname
head line	Anfangszeile
home-Directory	Home-Dateiverzeichnis
INODE	Indexeintrag
inumber	Indexnummer
kernel	Systemkern
keyboard	Tastatur
keyword parameter	Kennwortparameter
link	Verweis
login-Directory	Login-Dateiverzeichnis
meta character	Metazeichen
mode	Zugriffsrecht
new line	neue Zeile
null command	Leeres Kommando
option	Schalter
owner	Eigentümer
parentheses	runde Klammern
password	Kennwort
path	Pfad
pathname	Pfadname
pattern	Muster
pipeline	Pipeline

positional parameter	Stellungsparameter
procedure	Prozedur
process	Prozeß
prompt	Bereit-Zeichen
prompt	Eingabeaufforderungszeichen
protection bit	Schutzbit
quoting mechanism	Apostrophen-Mechanismus
read permission	Leseerlaubnis
record separator	Zeilentrenner
redirection	umleiten
regular expression	regulärer Ausdruck
relational expression	Vergleichsausdruck
reminder	Gedächtnisstütze
root Directory	Root-Dateiverzeichnis
screen	Bildschirm
script	Prozedur
search string	Suchzeichenfolge
shell	Shell
source code	Quellcode
special character	Sonderzeichen
special file	Geräte-datei
standard input	Standard-Eingabe
standard output	Standard-Ausgabe
string	Zeichenfolge
super user/administrator	Systemverwalter
super-user	Administrator
terminal	Datensichtstation
user	Benutzer
user identification (UID)	Benutzernummer
user-defined variable	Kennwortparameter
working directory	aktuelles Dateiverzeichnis
write permission	Schreiberlaubnis

Literatur

Betriebssystem SINIX
Buch 2
Menüs
Bestellnummer: U1902-J-Z95-3

Betriebssystem SINIX
TRANSIN für PC-X
Bestellnummer: U1903-J-Z95-2

TRANSIN für PC-MX
Bestellnummer: U2303-J-Z95-1

Folgende Bücher haben wir beim Erstellen dieses Buchs verwendet:

R. Thomas und J. Yates
A User Guide to the Unix System
Berkley: Osborne/Mc Graw-Hill 1982

UNIX Time-Sharing System. UNIX Programmer's Manual Vol.1
Seventh Edition, Murray Hill: Bell Telephone Laboratories 1979

UNIX Time-Sharing System. UNIX Programmer's Manual, Vol. 2A
Seventh Edition, Murray Hill: Bell Telephone Laboratories 1979

UNIX Time-Sharing System. UNIX Programmer's Manual, Vol. 2B
Seventh Edition, Murray Hill: Bell Telephone Laboratories 1979

R. Gauthier
Using the UNIX System
Reston Publishing Co. (Prentice Hall) 1981

D.M. Ritchie
The UNIX Time-Sharing System: A Retrospective
Bell sys. TEch. J. 57(6) 1947-69, 1978

M. Banaham, A. Reuter
UNIX-the book
Wilmslow, UK, Sigma Technical Press, 1982

B.W. Kernighan, D.M. Ritchie
Programmieren in C
Deutsche Ausgabe von Prof. Dr. A.T. Schreiner, Dr. Ernst Janich
Carl Hanser Verlag München, 1983

Stichwörter

Ablaufanweisungen 3-18, 3-35
Addition (dc) 6-82
Addition (expr) 6-121
Adresse (ed) 6-102f
Adresse (sed) 6-228
ändern (ed) 6-107
Aktion (awk) 6-27
aktive Benutzerkennungen (who) 6-293
aktuelle Zeile (ed) 6-101
aktuelle Zeilennummer (sed) 6-228
Aktuelles Dateiverzeichnis 2-4
aktuelles Dateiverzeichnis (cd) 6-39
Akzent Gravis 3-12
Anfangszeilen ausgeben (head) 6-138
anfügen (ed) 6-107
anfügen (sed) 6-229
anfügen an Musterspeicher (sed) 6-231
Anführungszeichen (6) 6-27
Angabe, wahlfreie (6) 6-27
anhängen an Haltespeicher (sed) 6-231
Anweisung (awk) 6-27
Anzahl der Dateien (quot) 6-218
Apostroph 3-3
Apostrophier-Mechanismus 3-12
Archiv (tar) 6-258
Archiv anlegen (tar) 6-259
Archiv lesen (tar) 6-260
Archivieren auf Band oder Diskette (tar) 6-258
Arithmetische Sprache (bc) 6-29
Art einer Datei (file) 6-129
auflisten (sed) 6-231
Aufräumarbeiten 5-29
Aufruf der Prozeduren 3-23
aufteilen (ed) 6-112
Auftragslage (lpr) 6-156
Ausdruck, regulärer (awk) 6-27
Ausdruck, regulärer (ed) 6-104
Ausdrücke (awk) 6-27
Ausdrücke auswerten (expr) 6-120
Ausführberechtigung 2-20
Ausführung verzögern (sleep) 6-238
Ausführungserlaubnis (chmod) 6-59
Ausführungszeitpunkt festsetzen (um) 6-27
Ausführungszeitpunkt festsetzen (at) 6-283
Ausgabe anhalten (6) 6-27
Ausgabe fortsetzen (6) 6-27

Ausgabeoperationen des Dateisystems (sync) 6-254
ausgeben (ed) 6-111
ausgeben (ed) 6-110
ausgeben (sed) 6-232
Ausgeben auf Standard-Ausgabe und Datei (tee) 6-269
ausgeben bis "neue Zeile" (sed) 6-232
ausgeben, mehrspaltig (pr) 6-199
ausgeben, Zeichenfolgen (echo) 6-97
austauschen (sed) 6-234

Bandarchiv (tar) 6-258
Baumstruktur 2-2
Bedienbereich (ced) 6-44
Bedingung "falsch" (false) 6-123
Bedingung "wahr" (true) 6-280
Bedingungen (awk) 6-27
Bedingungen (find) 6-134
Bedingungen prüfen (test) 6-271
Bedingungen verknüpfen (test) 6-274
beenden (sed) 6-233
beenden einer ced-Sitzung (ced) 6-53
Begrüßungsbildschirm 1-1, 5-33
Benutzergruppe (chgrp) 6-57
Benutzergruppe wechseln (newgrp) 6-186
Benutzerkennung 1-1, 1-2, 1-6, 2-15, 2-18, 5-1
Benutzerkennung (ls) 6-161
Benutzerkennung (passwd) 6-195
Benutzerkennung definieren 5-12
Benutzerkennung guest 1-2
Benutzerkennung vorübergehend wechseln (su) 6-249
Benutzerkennung wechseln (login) 6-148
Benutzerklassen 2-20
Benutzernummer 2-14, 5-1
Benutzernummer (ls) 6-162
Benutzernummer (ps) 6-208
Benutzernummer (UID) 2-25, 2-27, 5-13
Benutzer- und Gerätedateien kopieren (copy) 6-67
Bereich (awk) 6-27
Bereich (sed) 6-228
Bereich kopieren (ced) 6-47
Bereich löschen (ced) 6-47
Bereich speichern (ced) 6-47
Bereich zurückholen (ced) 6-47
Bereit-Zeichen (6) 6-27
Beschreibung der Modi (ced) 6-47
Bildschirmausgabe steuern (more) 6-178
Bildschirminhalt neu ausgeben (ced) 6-56
Bildschirmorientiert 4-2
bildschirmorientierter Editor (ced) 6-41

Block verschieben 4-13
Block-Geräte-datei (find) 6-132
Blockungsfaktor (tar) 6-262
Blöcke 5-8
Briefkasten (mail) 6-164
Byte-Geräte-datei (find) 6-132

Cbreak-Modus (stty) 6-247
CED vorübergehend unterbrechen 4-36
CED-Editor 2-5, 4-1
CED-Sitzung beenden 4-32
ced-Sitzung, beenden (ced) 6-53
Console 5-1
core (pstat) 6-216
CPU-Zeit (time) 6-276

Darstellung (6) 6-27
Datei 4-1
Datei .profile und /etc/profile 2-7
Datei /etc/group 5-14
Datei /etc/passwd. 5-12
Datei aufbereiten zum Drucken (pr) 6-197
Datei aufteilen (split) 6-244
Datei erzeugen 2-5
Datei für ein Gerät 2-19
Datei kopieren 2-2
Datei löschen 2-6
Datei sichern 4-30
Datei übertragen 2-2
Datei, aktuelle (ced) 6-47
Datei, Eigenschaften (test) 6-272
Datei, physikalisch 2-2
Dateiänderung, letzte (find) 6-132
Dateiänderung, letzte (ls) 6-161
Dateiänderung, letzte (settime) 6-236
Dateiänderung, letzte (tar) 6-262
Dateiänderung, letzte (touch) 6-277
Dateianfang (ced) 6-56
dateideskriptor (test) 6-273
Dateideskriptoren 5-32
Dateien am Drucker ausdrucken (lpr) 6-15
Dateien archivieren (far) 6-125
Dateien ausdrucken (print) 6-203
Dateien ausgeben (cat) 6-37
Dateien ausgeben, nacheinander (cat) 6-3
Dateien bestimmter Größe (quot) 6-218
Dateien durchblättern (more) 6-178
Dateien durchsuchen (egrep) 6-116
Dateien durchsuchen (fgrep) 6-126

Dateien durchsuchen (grep) 6-135
Dateien für Geräte 2-9
Dateien gruppenweise kopieren (copy) 6-67
Dateien in ein anderes Dateiverzeichnis übertragen (mv) 6-184
Dateien kopieren (copy) 6-71
Dateien löschen 2-11
Dateien löschen (rm) 6-221
Dateien markieren (ls) 6-162
Dateien mischen (sort) 6-239
Dateien parallel bearbeiten (ced) 6-48
Dateien umbenennen (mv) 6-183
Dateien unterscheiden (dateityp) 6-77
Dateien unterscheiden (file) 6-129
Dateien verbinden nach Vergleichsfeldern (join) 6-139
Dateien vergleichen, drei (diff3) 6-91
Dateien vergleichen, zwei (diff) 6-87
Dateien verwalten (make) 6-168
Dateien zeichenweise vergleichen (cmp) 6-63
Dateien zeilenweise vergleichen (diff) 6-87
Dateien, Anzahl (quot) 6-218
Dateien, Informationen (ls) 6-160
Dateien, offene (pstat) 6-216
Dateien, sortieren (sort) 6-239
Dateien, überschreiben (tar) 6-263
Dateigröße (ls) 6-161
Dateinhalt hexadezimal (xd) 6-297
Dateiname (ed) 6-101
Dateiname ohne Pfad (basename) 6-27
Dateiname, aktueller (ed) 6-108
Dateinamen 2-2, 2-11,
Dateinamen (6) 6-27
Dateinamen Sonderzeichen 2-9,
Dateinamen, abgekürzte (echo) 6-98
Dateinamen, Sonderzeichen (6) 6-27
Dateinamen, Sonderzeichen (tar) 6-264
Datei-starten 3-62
Dateisystem 2-1
Dateisystem (tar) 6-258
Dateisystem abgewiesen 5-26
Dateisystem abhängen 5-26
Dateisystem auf Diskette 5-19, 5-30
Dateisystem auf Diskette kopieren 5-6
Dateisystem prüfen auf Belegung (quot) 6-218
Dateisystem, Diskettenarchiv (far) 6-125
Dateityp (dateityp) 6-77
Dateityp (file) 6-129
Dateiverweise 2-13
Dateiverzeichnis 2-2
Dateiverzeichnis (ln) 6-146

Dateiverzeichnis /dev 2-9
Dateiverzeichnis /etc 5-2
Dateiverzeichnis /usr/lib 5-2
Dateiverzeichnis durchsuchen 3-36
Dateiverzeichnis einrichten (mkdir) 6-178
Dateiverzeichnis erzeugen 2-3
Dateiverzeichnis kennzeichnen 2-3
Dateiverzeichnis löschen 2-4
Dateiverzeichnis markieren (ls) 6-162
Dateiverzeichnis überprüfen 5-10
Dateiverzeichnis wechseln (cd) 6-39
Dateiverzeichnis, aktuelles (cd) 6-39
Dateiverzeichnis, aktuelles (pwd) 6-217
Dateiverzeichnis, Eigenschaften (test) 6-272
Dateiverzeichnis, Eigenschaften (ls) 6-161
Dateiverzeichnis, Home (cd) 6-39
Dateiverzeichnis, Login (cd) 6-39
Dateiverzeichnis, Zugriffsrechte (mkdir) 6-177
Dateiverzeichnisse durchsuchen (find) 6-131
Dateiverzeichnisse kopieren (copy) 6-67
Dateiverzeichnisse löschen (rmdir) 6-223
Dateiverzeichnisse umbenennen (mv) 6-183
Dateiverzeichnisse, Informationen (ls) 6-160
Dateizugriff, letzter (ind) 6-132
Dateizugriff, letzter (ls) 6-161
Dateizugriff, letzter (settime) 6-236
Daten eingeben (6) 6-27
Datensichtstation blockiert 5-44
Datum, deutsch (datum) 6-79
Datum, englisch (date) 6-75
Dialog mit anderen Benutzern (write) 6-295
Diskette (far) 6-125
Diskette formatieren 5-17
Disketten 2-9, 5-16
Diskettenarchiv (tar) 6-258
Diskettenlaufwerk 5-20
Diskettenlaufwerk (tar) 6-258
display (echo) 6-98
Division (dc) 6-82
Division (expr) 6-121
Dokument 4-1
Dokument wechseln 4-28, 4-29
Druckauftrag (lpr) 6-152
Druckauftrag löschen (lpr) 6-153
Druckauftrag, Zustand (lpr) 6-154
Drucker 2-9, 5-46
Drucker (lpr) 6-152
Drucker (print) 6-203
Drucker 9004 5-43

Drucker freigeben (lpr) 6-157
Drucker sperren (lpr) 6-157
Druckerbetrieb 5-39, 5-40
Druckerbetrieb (lpr) 6-157
Druckerverwaltung 5-40
Druckerverwaltung (lpr) 6-152
Druckerverwaltungsprozeß 5-42
Druckerzustände (lpr) 6-154
durchblättern von Dateien (more) 6-178

Echo-Modus (stty) 6-247
Eckpunkt markieren 4-15
Editor im Prozedurbetrieb (sed) 6-226
Editor, bildschirmorientierter (ced) 6-4
Editor, zeilenorientierter (ed) 6-100
ed-Script (diff3) 6-92
ed-Skript erstellen (diff) 6-87
Eigenschaften der Datensichtstation ändern (stty) 6-246
Eigenschaften von Datensichtstationen 5-44
Eigentümer (ls) 6-161
Eigentümer ändern (chown) 6-62
einfügen (ced) 6-48
einfügen (ed) 6-109
einfügen (sed) 6-231
einfügen von Leerzeilen (ced) 6-47
Eingabe (6) 6-27
Eingabe beendet 1-4
Eingabe korrieren 5-48
Eingabe prüfen 3-37
Eingabe, nächste (sed) 6-232
Eingabebereich festlegen 4-20, 4-25
Eingabefehler 1-1
Eingabefehler korrigieren 1-5
Eingabemodus (ed) 6-101
Eingabezeile (sed) 6-227
eingebaute Funktionen (awk) 6-27
eingeben oder ändern (ced) 6-41
Einheiten umrechnen (units) 6-287
einlesen (ed) 6-108
Empfangsbereitschaften 3-3
Endabschnitt einer Datei ausgeben (tail) 6-256
Ende-Status (6) 6-27
Ende-Status 0 (true) 6-280
Ende-Status 1 (false) 6-123
Ende-Status abfragen 3-43, 3-56
Ende-Status auswerten 3-57
Endlos-Schleife (false) 6-124
Endlos-Schleife (true) 6-280
entschlüsseln, Text (crypt) 6-73

Entschlüsselung (ed) 6-114
Entschlüsselungsmodus (ed) 6-100
Entwerten Sonderzeichen 2-12
Erinnerungsdienst (calendar) 6-35
Erinnerungsdienst (kalender) 6-142
ersetzen (ed) 6-112
ersetzen (sed) 6-233
ersetzen Haltespeicher (sed) 6-231
ersetzen im Musterspeicher (sed) 6-231
ersetzen Zeichen (sed) 6-234
Ersetzungszeichenkette (ed) 6-112
erweiterte reguläre Ausdrücke (egrep) 6-116
Exponentialrechnung (dc) 6-82
Exportieren 3-33

Fehlermeldung (lpr) 6-158
Fehlermeldungen (6) 6-27
Fehlermeldungen (ed) 6-115
Feld (join) 6-140
Feld (uniq) 6-285
Felder (awk) 6-27
Felder (sort) 6-240
Felder, arrays (awk) 6-27
Felder, assoziative (awk) 6-27
Feldtrenner (awk) 6-27
Fenster (ced) 6-44
Fenster positionieren (ced) 6-49
Fenster verschieben 4-35
Folgediskette (far) 6-125
Folgediskette (tar) 6-263
Form des Ausdrucks (lpr) 6-152
for-Schleife beenden 3-35
Funktionen (bc) 6-29
Funktionen, eingebaute (awk) 6-27
Funktionstasten 4-3, 4-13

Gedächtnisstütze (calendar) 6-35
Gedächtnisstütze (kalender) 6-142
geheime Post (enroll) 6-119
geheime Post lesen (xget) 6-299
geheime Post senden (xsend) 6-301
Geräteeintrag 5-41
Geräteklasse (Majornummer) 2-16
Gerätename 5-41
Gerätenummer (Minornummer) 2-16
Gesamt-Pfadnamen 2-6, 2-16, 2-17
Gleichheitszeichen 3-28
Groß- und Kleinbuchstaben 1-4
Großbuchstaben 1-2

Gruppe von Dateien (make) 6-168
Gruppen 2-19
Gruppenidentifikation (newgrp) 6-186
Gruppenkennwort 5-15
Gruppenname 5-15
Gruppenname (ls) 6-162
Gruppenname (newgrp) 6-186
Gruppennummer 2-14
Gruppennummer (GID) 2-25, 5-13, 5-15
Gruppennummer (ls) 6-162
Gruppennummer ändern (chgrp) 6-57

Hardwareuhr 5-35, 5-36
HELP-Taste 4-6
hexadezimal ausgeben (xd) 6-297
Hintergrundprozeß 3-5, 3-35
Hochfahren des Systems 5-32
HOME 2-7
Home-Dateiverzeichnis 2-4

Identifikation 2-15
Indexeinträge 2-13, 2-15, 5-8
Indexeintrag (find) 6-132
Indexeintrag (mv) 6-184
Indexeintrag-Liste 2-13
Indexeintrag-Tabelle (pstat) 6-212
Indexnummer 2-3, 2-13, 5-8, 5-9, 5-10
Indexnummer (find) 6-132
Indexnummer (ls) 6-161
Indexnummer (mv) 6-184
Inhaltsverzeichnis des Archivs (tar) 6-260
Inkonsistenzen 5-50
interner Zwischenspeicher 4-30

Kalender (cal) 6-33
Keller (dc) 6-84
Kennwort (login) 6-148
Kennwort (passwd) 6-195
Kennwort ändern 1-6
Kennwort definieren 5-12
Kennwort vergessen 5-46
Kennworte 1-1, 1-6, 2-18
Kennwortparameter 3-14, 3-20, 3-23
Kennwortparameter übergeben 3-66
kennzeichnen Datei 2-5
Kindprozeß 3-33
Kleinbuchstaben 1-2
Kommando ausführen (ced) 6-49
Kommando ausführen (find) 6-133

Kommando eingeben 1-4, 3-3
Kommando später ausführen (um) 6-283
Kommando, Laufzeit (time) 6-276
Kommandoabbruch (ed) 6-115
Kommandoeingabe abschließen 1-4
Kommando-Liste 3-4, 3-35
Kommandoliste (ed) 6-109
Kommandoname (6) 6-27
Kommandos ausführen 3-50, 3-51
Kommandos der Shell 3-4, 3-47
Kommandos übergeben 3-51
Kommandos zusammenfassen 3-49
Kommandozeile 3-3, 3-21, 3-28
Kommandozeile übergeben 3-68
Kommandozeilen innerhalb einer Prozedur 3-28
Kommentarzeile schreiben 3-48
Konfigurationsdatei 5-40, 5-41
Konfigurationsdatei (lpr) 6-157
Konstante (6) 6-27
Konstante (units) 6-288
Kopfzeile (ced) 6-44
Kopfzeile (print) 6-203
kopieren (ed) 6-113
kopieren, Bereich (ced) 6-47
kopieren, Dateien (copy) 6-68
kopieren, Dateiverzeichnisse (copy) 6-67
korrigieren (ced) 6-45
Kurzbeschreibung (ced) 6-46

Laufzeit eines Kommandos (time) 6-276
LED-Leuchtkette 5-49
Leeres Kommando (false) 6-123
leeres Kommando (sed) 6-235
Leerzeichen (6) 6-27
Leerzeilen, einfügen (ced) 6-47
Leseberechtigung 2-19
Leseerlaubnis (chmod) 6-59
lesen (sed) 6-233
lexikalisch sortieren (sort) 6-240
löschen (ed) 6-108
löschen (sed) 6-230
löschen bis "neue Zeile" (sed) 6-230
Löschen von temporären Dateien 5-32
löschen Zeichen (ced) 6-56
löschen Zeile (ced) 6-56
löschen, Bereich (ced) 6-47
löschen, Dateien (rm) 6-221
löschen, Dateiverzeichnisse (rmdir) 6-22
Login 1-2, 2-7, 2-18, 3-3

Login-Dateiverzeichnis 2-4, 2-7
Login-Dateiverzeichnis (su) 6-249
Login-Shell 3-20, 3-26, 3-33, 3-60, 5-37
logische Dateisysteme 5-7

Majornummer 5-28
makefile (make) 6-168
makefile, definieren (make) 6-169
Makro-Definitionen (make) 6-171
markieren (ed) 6-110
markieren (sed) 6-235
markieren einer Zeile (ced) 6-47
mehrfache Zeilen auflisten (uniq) 6-285
mehrfache Zeilen suchen (uniq) 6-285
Meldungen erzeugen (echo) 6-97
Meldungen verhindern oder erlauben (mesg) 6-174
minornummer 5-28
mischen von Dateien (sort) 6-239
Modifikation 2-15
Modulofunktion (dc) 6-82
Modulofunktion (expr) 6-121
Modus (ced) 6-47
Modus auswählen (ced) 6-46
Multiplikation (dc) 6-82
Multiplikation (expr) 6-121
Muster (awk) 6-27
Muster (egrep) 6-116
Muster (fgrep) 6-126
Muster (grep) 6-135
Musterspeicher (sed) 6-227
Mustervergleich (awk) 6-27

Nachricht, verschlüsselt (enroll) 6-119
Nachrichten (Post) (mail) 6-164
Nachrichten senden (write) 6-295
Nachrichten verschlüsseln (xget) 6-299
nächste Zeile (ed) 6-114
Negation (sed) 6-234
neue Zeile (stty) 6-247
nicht abdruckbare Zeichen (ed) 6-100
nicht abdruckbare Zeichen (ls) 6-162
Notbremse (ced) 6-56

Objekt (make) 6-169
Operatoren (awk) 6-27
Operatoren (bc) 6-31
Operatoren (egrep) 6-118

Parameter für Shell-Prozeduren 3-20
Parameter vordefinieren 3-30
passende Zeichenfolge (ed) 6-104
PATH 2-7
Pfadname 2-17, 5-9, 6-5
Pfadname (basename) 6-27
Pfadname (pwd) 6-217
Pfadname Ihrer Datensichtstation (tty) 6-281
Pfadnamen (6) 6-27
Pfadnamen (find) 6-133
physikalisches Dateisystem 5-7, 5-11
physikalisches Dateisystem erzeugen 5-20
Pipeline Beispiel 3-10
Pipeline dokumentieren (tee) 6-269
Pipeline Prozeß 3-9
Pipelines 3-4, 3-8
Pipe-Zeichen 3-8
Plattenbelegung 5-11
Plattenblockadresse 2-13
Post empfangen (mail) 6-164
Post senden (mail) 6-164
Post, geheime (enroll) 6-119
Priorität ändern, Druckauftrag (lpr) 6-153
Priorität von Kommandos ändern (nice) 6-188
Programmsystem (make) 6-168
Protokoll der Aktionen (tar) 6-260
Protokoll der Sitzung (script) 6-224
Prozedur anhalten 3-63
Prozeß 3-4, 3-47
Prozeß abbrechen 5-49
Prozeßabschluß 3-76
Prozeßdaten (ps) 6-207
Prozesse 3-32
Prozesse beenden (kill) 6-144
Prozesse zeitweise stilllegen (sleep) 6-238
Prozeßeigentümer (ps) 6-208
Prozeßhierarchie 3-33
Prozeßkonzept 3-32
Prozeßnummer 3-35
Prozeßnummer (ps) 6-207
Prozessnummern (kill) 6-144
Prozeßstatus (ps) 6-208
Prozeßtabelle (pstat) 6-214
Prozeßzeiten 3-72
Prüfsumme einer Datei (sum) 6-252

Raw-Modus (stty) 6-246
Rechenfunktionen (bc) 6-29
Rechteck bearbeiten (ced) 6-52

Rechtecke 4-2, 4-18
Rechtecke verschieben 4-15
Register (dc) 6-82
reguläre Ausdrücke (awk) 6-27
reguläre Ausdrücke (more) 6-179
reguläre Ausdrücke, erweiterte (awk) 6-27
reguläre Ausdrücke, erweiterte (egrep) 6-116
regulärer Ausdruck (ed) 6-112
regulärer Ausdruck (ed) 6-104
regulärer Ausdruck (grep) 6-136
regulärer Ausdruck (sed) 6-233
Reihenfolge von Zeichen umkehren (rev) 6-220
Relativ-Pfadname 2-17, 2-18
RESET-Taster 5-49
Returncode (siehe: Ende-Status)
root-Dateisystem 5-33
Root-Dateiverzeichnis 2-4, 2-17
root-Kennwort 5-1
root-Kennwort restaurieren 5-47
Routinearbeiten 5-29
rückgängig machen (ed) 6-113

S-Bit 2-25, 2-27, 2-28
schalter (6) 6-27
Schalter für die Shell setzen 3-66
Schleifen beenden 3-36
Schleifen steuern 3-36
Schlüssel (crypt) 6-73
Schlüssel (ed) 6-114
Schlüssel (enroll) 6-119
Schnelle Schreibmarkenbewegungen 4-10
Schrägstrich 2-18
Schreibberechtigung 2-20
schreiben (ed) 6-113
schreiben (sed) 6-234
Schreiberlaubnis (chmod) 6-59
Schreibmarke 1-2
Schreibmarke bewegen 4-4
Schreibmarke bewegen (ced) 6-46
Schreibmarke im Dokument bewegen 4-2
Schutzbits 2-19, 2-20
Schutzbits setzen 2-20
Schutzbits, Beispiele dafür 2-22
Seitenende (pr) 6-198
Seitenkopf (pr) 6-198
Seitennummer (print) 6-203
set-Kommando 3-28
Shell 1-3, 2-7, 3-1
Shell aufrufen (ced) 6-55

Shell aufrufen (ed) 6-114
Shell beenden 3-66
Shell beendet 3-52
Shell ersetzt 3-52
Shell/Sub-Shell 3-55
Shell-Prozedur beenden 3-55
Shell-Prozeduren 3-1, 3-18, 3-32
Shell-Umgebung (login) 6-148
Shell-Variablen 2-7
SIDA-Anweisung 5-4
Signal gesendet 3-73
Signale ignorieren (nohup) 6-190
Signale senden (kill) 6-144
Signalnummern 3-73
SINIX verlassen 1-6
SINIX-Kommandos aufrufen 4-20
Skript (sed) 6-226
Software installieren 5-38
Sonderzeichen 2-11, 2-12, 3-12, 3-17, 3-3, 5-48
Sonderzeichen (tr) 6-279
Sonderzeichen für Dateinamen (6) 6-27
Sonderzeichen für Dateinamen (tar) 6-264
sortieren in umgekehrter Reihenfolge (sort) 6-24
sortieren und mischen von Dateien (sort) 6-239
sortieren, lexikalisch (sort) 6-240
sortieren, nach Zahlenwerten (sort) 6-240
Sortierfeld (sort) 6-240
Sortierfelder, mehrere (sort) 6-241
Sortierrichtung umkehren (ls) 6-161
sortierte Dateien vergleichen (comm) 6-65
Spalten vertauschen 4-14
Speichern, Bereichs (ced) 6-47
Speicherplatzbelegung im Dateisystem (df) 6-85
Speicherplatzbelegung von Dateien (du) 6-95
Standard-Ausgabe (6) 6-27
Standard-Ausgabe (test) 6-273
Standard-Ein-/Ausgabe 3-5
Standard-Eingabe 3-8, 6-4
Standard-Eingabe (6) 6-27
Standard-Eingabe (test) 6-273
Standard-Eingabe für eine Prozedur umlenken 3-53
Standard-Eingabe umgeleitet 3-34
Standard-Eingabe umleiten 3-7
Standard-Eingabe umlenken 3-52
Standardeinstellung der Schutzbits 2-21
Standardeinstellung, Schutzbits 3-75
Standard-Fehlerausgabe (6) 6-27
Standard-Fehlerausgabe (test) 6-273
Standard-Funktionen (ced) 6-56

Standard-Variablen 3-15
Standard-Variablen für die Shell 3-16
Standardwerte 3-15
Standardwerte setzen 3-30
Statements (bc) 6-31
Statistik, Text (prep) 6-201
Stellungsparameter 3-20, 3-21
Stellungsparameter aufrufen 3-21
Stellungsparameter übergeben 3-21, 3-22
Stellungsparameter verschieben 3-70
Sub-Shell 3-49
Sub-Shell (printenv) 6-205
Subtraktion (dc) 6-82
Subtraktion (expr) 6-121
suchen nach Zeichenfolgen (ced) 6-52
suchen, Muster (egrep) 6-116
suchen, Muster (fgrep) 6-126
suchen, Muster (grep) 6-135
Suchzeichenkette 4-38
Suchzeichenkette (ced) 6-41
Super-User 5-1
sync-Kommando 5-31
Syntax-Fehler finden 3-65
Systemabsturz 5-31
Systeminformation (pstat) 6-212
Systempuffer (sync) 6-254
Systemstart 5-32
Systemtabelle 5-50
System-Tabelle (pstat) 6-212
Systemuhr 5-35
Systemuhr stellen (date) 6-75
Systemuhr stellen (datum) 6-79
Systemverwalter 5-1
Systemverwalterfunktionen (su) 6-250
Systemverwaltermenü 5-12
Systemzeit 5-35, 5-36

Tabulatorpositionen (ced) 6-46
Tabulatorzeichen (make) 6-170
Tabulatorzeichen (stty) 6-247
Tastatur 1-1, 1-3
Tastatur deutsch 5-48
Tastatur international 5-48
Tastaturblocks 1-5
Taste Text zuweisen 4-37
Tasten belegen 4-39
Tasten im CED 4-3
Tasten mit besonderer Bedeutung (ced) 6-55
Tasten programmieren (ced) 6-51

Tastenprogrammierung anzeigen (ced) 6-54
Teilbaum (du) 6-95
Termine (calendar) 6-35
Termine (kalender) 6-142
Text einfügen 4-2
Text eingeben 4-2
Text eingeben (ced) 6-50
Text statistisch aufbereiten (prep) 6-201
Text ver- oder entschlüsseln (crypt) 6-73
Texte suchen 4-33
Textmarkierung (ced) 6-53
Texttabelle (pstat) 6-213
Textverarbeitung (awk) 6-27
Tischrechner (dc) 6-81
Treiberprogramm 5-43
Trennzeichen (awk) 6-27
Trennzeichen (join) 6-140
Trennzeichen (pr) 6-199
Trennzeichen (sort) 6-240

Übergabe von Variablen 3-26, 3-28
Überschreiben von Dateien (tar) 6-263
Uhrzeit, deutsch (datum) 6-79
Uhrzeit, englisch (date) 6-75
Umleitung der Ausgabe 3-7
Umrechnen von Einheiten (units) 6-287
Umrechnungsfaktoren (units) 6-287
Umwandeln von Kennwort- in Stellungsparameter 3-28
Unterbrechungssignale 3-73
Unter-Dateiverzeichnisse (ls) 6-162

Variable definieren 3-25, 3-26
Variable exportieren 3-26
Variable global prüfen 3-67
Variablen (awk) 6-27
Variablen abfragen (echo) 6-97
Variablen als Kennwortparameter lesen 3-24
Variablen exportiert 2-7
Variablen für die Shell 3-15
Variablen Login-Shell 3-23
Variablen mit Standardwerten 3-20
Variablen schützen 3-65
Variablen- und Parameterbereich 3-20, 3-25
Variablen verändert 2-9
Variablen weiterreichen 3-60
Variablen zurückgeben 3-34
Variablenwerte ausgeben (printenv) 6-205
Vaterprozeß 3-33
verändern (sed) 6-230

verbinden (ed) 6-110
Vergleich (awk) 6-27
Vergleiche von Zeichenfolgen (test) 6-273
vergleichen, Dateien (cmp) 6-63
Vergleichsfelder (join) 6-139
verlassen (ed) 6-111
verschieben (ed) 6-110
verschlüsseln, Nachrichten (xget) 6-299
verschlüsseln, Text (crypt) 6-73
verschlüsselte Nachricht (enroll) 6-119
Verschlüsselung (ed) 6-114
Versionsnummern (what) 6-292
vertikal einfügen 4-18
Verweis (copy) 6-68
Verweis (du) 6-96
Verweis (find) 6-132
Verweis (ln) 6-146
Verweis (ls) 6-161
Verweis (mkdir) 6-176
Verweis (rm) 6-221
Verweis (tar) 6-262
Verweise für ein Dateiverzeichnis 2-14
Verweise für eine Datei 2-14
Verweiszähler 5-8
Verzweigen (sed) 6-234
verzweigen (sed) 6-230
vordefinierte Variablen (awk) 6-27

Währungen (units) 6-288
Warteschlange (lpr) 6-154
wechseln in die Shell 4-2
while-Schleife beenden 3-36
Wort (prep) 6-201
Worte zählen (wc) 6-290
Wortzähler (prep) 6-201

X-Bit 2-23

Zählen: Zeilen, Worte und Zeichen (wc) 6-290
Zahlenwerte sortieren (sort) 6-240
Zeichen Einfügen, Löschen 4-4
Zeichen ersetzen (sed) 6-234
Zeichen ersetzen (tr) 6-278
Zeichen zählen (wc) 6-290
Zeichen, nicht abdruckbare (ls) 6-162
Zeichen, nicht druckbare (ed) 6-100
Zeichenfolge (echo) 6-97
Zeichenfolge suchen (ced) 6-52
Zeichenfolge suchen (fgrep) 6-126

Zeichenfolge, passende (ed) 6-104
Zeichenfolgen (tr) 6-279
Zeichenfolgen ausgeben (echo) 6-97
Zeichenfolgen zusammenfassen 2-10
Zeichenfolgen, Vergleich (test) 6-273
Zeichenketten suchen 4-33
Zeichenlöscher (stty) 6-247
Zeichensatz 5-48
Zeile bearbeiten (ced) 6-54
Zeile fortsetzen (6) 6-27
Zeile markieren (ced) 6-47
Zeile, aktuelle (ed) 6-101
Zeile, gelesene (sed) 6-227
Zeile, Maximallänge (ed) 6-101
Zeile, überlang (ed) 6-110
Zeilen (uniq) 6-285
Zeilen bearbeiten 4-24
Zeilen durchsuchen (ed) 6-112
Zeilen Einfügen, Löschen 4-4
Zeilen mit bestimmtem Anfang suchen (look) 6-150
Zeilen verschieben 4-19
Zeilen zählen (wc) 6-290
Zeilen, Worte und Zeichen zählen (wc) 6-290
Zeilenabschluß (stty) 6-247
Zeilenbereich (ced) 6-47
Zeilenbereich bearbeiten 4-28
Zeilenbereich verschieben 4-12
Zeilennummer (ed) 6-114
Zeilennummer (num) 6-192
Zeilennummer (sed) 6-227
Zeilennummer (sed) 6-235
zeilenorientierter Editor (ed) 6-100
Zeilentrenner (awk) 6-27
Zeit der letzten Änderung (find) 6-132
Zeit der letzten Änderung (ls) 6-161
Zeit der letzten Änderung (settime) 6-23
Zeit der letzten Änderung (tar) 6-262
Zeit der letzten Änderung (touch) 6-277
Zeit des letzten Zugriffs (ind) 6-132
Zeit des letzten Zugriffs (ls) 6-161
Ziel (make) 6-169
Zugriffschutz 2-19
Zugriffsrechte (tar) 6-263
Zugriffsrechte ändern (chmod) 6-58
zurückholen, Bereich (ced) 6-47
zusammenfassen (sed) 6-235
Zustand von Druckaufträgen (lpr) 6-154
Zwischenspeicher 4-13

