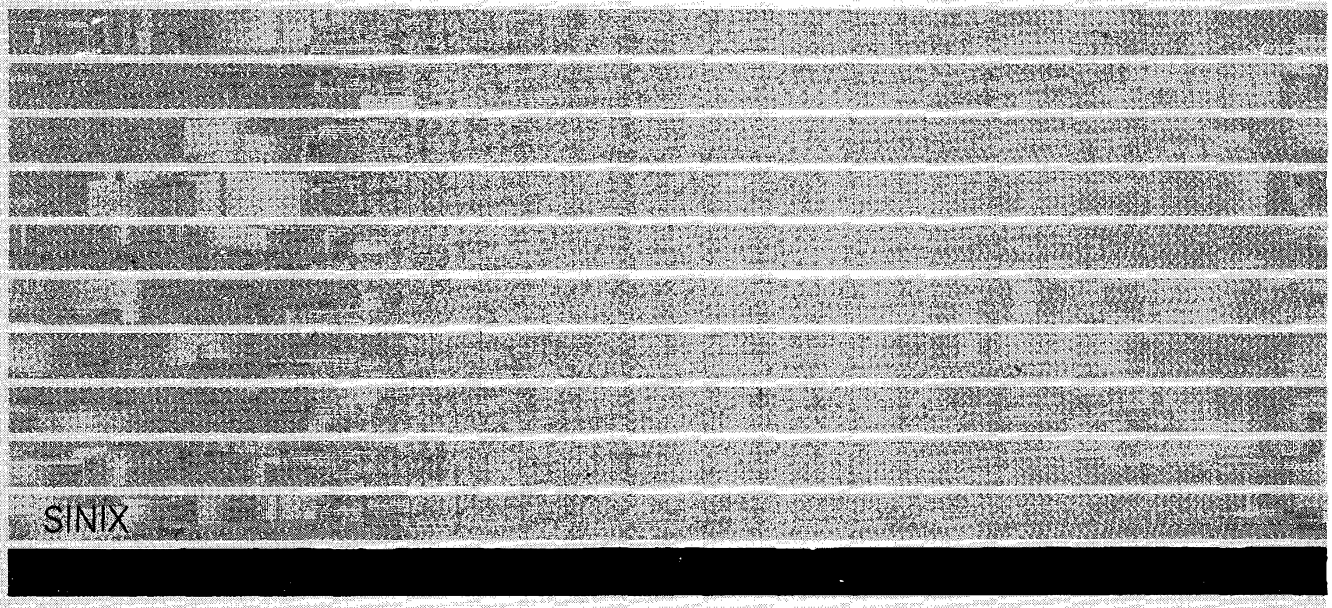
SIEMENS
NIXDORF



SINIX V5.40

Leitfaden für Benutzer

Benutzerhandbuch

Sie haben

uns zu diesem Handbuch etwas mitzuteilen?
Schicken Sie uns bitte Ihre Anregungen unter
Angabe der Bestellnummer dieses Handbuches.

Siemens Nixdorf Informationssysteme AG
Manualredaktion STM QM2
Otto-Hahn-Ring 6
W-8000 München 83

Fax: (089) 636-40443

email im EUnet:
man@sieqm2.uucp

Dieses Handbuch ist auf chlorfrei gebleichtem Papier gedruckt.

SINIX V5.40

Leitfaden für Benutzer

Benutzerhandbuch

Ausgabe Dezember 1990 (SINIX V5.40)

Wollen Sie mehr wissen...

...über dieses Produkt
...oder ein anderes Thema der Informationstechnik?

Unsere Training Center stehen für Sie bereit.
Besuchen Sie uns in Berlin, Essen, Frankfurt/Main oder Hamburg,
in Hannover, Mainz, München, Stuttgart, Wien oder Zürich.

Auskunft und Informationsmaterial erhalten Sie über:

München (089) 636-2009
oder schreiben Sie an:

Siemens Nixdorf Training Center
Postfach 830951, W-8000 München 83

X/Open XPG3-konform
Warenzeichen beantragt

Herausgegeben von
Siemens Nixdorf Informationssysteme AG

Inhalt

Vorwort

Vorwort 1
Beschreibungsformat 5

1 Grundlagen

Zur Grundkonzeption von UNIX 1-1
Zur Funktionsweise von UNIX 1-3

2 Einführung in die Benutzung von UNIX

Einstieg 2-1
Das Terminal 2-2
Der Benutzername 2-11
Herstellen der Verbindung mit UNIX 2-12

3 Das Dateisystem

Einführung 3-1
Der Aufbau des Dateisystems 3-2
Ihre Position im Dateisystem 3-4
Organisation eines Verzeichnisses 3-17
Zugriff auf und Manipulation von Dateien 3-31

4 Überblick über die Einführungskurse

Einführung 4-1
Verwaltung des System-Office 4-2

Editieren von Text 4-3
Die Shell 4-9
Programmieren mit awk 4-14
Elektronische Kommunikation 4-15
Programmieren unter UNIX 4-16

5 Einführung in FACE
Was ist FACE? 5-1
Einstieg 5-3
Benutzung des FACE-Office 5-44
Weitere FACE-Funktionen 5-82

6 Der Zeileneditor ed
Einführung in den Zeileneditor 6-1
Benutzerhinweise 6-3
Einstieg 6-4
Übung 1 6-14
Allgemeines Format der ed-Kommandos 6-15
Zeilenadressierung 6-16
Übung 2 6-29
Abrufen des Dateiinhalts 6-30
Erstellen von Text 6-33
Übung 3 6-40
Löschen von Text und Aufheben von Änderungen 6-42
Ersetzen von Text 6-46
Übung 4 6-54
Mustervergleiche mit Sonderzeichen 6-56
Übung 5 6-66
Versetzen von Text 6-68
Übung 6 6-77
Weitere nützliche Kommandos und Informationen 6-78
Übung 7 6-88
Auflösung der Übungen 6-89

7 Der Bildschirmeditor vi

Einführung 7-1
Einstieg 7-4
Anlegen einer Datei 7-7
Editieren von Text im Kommandomodus 7-10
Beenden von `vi` 7-18
Übung 1 7-21
Positionierung des Cursors auf dem Bildschirm 7-22
Cursor an eine aktuell nicht angezeigte Stelle bewegen 7-40
Übung 2 7-51
Erstellen von Text 7-53
Übung 3 7-58
Löschen von Text 7-59
Übung 4 7-66
Ändern von Text 7-67
Text ausschneiden und einfügen 7-74
Übung 5 7-79
Spezielle Kommandos 7-80
Arbeiten mit Zeileneditor-Kommandos unter `vi` 7-83
Beenden von `vi` 7-89
Spezielle `vi`-Optionen 7-92
Übung 6 7-95
Auflösung der Übungen 7-96

8 Der Druck-Service LP

Einführung 8-1
Steuerung des Druckprozesses 8-5
Formatierung der Druckausgabe mit dem Kommando lp 8-15
Die Kommandos des Druck-Service LP 8-28

9 Die Shell

Einführung 9-1
Die Kommandosprache der Shell 9-2
Übungen zur Kommandosprache 9-39
Shell-Programmierung 9-41
Änderung Ihrer Benutzerumgebung 9-93
Übungen zur Shell-Programmierung 9-99

10 Die Programmiersprache awk

Einführung 10-1
Grundlagen der Programmiersprache `awk` 10-2
Muster 10-13
Aktionen 10-22
Die Ausgabe 10-43
Die Eingabe 10-49
`awk` mit anderen Kommandos und der Shell 10-55
Anwendungsbeispiele 10-58
`awk` - Kurzübersicht 10-64

11 Die elektronische Post

Einleitung 11-1
Austauschen von Nachrichten 11-2
Das Kommando `mail` 11-3
Das Kommando `mailx` 11-17
Überblick über das Kommando `mailx` 11-18
Kommando-Optionen 11-20
Senden von Nachrichten: Die Tilde-Kommandos 11-21
Empfangen von Nachrichten 11-33
Die Datei `.mailrc` 11-42

12 Kommunikation mit fernen Systemen

Einführung 12-1
Übermittlung von Dateien 12-2
Das Netzwerk 12-18

A Kurzübersicht über das Dateisystem

Das UNIX-Dateisystem A-1
Die Verzeichnisse des UNIX-Betriebssystems A-4

B UNIX-Kommandoübersicht

Grundlegende UNIX-Kommandos B-1

C FACE-Kurzübersicht

Einführung C-1
Kommandos und Funktionen für das Command Menu C-3
Kommandos und Funktionen für Dateien und Kataloge C-4
Kommandos und Funktionen für Ausschnitte C-16
Das Menü Programs C-19
Verschiedene Funktionen C-25

D Kurzübersicht über die ed-Kommandos

ed-Kurzübersicht D-1

E **Kurzübersicht über die vi-Kommandos**
vi-Kurzübersicht E-1

F **Die Kommandosprache der Shell - Kurzübersicht**
Die Kommandosprache der Shell - Kurzübersicht F-1

G **Einstellung des Terminals**
Einstellung der Variablen TERM G-1
Beispiel G-4
Arbeiten mit Shell-Fenstern G-6

GL **Fachwörter**
Fachwörter GL-1

Bilder und Tabellen

Bild 1-1: Modell des UNIX-Systems 1-3
Bild 1-2: Die Funktionen des Systemkerns 1-4
Bild 1-3: Ausführung eines UNIX-Kommandos 1-9
Bild 1-4: Die hierarchische Struktur des Dateisystems 1-10
Bild 1-5: Beispiel für ein Dateisystem 1-13
Bild 2-1: Eingabe unter UNIX 2-5
Bild 2-2: Fehlersuche bei Problemen mit der Anmeldung* 2-18
Bild 3-1: Beispiel für ein Dateisystem 3-3
Bild 3-2: Verzeichnis der Home-Verzeichnisse 3-5
Bild 3-3: `pwd`-Kurzübersicht 3-7
Bild 3-4: Der absolute Pfadname des Verzeichnisses `/home/starship` 3-10
Bild 3-5: Der relative Pfadname des Verzeichnisses `draft` 3-12
Bild 3-6: Der relative Pfadname von `starship` nach `outline` 3-13
Bild 3-7: Beispiele für Pfadnamen 3-15
Bild 3-8: `mkdir`-Kurzübersicht 3-18
Bild 3-9: Die Ausgabe des Kommandos `ls -l` 3-24
Bild 3-10: `ls`-Kurzübersicht 3-25
Bild 3-11: `cd`-Kurzübersicht 3-28
Bild 3-12: `rmdir`-Kurzübersicht 3-30
Bild 3-13: Grundlegende Kommandos für das Arbeiten mit Dateien 3-31
Bild 3-14: `cat`-Kurzübersicht 3-35
Bild 3-15: Die Optionen bei `pg` 3-36
Bild 3-16: `pg`-Kurzübersicht 3-40
Bild 3-17: `pr`-Kurzübersicht 3-43
Bild 3-18: `cp`-Kurzübersicht 3-46
Bild 3-19: `mv`-Kurzübersicht 3-49
Bild 3-20: `rm`-Kurzübersicht 3-51
Bild 3-21: `wc`-Kurzübersicht 3-54
Bild 3-22: `chmod`-Kurzübersicht 3-61
Bild 3-23: `diff`-Kurzübersicht 3-65
Bild 3-24: `grep`-Kurzübersicht 3-67
Bild 3-25: `sort`-Kurzübersicht 3-70
Bild 5-1: Die Funktionsbereiche des FACE-Bildschirms 5-5
Bild 5-2: Das Menü AT&T FACE 5-8
Bild 5-3: Benannte Tasten und ihre Ersatztasten 5-12

Bild 5-4: Funktionstasten in einem Menü 5-13
Bild 5-5: Steuerung des Cursors in einem Menü 5-15
Bild 5-6: Zusätzliche Tasten zur Steuerung des Cursors in einem verschiebbaren Menü 5-17
Bild 5-7: Die Schablone Display Frames Form 5-21
Bild 5-8: Funktionstasten in einer Schablone 5-22
Bild 5-9: Steuerung des Cursors in einer Schablone 5-24
Bild 5-10: Das Command Menu 5-36
Bild 5-11: Steuerung des Cursors in Help- und anderen Text-Ausschnitten 5-41
Bild 5-12: Standard-Format eines Menüs mit Dateien und Katalogen 5-45
Bild 5-13: Die Find-Schablone 5-65
Bild 5-14: Das Menü Preferences 5-73
Bild 7-1: Anzeige einer Datei in einem `vi`-Fenster 7-2
Bild 7-2: Tastatur mit Pfeilen zur Anzeige der Richtung der Cursor-Bewegungen 7-11
Bild 8-1: Die zentralen Komponenten eines Druckauftrags 8-3
Bild 8-2: Druckkommandos und ihre Funktionen 8-28
Bild 8-3: `lp`-Kurzübersicht 8-29
Bild 8-4: `lpstat`-Kurzübersicht 8-31
Bild 8-5: `cancel`-Kurzübersicht 8-33
Bild 8-6: `enable`-Kurzübersicht 8-34
Bild 8-7: `disable`-Kurzübersicht 8-35
Bild 9-1: Format eines Here-Dokuments 9-64
Bild 9-2: Format der `for`-Schleife 9-69
Bild 9-3: Format der `while`-Schleife 9-73
Bild 9-4: Format der `if...then`-Bedingungsanweisung 9-76
Bild 9-5: Format der `if...then...else`-Bedingungsanweisung 9-78
Bild 9-6: Die `case...esac`-Bedingungsanweisung 9-84
Bild 10-1: `awk`-Programm - Aufbau und Beispiel 10-2
Bild 10-2: Die Beispieldatei `countries` 10-5
Bild 11-1: `uname`-Kurzübersicht 11-10
Bild 11-2: `uuname`-Kurzübersicht 11-10
Bild 11-3: `mail`-Kurzüberblick (Senden von Nachrichten) 11-12
Bild 11-4: `mail`-Kurzübersicht (Lesen von Nachrichten) 11-15
Bild 11-5: Beispiel für eine `.mailrc`-Datei 11-43
Bild 12-1: `uucp`-Kurzübersicht 12-8
Bild 12-2: `uuto`-Kurzübersicht 12-11
Bild 12-3: `uustat`-Kurzübersicht 12-13
Bild 12-4: `uupick`-Kurzübersicht 12-17
Bild 12-5: `ct`-Kurzübersicht 12-21
Bild 12-6: Kommandofolgen, die bei `cu` **verwendet werden** 12-24

Bild 12-7: cu-Kurzübersicht 12-27
Bild 12-8: uux-Kurzübersicht 12-29
Bild A-1: Aufbau des Dateisystems A-2
Tabelle 4-1: Vergleich des Zeilen- und des Bildschirmeditors (ed und vi) 4-7
Tabelle 6-1: Kommandos des Editors ed 6-13
Tabelle 6-2: Symbole und Kommandos zur Zeilenadressierung 6-28
Tabelle 6-3: Abrufen von Text über Zeilenadressen - Beispiele 6-31
Tabelle 6-4: Kommandos zum Abrufen von Text 6-32
Tabelle 6-5: Kommandos zum Erstellen von Text 6-39
Tabelle 6-6: Kommandos zum Löschen von Text 6-45
Tabelle 6-7: Sonderzeichen 6-65
Tabelle 6-8: Die ed-Kommandos zum Versetzen von Text 6-76
Tabelle 6-9: Weitere nützliche Kommandos 6-87
Tabelle 7-1: Die Kommandos des Editors vi 7-20
Tabelle 7-2: Die vi-Positionierungs-Kommandos 7-36
Tabelle 7-3: Weitere vi-Kommandos zur Cursor-Positionierung 7-49
Tabelle 7-4: Die vi-Kommandos zum Erstellen von Text 7-57
Tabelle 7-5: Kommandos zum Löschen von Text 7-65
Tabelle 7-6: vi-Kommandos zum Überschreiben von Text 7-73
Tabelle 7-7: Das Kommando y 7-76
Tabelle 7-8: vi-Kommandos zum Ausschneiden und Einfügen von Text 7-78
Tabelle 7-9: Die speziellen vi-Kommandos 7-82
Tabelle 7-10: Die Zeileneditor-Kommandos 7-88
Tabelle 7-11: Die Kommandos zum Beenden von vi 7-91
Tabelle 7-12: Die speziellen vi-Optionen 7-94
Tabelle 9-1: Sonderzeichen der Shell-Kommandosprache 9-3
Tabelle 9-2: echo-Kurzübersicht 9-5
Tabelle 9-3: Metazeichen - Kurzübersicht 9-10
Tabelle 9-4: banner-Kurzübersicht 9-15
Tabelle 9-5: spell-Kurzübersicht 9-19
Tabelle 9-6: cut-Kurzübersicht 9-24
Tabelle 9-7: date-Kurzübersicht 9-26
Tabelle 9-8: batch-Kurzübersicht 9-29
Tabelle 9-9: at-Kurzübersicht 9-32
Tabelle 9-10: ps-Kurzübersicht 9-34
Tabelle 9-11: kill-Kurzübersicht 9-36
Tabelle 9-12: nohup-Kurzübersicht 9-38
Tabelle 9-13: Kurzübersicht über das Shell-Programm dl 9-44
Tabelle 9-14: bbday-Kurzübersicht 9-48
Tabelle 9-15: whoson-Kurzübersicht 9-49

Tabelle 9-16: Kurzübersicht über das Shell-Programm `get.num` 9-51
Tabelle 9-17: Kurzübersicht über das Shell-Programm `show.param` 9-53
Tabelle 9-18: Kurzübersicht über das Shell-Programm `mknum` 9-58
Tabelle 9-19: Kurzübersicht über das Shell-Programm `num.please` 9-58
Tabelle 9-20: Kurzübersicht über das Shell-Programm `t` 9-60
Tabelle 9-21: Kurzübersicht über das Shell-Programm `log.time` 9-62
Tabelle 9-22: `gbday`-Kurzübersicht 9-65
Tabelle 9-23: `ch.text`-Kurzübersicht 9-67
Tabelle 9-24: Kurzübersicht über das Shell-Programm `mv.file` 9-72
Tabelle 9-25: Kurzübersicht über das Shell-Programm `search` 9-79
Tabelle 9-26: Kurzübersicht über das Shell-Programm `mv.ex` 9-82
Tabelle 9-27: Kurzübersicht über das Shell-Programm `set.term` 9-87
Tabelle 9-28: `tail`-Kurzübersicht 9-95
Tabelle 10-1: `awk`-Vergleichsoperatoren 10-14
Tabelle 10-2: `awk` - Reguläre Ausdrücke 10-19
Tabelle 10-3: `awk` - Vordefinierte Variablen 10-22
Tabelle 10-4: `awk` - Vordefinierte arithmetische Funktionen 10-26
Tabelle 10-5: `awk` Vordefinierte Zeichenketten-Funktionen 10-28
Tabelle 10-6: `awk printf` Konvertierungszeichen 10-45
Tabelle 10-7: Die Funktion `getline` 10-53

Vorwort

Die Informationen in diesem Leitfaden gliedern sich in zwei Hauptteile: Einen Überblick über das Betriebssystem UNIX sowie eine Reihe von didaktisch aufgebauten Einführungskursen zu den wichtigsten UNIX-Werkzeugen. Die beiden Teile werden im folgenden kurz beschrieben. Der nächste Abschnitt dieses Vorworts enthält Informationen über den Inhalt der Anhänge dieses *Leitfadens* sowie über das Fachwörterverzeichnis. Im letzten Abschnitt dieses Vorworts, "Beschreibungsformat,"
werden schließlich die drucktechnischen Gestaltungsmittel erläutert, die in diesem *Leitfaden* durchgängig zur Anwendung kamen. Diesen Abschnitt können Sie beim Durcharbeiten dieses *Leitfadens* als Referenz benutzen.

Überblick über das UNIX-System

Der erste Teil des *Leitfadens* umfaßt die Kapitel 1–3, die Sie in die Grundprinzipien des UNIX-Systems einführen. Da die Kapitel aufeinander aufbauen, sollten Sie sie der Reihe nach durcharbeiten.

- Kapitel 1, "Grundlagen", enthält einen Überblick über das Betriebssystem.
- Kapitel 2, "Einführung in die Benutzung von UNIX", enthält die allgemeinen Richtlinien zum Arbeiten mit UNIX. Im einzelnen geht es dabei um die Benutzung Ihres Terminals, das Einrichten eines Benutzereintrags sowie den Verbindungsaufbau zum UNIX-System.
- Kapitel 3, "Das Dateisystem", führt Sie in den Umgang mit dem Dateisystem ein. Dabei geht es u.a. um die Kommandos zum Aufbau einer eigenen Verzeichnishierarchie, den Zugriff auf und die Manipulation der Unterverzeichnisse und Dateien innerhalb dieser Verzeichnishierarchie sowie die Überprüfung des Inhalts von anderen Verzeichnissen im Dateisystem, auf die Sie Zugriff haben.

Einführungskurse

Im zweiten Teil des *Leitfadens* finden Sie Einführungskurse zu den folgenden Themen: Eine neue Schnittstelle zum UNIX-System, der Zeilen- und der Bildschirmeditor, die Kommandosprache der Shell, die Programmiersprache `awk` sowie die Werkzeuge zur elektronischen Datenübermittlung. Den besten Lerneffekt erzielen Sie, indem Sie die Beispiele und Übungen in jeder Einführung vollständig nachvollziehen. Die Einführungen bauen auf den Grundkonzepten auf, die in den Kapiteln 1–3 vorgestellt worden sind.

- Kapitel 4, "Überblick über die Einführungskurse", führt in die acht Kapitel in der zweiten Hälfte des *Leitfadens* auf. Dabei werden Möglichkeiten des UNIX-Systems wie die Kommandoausführung, das Editieren von Texten, die elektronische Datenübertragung, die Programmierung unter UNIX und die Hilfsmittel für die Programmentwicklung kurz angeschnitten.
- Kapitel 5, "Einführung in FACE", stellt Ihnen eine benutzerfreundliche Schnittstelle zum UNIX-System vor.
- Kapitel 6, "Der Zeileneditor `ed`", informiert Sie darüber, wie Sie mit dem Zeileneditor `ed` Text auf einem Datensichtgerät oder einem Drucker-Terminal erstellen und ändern können.
- Kapitel 7, "Der Bildschirmeditor `vi`", informiert Sie darüber, wie Sie mit dem Bildschirmeditor `vi` Text auf einem Datensichtgerät erstellen und ändern können.

Hinweis: Der Bildschirmeditor `vi` basiert auf einem Programm, das an der University of California, Berkeley, California entwickelt worden ist. Eigentümer und Lizenzgeber dieses Programms ist die University of California.

- Kapitel 8, "Der Druck-Service LP", führt Sie in die Ausgabe von Dateien auf Papier mit Hilfe des Druck-Service LP ein. Außerdem werden weitere Möglichkeiten des Druck-Service, wie das Aktivieren und Deaktivieren eines Druckers, die Änderung der Formatierung der Druckseite sowie die Steuerung des Druckprozesses vorgestellt.
- Kapitel 9, "Die Shell", führt Sie in das Arbeiten mit der Shell sowohl in ihrer Funktion als Kommandointerpreter als auch in ihrer Funktion als Programmiersprache, mit der Shell-Programme erstellt werden können, ein.

- Kapitel 10, "Die Programmiersprache `awk`", führt Sie in die Syntax, lexikalischen Einheiten und Ausdrücke der Programmiersprache sowie deren Benutzung zur Verarbeitung von Programmen ein.
- Kapitel 11, "Die elektronische Post", enthält eine umfassende Einführung in die elektronische Post und die zugehörigen Kommandos.
- Kapitel 12, "Kommunikation mit fernen Systemen", führt Sie in die Übermittlung von Nachrichten und Dateien an andere Benutzer ein, die entweder an Ihrem oder an einem anderen UNIX-System arbeiten.

Zum Nachschlagen

Zusätzlich enthält dieser Leitfaden sieben Anhänge mit Kurzübersichten und ein Fachwörterverzeichnis mit Definitionen zum Thema UNIX zum Nachschlagen.

- Anhang A, "Kurzübersicht über das Dateisystem", führt Sie in die Informationsspeicherung unter UNIX ein.
- Anhang B, "UNIX-Kommandoübersicht", enthält in alphabetischer Reihenfolge jedes UNIX-Kommando, das in diesem *Leitfaden* beschrieben wird.
- Anhang C, "FACE-Kurzübersicht", enthält Kurzinformationen über die FACE-Schnittstelle, die in Kapitel 5 beschrieben wird.
- Anhang D, "Kurzübersicht über die `ed`-Kommandos", enthält Kurzinformationen über den Zeileneditor `ed` (ausführliche Informationen finden Sie in Kapitel 6, "Der Zeileneditor `ed`.") Die Kommandos sind wie in Kapitel 6 zu Funktionsgruppen zusammengefaßt.
- Anhang E, "Kurzübersicht über die `vi`-Kommandos", enthält Kurzinformationen über den Bildschirmeditor `vi`, der in Kapitel 7, "Der Bildschirmeditor `vi`", beschrieben wird. Die Kommandos sind wie in Kapitel 7 zu Funktionsgruppen zusammengefaßt.
- Anhang F, "Die Kommandosprache der Shell - Kurzübersicht", faßt die Kommandosprache der Shell sowie ihre Syntax und Programmstrukturen zusammen, wie sie in Kapitel 9, "Die Shell ," beschrieben worden sind.

- Anhang G, "Einstellung des Terminals", erklärt die Konfiguration Ihres Terminals für die Arbeit mit dem UNIX-System sowie die Eröffnung von mehreren Fenstern auf dem Bildschirm von Terminals mit Fensterdarstellung.
- Im Fachwörterverzeichnis werden die Fachbegriffe erklärt, die in diesem Leitfaden vorkommen.

Beschreibungsformat

Das nachfolgend beschriebene Beschreibungsformat wird in diesem *Leitfaden* durchgängig verwendet.

`Computerschrift`	Die Eingabe des Benutzers (z.B. Kommandos, Optionen und Argumente in der Kommandozeile, die Namen von Umgebungsvariablen sowie die Namen von Verzeichnissen und Dateien) wurden in `Computerschrift` gedruckt. Auch die von UNIX erzeugte Ausgabe (z.B. die Eingabeaufforderung und das Ergebnis von Kommandos) wurden in `Computerschrift` gedruckt.
Kursivschrift	Die Namen von Variablen, für die Werte eingesetzt werden müssen (z.B. *paßwort*), sowie Buchtitel wurden *kursiv* gedruckt.
<>	Die Eingabe, die nicht auf dem Bildschirm erscheint (z.B. Paßwörter, Tabulatorzeichen oder `RETURN`) wurden zwischen spitze Klammern gesetzt.
<^*zeichen*>	Steuerzeichen wurden zwischen spitze Klammern gesetzt, da sie bei der Eingabe nicht auf dem Bildschirm erscheinen. Der Zirkumflex (^) steht für die CONTROL-Taste (die normalerweise mit `CTRL` beschriftet ist). Zur Eingabe eines Steuerzeichens halten Sie die CONTROL-Taste gedrückt und geben gleichzeitig *zeichen* ein. So bedeutet beispielsweise <^d>, daß Sie die CONTROL-Taste gedrückt halten und gleichzeitig die Buchstabentaste D betätigen müssen; der Buchstabe D erscheint nicht auf dem Bildschirm.
[ ]	Optionale Kommandoptionen und -argumente (z.B. `[-msCj]`) sind in eckige Klammern eingeschlossen.

|	Mit dem senkrechten Strich werden optionale Argumente, von denen Sie nur eines eingeben müssen, voneinander getrennt. Bei einer Kommandozeile im Format *kommando* [*arg1* | *arg2*] können Sie beim Aufruf von *kommando* entweder *arg1* oder *arg2* benutzen.
...	Drei Punkte hinter einem Argument zeigen Ihnen, daß dieses Argument in ein und derselben Kommandozeile zwei- oder mehrmals benutzt werden kann.
↑	Pfeile auf dem Bildschirm (in den Beispielen in Kapitel 7, "Der Bildschirmeditor `vi`") zeigen Ihnen die Position des Cursors.
kommando(zahl)	Folgt auf einen Kommandonamen eine in Klammern eingeschlossene Zahl, so wird damit auf den Teil eines UNIX-Referenzhandbuchs Bezug genommen, in dem dieses Kommando beschrieben wird (es gibt drei Referenzhandbücher: das *Referenzhandbuch für Benutzer, Referenzhandbuch für Systemverwalter* sowie das *Referenzhandbuch für Programmierer*). Die Angabe `cat(1)` bezieht sich beispielsweise auf den Eintrag in Abschnitt 1 (des *Referenzhandbuch für Benutzer*), in dem das Kommando `cat` beschrieben wird.

In den Kommandobeispielen steht das Dollar-Symbol (`$`) für die Eingabeaufforderung der Shell. Allerdings wird in manchen Systemen eine andere Eingabeaufforderung benutzt. Unabhängig vom Symbol, das in Ihrem System verwendet wird, sollten Sie stets beachten, daß die Eingabeaufforderungen vom System generiert werden. Daher dürfen Sie sie in keinem Fall eingeben, selbst wenn sie zu Beginn einer Kommandozeile angegeben werden (mit dem Dollar-Symbol (`$`) wird auch auf den Wert von Stellungsparametern und Schlüsselwortparametern Bezug genommen; ausführliche Informationen hierzu finden Sie in Kapitel 9, "Die Shell ").

Die Beispiele werden in allen Kapiteln auf Voll- und Teilansichten von Bildschirmanzeigen gezeigt, um Ihnen einen Eindruck davon zu vermitteln, wie der Bildschirm Ihres Terminals bei Ihrem Dialog mit dem UNIX-System aussieht. Diese Beispiele demonstrieren Ihnen die Benutzung der UNIX-Systemeditoren, das Erstellen von kurzen Programmen sowie die Ausführung von Kommandos. Die Eingabe (die von Ihnen eingegebenen Zeichen) und die Ausgabe (die von UNIX ausgegebenen Zeichen) werden auf diesen Bildschirmen dem oben aufgeführten Beschreibungsformat entsprechend gezeigt. Sämtliche Beispiele lassen sich auf jedem beliebigen Terminal-Typ nachvollziehen.

Die Kommandos, die in jedem Abschnitt eines Kapitels beschrieben werden, werden am Ende des jeweiligen Abschnitts noch einmal kurz zusammengefaßt. Eine nach thematischen Gesichtspunkten gegliederte Kurzübersicht über die `vi`-Kommandos finden Sie in Anhang E. Am Ende einiger Beispiele finden Sie auch Übungen, mit denen Sie die Kommandos ausprobieren können. Die Auflösung sämtlicher Übungen finden Sie am Ende des jeweiligen Kapitels.

Hinweis: Der Text des *Leitfaden für Benutzer* wurde mit den im *Leitfaden* beschriebenen UNIX-Texteditoren erstellt und mit dem Softwarepaket DOCUMENTER'S WORKBENCH formatiert.

1 Grundlagen

Zur Grundkonzeption von UNIX 1-1

Zur Funktionsweise von UNIX 1-3
Der Systemkern 1-4
Die Shell 1-5
Kommandos 1-5
- Zur Funktionsweise von Kommandos 1-6
- Aufrufen von Kommandos 1-6
- Ablauf einer Kommandoausführung 1-9

Das Dateisystem 1-9
- Normale Dateien 1-11
- Verzeichnisse 1-11
- Gerätedateien 1-12
- Symbolische Verweise 1-12
- Der Aufbau des Dateisystems 1-12

Zur Grundkonzeption von UNIX

Das Betriebssystem UNIX besteht aus einer Gruppe von Programmen (bzw. Software) zur Ansteuerung des Rechners. Es bildet einerseits das Bindeglied zwischen dem Rechner und Ihnen und stellt Ihnen andererseits Werkzeuge zur Erleichterung Ihrer Arbeit zur Verfügung. Aufgrund seiner Konzeption als unkomplizierte, leistungsfähige und flexible Rechnerumgebung bietet es zahlreiche Vorteile:

- Ein universelles System, das ein breites Spektrum an Aufgaben bzw. Anwendungen durchführen kann.
- Eine dialogfähige Umgebung, in der Sie direkt mit dem Rechner kommunizieren können und auf Ihre Anforderungen und Eingaben sofort Reaktionen erhalten.
- Eine Mehrbenutzerumgebung, in der Sie die Ressourcen des Rechners ohne Abstriche an der Produktivität mit anderen Benutzern gemeinsam nutzen können.

  Ein solches System wird Teilnehmer- oder Timesharing-System genannt. Dabei führt UNIX den Dialog mit den einzelnen Benutzern in einem so schnellen Wechsel durch, daß es scheinbar mit allen Benutzern gleichzeitig kommuniziert.
- Eine Mehrprozeß- bzw. Multitasking-Umgebung, die Ihnen die gleichzeitige Ausführung von mehreren Programmen ermöglicht.

Das UNIX-System besteht aus vier Hauptkomponenten:

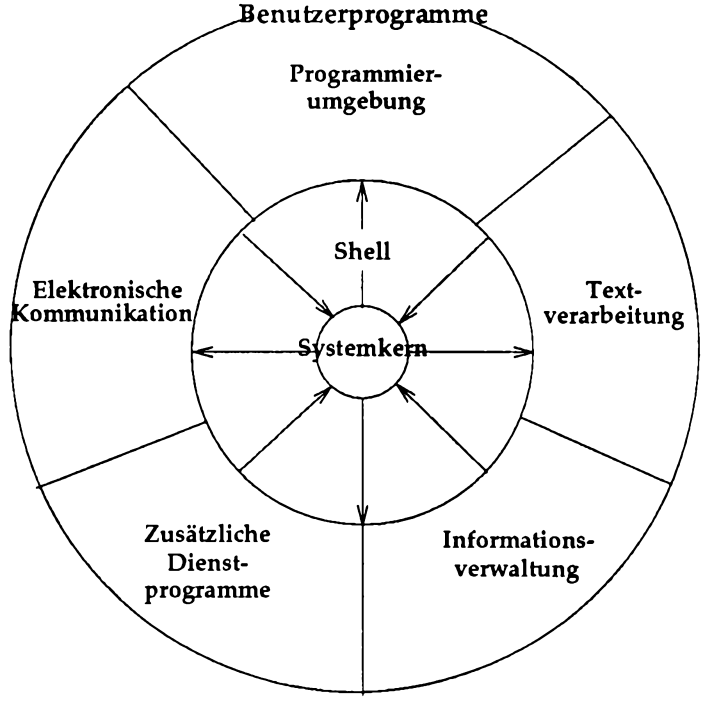
Systemkern	Der Systemkern ist das zentrale Programm des Betriebssystems. Er ist für die Koordination der rechner-internen Funktionen (z.B. die Zuordnung der Systemressourcen) zuständig. Der Systemkern arbeitet "unsichtbar"+ - Sie werden ihn während Ihrer Arbeit mit dem System nicht bemerken.
Shell	Die Shell ist ein Programm, das Ihre Kommandos interpretiert und ausführt und so die Verbindung zwischen Ihnen und dem Kern herstellt. Da sie Ihre Eingaben einliest und Ihnen Meldungen schickt, wird sie als interaktiv oder dialogfähig bezeichnet.

Kommandos	Kommandos sind die Namen der Programme, die der Rechner für Sie ausführen soll. Programm-Pakete werden als Werkzeuge bezeichnet. Das UNIX-System stellt Werkzeuge für verschiedene Aufgaben (z.B. das Erstellen und Ändern von Texten, das Schreiben von Programmen, die Entwicklung von Software-Werkzeugen sowie den Informationsaustausch mit anderen Benutzern über den Rechner) zur Verfügung.
Dateisystem	Das Dateisystem ist die Zusammenfassung aller auf Ihrem Rechner zur Verfügung stehenden Dateien. Es ermöglicht Ihnen das problemlose Abspeichern und Auffinden von Informationen.

Zur Funktionsweise von UNIX

Bild 1-1 ist ein Modell des UNIX-Systems. Jeder Kreis stellt eine der Hauptkomponenten von UNIX dar: den Systemkern, die Shell sowie die Benutzerprogramme bzw. Kommandos. Die Pfeile deuten die Aufgabe der Shell an, die Verbindung zwischen Ihnen und dem Kern herzustellen. In den nachfolgenden Abschnitten dieses Kapitels werden die einzelnen Komponenten beschrieben, außerdem eine weitere wichtige Einrichtung des UNIX-Systems, das Dateisystem.

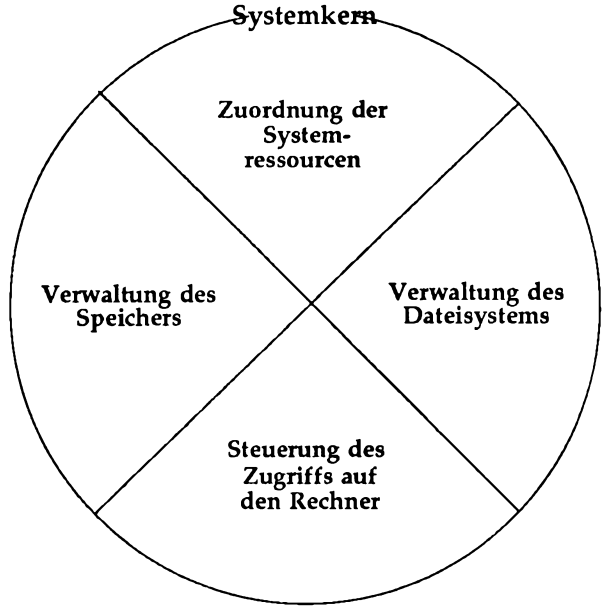
Bild 1-1: Modell des UNIX-Systems



Der Systemkern

Das zentrale Programm des UNIX-Systems wird Systemkern genannt. Der Systemkern steuert den Zugriff auf den Rechner, verwaltet Rechnerspeicher und Dateisystem und ordnet den Benutzern die Rechnerressourcen zu. Bild 1-2 enthält eine schematische Darstellung der Funktionen des Systemkerns.

Bild 1-2: Die Funktionen des Systemkerns



Die Shell

Die Shell ist ein Programm, das Ihnen die Kommunikation mit dem Betriebssystem ermöglicht. Die Shell liest die von Ihnen eingegebenen Kommandos, interpretiert sie und gibt sie als Aufforderung zur Ausführung von anderen Programmen, zum Zugriff auf Dateien oder zur Generierung einer Ausgabe weiter. Die Shell ist außerdem eine leistungsfähige, C-ähnliche Programmiersprache, mit der der Ablauf von Programmen in Abhängigkeit von Bedingungen gesteuert werden kann. Das Modell des UNIX-Systems in Bild 1-1 zeigt, wie die Kommunikation zwischen Ihnen und dem Systemkern über die Shell abläuft.

Kapitel 4 beschreibt die allgemeinen Eigenschaften der Shell. Kapitel 9 enthält Informationen darüber, wie Sie einfache Shellprogramme, sogenannte Shell-Prozeduren, schreiben und Ihre Umgebung auf Ihre Anforderungen zuschneiden können.

Kommandos

Ein Programm ist eine Gruppe von Anweisungen an den Rechner. Programme, die vom Rechner vor der Ausführung nicht umgesetzt werden müssen, werden ausführbare Programme oder Kommandos genannt. Als UNIX-Benutzer stehen Ihnen in der Regel zahlreiche Standardprogramme und -werkzeuge zur Verfügung. Für den Programmierer und Software-Entwickler bietet UNIX zu diesem Zweck Systemaufrufe, Unterprogramme und andere Werkzeuge. Natürlich können Sie auch Ihre selbst geschriebenen Programme benutzen.

Dieses Kapitel stellt Ihnen eine Reihe von UNIX-Programmen und -Werkzeugen vor, die Sie bei Ihrer täglichen Arbeit mit UNIX benutzen werden. Weitere Informationen über diese Programme und andere Standardprogramme finden Sie im *Referenzhandbuch für Benutzer*. Werkzeuge und Unterprogramme zur Programmierung und Software-Entwicklung werden im *Referenzhandbuch für Programmierer* beschrieben. Bestellhinweise für die UNIX-Dokumentation von AT&T finden Sie in *Documentation Roadmap*.

Diese Handbücher stehen in einigen Systemen auch online zur Verfügung (die Online-Dokumentation ist im Dateisystem Ihres Rechners gespeichert). Mit dem Kommando `man` (für manual page) können Sie Seiten aus der Online-Dokumentation abrufen. Ausführliche Informationen zur Benutzung des Kommandos `man` finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `man(1)`

Zur Funktionsweise von Kommandos

Im Modell des UNIX-Systems in Bild 1-1 sind die Systemprogramme und Werkzeuge im äußeren Kreis in verschiedene Funktionsgruppen eingeteilt. Dabei handelt es sich um folgende Funktionsgruppen:

Programmierumgebung	Viele UNIX-Programme stellen dem Benutzer Dienstprogramme sowie Schnittstellen zwischen dem UNIX-System und den Programmiersprachen und somit eine bedienungsfreundliche Programmierumgebung zur Verfügung.
Textverarbeitung	Zum System gehören Zeilen- und Bildschirm-Editoren zur Erstellung und Änderung von Texten, eine Rechtschreibprüfung zum Auffinden von (englischen) Schreibfehlern sowie ein Textformatierungssystem zur Erzeugung von druckfertigen Vorlagen.
Informationsverwaltung	Das System stellt zahlreiche Programme zur Verfügung, mit denen Dateien und Verzeichnisse angelegt, organisiert und gelöscht werden können.
Zusätzliche Dienstprogramme	Mit weiteren Werkzeugen können Graphiken erstellt und Rechenfunktionen durchgeführt werden.
Elektronische Kommunikation	Zum Austausch von Informationen mit anderen Benutzern und UNIX-Systemen stehen eine Reihe von Programmen wie z.B. `mail` zur Verfügung.

Aufrufen von Kommandos

Damit UNIX Ihre Anforderungen versteht, muß jedes Kommando im korrekten Format bzw. in der korrekten Kommandozeilensyntax eingegeben werden. Die Kommandozeilensyntax definiert die Reihenfolge, in der Sie die Komponenten einer Kommandozeile eingeben müssen. Ähnlich wie in der natürlichen Sprache ist auch bei der Eingabe der Komponenten einer Kommandozeile eine

bestimmte, durch die Syntax vorgegebene Reihenfolge einzuhalten; andernfalls kann die UNIX-Shell
Ihre Anforderung nicht interpretieren. Nachfolgend sehen Sie ein Beispiel für die Syntax einer UNIX-Kommandozeile.

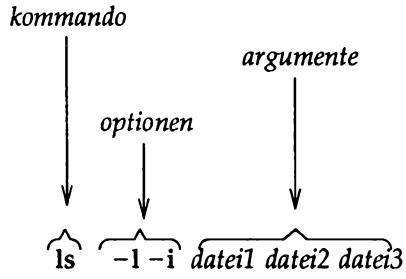
kommando option(en) argument(e)<CR>

Jede UNIX-Kommandozeile muß aus mindestens zwei Komponenten bestehen: dem Kommandonamen und der Taste RETURN (um anzugeben, daß die Taste RETURN betätigt werden muß, wird in diesem *Leitfaden* durchgängig das Format <CR> verwendet). Zusätzlich kann eine Kommandozeile Optionen und/oder Argumente enthalten. Was sind Kommandos, Optionen und Argumente?

- Ein *Kommando* ist der Name des Programms, das Sie ausführen möchten.
- Mit einer *Option* können Sie den Programmablauf beeinflussen.
- Ein *Argument* gibt die Daten an, die durch das Kommando verarbeitet werden sollen (im Normalfall ein Verzeichnis- oder Dateiname).

In Kommandozeilen, die Optionen und/oder Argumente enthalten, müssen die einzelnen Komponenten durch mindestens ein Leerzeichen voneinander getrennt sein (ein Leerzeichen können Sie über die Leertaste eingeben). Enthält ein Argument ein Leerzeichen, so ist es zwischen Anführungszeichen zu setzen. Wenn Ihr Kommando z.B. das Argument `sample 1` enthält, sieht die Eingabe folgendermaßen aus:
`"sample 1"`. Ohne die Anführungszeichen interpretiert die Shell `sample` und `1` als zwei separate Argumente.

Bei einigen Kommandos sind mehrere Optionen und/oder Argumente in einer Kommandozeile zulässig. Betrachten Sie dazu das folgende Beispiel:



In diesem Beispiel wird das Kommando `ls` mit Optionen (`-l` und `-i`) verwendet, um Informationen über die Dateien *datei1*, *datei2* und *datei3* abzurufen. Die Option `-l` gibt an, daß die Informationen im ausführlichen ("long") Format ausgegeben werden, also auch Angaben über die Zugriffsrechte, den Eigentümer und die Größe enthalten sollen. Mit der Option `-i` wird die I-Node-Nummer angegeben (derartige Optionen können Sie unter UNIX in der Regel direkt und in beliebiger Reihenfolge hintereinanderschreiben; bei diesem Beispiel wäre also auch die Eingabe `-il` oder `-li` zulässig). Zusätzlich wurden drei Dateien (*datei1*, *datei2* und *datei3*) als Argumente angegeben. Die Argumente dürfen im Gegensatz zu den meisten Optionen auf keinen Fall aneinandergereiht werden.

Die folgenden Beispiele zeigen, wie die einzelnen Komponenten einer Kommandozeile aufeinanderfolgen, und an welcher Stelle Leerzeichen eingefügt werden müssen:

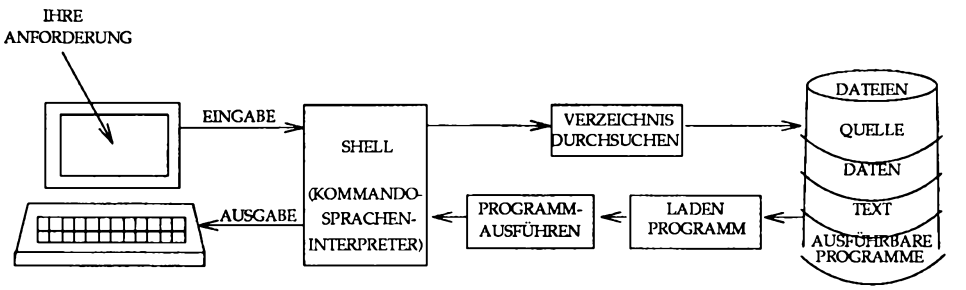
Falsch	Richtig
`ls`*datei*	`ls` *datei*
`ls-l`*datei*	`ls -l` *datei*
`ls -l i` *datei*	`ls -li` *datei*
	oder
	`ls -l -i` *datei*
`ls` *datei1datei2*	`ls` *datei1 datei2*

Beachten Sie stets, daß jede Kommandozeile, unabhängig von Ihrer Länge, mit der Taste RETURN beendet werden muß.

Ablauf einer Kommandoausführung

Bild 1-3 zeigt das Flußdiagramm einer Kommandoausführung unter UNIX.

Bild 1-3: Ausführung eines UNIX-Kommandos

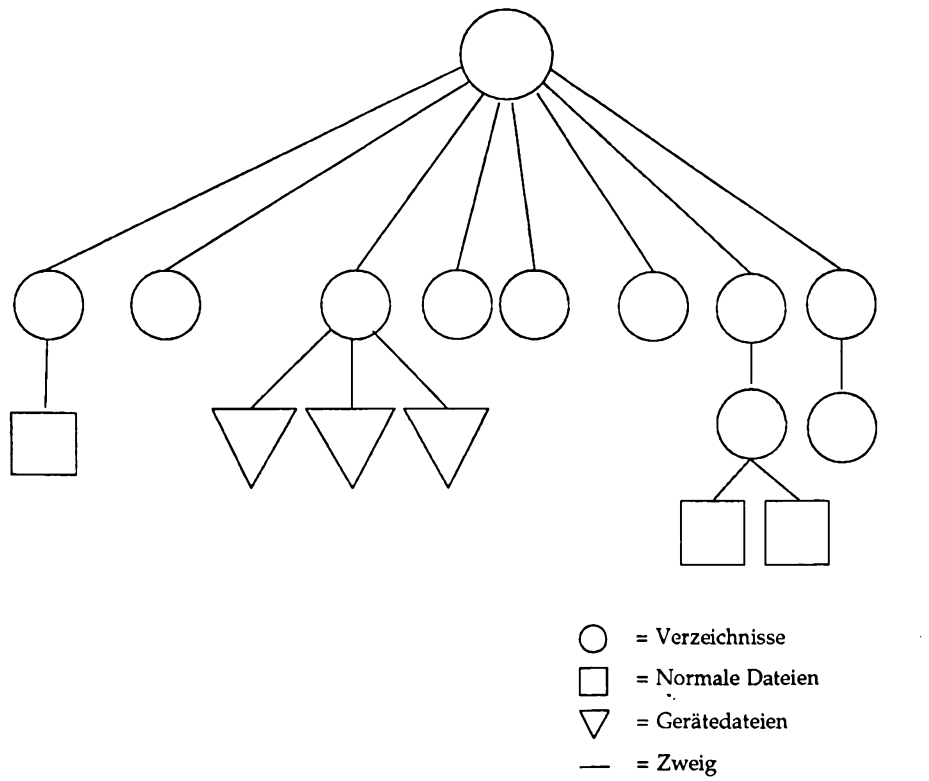


Damit ein Kommando ausgeführt wird, geben Sie eine Kommandozeile ein, sobald auf dem Bildschirm eine Eingabeaufforderung (z.B. $) erscheint. Die Shell liest Ihr Kommando als Eingabe ein, durchsucht ein oder mehrere Verzeichnisse nach dem von Ihnen angegebenen Programm und leitet Ihre Anforderung, zusammen mit der Programmanforderung, an den Systemkern weiter. Der Systemkern arbeitet die Anweisungen des Programms ab und führt das von Ihnen angegebene Kommando aus. Nach der Beendigung des Programms zeigt die Shell durch die Ausgabe einer Eingabeaufforderung an, daß sie bereit ist, Ihr nächstes Kommando entgegenzunehmen.

Das Dateisystem

Das Dateisystem ist der Eckpfeiler des UNIX- Systems. Mit dem Dateisystem steht Ihnen eine Methode zur logischen Organisation, Suche und Verwaltung von Informationen zur Verfügung. Die Struktur des Dateisystems ist hierarchisch. Könnte man sie sichtbar machen, sähe sie wie ein Organisationsplan oder ein umgedrehter Baum aus (siehe Bild 1-4).

Bild 1-4: Die hierarchische Struktur des Dateisystems



Die Datei ist die grundlegende Einheit des UNIX- Systems. Eine Datei kann eine normale Datei sein, ein Verzeichnis, eine Gerätedatei oder ein symbolischer Verweis (siehe Kapitel 3, "Das Dateisystem").

Normale Dateien

Eine normale Datei ist eine Ansammlung von Zeichen, die das System als Einheit behandelt. In normalen Dateien können Sie beliebige Informationen speichern, die Sie für den späteren Gebrauch aufbewahren möchten. Sie können Texte für Briefe oder Berichte enthalten, den Code für Ihre Programme oder Kommandos zum Aufrufen Ihrer Programme. In eine Datei, die Sie einmal angelegt haben, können Sie Daten einfügen; ebenso können Sie Daten aus ihr entfernen oder sie vollkommen löschen, wenn Sie sie nicht mehr benötigen.

Verzeichnisse

Ein Verzeichnis ist ebenfalls eine Datei, kann aber selbst Dateien und Verzeichnisse enthalten. Normalerweise besteht ein thematischer Zusammenhang zwischen den Dateien innerhalb ein und desselben. Verzeichnisses. So können z.B. in einem Verzeichnis mit dem Namen `sales` die Dateien `jan`, `feb`, `mar` mit den monatlichen Verkaufszahlen enthalten sein. Sie können Verzeichnisse anlegen, in denen Sie ebenfalls Dateien anlegen; die Dateien in einem Verzeichnis können Sie ebenso wie die Verzeichnisse selbst jederzeit löschen.

Alle Verzeichnisse, die Sie anlegen und besitzen, befinden sich in Ihrem Home-Verzeichnis. Dieses Verzeichnis wird Ihnen vom System gleichzeitig mit Ihrem Benutzernamen zugewiesen. Es steht allein unter Ihrer Kontrolle. Kein anderer Benutzer kann ohne Ihre explizite Erlaubnis die darin enthaltenen Dateien lesen oder Daten in sie schreiben. Auch die Struktur des Verzeichnisses wird von Ihnen bestimmt.

Zusätzlich führt das UNIX-System eine Reihe von Verzeichnissen zur eigenen Verwendung. Die Struktur dieser Verzeichnisse ist auf allen UNIX-Systemen im Großen und Ganzen gleich. Diese Verzeichnisse, zu denen auch verschiedene wichtige Systemverzeichnisse gehören, befinden sich in der Dateihierarchie direkt unter dem Root-Verzeichnis. Das Root-Verzeichnis (mit / gekennzeichnet) ist der Dreh- und Angelpunkt der UNIX- Dateistruktur. Alle Verzeichnisse und Dateien befinden sich in hierarchischer Anordnung darunter.

Gerätedateien

Die Gerätedateien sind die außergewöhnlichste Einrichtung des Dateisystems. Eine Gerätedatei steht für ein physisches Gerät wie z.B. ein Terminal, ein Plattenlaufwerk, ein Magnetbandlaufwerk oder den Anschluß für eine Übertragungsleitung. Die Gerätedateien werden vom System wie normale Dateien zur Daten-Ein- und Ausgabe benutzt. Allerdings aktivieren die Lese- bzw. Schreibanforderungen des Systems nicht den normalen Dateizugriffsmechanismus; stattdessen wird das der Datei zugeordnete Gerätesteuerungsprogramm aktiviert, was z.B. eine Bewegung des Schreib-/Lesekopfs oder einen schnellen Bandvorlauf zur Folge hat.

Symbolische Verweise

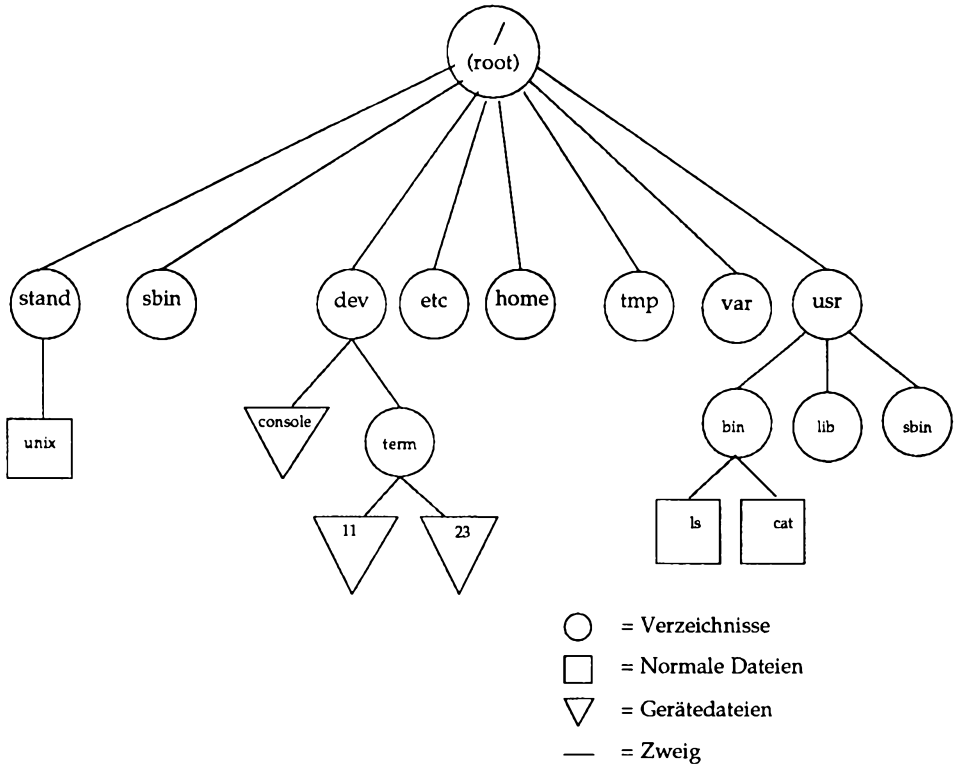
Symbolische Verweise sind Dateien, die auf andere Dateien zeigen. Weitere Informationen über diese Dateien und ihre Verwendung finden Sie im Kapitel 6 des *Application Programmer's Guide.*

Der Aufbau des Dateisystems

In einigen Betriebssystemen müssen Sie für eine Datei einen Dateityp angeben, der festlegt, daß diese Datei in einer bestimmten Weise verwendet wird. Das erfordert von Ihnen die Überlegung, wie die Dateien gespeichert werden, da es sich um sequentielle, binäre oder um Dateien mit wahlfreiem Zugriff handeln kann. UNIX behandelt jedoch alle Dateien gleich. Das macht es so einfach, mit der Dateistruktur von UNIX zu arbeiten. So müssen Sie z.B. für Ihre Dateien keinen Speicherbedarf angeben, da dies automatisch durch das System geschieht. Ein weiterer Vorteil ist, daß Sie ein Gerät, auf das Sie oder eines Ihrer Programme zugreifen wollen (z.B. einen Drucker), genauso angeben können wie eine beliebige andere Datei. Unter UNIX gibt es für Ihre gesamte Ein- und Ausgabe nur eine einzige Schnittstelle, was Ihren Dialog mit dem System vereinfacht.

In Bild 1-4 wird ein Beispiel für ein typisches Dateisystem gezeigt. Beachten Sie, daß das Root-Verzeichnis verschiedene wichtige Systemverzeichnisse enthält.

Bild 1-5: Beispiel für ein Dateisystem



`/stand` Enthält ladbare Programme und Daten-Dateien, die beim Urladen verwendet werden.

`/sbin` Enthält wichtige ausführbare Programme, die beim Urladen und bei der manuellen Wiederherstellung des Systems verwendet werden.

`/dev` Enthält Gerätedateien für Peripheriegeräte wie die Konsole, den Zeilendrucker, Benutzer-Terminals und Platten.

`/etc`	Enthält maschinenspezifische Konfigurationsdateien für die Systemverwaltung und Systemverwaltungsdatenbanken.
`/home`	Die Root eines Teilbaums für Benutzerverzeichnisse.
`/tmp`	Enthält temporäre Dateien.
`/var`	Die Root eines Teilbaums für Dateien, die ständig geändert werden (z.B. die Protokolldateien).
`/usr`	Enthält weitere Verzeichnisse wie z.B. `lib` und `bin`.

Zusammenfassend bilden die von Ihnen erstellten Verzeichnisse und Dateien jenen Teil des Dateisystems, der unter Ihrer Kontrolle steht. Die übrigen Teile des Dateisystems, die vom Betriebssystem bereitgestellt und verwaltet werden (z.B. `/sbin`, `/dev`, `/etc`, `/tmp` und `/usr`), haben auf allen UNIX-Systemen im Großen und Ganzen dieselbe Struktur.

In anderen Kapiteln dieses Leitfadens werden Sie mehr über das Dateisystem erfahren. In Kapitel 3 wird z.B. erklärt, wie man dem Dateisystem eine Verzeichnisstruktur gibt, wie man auf Dateien zugreift und sie manipuliert. Kapitel 5 und 6 enthalten eine ausführliche Einführung in das Anlegen und Editieren von Dateien.

In diesem Kapitel wurde die Grundkonzeption des UNIX-Systems erklärt. Die folgenden Kapitel zeigen Ihnen, wie Sie diese Konzeption in die Praxis umsetzen.

2 Einführung in die Benutzung von UNIX

Einstieg 2-1

Das Terminal 2-2
Konfiguration des Terminals 2-2
Einteilung und Belegung der Tastatur 2-3
Regeln für die Eingabe 2-4
- Die Eingabeaufforderung 2-6
- Korrektur von Eingabefehlern 2-6
- Sonderzeichen als darstellbare Zeichen 2-8
- Eingabegeschwindigkeit 2-9
- Abbrechen eines Kommandos 2-9
- Die Steuerzeichen 2-9

Der Benutzername 2-11

Herstellen der Verbindung mit UNIX 2-12
Die Anmeldung 2-13
Das Paßwort 2-13
Bei Problemen mit der Anmeldung 2-16
Einfache Kommandos 2-19
Abmelden 2-20

Einstieg

Dieses Kapitel ist eine Einführung in die Benutzung des UNIX-Systems. Im einzelnen wird die Konfiguration des Terminals erklärt, die Bedienung der Tastatur, die Wahl eines Benutzernamens, die Anmeldung beim/Abmeldung vom System sowie die Eingabe von einfachen Kommandos.

Für eine Anmeldung bei UNIX ist folgendes erforderlich:

- Ein Terminal.
- Ein Benutzername, der Sie als berechtigten Benutzer ausweist.
- Ein Paßwort zum Schutz Ihres Benutzernamens.
- Eine Anleitung zum Anwählen des Hostrechners und den Zugriff auf UNIX, wenn Ihr Terminal nicht direkt mit dem Rechner verbunden ist.

Die Beschreibung der Kommandos in diesem Kapitel hält sich an das Beschreibungsformat, das in diesem *Leitfaden* durchgängig verwendet wird. Informationen zum Beschreibungsformat finden Sie im Vorwort.

Das Terminal

Das Terminal ist ein Gerät zur Ein- und Ausgabe von Zeichen: Sie geben daran Ihre Anforderungen an das UNIX-System ein, und UNIX benutzt es wiederum zur Ausgabe von Ergebnissen und Meldungen. Bei den Terminals unterscheidet man grundsätzlich zwischen Datensichtgeräten und Drucker-Terminals.

Auf einem Datensichtgerät wird die Ein- bzw. Ausgabe auf einem Bildschirm angezeigt; bei einem Drucker-Terminal wird sie auf Endlospapier ausgedruckt. Dieser Unterschied wirkt sich in der Regel jedoch nicht auf die Eingabe des Benutzers und die vom System generierte Ausgabe aus. Die Anweisungen in diesem Leitfaden beziehen sich zwar auf Datensichtgeräte, können aber, sofern keine anderslautenden Angaben gemacht werden, direkt auf Drucker-Terminals übertragen werden.

Konfiguration des Terminals

Jedes Terminal muß, unabhängig vom Typ, ordnungsgemäß konfiguriert sein, um mit dem UNIX-System Daten austauschen zu können. Wenn Sie bisher noch nie die Optionen eines Terminals eingestellt haben, sollten Sie möglicherweise einen erfahrenen Benutzer um Hilfe bitten.

Die Konfiguration des Terminals hängt von der Art des verwendeten Geräts ab. Bei einer Reihe von Terminals erfolgt die Konfiguration hardwaremäßig über Schalter; bei anderen geschieht dies direkt über bestimmte Funktionstasten der Tastatur. Die Informationen über die Konfiguration Ihres speziellen Terminals schlagen Sie bitte in der vom Hersteller mitgelieferten Dokumentation nach.

Bevor Sie versuchen, sich bei UNIX anzumelden, sollten Sie zur Überprüfung der Konfiguration Ihres Terminals die folgenden Schritte durchführen:

1. Schalten Sie das Terminal ein.
2. Schalten Sie Ihr Terminal auf Fern- bzw. Online-Betrieb. Dadurch wird sichergestellt, daß das Terminal direkt durch den Rechner angesteuert wird.
3. Stellen Sie sicher, daß das Terminal auf Vollduplex-Betrieb geschaltet ist. Das UNIX- System arbeitet im Vollduplex-Betrieb. Unter Vollduplex versteht man ein Kommunikationsprotokoll, bei dem beide Seiten gleichzeitig senden und empfangen können; dies wird in der Regel durch die Verwendung von unterschiedlichen Signalfrequenzen ermöglicht. Bei einer Vollduplex-Verbindung können Sie also selbst dann Informationen

an UNIX senden, wenn es gerade Informationen an Sie übermittelt (bei Halbduplex-Betrieb muß die eine Seite auf Empfang geschaltet sein, wenn die andere Seite sich im Sende-Modus befindet).

4. Falls Ihr Terminal nicht direkt mit dem Rechner verbunden ist, müssen Sie überprüfen, ob das verwendete Modem auf Vollduplex-Betrieb geschaltet ist.
5. Stellen Sie die Zeichengenerierung auf Kleinschreibweise.
6. Deaktivieren Sie die Paritätsprüfung auf Ihrem Terminal. UNIX verwendet im Gegensatz zu einer Reihe anderer Betriebssysteme keine Paritätsprüfung zur Fehlerüberprüfung.
7. Setzen Sie die Übertragungsgeschwindigkeit fest. Die Übertragungsgeschwindigkeit (Baud-Rate) ist die Geschwindigkeit, in der die Datenübertragungen zwischen Rechner und Terminal stattfinden. Die Übertragungsgeschwindigkeit wird in Baud oder Zeichen pro Sekunde angegeben (ein Terminal mit einer Übertragungsgeschwindigkeit von 4800 Baud sendet und empfängt beispielsweise 480 Zeichen pro Sekunde). Je nach Rechner und Terminal kann eine Übertragungsgeschwindigkeit zwischen 300 und 19200 Baud eingestellt werden. Allerdings können die Daten von einigen Rechnern mit einer noch höheren Geschwindigkeit verarbeitet werden.

Einteilung und Belegung der Tastatur

Die Einteilung und Belegung der Tastatur unterscheidet sich von Terminal zu Terminal. Gemeinsam ist allen Terminals die Benutzung eines aus 128 Zeichen bestehenden Zeichensatzes, des sogenannten ASCII-Zeichensatzes (ASCII ist ein Akronym für American Standard Code for Information Interchange - amerikanischer Standardcode für Informationsaustausch). Die einzelnen Tasten sind dabei so beschriftet, daß ihre Belegung für den Benutzer ersichtlich ist (z.B. mit den Buchstaben des Alphabets); jeder Taste ist ein ASCII-Code zugeordnet, der vom Rechner verstanden wird.

Die Tastatur eines typischen ASCII-Terminals entspricht weitgehend der einer Schreibmaschine. Zusätzlich sind jedoch Tasten mit speziellen Funktionen vorhanden, mit denen beispielsweise ein Programmlauf unterbrochen werden kann. Die Tasten lassen sich in folgende Gruppen aufteilen:

- Die Buchstaben-Tasten (sowohl Groß- als auch Kleinschreibung).
- Die numerischen Tasten (von 0 bis 9).
- Tasten zur Generierung bestimmter Symbole (z.B. ! @ # $ % ^ & ( ) _ - + = ~ ' { } [ ] \ : ; " ' < > , ? /)
- Tasten mit vordefinierter Funktion (wie <RETURN> und <BREAK>), und Abkürzungen (wie <DEL> für delete (Löschen), <CTRL> für control (Steuerung) und <ESC> für escape (Ende).

Außer den oben aufgeführten Tasten weist eine Terminal-Tastatur Tasten auf, die speziell auf das Arbeiten am Rechner zugeschnitten worden sind. Die Funktion dieser Tasten geht aus ihrer Beschriftung (Zeichen oder Symbole) hervor. Allerdings unterscheidet sich ihre Anordnung von Tastatur zu Tastatur - für die Tastenanordnung ist kein Standard eingeführt worden.

Regeln für die Eingabe

Damit Sie mit dem UNIX-Betriebssystem arbeiten können, müssen Sie bei der Eingabe bestimmte Regeln beachten. Die UNIX-Kommandos müssen in Kleinbuchstaben eingegeben werden (mit Ausnahme der Kommandos, die in der Dokumentation explizit in Großschreibweise angegeben werden). Über bestimmte Tasten oder Tastenkombinationen können eingegebene Buchstaben oder Zeilen gelöscht und andere Funktionen durchgeführt werden. Bild 2-1 enthält eine Übersicht über diese Funktionen und die zugehörigen Tasten(kombinationen). Dabei ist zu beachten, daß es sich bei den hier angegebenen Tasten lediglich um die Standard- Möglichkeiten handelt - die meisten Funktionen können über verschiedene Tasten durchgeführt werden. Auf den folgenden Seiten werden einige dieser Tasten genauer erklärt.

Bild 2-1: Eingabe unter UNIX

Taste(n)*	Funktion
$	Die Eingabeaufforderung des Systems. Hier können Sie ein Kommando eingeben.
<BACKSPACE> oder <^h>	Löschen eines Zeichens.
@	Vollständiges Löschen einer Zeile.
<BREAK>	Beendigung der Ausführung eines Programms oder Kommandos.
<DEL>	Löschen der aktuellen Kommandozeile.
<ESC>	In Kombination mit einem anderen Zeichen wird eine spezielle Funktion durchgeführt (die sogenannte Escape-Sequenz).
	Beim Editor `vi` wird der Eingabemodus beendet und der Kommandomodus wieder eingeschaltet.
<CR>	Die Taste <RETURN>. Eine Eingabezeile wird abgeschlossen, und der Cursor wird in eine neue Zeile bewegt.
<^d>†	Beendigung der Eingabe an das System und Abmeldung.
<^h>	Korrekturfunktion auf Terminals ohne Korrekturtaste.
<^i>	Horizontaltabulatorzeichen auf Terminals ohne Tabulatortaste.
<^s>	Vorübergehendes Anhalten der Bildschirmausgabe.
<^q>	Fortsetzung der Bildschirmausgabe, nachdem sie über die Tasten <^s> angehalten worden war.

* Nicht darstellbare Zeichen stehen in eckigen Klammern (< >).

† Zeichen, vor denen ein Zirkumflex (^) steht, werden Steuerzeichen genannt. Ein Steuerzeichen wird eingegeben, indem man die Taste <CTRL> gedrückt hält und gleichzeitig die angegebene Buchstabentaste betätigt.

Die Eingabeaufforderung

Das Standard-Symbol, mit dem UNIX seine Empfangsbereitschaft signalisiert, ist das Dollarzeichen ($). Wenn diese Eingabeaufforderung auf Ihrem Bildschirm erscheint, wartet UNIX auf Ihre Anweisungen. Sie können jetzt ein Kommando eingeben und anschließend die Taste <RETURN> betätigen.

Das Dollarzeichen ($) ist lediglich das Standard-Symbol für die Eingabeaufforderung - Sie können jederzeit zu einer ein- oder mehrstelligen Eingabeaufforderung Ihrer Wahl wechseln (siehe Kapitel 7).

Korrektur von Eingabefehlern

Zur Korrektur eines Eingabefehlers gibt es verschiedene Möglichkeiten. Mit dem Klammeraffen @ können Sie die Zeile löschen, in der Sie sich aktuell befinden. Über die Tasten <BACKSPACE> und <^h> löschen Sie das zuletzt eingegebene Zeichen. Auch diese Tasten und Zeichen stellen lediglich die standardmäßig vorgegebenen Möglichkeiten dar; die verschiedenen Funktionen können jederzeit auf andere Tasten gelegt werden (siehe hierzu Abschnitt "Neuzuordnung der Löschfunktionen" in diesem Kapitel und ""Konfiguration des Terminals" in Kapitel 7).

Löschen der aktuellen Zeile: Das Zeichen @

Mit dem Klammeraffen (@) können Sie die Zeile löschen, in der Sie sich aktuell befinden. Sobald Sie diese Taste betätigen, erscheint das Zeichen @ am Ende der Zeile, und der Cursor springt in die nächste Zeile. Die fehlerhafte Zeile wird dabei nicht gelöscht, sondern vom System ignoriert.

Das Zeichen @ kann nur für die aktuelle Zeile verwendet werden. Achten Sie deshalb darauf, daß Sie @ wirklich eingegeben haben, bevor Sie die Taste <RETURN> betätigen. Die folgende Kommandozeile enthält einen Eingabefehler; mit @ können Sie das Kommando stornieren:

```
whooo@
who<CR>
```

Löschen des zuletzt eingegebenen Zeichens: <^h> und <BACKSPACE>

Zum Löschen von einem oder mehreren der zuletzt eingegebenen Zeichen in der aktuellen Zeile kann man die Taste <BACKSPACE> oder die Tastenkombination <^h> verwenden. In beiden Fällen wird der Cursor zum vorhergehenden Zeichen bewegt. Sie haben jetzt die Möglichkeit, dieses Zeichen zu

überschreiben; das alte Zeichen wird dabei gelöscht. Das Löschen von Zeichen ist mit diesen Tasten also überhaupt kein Problem.

Nach dieser Methode können Sie beliebig viele Zeichen löschen: Sie müssen lediglich <BACKSPACE> oder <^h> entsprechend oft betätigen. Das folgende Beispiel zeigt, wie Sie zwei Zeichen durch die zweifache Betätigung der Taste <BACKSPACE> löschen können:

```
dattw<BACKSPACE><BACKSPACE>e<CR>
```

Nach der Korrektur der Eingabe handelt es sich hier um das UNIX-Kommando `date`.

Neuzuordnung der Lösch-Funktionen

Wie bereits erwähnt, können die Funktionen zum Löschen von Zeilen und Zeichen auch auf andere Tasten gelegt werden. Soll diese Neuzuordnung nur während Ihrer aktuellen UNIX-Sitzung gültig sein, so ist die Eingabe eines Shell-Kommandos ausreichend. Mit Ihrer Abmeldung vom System wird dann die Standardzuordnung wiederhergestellt. Eine permanente Neuzuordnung wird über eine Änderung der Datei `.profile` vorgenommen. Informationen über die temporäre oder permanente Neubelegung der Tastatur sowie die Datei `.profile` finden Sie in Kapitel 7.

Bei einer Neubelegung der Tastatur müssen Sie die folgenden drei Punkte beachten: Wird eine Funktion von einer Standard- auf eine andere Taste gelegt, entfernen Sie diese Funktion gleichzeitig von der Standardtaste. Legen Sie z.B. die Löschfunktion von der Taste <BACKSPACE> auf das Zeichen # um, können Sie mit der Taste <BACKSPACE> keine Zeichen mehr löschen. Ein und dieselbe Funktion kann in keinem Fall auf zwei Tasten gelegt werden.

Die neue Tastaturbelegung wird von allen anderen installierten UNIX-Programmen übernommen, vorausgesetzt, sie unterstützen die entsprechenden Funktionen. So kann beispielsweise unter dem dialogfähigen Editor `ed` (siehe Kapitel 5) zum Löschen von Text dieselbe Taste benutzt werden, mit der Sie auch, wie in diesem Abschnitt beschrieben, in einer Shell-Kommandozeile Eingabefehler löschen. Deshalb müssen Sie, nachdem Sie diese Funktion auf die Taste # gelegt haben, zum Löschen von Zeichen auch in einer `ed`-Sitzung die Taste # benutzen. Die Funktion läßt sich dann nicht mehr über die Taste <BACKSPACE> durchführen.

Außerdem ist zu beachten, daß Änderung an Ihrer Login-Prozedur `.profile` erst nach der nächsten Anmeldung wirksam sind. Fehler bei der Eingabe Ihres Benutzernamens oder Ihres Paßworts müssen deshalb weiterhin über die Taste <BACKSPACE> korrigiert werden.

Bei allen Tasten ist zu beachten, daß sie nur für die aktuelle Zeile gültig ist. Betätigen Sie also erst dann die Taste <RETURN>, nachdem Sie Ihre Eingabe genau auf Fehler überprüft und alle Fehler korrigiert haben.

Sonderzeichen als darstellbare Zeichen

Die Sonderzeichen, mit denen einige Tasten belegt sind, können auch als darstellbare Zeichen benutzt werden. Da UNIX diese Sonderzeichen als Kommandos interpretiert, müssen Sie das System darüber informieren, wenn es die Sonderfunktion der Taste ignorieren oder aufheben soll. Dazu verwenden Sie den Gegenschrägstrich (\). Jedem Sonderzeichen, das als darstellbares Zeichen verwendet werden soll, stellen Sie einen Gegenschrägstrich (\) voran. Damit signalisieren Sie dem System, daß es die spezielle Bedeutung dieses Zeichens ignorieren soll und das Zeichen als darstellbares Zeichen in den Text einfügen soll.

Wenn Sie z.B. in eine Datei den Satz

> He bought three pounds @ $.05 cents each.

einfügen möchten, müssen Sie verhindern, daß UNIX das Zeichen `@` als Aufforderung zum Löschen eines Zeichens interpretiert. Hierzu setzen Sie vor das Zeichen `@` einen Gegenschrägstrich (\); andernfalls löscht das System alle Wörter, die dem Zeichen `@` in der aktuellen Zeile vorangehen. Ihr Satz sieht dann folgendermaßen aus:

```
$.05 cents each.
```

Dies können Sie vermeiden, indem Sie den Satz wie folgt eingeben:

```
He bought three pounds \@ $.05 cents each.
```

Eingabegeschwindigkeit

Sobald auf dem Terminal die Eingabeaufforderung erscheint, können Sie Ihr Kommando in beliebiger Geschwindigkeit eingeben; dies gilt selbst dann, wenn UNIX gerade ein Kommando verarbeitet oder dessen Ergebnis ausgibt. Da Ihre Eingabe und die Ausgabe des Systems gleichzeitig auf dem Bildschirm erscheinen, kann es vorkommen, daß die auf dem Bildschirm erscheinenden Ein- und Ausgaben sich gegenseitig überlagern. So störend dies für Sie auch sein mag - auf die Verarbeitung der Daten durch UNIX hat dies keinerlei Auswirkung, da das UNIX-System Ihre Eingabe selbst dann in seinen Tastaturpuffer einliest, wenn es gerade mit einer Ausgabe beschäftigt ist (die Kommunikation findet, wie bereits erwähnt, im Vollduplex-Betrieb statt). Dadurch kann das System die Ein- und Ausgabe gleichzeitig verarbeiten. Das System nimmt also Ihre Eingabe (Ihre nächste Anforderung) entgegen und speichert sie ab, während es seine Ausgabe (die Antwort auf Ihre letzte Anforderung) auf dem Bildschirm anzeigt.

Abbrechen eines Kommandos

Die Verarbeitung der meisten Kommandos kann über die Taste <BREAK> oder <DELETE> abgebrochen werden. Daraufhin bricht UNIX das Programm ab und gibt eine neue Eingabeaufforderung auf dem Bildschirm aus, um Ihnen seine Empfangsbereitschaft zu signalisieren.

Die Steuerzeichen

Stellen Sie zunächst die Position der mit CONTROL oder CTRL beschrifteten CONTROL-Taste auf Ihrer Tastatur fest. In vielen Fällen befindet sie sich links von der A-Taste oder unterhalb der Y-Taste. Diese Taste wird zusammen mit anderen Zeichen zur physischen Beeinflussung der Eingabezeile benutzt. Kommandos, die auf diese Weise eingegeben werden, nennt man Steuerzeichen. Mit einigen dieser Steuerzeichen werden sehr einfache Aufgaben durchgeführt, z.B. das Zurücksetzen des Cursors um eine Stelle oder die Eingabe von Tabulatorzeichen. Mit anderen Steuerzeichen werden UNIX-spezifische Kommandos eingegeben. Über eines dieser Steuerzeichen (Standard: CONTROL-s) können Sie beispielsweise die Bildschirmausgabe anhalten.

Zur Eingabe eines Steuerzeichens halten Sie die CONTROL-Taste gedrückt, während Sie die entsprechende Buchstabentaste betätigen. Die meisten dieser Steuerzeichen erscheinen bei ihrer Eingabe nicht auf dem Bildschirm. Sie sind deshalb in diesem Leitfaden in spitze Klammern eingeschlossen (siehe dazu

Abschnitt "Beschreibungsformat" im Vorwort). Für die CONTROL-Taste wird ein Zirkumflex (^) eingegeben, gefolgt von der entsprechenden Buchstabemtaste. So steht <^s> beispielsweise für das Steuerzeichen Control-s.

Am häufigsten werden die Steuerzeichen verwendet, mit denen die Bildschirmausgabe gesteuert und die Abmeldung vom System vorgenommen wird. Um zu verhindern, daß die Informationen zu schnell über den oberen Bildschirmrand hinausjagen, geben Sie das Steuerzeichen <^s> ein; hiermit können Sie die Bildschirmausgabe anhalten. Sobald Sie die nächste Bildschirmseite abrufen möchten, geben Sie das Steuerzeichen <^q> ein.

Zur Abmeldung vom UNIX-System geben Sie das Steuerzeichen <^d> ein (siehe Abschnitt "Abmelden " in diesem Kapitel).

Mit weiteren Steuerzeichen können Sie unter dem UNIX-System Funktionen aufrufen, für die auf einigen Terminals keine speziellen Tasten vorhanden sind. Gibt es auf Ihrer Tastatur beispielsweise keine Korrekturtaste, können Sie stattdessen das Steuerzeichen <^h> verwenden. Ebenso können mit dem Steuerzeichen <^i> Tabulatorzeichen eingegeben werden, wenn keine spezielle Tabulatortaste vorhanden ist (die Umsetzung der Tabulatorfunktion wird im Abschnitt "Bei Problemen mit der Anmeldung" in diesem Kapitel beschrieben).

Nachdem Sie nun das Terminal ordnungsgemäß konfiguriert haben und sich mit der Tastatur vertraut gemacht haben, fehlt Ihnen zum Arbeiten mit UNIX nur noch ein Schritt: Sie brauchen einen Benutzernamen.

Der Benutzername

Den Benutzernamen müssen Sie bei jeder Anmeldung beim System eingeben. Anhand dieses Namens prüft das System, ob Sie eine Berechtigung zum Arbeiten am System besitzen. Das System hält Datum und Zeitpunkt Ihrer Anmeldung jedesmal fest.

Damit Sie in den Besitz eines gültigen Benutzernamens gelangen, muß Ihr Systemverwalter zunächst einen Benutzereintrag für Sie anlegen. Für die Zusammensetzung des Benutzernamens gibt es nur wenige Regeln. Im Normalfall ist er drei bis acht Zeichen lang. Er kann Groß- und Kleinbuchstaben, Zahlen sowie den Unterstrich (_) enthalten; allerdings darf er nicht mit einer Zahl beginnen.

Bei der Wahl Ihres Benutzernamens werden Sie sich wahrscheinlich nach den Gepflogenheiten an Ihrem lokalen System richten. Vielleicht ist es an Ihrem System beispielsweise üblich, daß alle Benutzer als Benutzernamen ihre Initialen, ihren Nachnamen oder einen Spitznamen wählen. Zulässig sind beispielsweise die Benutzernamen `starship`, `mary2` und `jrms`.

Herstellen der Verbindung mit UNIX

Hinweis: In diesem Abschnitt wird davon ausgegangen, daß Ihr Terminal mit dem Rechner direkt oder über eine Telephonleitung verbunden ist. Zur Anmeldung über eine Telephonleitung gibt es eine Vielzahl von Möglichkeiten (in manchen Fällen ist beispielsweise aus Sicherheitsgründen eine spezielle Telephonnummer oder ein spezieller Sicherheitscode notwendig); im folgenden wird eine Anmeldeprozedur beschrieben, die für die meisten Fälle geeignet sein dürfte. Falls Sie die Verbindung mit Ihrem UNIX-System von einem Terminal aus aufnehmen möchten, das sich nicht am gleichen Standort wie der Rechner befindet, erkundigen Sie sich beim Systemverwalter nach der notwendigen Anmeldeprozedur.

Schalten Sie Ihr Terminal ein. Wenn es direkt mit dem Rechner verbunden ist, wird in der oberen linken Bildschirmecke die Eingabeaufforderung `login:` dargestellt.

Wenn Sie mit dem Rechner über eine Telephonleitung Daten austauschen möchten, muß zunächst die Verbindung hergestellt werden. Dies kann folgendermaßen geschehen:

1. Geben Sie die Telephonnummer ein, um die Verbindung mit dem UNIX-System herzustellen. Auf dem Bildschirm erscheint daraufhin eine der folgenden Meldungen:

    1. `BUSY` Dies bedeutet, daß die Leitung gerade belegt ist. Versuchen Sie, den Rechner erneut anzuwählen.

    2. `NO ANSWER` Dies bedeutet normalerweise, daß das System wegen einer mechanischen Störung oder Problemen mit der Elektronik keine Verbindung herstellen kann. Überprüfen Sie die Verbindungen zwischen Ihrem Terminal, dem Modem und der Telephonleitung. Wählen Sie daraufhin den Rechner erneut an.

    3. `ONLINE` Dies zeigt Ihnen, daß die Verbindung mit dem System erfolgreich hergestellt worden ist.

2. Betätigen Sie nun die Taste `<RETURN>`. Das System fordert Sie daraufhin mit `login:` zur Anmeldung auf.

3. Möglicherweise erscheinen auf dem Bildschirm eine Reihe von Zeichen, die keinen Sinn ergeben. Dies liegt dann daran, daß die von Ihnen angewählte Übertragungsleitung mehrere Übertragungsgeschwindigkeiten unterstützt und UNIX versucht, die Daten mit einer falschen

Übertragungsgeschwindigkeit an Ihr Terminal zu schicken. In einem solchen Fall betätigen Sie die Taste <BREAK> oder <RETURN>, um zu einer anderen Übertragungsgeschwindigkeit umzuschalten. Erscheint dann immer noch nicht die Eingabeaufforderung `login:` (warten Sie einige Sekunden), so versuchen Sie es erneut über die Taste <BREAK> oder <RETURN>.

Die Anmeldung

Sobald auf dem Bildschirm die Eingabeaufforderung `login:` erscheint, geben Sie Ihren Benutzernamen ein und betätigen Sie dann die Taste <RETURN>. Beim Benutzernamen `starship` sieht Ihre Anmeldungs-Zeile folgendermaßen aus:

```
login: starship<CR>
```

Einen Eingabefehler, den Sie während der Eingabe Ihres Benutzernamens gemacht haben, können Sie, wie bereits erwähnt, über die Korrekturtaste <BACKSPACE> oder das Zeichen @ beheben.

Hinweis: Beachten Sie, daß der Benutzername in Kleinbuchstaben eingegeben werden muß. Falls Sie bei der Anmeldung Großbuchstaben verwenden, geht UNIX davon aus, daß alle weiteren Eingaben in Großschreibweise erfolgen, und erzeugt auch seine Ausgabe in Großbuchstaben.

Das Paßwort

Als nächstes fordert das System Sie zur Eingabe Ihres Paßworts auf. Geben Sie Ihr Paßwort ein und betätigen Sie die Taste <RETURN>. Auch hier können Sie einen Fehler bei der Eingabe des Paßworts über die Taste <BACKSPACE> oder das Zeichen @ beheben. Ihr Paßwort erscheint unter UNIX aus Sicherheitsgründen nicht auf dem Bildschirm.

Nachdem sowohl der Benutzername als auch das Paßwort vom System akzeptiert worden sind, kann es vorkommen, daß an dieser Stelle eine Tagesmeldung (message of the day) und/oder aktuelle Informationen erscheinen. Darauf folgt das Standard-Symbol, das Sie zur Eingabe eines Kommandos auffordert, nämlich das Dollar-Zeichen ($). (die Tagesmeldung kann z.B. den Zeitplan für die Systemwartung enthalten. Die aktuellen Informationen können z.B. ein

neues Systemwerkzeug ankündigen). Nach Ihrer Anmeldung sieht Ihr Bildschirm in etwa wie folgt aus:

```
login:  starship<CR>
password:
UNIX system news
$
```

Bei Eingabefehlern während der Anmeldung erscheint auf dem Bildschirm die Meldung `login incorrect`; Sie können dann erneut versuchen, sich bei der Eingabeaufforderung `login:` anzumelden.

```
login: starship<CR>
password:
login incorrect
login:
```

Bei Ihrer ersten Anmeldung bei UNIX weicht der Ablauf Ihrer Anmeldeprozedur möglicherweise von der Beschreibung in diesem Abschnitt ab. Dies liegt möglicherweise daran, daß Ihr Systemverwalter Ihnen beim Einrichten Ihres Benutzereintrags aus Sicherheitsgründen ein temporäres Paßwort zugewiesen hat; in einem solchen Fall können Sie sich erst dann beim System anmelden, nachdem Sie ein neues Paßwort ausgewählt haben.

Der Zwang zur Wahl eines Paßworts, das nur Ihnen bekannt ist, stellt einen entscheidenden Beitrag zur Erhöhung der Systemsicherheit dar. Das Paßwort dürfen Sie auf keinen Fall weitergeben, um den Schutz der Systemressourcen und Ihrer eigenen Daten zu gewährleisten.

Der tatsächliche Ablauf Ihrer Anmeldeprozedur hängt von der Einrichtung Ihres Rechnersystems durch den Systemverwalter ab. Im Großen und Ganzen wird die Prozedur jedoch bei einer Erstanmeldung aus den folgenden Schritten bestehen:

1. Stellen Sie die Verbindung her. UNIX zeigt daraufhin die Eingabeaufforderung `login:` an. Geben Sie Ihren Benutzernamen ein und betätigen Sie die Taste `<RETURN>`.
2. UNIX fordert Sie nun zur Eingabe Ihres Paßworts auf. Geben Sie das temporäre Paßwort ein und betätigen Sie die Taste `<RETURN>`.
3. Das System informiert Sie darüber, daß das temporäre Paßwort nicht mehr gültig ist und ein neues gewählt werden muß.
4. Zunächst fordert das System Sie zur Eingabe Ihres alten Paßworts auf. Geben Sie Ihr temporäres Paßwort ein.
5. Daraufhin fordert das System Sie zur Eingabe Ihres neuen Paßworts auf. Geben Sie das gewählte Paßwort ein.

   Für die Bildung von Paßwörtern gelten folgende Regeln:

   - Ein Paßwort muß aus mindestens sechs Zeichen bestehen. Bei mehr als acht Zeichen werden nur die ersten acht Zeichen vom System gelesen.
   - Jedes Paßwort muß mindestens zwei Buchstaben sowie mindestens ein numerisches oder Sonderzeichen enthalten. Buchstaben können sowohl in Groß- als auch Kleinschreibweise eingegeben werden.
   - Das Paßwort darf nicht mit dem Benutzernamen identisch sein. Unzulässig sind auch Paßwörter, die lediglich eine leicht abgewandelte Variante des Benutzernamens darstellen (z.B. durch Vertauschen einiger Zeichen oder einfaches Umdrehen des Benutzernamens entstanden sind). Groß- und Kleinbuchstaben werden bei diesem Vergleich als identisch betrachtet.
   - Das neue Paßwort muß sich vom alten durch mindestens drei Zeichen unterscheiden. Auch bei diesem Vergleich werden Groß- und Kleinbuchstaben als identisch betrachtet.

   Gültige Paßwörter sind z.B.: `mar84ch`, `Jonath0n`, und `BRAV3S`.

   Hinweis: Möglicherweise gelten auf Ihrem UNIX-System für die Bildung von Paßwörtern andere Regeln. Auskünfte darüber kann Ihnen Ihr Systemverwalter erteilen.

6. UNIX fordert Sie zur Kontrolle zur erneuten Eingabe Ihres neuen Paßworts auf. Geben Sie das neue Paßwort erneut ein.
7. Entspricht die zweite Eingabe des Paßworts nicht der ersten, werden Sie vom System darauf aufmerksam gemacht und zur Wiederholung der Prozedur aufgefordert. Bei einigen Systemen wird allerdings die Verbindung abgebrochen, wenn die zweite Eingabe des Paßworts nicht mit der ersten übereinstimmt. In diesem Fall müssen Sie die Anmeldeprozedur wieder bei Schritt 1 beginnen. Bei einer Übereinstimmung der beiden Paßwörter zeigt das System die Eingabeaufforderung an.

Der folgende Bildschirm zeigt den vollständigen Ablauf (Schritte 1 bis 6) einer Erstanmeldung bei UNIX.

```
login: starship  <CR>
password: <CR>
Your password has expired.
Choose a new one.
Old password: <CR>
New password: <CR>
Re-enter new password: <CR>
$
```

Bei Problemen mit der Anmeldung

Im Normalfall gibt es bei der Anmeldung auf dem Terminal keine Probleme, vorausgesetzt, es ist richtig konfiguriert. Manchmal treten jedoch unvorhersehbare Probleme auf. So kann es beispielsweise vorkommen, daß die Wagenrücklauftaste nicht ordnungsgemäß funktioniert.

Einige Probleme lassen sich einfach dadurch lösen, daß man das System abschaltet und sich neu anmeldet. Falls das Problem damit nicht behoben ist, sollten Sie zuerst folgende Fehlerquellen überprüfen und erst dann eine Neuanmeldung versuchen:

Tastatur	Tasten mit der Beschriftung CAPS, LOCAL, und BLOCK dürfen nicht aktiviert (in Feststell-Position) sein. Zum Lösen der Tasten müssen Sie sie lediglich einmal betätigen.
Modem	Wenn Ihr Terminal mit dem System über eine Telephonleitung verbunden ist, so überprüfen Sie, ob die Übertragungsgeschwindigkeit und der Vollduplex-Betrieb ordnungsgemäß eingestellt sind.
Schalter	Auf einigen Terminals müssen zur Herstellung der Kompatibilität zum UNIX-System Schalter gesetzt werden. Stellen Sie sicher, daß diese Schalter korrekt gesetzt sind.

Informationen zur Überprüfung der Konfiguration Ihres Terminals finden Sie im Abschnitt "Konfiguration des Terminals" in diesem Kapitel. Bei Fragen zu Ihrer Tastatur, Ihrem Terminal oder Ihrem Modem schlagen Sie bitte in der entsprechenden Dokumentation nach.

Bild 2-2 zeigt eine Reihe von Problemen bei der Anmeldung mit ihren Ursachen und Lösungsmöglichkeiten. Bei weiteren Fragen wenden Sie sich bitte an Ihren Systenmverwalter.

Bild 2-2: Fehlersuche bei Problemen mit der Anmeldung*

Problem †	Mögliche Ursache	Maßnahme
Sinnlose Zeichen	Unkorrekte Übertragungsgeschwindigkeit des UNIX-Systems.	Betätigen Sie die Taste <RETURN> oder <BREAK>.
Ein-/Ausgabe in GROSS-BUCHSTABEN.	Das Terminal ist auf GROSS-BUCHSTABEN eingestellt.	Melden Sie sich ab und stellen Sie die Zeichengenerierung auf Kleinbuchstaben um.
Die Eingabe erscheint in GROSSBUCHSTABEN, die Ausgabe in kleinbuchstaben.	Die Taste CAPS (bzw. CAPS LOCK) ist aktiviert	Betätigen Sie die Taste CAPS bzw. CAPS LOCK) um sie zu freizugeben.
Die Eingabe erscheint doppelt.	Das Terminal ist auf HALBDUPLEX geschaltet.	Schalten Sie das Terminal in den VOLLDUPLEX- Betrieb um.
Die Tabulatortaste funktioniert nicht ordnungsgemäß.	Die Tabulatorzeichen werden nicht korrekt in Leerzeichen umgesetzt.	Geben Sie `stty -tabs` ‡ ein.
Die Verbindung kann nicht hergestellt werden, obwohl beim Anwählen ein hoher Ton zu hören ist.	Das Terminal ist auf LOKAL- oder OFF-LINE- Betrieb geschaltet.	Schalten Sie das Terminal auf ON-LINE- Betrieb um und wiederholen Sie die Anmeldung.
Die Übertragungsleitung (Verbindung zum UNIX- System) wird wiederholt unterbrochen.	Die Telephonleitung oder der Anschluß ist gestört.	Wenden Sie sich an den Systemverwalter.

* Viele Probleme sind auf eine unkorrekte Konfiguration zurückzuführen. Überprüfen Sie vor Ihrer Anmeldung die Konfiguration anhand des Abschnitts "Konfiguration des Terminals", um derartige Fehler auszuschließen.

† Einige Probleme können auf Ihr spezielles Terminal oder das benutzte Modem zurückzuführen sein. Schlagen Sie in der entsprechenden Dokumentation nach, falls die Probleme mit den vorgeschlagenen Maßnahmen nicht gelöst werden können.

‡ Mit der Eingabe von `stty -tabs` werden die eingegebenen Tabulatorzeichen nur während der aktuellen Sitzung in die entsprechende Anzahl von Leerzeichen umgesetzt. Zur permanenten Einstellung dieser Funktion müssen Sie die Zeile `stty-tabs` in Ihre Login-Prozedur `.profile` einfügen (siehe Kapitel 7).

Einfache Kommandos

Wenn die Eingabeaufforderung am Bildschirm erscheint, hat UNIX Sie als berechtigten Benutzer akzeptiert. Das System wartet nun darauf, daß Sie durch die Eingabe eines Kommandos ein Programm anfordern.

Versuchen Sie es beispielsweise mit dem Kommando `date` Geben Sie nach der Eingabeaufforderung das Kommando ein und betätigen Sie die Taste <RETURN>. UNIX greift nun auf ein Programm mit dem Namen `date` zu, führt es aus und zeigt das Ergebnis auf dem Bildschirm an:

```
$ date<CR>
Tues Jul 18 14:49:07 EDT 1989
$
```

Das Kommando `date` bewirkt also, daß das Datum sowie die Uhrzeit im 24-Stunden-Format ausgegeben werden.

Geben Sie jetzt das Kommando `who` ein und betätigen Sie die Taste <RETURN>. Ihre Bildschirmanzeige sieht dann in etwa wie folgt aus:

```
$ who<CR>
starship        term/00       Jul 18       8:53
mlf             term/02 Jul 188:56
jro             term/05       Jul 18       8:54
ral             term/06       Jul 18       8:56
$
```

Das Kommando `who` listet die Benutzernamen aller Benutzer auf, die gegenwärtig an Ihrem System arbeiten. Die Einträge mit dem Bezeichner tty gehören zu Gerätedateien, die den Terminals der verschiedenen Benutzer zugeordnet sind. Außerdem werden Sie über Datum und Uhrzeit der

Anmeldung der einzelnen Benutzer informiert.

Abmelden

Zum Abschluß einer UNIX-Sitzung geben Sie bei der UNIX-Eingabeaufforderung das Steuerzeichen <^d> ein (zur Erinnerung: ein Steuerzeichen wie <^d> wird eingegeben, indem man die die CONTROL-Taste gedrückt hält und gleichzeitig die entsprechende Buchstabentaste betätigt. Steuerzeichen sind nicht-darstellbare Zeichen und werden somit nicht auf dem Bildschirm angezeigt). Nach ein paar Sekunden erscheint erneut die UNIX-Eingabeaufforderung `login:`.

```
$ <^d>
login:
```

Dies zeigt Ihnen, daß Ihre Abmeldung erfolgreich durchgeführt worden ist. Jetzt kann sich ein neuer Benutzer beim System anmelden.

Hinweis: Sie sollten sich immer mit <^d> von UNIX abmelden, bevor Sie das Terminal abschalten oder die Telephonverbindung abbrechen; andernfalls werden Sie nicht wirklich vom System abgemeldet.

3 Das Dateisystem

Einführung 3-1

Der Aufbau des Dateisystems 3-2

Ihre Position im Dateisystem 3-4
Ihr Home-Verzeichnis 3-4
Ihr aktuelles Verzeichnis 3-6
Pfadnamen 3-7
- Absolute Pfadnamen 3-8
- Relative Pfadnamen 3-11
- Benennung von Verzeichnissen und Dateien 3-16

Organisation eines Verzeichnisses 3-17
Anlegen von Verzeichnissen: Das Kommando `mkdir` 3-17
Abrufen des Verzeichnisinhalts: Das Kommando `ls` 3-19
- Einige wichtige `ls`-Optionen 3-21

Aktuelles Verzeichnis wechseln: Das Kommando `cd` 3-26
Löschen von Verzeichnissen: Das Kommando `rmdir` 3-28

Zugriff auf und Manipulation von Dateien 3-31
Grundlegende Kommandos 3-31
- Inhalt einer Datei ausgeben: Die Kommandos `cat`, `pg` und `pr` 3-32
- Kopie einer Datei erstellen: Das Kommando `cp` 3-44
- Dateien versetzen und umbenennen: Das Kommando `mv` 3-47

- Datei löschen: Das Kommando `rm` 3-49
- Anzahl der Zeilen, Wörter und Zeichen in einer Datei ausgeben: Das Kommando `wc` 3-51
- Zugriffsschutz für Dateien: Das Kommando `chmod` 3-54

Kommandos für Fortgeschrittene 3-61

- Vergleich von Dateien: Das Kommando `diff` 3-63
- Mustervergleich in Dateien: Das Kommando`grep` 3-65
- Dateien sortieren und mischen: Das Kommando `sort` 3-68

Einführung

Die effektive Benutzung des UNIX-Dateisystems setzt Kenntnisse über seine Struktur und Ihre Einordnung innerhalb dieser Struktur voraus, außerdem Kenntnisse darüber, wie sich Ihre Einordnung bei einem Wechsel innerhalb des Dateisystems ändert. Dieses Kapitel führt Sie in die Benutzung des Dateisystems ein.

Die ersten beiden Abschnitte ("Der Aufbau des Dateisystems" und "Ihre Position im Dateisystem") enthalten allgemeinere Informationen zum Dateisystem. In den übrigen Abschnitten dieses Kapitels lernen Sie die UNIX-Kommandos kennen, mit denen Sie Ihre eigene Verzeichnisstruktur aufbauen, auf Unterverzeichnisse und die darin enthaltenen Dateien zugreifen, sie manipulieren und den Inhalt anderer Verzeichnisse, für die Sie die Zugriffsberechtigung haben, abrufen können.

Jedes Kommando wird in einem eigenen Unterabschnitt behandelt. Am Ende jedes Unterabschnitts befindet sich eine Tabelle mit der Zusammenfassung des jeweiligen Kommandos, so daß Sie die Syntax und Funktion des Kommandos später schnell nachschlagen können. Viele der aufgeführten Kommandos haben noch zusätzliche, kompliziertere Anwendungsmöglichkeiten, die jedoch dem erfahrenen Anwender vorbehalten sind und in anderen UNIX-Handbüchern beschrieben werden. Die hier aufgeführten Kommandos bilden die Grundlage für einen effektiven Umgang mit dem Dateisystem. Wenden Sie das jeweils vorgestellte Kommando gleich an, um mit ihm vertraut zu werden.

Der Aufbau des Dateisystems

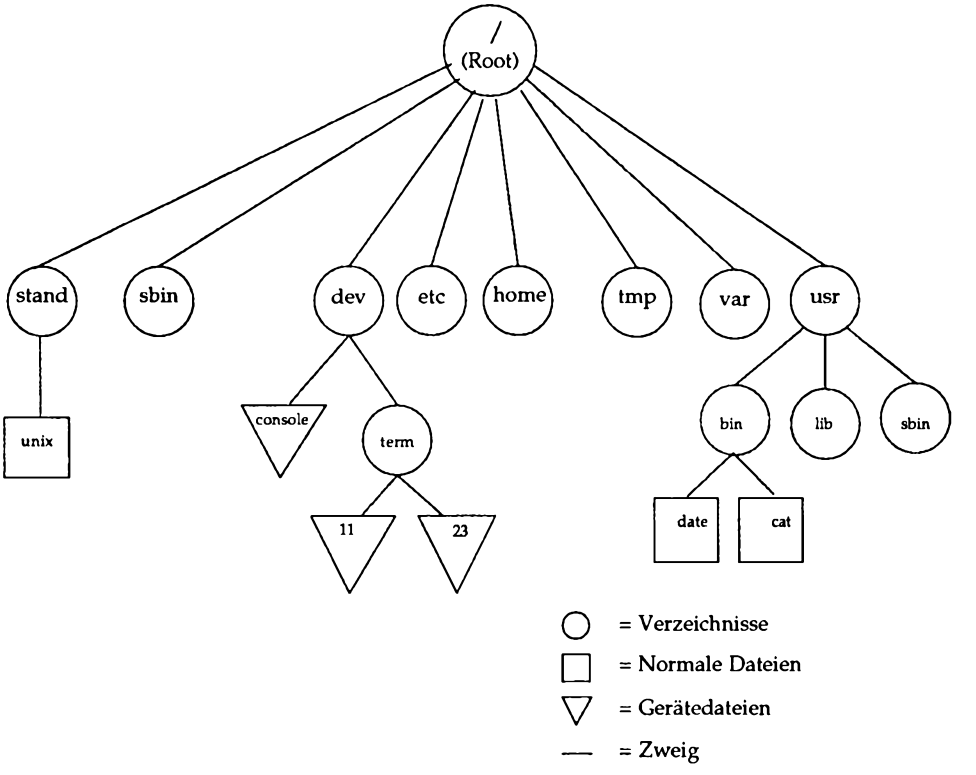
Das Dateisystem besteht aus normalen Dateien, Gerätedateien, symbolischen Verweisen und Verzeichnissen. Diese Komponenten ermöglichen das "elektronische" Organisieren, Auffinden und Verwalten von Informationen. Im Kapitel 1 wurden die Eigenschaften der Verzeichnisse und Dateien kurz erklärt. In diesem Kapitel finden Sie nochmals einen kurzen Überblick über diese Eigenschaften, dann eine Einführung in das Arbeiten mit Dateien und Verzeichnissen.

- Eine normale Datei ist eine Ansammlung von Zeichen, die auf Platte oder Diskette gespeichert sind. Sie kann z.B. den Text für einen Bericht oder den Code für ein Programm enthalten.
- Eine Gerätedatei steht für ein physisches Gerät wie z.B. ein Terminal oder eine Platte.
- Ein symbolischer Verweis ist eine Datei, die auf eine andere Datei zeigt.
- Ein Verzeichnis ist eine Ansammlung von Dateien und weiteren Verzeichnissen (sogenannte Unterverzeichnisse). Sie können Dateien nach beliebigen Kriterien zu Verzeichnissen zusammenfassen. So bietet es sich in einer Firma beispielsweise an, für jedes angebotene Produkt oder die Korrespondenz mit den verschiedenen Firmen ein eigenes Verzeichnis anzulegen.

Alle Verzeichnisse und Dateien sind in einer baumförmigen, hierarchischen Struktur angeordnet. Im Bild 3-1 sehen Sie ein Beispiel für eine Dateihierarchie, an deren Ausgangspunkt ein Verzeichnis namens Root (mit / gekennzeichnet) steht. Wenn Sie die bei Root beginnenden Zweige weiterverfolgen, gelangen Sie zu anderen wichtigen Systemverzeichnissen. Von diesen Verzeichnissen aus können Sie wiederum alle anderen Verzeichnisse und Dateien des Dateisystems erreichen.

In dieser Hierarchie nennt man die Beziehung zwischen dem Verzeichnis und den darin enthaltenen Dateien und Unterverzeichnissen eine Vater/Sohn-Beziehung. Eine derartige Beziehung kann auf vielen Ebenen der Dateihierarchie vorliegen. Für die Anzahl der Dateien und Verzeichnisse, die Ihren Verzeichnissen enthalten sind, gibt es keinerlei Einschränkungen; dasselbe gilt für die Anzahl der Verzeichnisebenen. Daraus ergibt sich für die Organisation Ihres Dateisystems eine Vielzahl von Möglichkeiten. Bild 3-1 zeigt ein Beispiel für ein komplexes Dateisystem.

Bild 3-1: Beispiel für ein Dateisystem



Ihre Position im Dateisystem

Ihr gesamter Dialog mit dem UNIX-System findet von einer bestimmten Position innerhalb des hierarchischen Dateisystems statt. Nach Ihrer Anmeldung werden Sie von UNIX automatisch an eine bestimmte Stelle innerhalb des Dateisystems gesetzt. Von dieser Stelle aus können Sie sich durch die gesamte Hierarchie bewegen, um in einem Ihrer Verzeichnisse oder mit einer Ihrer Dateien zu arbeiten. Zusätzlich können Sie auch auf die Verzeichnisse und Dateien, für die Sie nicht als Eigentümer eingetragen sind, zugreifen - vorausgesetzt, Sie haben dafür die entsprechende Berechtigung.

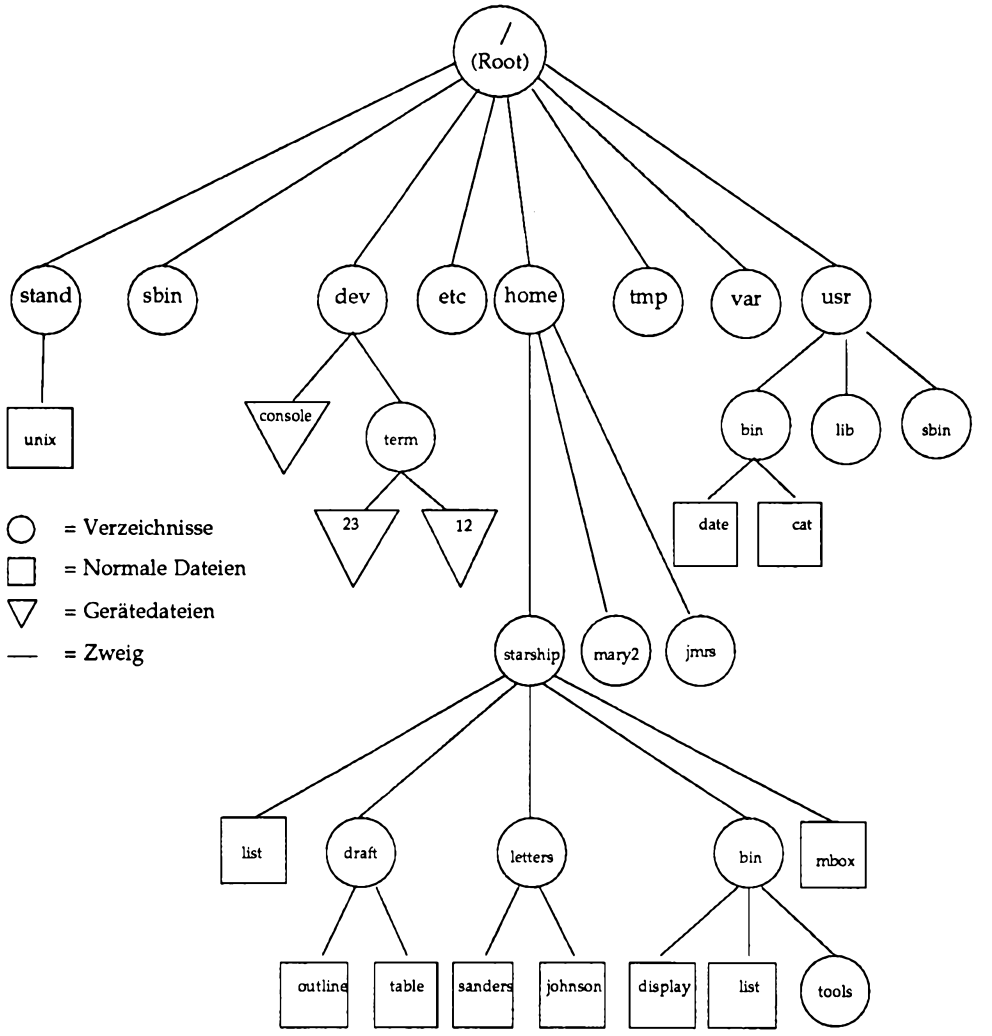
Der folgende Abschnitt beschreibt Ihre Position innerhalb des hierarchischen Dateisystems und wie sich diese Position verändert, während Sie sich durch das Dateisystem bewegen.

Ihr Home-Verzeichnis

Nachdem Sie sich erfolgreich bei UNIX angemeldet haben, versetzt UNIX Sie an eine bestimmte Stelle innerhalb des Dateisystems, nämlich in das sogenannte Home-Verzeichnis. Der Name Ihres Home-Verzeichnisses entspricht im Normalfall Ihrem Benutzernamen, der Ihnen beim Einrichten Ihres Benutzereintrags zugeteilt worden ist. Jeder Benutzer, dem ein Benutzername zugeordnet worden ist, hat im Dateisystem sein eigenes Home-Verzeichnis.

UNIX bewahrt den Überblick über die verschiedenen Home-Verzeichnisse, indem es ihre Namen und Struktur in einem oder mehreren Systemverzeichnissen protokolliert. So stehen z.B. die Home-Verzeichnisse der Benutzernamen `starship`, `mary2` und `jmrs` alle in einem Systemverzeichnis namens `home`. Bild 3-2 zeigt die Beziehungen zwischen einem Systemverzeichnis wie `home` und den anderen wichtigen UNIX-Systemverzeichnissen, die im Kapitel 1 beschrieben worden sind.

Bild 3-2: Verzeichnis der Home-Verzeichnisse



In Ihrem Home-Verzeichnis können Sie einerseits Dateien anlegen, andererseits weitere Verzeichnisse (sogenannte Unterverzeichnisse), in die Sie wiederum Dateien einbringen können. Sie können Ihre Dateien und Verzeichnisse nicht nur versetzen und löschen, sondern auch festlegen, ob andere Benutzer darauf zugreifen dürfen oder nicht. Da Sie der Eigentümer Ihres Home-Verzeichnisses sind, haben Sie die Verantwortung für seinen gesamten Inhalt. Ihr Home-Verzeichnis ist die Plattform, von der Sie sowohl alle darin enthaltenen (untergeordneten) Dateien und Verzeichnisse als auch den darüberliegenden Teil des Dateisystems bis hinauf zum Root-Verzeichnis überblicken können.

Ihr aktuelles Verzeichnis

Solange Sie in Ihrem Home-Verzeichnis arbeiten, nennt man dieses Verzeichnis Ihr aktuelles Verzeichnis bzw. Ihr Arbeitsverzeichnis. Sobald Sie in ein anderes Verzeichnis wechseln, wird dieses neue Verzeichnis zu Ihrem aktuellen Verzeichnis.

Das UNIX-Kommando `pwd` (für print working directory) zeigt den Namen des Verzeichnisses an, in dem Sie sich gerade befinden. Wenn Sie beispielsweise den Benutzernamen `starship` haben und bei der ersten Eingabeaufforderung nach Ihrer Anmeldung das Kommando `pwd` eingeben, erhalten Sie vom System die folgende Antwort:

```
$pwd<CR>
/home/starship
$
```

Das System gibt sowohl das Verzeichnis aus, in dem Sie gerade arbeiten (`starship`), als auch die Position dieses Verzeichnisses innerhalb des Dateisystems. Durch den Pfadnamen `/home/starship` erfahren Sie, daß das Root-Verzeichnis (das durch den ersten Schrägstrich (/) angegeben wird) das Verzeichnis `home` enthält, in dem sich wiederum das Unterverzeichnis `starship` befindet (die Schrägstriche dienen - mit Ausnahme des ersten Schrägstrichs in der Zeile - zur Abgrenzung der Verzeichnisse und Dateien untereinander und zeigen die Position des jeweiligen Verzeichnisses relativ zum Root-Verzeichnis). Ein Verzeichnisname, aus dem sämtliche Verzeichnisse hervorgehen, die zwischen dem Verzeichnis und dem Root-Verzeichnis liegen, wird absoluter

Pfadname genannt. Auf den folgenden Seiten werden wir den Aufbau des absoluten Pfadnamens analysieren und nachvollziehen, damit Sie Ihre ersten "Gehversuche" im Dateisystem unternehmen können.

Beachten Sie, daß Sie Ihre aktuelle Position im Dateisystem jederzeit mit dem Kommando `pwd` ermitteln können. Dies ist besonders hilfreich, wenn Sie versuchen, eine Datei zu lesen oder zu kopieren, und UNIX Sie in einer Meldung informiert, daß diese Datei nicht vorhanden ist.
Möglicherweise werden Sie sich zu Ihrer Überraschung in einem ganz anderen Verzeichnis befinden, als Sie ursprünglich angenommen hatten.

In Bild 3-3 enthält einen Überblick über die Syntax und Funktion des Kommandos `pwd`.

Bild 3-3: `pwd`-Kurzübersicht

Kurzbeschreibung `pwd` – Absoluten Namen des Arbeitsverzeichnisses ausgeben.		
Kommando	*Optionen*	*Argumente*
`pwd`	keine	keine
`Funktion:`	`pwd` gibt den absoluten Pfadnamen des Verzeichnisses aus, in dem Sie sich gerade befinden.	

Pfadnamen

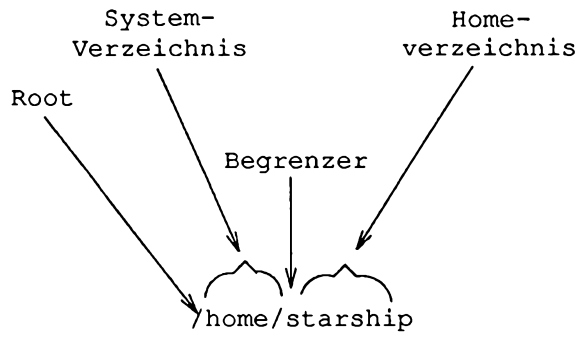
Unter UNIX besitzt jede Datei und jedes Verzeichnis einen eindeutigen Pfadnamen. Der Pfadname gibt die Position der Datei bzw. des Verzeichnisses innerhalb des Dateisystems an und zeigt, wie man darauf zugreifen kann. Damit Sie sich erfolgreich im Dateisystem bewegen können, müssen Sie wissen, wie die Angaben im Pfadnamen zu interpretieren sind. Zunächst müssen Sie zwischen zwei Arten von Pfadnamen unterscheiden: den absoluten und den relativen Pfadnamen.

Absolute Pfadnamen

Ein absoluter Pfadname zeichnet den Weg nach, der beim Root-Verzeichnis beginnt und über ein oder mehr Verzeichnisse zu einer bestimmten Datei oder zu einem bestimmten Verzeichnis führt. Mit Hilfe des absoluten Pfadnamens können Sie zu jeder Datei oder jedem Verzeichnis in Ihrem UNIX-System gelangen.

Da der absolute Pfadname in jedem Fall an der Root des Dateisystems beginnt, ist sein erstes Zeichen auf jeden Fall ein Schrägstrich (/). Der letzte Name in einem absoluten Pfadnamen ist entweder eine Datei- oder ein Verzeichnisname. Alle anderen Namen im Pfadnamen müssen sich auf Verzeichnisse beziehen.

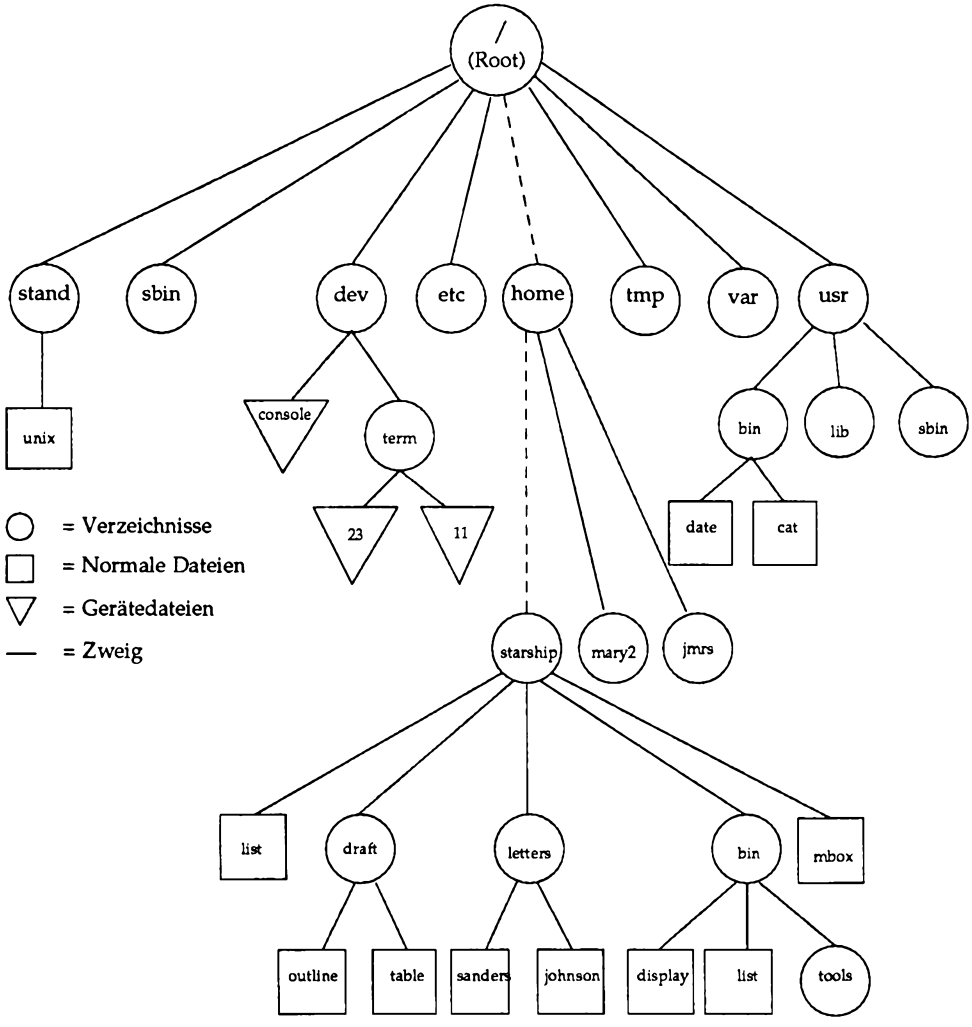
Das folgende Beispiel zeigt, wie ein absoluter Pfadname aufgebaut ist, und wie er Ihnen den Weg zu Ihrer Datei oder Ihrem Verzeichnis weist. Wenn Sie sich beispielsweise im Verzeichnis `starship` im Verzeichnis `/home` befinden, erhalten Sie über das Kommando `pwd` den absoluten Pfadnamen Ihres Arbeitsverzeichnisses (in diesem Fall `/home/starship`). Analysieren Sie die Komponenten dieses Pfadnamens anhand des folgenden Diagramms und der darauffolgenden Beschreibung:



/ (als erstes Zeichen)	=	Der Schrägstrich als erstes Zeichen im Pfadnamen steht für die Root des Dateisystems.
`home`	=	Das Systemverzeichnis unmittelbar unter der Root, auf das die Root zeigt bzw. in das die Root verzweigt.
/ (nicht als erstes Zeichen)	=	Der nächste Schrägstrich dient zur Abgrenzung der Verzeichnisnamen `home` und `starship`.
`starship`	=	Das aktuelle Arbeitsverzeichnis.

Verfolgen Sie jetzt den absoluten Pfad zum Verzeichnis `/home/starship` anhand der gestrichelten Linien in Bild 3-4.

Bild 3-4: Der absolute Pfadname des Verzeichnisses `/home/starship`



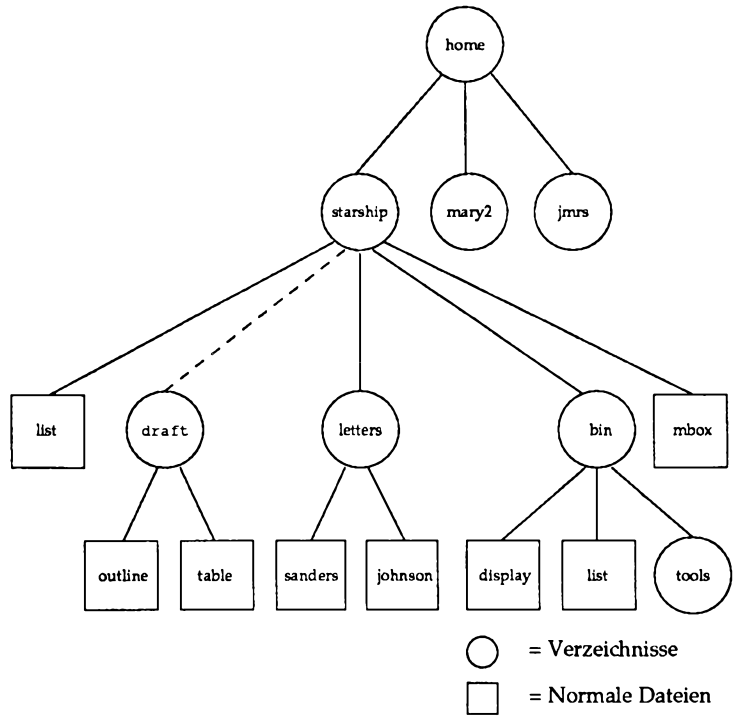
Relative Pfadnamen

Der Zugriffspfad, den ein relativer Pfadname angibt, beginnt im aktuellen Arbeitsverzeichnis. Er führt über eine Reihe von Verzeichnissen, die in der Verzeichnishierarchie entweder oberhalb oder unterhalb des aktuellen Arbeitsverzeichnisses liegen, zu einer bestimmten Datei bzw. einem bestimmten Verzeichnis. Wenn Sie die Hierarchie von Ihrem Arbeitsverzeichnis aus nach unten steigen, gelangen Sie zu Ihren eigenen Dateien oder Verzeichnissen. Wenn Sie die Hierarchie vom Arbeitsverzeichnis aus nach oben steigen, gelangen Sie über verschiedene Vaterverzeichnisse zum Ursprung aller Systemverzeichnisse, dem Root-Verzeichnis. Von dort aus können Sie in das gesamte Dateisystem gelangen.

Ein relativer Pfadname beginnt entweder mit einem Verzeichnisnamen, einem Dateinamen, einem Punkt (`.`) oder zwei Punkten (`..`). Der Punkt (`.`) steht für Ihr aktuelles Verzeichnis. Zwei Punkte (`..`) stehen für das Verzeichnis, das Ihrem aktuellen Verzeichnis in der Dateisystemhierarchie unmittelbar übergeordnet ist. Das durch `..` dargestellte Verzeichnis wird das Vaterverzeichnis von `.` genannt.

Angenommen, Sie befinden sich in unserem Beispieldateisystem im Verzeichnis `starship`; `starship` enthalte die Verzeichnisse `draft` und `letters`, `bin` sowie die Datei `mbox`. Der relative Pfadname dieser Datei und dieser Verzeichnisse ist dann einfach der eigene Name, z.B. `draft` oder `mbox`. Bild 3-5 zeichnet den relativen Pfad von `starship` zu `draft` nach.

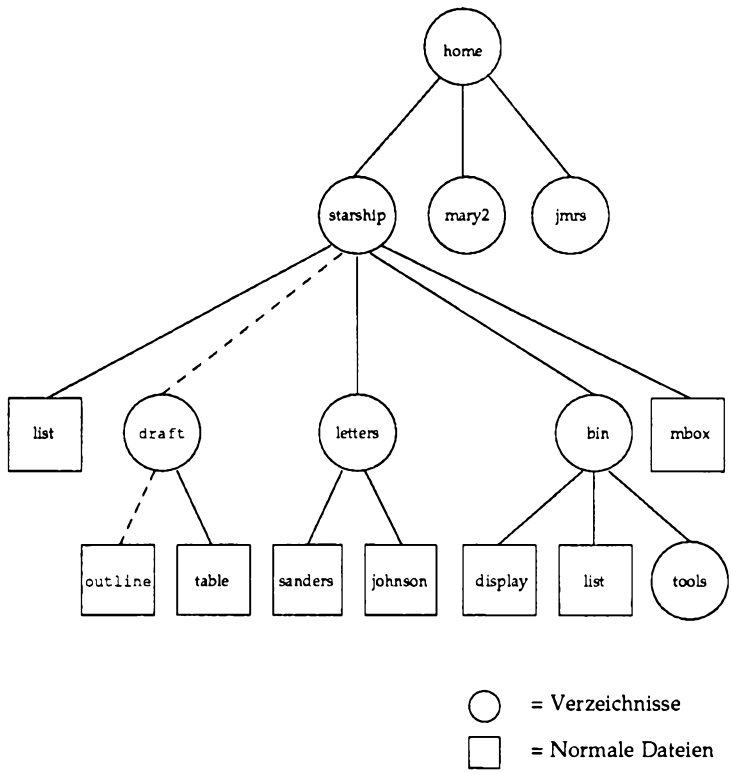
Bild 3-5: Der relative Pfadname des Verzeichnisses `draft`



Das Verzeichnis `draft`, das sich im Verzeichnis `starship` befindet, enthält die Dateien `outline` und `table`. Der relative Pfadname von `starship` zur Datei `outline` ist `draft/outline`.

Bild 3-6 zeichnet diesen relativen Pfad nach. Beachten Sie, daß der Schrägstrich (`/`) in diesem Pfadnamen das Verzeichnis `draft` von der Datei `outline` trennt. Der Schrägstrich (`/`) ist hier also ein Begrenzer, der anzeigt, daß die Datei `outline` dem Verzeichnis `draft` untergeordnet ist. D. h., die Datei `outline` ist ein Sohn des (Vater-)Verzeichnisses `draft`.

Bild 3-6: Der relative Pfadname von `starship` nach `outline`



Bisher beschränkte sich die Diskussion der relativen Pfadnamen auf die Angabe der Namen der Dateien oder Verzeichnisse, die zu Ihrem aktuellen Arbeitsverzeichnis gehören bzw. die Söhne des aktuellen Arbeitsverzeichnisses sind. Sie wissen jetzt, wie man in der Verzeichnishierarchie Ebene für Ebene nach unten steigt, bis man sein Ziel erreicht hat. Sie können jedoch genauso in der Dateihierarchie nach oben steigen, um danach zu Verzeichnissen und Dateien weiter oben oder weiter unten in der Verzeichnishierarchie zu gelangen.

Um zum Vaterverzeichnis Ihres aktuellen Verzeichnisses zu wechseln, können Sie das Kürzel `..` benutzen. Wenn Sie sich in unserem Beispieldateisystem also aktuell im Verzeichnis `draft` befinden, so ist `..` der Pfadname zum Verzeichnis `starship` und `../..` der Pfadname zum Vaterverzeichnis von `starship`, nämlich `home`.

Vom Verzeichnis `draft` aus können Sie auch einen Pfad zur Datei `sanders` aufbauen. Hierzu müssen Sie den folgenden Pfadnamen eingeben: `../letters/sanders`. Dabei gelangen Sie mit `..` zum Verzeichnis `starship`, dann mit `letters` und `sanders` über das Verzeichnis `letters` zur Datei `sanders` auf einer niedrigeren Ebene der Verzeichnishierarchie.

Beachten Sie, daß Sie anstelle eines relativen Pfadnamens in jedem Fall auch einen absoluten Pfadnamen angeben können.

Bild 3-7 enthält Beispiele für relative und absolute Pfadnamen.

Bild 3-7: Beispiele für Pfadnamen

Pfadname	Bedeutung
/	Absoluter Pfadname des Root-Verzeichnisses
/usr/bin	Absoluter Pfadname des Verzeichnisses bin, das in usr, einem Sohnverzeichnis von Root, enthalten ist (und die meisten ausführbaren Programme und Dienstprogramme enthält).
/home/starship/bin/tools	Absoluter Pfadname des Verzeichnisses tools, einem Sohnverzeichnis von bin, das wiederum im Verzeichnis starship enthalten ist; das Verzeichnis starship befindet sich im Verzeichnis home, das wiederum der Root untergeordnet ist.
bin/tools	Relativer Pfadname zur Datei bzw. zum Verzeichnis tools im Verzeichnis bin. Wenn / das aktuelle Verzeichnis ist, so sucht UNIX nach dem Pfad /usr/bin/tools. Ist das aktuelle Verzeichnis dagegen starship, so sucht UNIX nach dem absoluten Pfadnamen /home/starship/bin/tools.
tools	Relativer Pfadname der Datei bzw. des Verzeichnisses tools im aktuellen Verzeichnis.

Sie werden sicher etwas Übung benötigen, bis Sie sich mit Hilfe der Pfadnamen sicher im Dateisystem bewegen können. Lassen Sie sich jedoch nicht entmutigen - für das Erlernen einer neuen Konzeption ist das vollkommen selbstverständlich!

Benennung von Verzeichnissen und Dateien

Sie können Ihren Verzeichnissen oder Dateien beliebige Namen zuordnen, vorausgesetzt, Sie beachten folgende Regeln:

- Sie können alle Zeichen außer dem Schrägstrich (/) verwenden.
- Von der Verwendung einiger Zeichen und Tasten wird abgeraten. Dazu gehören die Leertaste, die Tabulatortaste, die Korrekturtaste und folgende Zeichen:

  ```
  ? @ # $ ^ & * ( ) ` [ ] \  |  ; ' " < >
  ```

  Ein Verzeichnis- oder Dateiname, in dem ein Leer- oder Tabulatorzeichen enthalten ist, muß in der Kommandozeile in Anführungszeichen gesetzt werden.
- Die Zeichen +, - oder . sollten nicht als erstes Zeichen eines Dateinamens verwendet werden.
- Groß- und Kleinbuchstaben werden unter UNIX unterschiedlich behandelt. Das Verzeichnis/die Datei mit dem Namen `draft` und das Verzeichnis/die Datei mit dem Namen `DRAFT` werden also als zwei unterschiedliche Verzeichnisse/Dateien betrachtet.

Im folgenden sind einige zulässige Datei- und Verzeichnisnamen aufgeführt:

```
memo        MEMO        section2    ref:list
file.d      chap3+4     item1-10    outline
```

In den übrigen Abschnitten des Kapitels lernen Sie verschiedene UNIX-Kommandos kennen, mit denen Sie Informationen über das Dateisystem abrufen können.

Organisation eines Verzeichnisses

Dieser Abschnitt stellt Ihnen vier UNIX-Kommandos vor, mit denen Sie eine Verzeichnisstruktur aufbauen und verwalten können: `mkdir`, `ls`, `cd` und `rmdir`.

`mkdir`	Im aktuellen Verzeichnis werden neue Verzeichnisse und Unterverzeichnisse angelegt.
`ls`	Die Namen aller Unterverzeichnisse und Dateien in einem Verzeichnis werden ausgegeben.
`cd`	Sie können in ein anderes Verzeichnis innerhalb des Dateisystems wechseln.
`rmdir`	Ein leeres Verzeichnis wird gelöscht.

Diese Kommandos können sowohl mit relativen als auch mit absoluten Pfadnamen verwendet werden. Bei den Kommandos `ls` und `cd` kann der Pfadname auch vollständig weggelassen werden. Eine ausführliche Beschreibung dieser Kommandos finden Sie in den folgenden vier Abschnitten.

Anlegen von Verzeichnissen: Das Kommando `mkdir`

Beim Anlegen der Unterverzeichnisse in Ihrem Home-Verzeichnis sollten Sie nach einem sinnvollen und logischen Schema vorgehen, das Ihnen das spätere Auffinden der Informationen in Ihren Dateien erleichtert. So können Sie die Dateien beispielsweise nach thematischen Gesichtspunkten auf die Verzeichnisse verteilen.

Mit dem Kommando `mkdir` (für make directory) können Sie ein neues Verzeichnis anlegen. Hierfür geben Sie einfach das Kommando zusammen mit dem Namen für das neue Verzeichnis (die neue Datei) ein. In unserem Beispiel hat der Eigentümer des Unterverzeichnisses `draft` das Verzeichnis `draft` angelegt, indem er im Home-Verzeichnis (`/home/starship`) das folgende Kommando eingegeben hat:

```
$ mkdir draft <CR>
$
```

Die zweite Eingabeaufforderung zeigt an, daß das Kommando erfolgreich ausgeführt worden ist.

Ebenfalls im Home-Verzeichnis hat der Benutzer nach derselben Methode zwei weitere Unterverzeichnisse angelegt, nämlich `letters` und `bin`:

```
$ mkdir letters<CR>
$ mkdir bin<CR>
$
```

Die drei Unterverzeichnisse (`draft`, `letters` und `bin`) können Sie auch alle gleichzeitig anlegen, indem Sie ihre Namen in dieselbe Kommandozeile schreiben:

```
$ mkdir draft letters bin<CR>
$
```

In einem neu angelegten Unterverzeichnis können weitere Unterverzeichnisse angelegt werden. Sie können Ihren Verzeichnissen und Dateien beliebige Namen geben; beachten Sie dabei jedoch die Regeln, die im Abschnitt "Benennung von Verzeichnissen und Dateien" aufgeführt sind.

Bild 3-8 enthält einen Überblick über Syntax und Funktion des Kommandos `mkdir`.

Bild 3-8: `mkdir`-Kurzübersicht

Kurzbeschreibung `mkdir` – Neues Verzeichnis anlegen.		
Kommando	*Optionen*	*Argumente*
`mkdir`	vorhanden	*verzeichnisname(n)*
`Funktion:`	`mkdir` legt ein neues Verzeichnis (Unterverzeichnis) an.	
`Hinweis:`	Nachdem das Verzeichnis angelegt worden ist, gibt das System eine Eingabeaufforderung (Standard: `$`) aus.	

Abrufen des Verzeichnisinhalts: Das Kommando `ls`

In allen Verzeichnissen des Dateisystems werden Informationen über die enthaltenen Dateien und Unterverzeichnisse geführt. Bei diesen Informationen handelt es sich beispielsweise um die Größe der Datei und das Datum der letzten Änderung. Diese Informationen über den Inhalt Ihres aktuellen Verzeichnisses und anderer Systemverzeichnisse können Sie mit dem Kommando `ls` (für list) abrufen.

Das Kommando `ls` gibt die Namen sämtlicher Dateien und Unterverzeichnisse aus, die in einem von Ihnen angegebenen Verzeichnis enthalten sind. Fehlt die Angabe eines Verzeichnisses, werden die Namen der Dateien und Verzeichnisse in Ihrem aktuellen Verzeichnis angezeigt. Machen Sie sich die Funktionsweise des Kommandos `ls` nochmals anhand des Dateisystems in Bild 3-2 klar.

Angenommen, Sie haben sich mit dem Benutzernamen `starship` bei UNIX angemeldet; wenn Sie jetzt das Kommando `pwd` eingeben, zeigt System den Pfadnamen `/home/starship` an. Geben Sie nun das Kommando `ls` ein und betätigen Sie die Taste `RETURN`, um einen Überblick über die Namen und Verzeichnisse und Dateien im aktuellen Verzeichnis zu erhalten. Das Ergebnis dieser Kommandofolge sieht dann folgendermaßen aus:

```
$ pwd<CR>
/home/starship
$ ls<CR>
bin
draft
letters
list
mbox
$
```

Das System zeigt also die Namen der Dateien und Verzeichnisse in Ihrem aktuellen Verzeichnis `starship` in alphabetischer Reihenfolge an (Verzeichnis- oder Dateinamen, die mit einer Zahl oder einem Großbuchstaben beginnen, würden dabei an erster Stelle stehen).

Sie können auch den Inhalt eines Verzeichnisses abrufen, das nicht Ihr aktuelles Verzeichnis ist, ohne Ihr aktuelles Verzeichnis zu verlassen. Dafür müssen Sie den Namen des Verzeichnisses folgendermaßen angeben:

`ls` *pfadname*<CR>

Das gewünschte Verzeichnis können Sie sowohl über den absoluten als auch über den relativen Pfadnamen angeben. So können Sie beispielsweise vom Verzeichnis `starship` aus den Inhalt von `draft` abrufen, indem Sie das Kommando `ls draft` eingeben und abschließend die Taste RETURN betätigen. Ihr Bildschirm sieht dann folgendermaßen aus:

```
$ ls draft<CR>
outline
table
$
```

Hier ist `draft` der relative Pfadname von einem Vater- (`starship`) zu einem Sohnverzeichnis (`draft`).

Mit Hilfe des relativen Pfadnamens können Sie auch den Inhalt eines Vaterverzeichnisses abrufen, wenn Sie sich in einem Sohnverzeichnis befinden. Dies ist am einfachsten durch die Eingabe von `..` (Punkt-Punkt) möglich. So gibt beispielsweise die folgende Kommandozeile den relativen Pfadnamen von `starship` zu `home` an:

```
$ ls ..<CR>
jmrs
mary2
starship
$
```

Dasselbe Ergebnis erzielen Sie mit dem absoluten Pfadnamen von der Root zum Verzeichnis `home`.

Wenn Sie das Kommando `ls /home` eingeben und die Taste RETURN betätigen, zeigt das System dieselbe Auflistung an.

Ähnlich können Sie mit dem Kommando `ls` und einem relativen oder absoluten Pfadnamen den Inhalt jedes beliebigen Systemverzeichnisses abrufen, für das Sie eine Zugriffsberechtigung haben.

Mit dem Kommando `ls` können Sie in einem umfangreichen Verzeichnis mit vielen Dateien schnell feststellen, ob sich eine bestimmte Datei in Ihrem aktuellen Verzeichnis befindet. Wenn Sie beispielsweise in Erfahrung bringen möchten, ob Ihr aktuelles Verzeichnis `draft` die Dateien `outline` und `notes` enthält, geben Sie das Kommando `ls` folgendermaßen ein:

```
$ ls outline notes<CR>
outline
notes: No such file or directory
$
```

Das System signalisiert Ihnen durch die Anzeige der Datei `outline`, daß diese Datei vorhanden ist; die Datei `notes` dagegen kann es nicht ausfindig machen.

Mit dem Kommando `ls` können Sie nicht den Inhalt einer Datei abrufen. Dazu gibt es die Kommandos `cat`, `pg` oder `pr`. Eine Beschreibung dieser Kommandos finden Sie im Abschnitt "Zugriff auf und Manipulation von Dateien" dieses Kapitels.

Einige wichtige `ls`-Optionen

Für das Kommando `ls` gibt es über ein Dutzend Optionen, mit denen Sie spezielle Informationen über eine Datei oder ein Unterverzeichnis abrufen können. In der Einarbeitungsphase sind hauptsächlich die Optionen `-a` und `-l` von Interesse. Weitere Optionen werden im *Referenzhandbuch für Benutzer* unter dem Eintrag `ls`(1) beschrieben.

Anzeige des vollständigen Verzeichnisinhalts

Einige wichtige Dateinamen in Ihrem Home- Verzeichnis beginnen mit einem Punkt (z.B. `.profile`). Diese Dateien werden beim Kommando `ls` standardmäßig nicht ausgegeben; damit sie ebenfalls angezeigt werden, müssen Sie das Kommando `ls` mit der Option `-a` eingeben.

Wenn Sie beispielsweise in Ihrem aktuellen Verzeichnis (`starship`) alle Dateien, einschließlich der Dateien, deren Namen mit einem Punkt (`.`) beginnen, abrufen möchten, erhalten Sie mit der Kommandozeile `ls -a` und der Betätigung der Taste `RETURN` folgende Anzeige:

```
$ ls -a<CR>
.
..
.profile
bin
draft
letters
list
mbox
$
```

Kurze Inhaltsübersicht

`-C` und `-F` gehören zu den wichtigsten Optionen des Kommandos `ls`. Bei der Eingabe dieser beiden Optionen in ein und derselben Kommandozeile werden die Unterverzeichnisse und Dateien eines Verzeichnisses in Spalten ausgegeben; ausführbare Dateien sind mit einem Stern (*) gekennzeichnet, Verzeichnisse mit einem Schrägstrich (/) und symbolische Verweise mit einem Klammeraffen (@). So können Sie den Inhalt Ihres Arbeitsverzeichnisses `starship` vollständig mit dem folgenden Kommando abrufen:

```
$ ls -CF<CR>
bin/            letters/        mbox
draft/          list*
$
```

Ausführliche Inhaltsübersicht

Besonders ausführliche Informationen erhalten Sie, wenn Sie das Kommando `ls` zusammen mit der Option `-l` verwenden. Mit dieser Option wird eine "lange" Liste mit Informationen wie z.B. über die Zugriffsrechte, die Anzahl der Verweise, den Eigentümer, den Gruppennamen, die Größe in Byte und den Zeitpunkt der letzten Änderung ausgegeben. Wenn Sie das Kommando `ls -l`

beispielsweise im Verzeichnis `starship` eingeben, erhalten Sie die folgende Ausgabe:

```
$ ls -l<CR>
total 30
drwxr-xr-x   3 starship     project          96  Oct 27  08:16  bin
drwxr-xr-x   2 starship     project          64  Nov  1  14:19  draft
drwxr-xr-x   2 starship     project          80  Nov  8  08:41  letters
-rwx------   2 starship     project       12301  Nov  2  10:15  list
-rw-------   1 starship     project          40  Oct 27  10:00  mbox
$
```

Die erste Zeile der Ausgabe (`total 30`) zeigt die Größe des ganzen Dateiverzeichnisses in Blöcken an. Jede weitere Zeile enthält Angaben zu einem Verzeichnis oder einer Datei im Verzeichnis `starship`. Das erste Zeichen jeder Zeile (`d`, `-`, `l`, `b` oder `c`) zeigt den Typ der Datei an.

`d` = (directory) Verzeichnis

`-` = Normale Datei

`l` = (link) Symbolischer Verweis

`b` = (block special file) Blockorientierte Gerätedatei

`c` = (character special file) Zeichenorientierte Gerätedatei

Wie aus diesen Informationen zu ersehen ist, enthält das Verzeichnis `starship` drei Verzeichnisse und zwei normale Dateien.

Mit den nächsten Zeichen (entweder Buchstaben oder Bindestriche) wird angegeben, wer für die Datei bzw. das Verzeichnis die Lese- und/oder Zugriffsberechtigung hat (weitere Informationen zum Thema Zugriffsberechtigungen finden Sie in der Beschreibung des Kommandos `chmod` im Abschnitt "Zugriff auf und Manipulation von Dateien" in diesem Kapitel).

Die nächste Zahl ist der Verweiszähler. Bei einer Datei gibt diese Zahl an, wieviele Verweise für diese Datei vorhanden sind, d.h. wieviele Benutzer mit dieser Datei verbunden sind. Bei einem Verzeichnis ergibt sich diese Zahl aus der Anzahl aller Verzeichnisse in diesem Verzeichnis plus zwei (für das Verzeichnis selbst und das Vaterverzeichnis).

Als nächstes wird der Benutzername des Eigentümers der Datei angegeben (in unserem Beispiel `starship`), gefolgt vom Gruppennamen der Datei oder des Verzeichnisses (`project`).

Die nächste Zahl gibt die Länge des Datei- bzw. Verzeichniseintrags in Byte an. Als nächstes wird der Monat, der Tag und der Zeitpunkt der letzten Änderung der Datei angegeben. Die letzte Spalte schließlich enthält den Namen des Verzeichnisses bzw. der Datei.

Bild 3-9 beschreibt die einzelnen Ausgabespalten des Kommandos `ls -l`.

Bild 3-9: Die Ausgabe des Kommandos `ls -l`

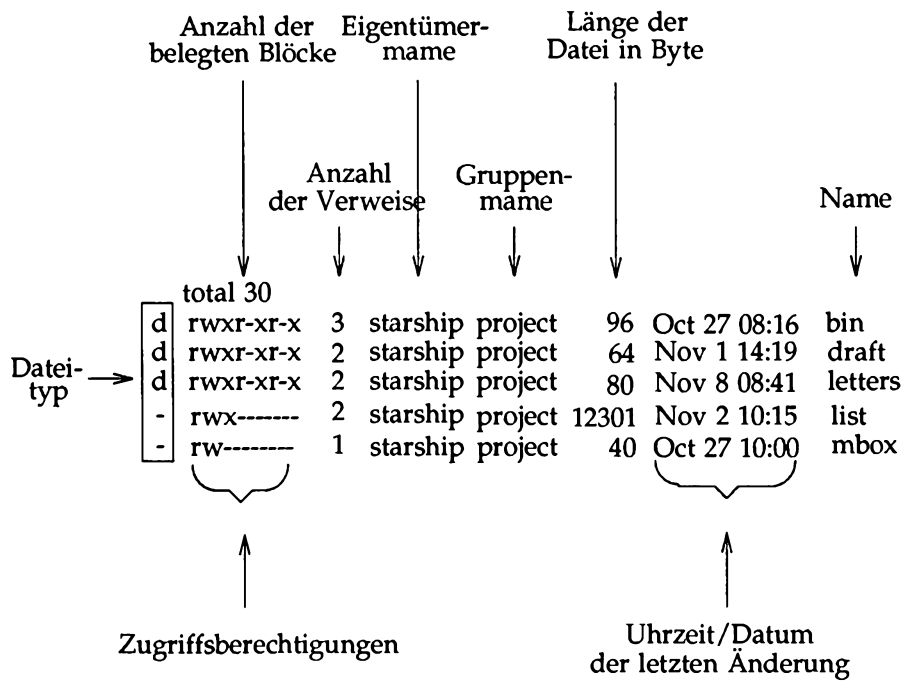


Bild 3-10 enthält einen Überblick über die Syntax und Funktion des Kommandos `ls` und zwei seiner Optionen.

Bild 3-10: `ls`-Kurzübersicht

Kurzbeschreibung `ls` – Inhalt eines Verzeichnisses ausgeben.		
Kommando	*Optionen*	*Argumente*
`ls`	`-a, -l,` u.v.a.*	*verzeichnisname(n)*
`Funktion:`	`ls` gibt die Namen der Dateien und Unterverzeichnisse in den angegebenen Verzeichnissen aus. Wenn kein Verzeichnis als Argument angegeben wird, so wird der Inhalt des Arbeitsverzeichnisses ausgegeben.	
`Optionen:`	`-a` Alle Einträge werden angezeigt, einschließlich der Namen, die mit . (Punkt) beginnen. `-l` Der Inhalt eines Verzeichnisses wird im ausführlichen Format angezeigt (u.a. mit Informationen über den Typ, die Zugriffsberechtigungen, die Größe in Byte und den Zeitpunkt der letzten Änderung).	
`Hinweis:`	Der Inhalt einer Datei kann mit dem Kommando `cat` auf dem Bildschirm abgerufen werden .	

* Einen Überblick über sämtliche Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `ls`(1).

Aktuelles Verzeichnis wechseln: Das Kommando `cd`

Unmittelbar nach Ihrer Anmeldung bei UNIX befinden Sie sich in Ihrem Home-Verzeichnis. Es bleibt Ihr aktuelles Arbeitsverzeichnis, solange Sie darin arbeiten. Sie können jedoch auch in ein anderes Verzeichnis wechseln. Dafür rufen Sie das Kommando `cd` (für change directory) auf. Dieses Kommando wird zusammen mit dem Pfadnamen des Verzeichnisses, in das Sie wechseln wollen, eingegeben.

`cd` *pfadname_verzeichnis_neu*<CR>

Als Argument für das Kommando `cd` kann jeder gültige Pfadname (absolut oder relativ) verwendet werden. Wird kein Pfadname angegeben, versetzt das Kommando Sie in Ihr Home-Verzeichnis. Nach einem Wechsel in ein neues Verzeichnis wird dieses Verzeichnis zu Ihrem aktuellen Verzeichnis.

Wenn Sie z.B. in unserem Beispieldateisystem vom Verzeichnis `starship` in dessen Sohnverzeichnis `draft` wechseln wollen, müssen Sie die Kommandozeile `cd draft` eingeben und anschließend die Taste `RETURN` betätigen (in diesem Fall ist `draft` der relative Pfadname zum gewünschten Verzeichnis). Sobald das System Ihnen die Beendigung der Kommandoausführung durch die Anzeige einer neuen Eingabeaufforderung signalisiert, können Sie Ihre neue Position im Dateisystem mit dem Kommando `pwd` überprüfen. Nachdem Sie dieses Kommando eingegeben und die Taste `RETURN` betätigt haben, sieht die Bildschirmanzeige folgendermaßen aus:

```
$ cd draft<CR>
$ pwd<CR>
/home/starship/draft
$
```

Auch im neuen Verzeichnis `draft` können Sie nun mit dem Kommando `mkdir` Unterverzeichnisse und mit den Editoren `ed` und `vi` neue Dateien anlegen (die Kapitel 6 und 7 enthalten eine ausführliche Einführung in die Kommandos `ed` und `vi`).

Auf die Dateien im Verzeichnis `draft` können Sie auch dann zugreifen, wenn es sich nicht um Ihr aktuelles Verzeichnis handelt. Über einen relativen oder absoluten Pfadnamen können Sie auf Dateien in beliebigen Verzeichnissen zugreifen. Angenommen, Sie befinden sich im Verzeichnis `draft` und möchten den Inhalt der Datei `sanders` im Verzeichnis `letters` (`/home/starship/draft`) mit dem Kommando `cat` abrufen. Dazu geben Sie in der Kommandozeile den absoluten Pfadnamen der Datei `sanders` an:

```
cat /home/starship/letters/sanders<CR>
```

Beim Kommando `cd` können auch absolute Pfadnamen angegeben werden. Um beispielsweise vom Verzeichnis `draft` in das Verzeichnis `letters` zu wechseln, geben Sie folgende Kommandozeile ein:

```
cd /home/starship/letters<CR>
```

Da `letters` und `draft` Sohnverzeichnisse von `starship` sind, können Sie dazu ebenso den relativen Pfadnamen `../letters` benutzen. Über `..` gelangen Sie zum Verzeichnis `starship`; der Rest des Pfadnamens führt Sie zu `letters`.

Bild 3-11 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `cd`.

Bild 3-11: `cd`-Kurzübersicht

Kurzbeschreibung `cd` – Arbeitsverzeichnis wechseln.		
Kommando	*Optionen*	*Argumente*
`cd`	keine	*verzeichnisname*
`Funktion:`	Mit `cd` wechselt Ihre Position im Dateisystem vom aktuellen Verzeichnis zum angegebenen Verzeichnis. Das Kommando `cd` ohne Angabe eines Verzeichnisnamens versetzt Sie in Ihr Home-Verzeichnis.	
`Hinweise:`	Nachdem die Shell Sie in das angegebene Verzeichnis versetzt hat, zeigt sie die Eingabeaufforderung (Standard: `$`) an. Um auf ein Verzeichnis zuzugreifen, das nicht in Ihrem Arbeitsverzeichnis enthalten ist, müssen Sie anstelle des Verzeichnisnamens den relativen oder absoluten Pfadnamen angeben.	

Löschen von Verzeichnissen: Das Kommando `rmdir`

Ein Verzeichnis, das Sie nicht mehr benötigen, können Sie mit dem Kommando `rmdir` (für remove a directory) löschen. Dieses Kommando hat die folgende Standardsyntax:

`rmdir` *verzeichnisname(n)*`<CR>`

In der Kommandozeile können auch mehrere Verzeichnisnamen angegeben werden.

Jedes Kommando wird ausführlich beschrieben und in einer Tabelle zusammengefaßt, so daß Sie Syntax und Funktionen des Kommandos schnell und einfach nachschlagen können.

Inhalt einer Datei ausgeben: Die Kommandos `cat`, `pg` und `pr`

Unter UNIX gibt es drei Kommandos zur Anzeige des Inhalts von ein oder mehreren Dateien: `cat`, `pg`, und `pr`. Das Kommando `cat` (für concatenate) erzeugt eine Ausgabe des Inhalts der angegebenen Datei(en). Diese Ausgabe wird auf Ihrem Bildschirm angezeigt, es sei denn, Sie leiten die Ausgabe des Kommandos `cat` in eine andere Datei oder zu einem neuen Kommando um.

Das Kommando `pg` ist vor allem dann hilfreich, wenn Sie den Inhalt einer umfangreicheren Datei lesen wollen: Sie können den Inhalt einer Datei damit bildschirm-seitenweise abrufen.
Das Kommando `pr` formatiert die angegebenen Dateien und zeigt sie entweder auf Ihrem Terminal an oder leitet sie bei der entsprechenden Anforderung zu einem Drucker um (siehe auch die Beschreibung zum Kommando `lp` im Kapitel 8, "Der Druck-Service LP").

Die folgenden Abschnitte beschreiben die Kommandos `cat`, `pg` und `pr`.

Inhalt von Dateien verknüpfen und anzeigen: Das Kommando `cat`

Das Kommando `cat` zeigt den Inhalt von ein oder mehreren Dateien auf dem Bildschirm an. Angenommen, Sie befinden sich in unserem Beispieldateisystem im Verzeichnis `letters` und wollen den Inhalt der Datei `johnson` abrufen. Wenn Sie die auf dem folgenden Bildschirm gezeigte Kommandozeile eingeben, erhalten Sie die nachfolgende Ausgabe:

Zugriff auf und Manipulation von Dateien

Dieser Abschnitt macht Sie mit verschiedenen UNIX-Kommandos bekannt, mit denen Sie auf Dateien im Dateisystem zugreifen und sie manipulieren können. Die Informationen in diesem Abschnitt sind in zwei Teile aufgeteilt: In grundlegende Informationen über den Umgang mit dem Dateisystem und Kommandos für fortgeschrittene Benutzer, mit denen auch komplexere Datenmanipulationen durchgeführt werden können.

Grundlegende Kommandos

In diesem Abschnitt werden die UNIX-Kommandos behandelt, die für den Zugriff auf Dateien und das Arbeiten mit Dateien in der Verzeichnisstruktur notwendig sind. Bild 3-13 enthält eine Übersicht über diese Kommandos.

Bild 3-13: Grundlegende Kommandos für das Arbeiten mit Dateien

Kommando	Funktion
`cat`	Der Inhalt einer Datei wird auf dem Bildschirm angezeigt.
`pg`	Der Inhalt einer Datei wird abschnitts- oder seitenweise auf dem Bildschirm angezeigt.
`pr`	Von einer Datei wird eine teilweise formatierte Version auf dem Terminal ausgegeben.
`cp`	Von einer vorhandenen Datei wird eine Kopie angefertigt.
`mv`	Eine Datei wird versetzt und umbenannt.
`rm`	Eine Datei wird gelöscht.
`wc`	Die Anzahl der Zeilen, Wörter und Zeichen einer Datei wird ausgegeben.
`chmod`	Die Zugriffsrechte für eine Datei (oder ein Verzeichnis) werden geändert.

Bild 3-12 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `rmdir`.

Bild 3-12: `rmdir`-Kurzübersicht

Kurzbeschreibung `rmdir` – Verzeichnis löschen.		
Kommando	*Optionen*	*Argumente*
`rmdir`	vorhanden	*verzeichnisname(n)*
`Funktion:`	`rmdir` löscht die angegebenen Verzeichnisse, vorausgesetzt, sie enthalten keine Dateien und/oder Unterverzeichnisse.	
`Hinweise:`	Wenn das Verzeichnis leer ist, wird sie gelöscht, und das System gibt die Eingabeaufforderung aus. Wenn das Verzeichnis Dateien und Unterverzeichnisse enthält, gibt das Kommando die Meldung `rmdir:` *verzeichnisname* `not empty` aus.	

Mit dem Kommando `rmdir` können Sie ein Verzeichnis nur dann löschen, wenn Sie der Eigentümer sind und das Verzeichnis leer ist. Damit Sie eine Datei aus dem Verzeichnis eines anderen Benutzers löschen können, muß der Eigentümer Ihnen die Schreibberechtigung für das Vaterverzeichnis der zu löschenden Datei erteilen.

Wenn Sie ein Verzeichnis löschen wollen, das noch Unterverzeichnisse und Dateien enthält (also noch nicht leer ist), gibt das Kommando `rmdir` die Meldung *verzeichnisname* `not empty` (Verzeichnis nicht leer) aus. Das Kommando kann erst ausgeführt werden, nachdem alle Unterverzeichnisse und Dateien gelöscht worden sind.

Angenommen, das Verzeichnis `memos` enthält das Unterverzeichnis `tech` sowie die beiden Dateien `june.30` und `july.31` (legen Sie diesen beide Dateien in Ihrem Home-Verzeichnis an, damit Sie mit dem Kommando `rmdir` experimentieren können). Wenn Sie jetzt versuchen, das Verzeichnis `memos` zu löschen (indem Sie in Ihrem Home-Verzeichnis das Kommando `rmdir` aufrufen), erhalten Sie die folgende Bildschirmanzeige:

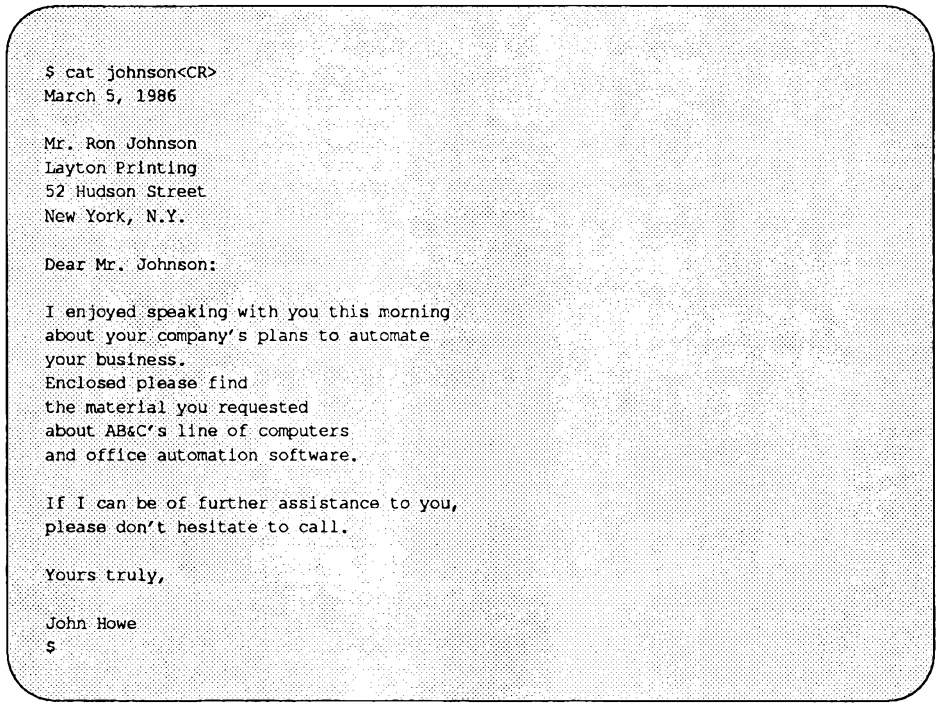
```
$ rmdir memos<CR>
rmdir:  memos not empty
$
```

Um das Verzeichnis `memos` löschen zu können, müssen Sie zuerst seinen Inhalt löschen, also das Unterverzeichnis `tech` und die beiden Dateien `june.30` und `july.31`. Zum Löschen des Unterverzeichnisses `tech` rufen Sie das Kommando `rmdir` auf. Das Löschen von Dateien wird im Abschnitt "Zugriff auf und Manipulation von Dateien" dieses Kapitels beschrieben.

Sobald das Verzeichnis `memos` leer ist, können Sie es auch selbst löschen. Zunächst müssen Sie jedoch in das Vaterverzeichnis dieses Verzeichnisses (Ihr Home-Verzeichnis) wechseln. Beim Aufrufen des Kommandos `rmdir` dürfen Sie sich nicht in dem Verzeichnis befinden, das gelöscht werden soll. Geben Sie in Ihrem Home-Verzeichnis folgendes ein:

```
rmdir memos<CR>
```

Wenn das Verzeichnis `memos` leer ist, wird es nun gelöscht. Die erfolgreiche Kommandoausführung können Sie auch hier an der Ausgabe der Eingabeaufforderung erkennen.



```
$ cat johnson<CR>
March 5, 1986

Mr. Ron Johnson
Layton Printing
52 Hudson Street
New York, N.Y.

Dear Mr. Johnson:

I enjoyed speaking with you this morning
about your company's plans to automate
your business.
Enclosed please find
the material you requested
about AB&C's line of computers
and office automation software.

If I can be of further assistance to you,
please don't hesitate to call.

Yours truly,

John Howe
$
```

Um den Inhalt von zwei (oder mehr) Dateien abzurufen, geben Sie einfach die Namen dieser Dateien in der Kommandozeile ein. Den Inhalt der Dateien `johnson` und `sanders` können Sie also mit dem folgenden Kommando abrufen:

```
$ cat johnson sanders<CR>
```

Das Kommando `cat` liest hintereinander die Dateien `johnson` und `sanders` und zeigt ihren Inhalt in derselben Reihenfolge auf Ihrem Terminal an:

```
$ cat johnson sanders<CR>
March 5, 1986

Mr. Ron Johnson
Layton Printing
52 Hudson Street
New York, N.Y.

Dear Mr. Johnson:

I enjoyed speaking with you this morning
.
.

Yours truly,

John Howe


March 5, 1986

Mrs. D.L. Sanders
Sanders Research, Inc.
43 Nassau Street
Princeton, N.J.

Dear Mrs. Sanders:

My colleagues and I have been following, with
great interest,
.
.

Sincerely,

John Howe
$
```

Informationen darüber, wie Sie die Ausgabe des Kommandos `cat` in eine andere Datei oder zu einem neuen Kommando umleiten können, finden Sie in Kapitel 9.

Bild 3-14 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `cat`.

Bild 3-14: `cat`**-Kurzübersicht**

Kurzbeschreibung `cat` – Dateiinhalte verknüpfen und anzeigen		
Kommando	*Optionen*	*Argumente*
`cat`	vorhanden*	*dateiname(n)*
`Funktion:`	Das Kommando `cat` liest die Namen aller in der Kommandozeile angegebenen Dateien und zeigt deren Inhalt an.	
`Hinweise:`	Wenn die angegebene Datei vorhanden und lesbar ist, wird ihr Inhalt auf dem Bildschirm angezeigt. Ist dies nicht der Fall, erscheint auf dem Bildschirm die Meldung: `cat: cannot open` *dateiname* (Datei kann nicht eröffnet werden). Den Inhalt eines Verzeichnisses können Sie mit dem Kommando `ls` abrufen.	

* Eine Übersicht über sämtliche Optionen und ihre Funktionen finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `cat`(1).

Inhalt einer Datei durchblättern: Das Kommando `pg`

Das Kommando `pg` (für page) ermöglicht das seitenweise Abrufen von ein oder mehreren Dateien auf dem Terminal. Dabei wird jeweils eine Bildschirmseite (ein großer Abschnitt) des Textes angezeigt, gefolgt von einer Eingabeaufforderung in Form eines Doppelpunktes (:). Mit diesem Zeichen signalisiert das Programm, daß es auf weitere Anweisungen wartet. Für die Eingabe gibt es jetzt verschiedene Möglichkeiten. Dazu gehören die Aufforderung an das Kommando, weitere Seiten der Datei anzuzeigen oder die Datei nach einer bestimmten Zeichenkette zu durchsuchen. Bild 3-15 enthält eine Kurzübersicht über einen Teil der verfügbaren Anweisungen.

Bild 3-15: Die Optionen bei `pg`

Kommando*	Funktion
`h`	Hilfe; Eine Liste der zur Verfügung stehenden `pg`†-Kommandos anzeigen.
`q` oder `Q`	`pg`-Lesemodus verlassen.
`<CR>`	Nächste Textseite anzeigen.
`l`	Nächste Textzeile anzeigen.
`d` oder `^d`	Eine weitere halbe Textseite anzeigen.
`.` oder `^l`	Aktuelle Textseite erneut anzeigen.
`f`	Nächste Textseite überspringen und darauffolgende Seite anzeigen.
`n`	Nächsten in der Kommandozeile enthaltene Datei anzeigen.
`p`	Vorhergehende Datei in der Kommandozeile anzeigen.
`$`	Letzte Textseite der aktuell angezeigten Datei ausgeben.
/muster	Datei in Vorwärtsrichtung nach dem angegebenen Muster durchsuchen.
`?`*muster*	Datei in Rückwärtsrichtung nach dem angegebenen Muster durchsuchen.

* Den meisten Kommandos kann eine Zahl vorangestellt werden. Dies gilt z.B. für <CR>, wobei +1 <CR> für "nächste Textseite anzeigen" steht, –1 <CR> für "vorhergehende Textseite anzeigen" und 1 <CR> für "erste Textseite anzeigen".

† Eine ausführliche Beschreibung sämtlicher `pg`-Kommandos finden Sie im *Referenzhandbuch für Benutzer.*

Das Kommando `pg` ist vor allem beim Abrufen einer umfangreichen Datei oder von mehreren Dateien von großem Nutzen: Da das Programm nach der Anzeige jeder Seite eine Pause einlegt, bleibt Ihnen genügend Zeit zum Lesen der Bildschirmanzeige. Die Länge der angezeigten Textseite hängt vom jeweiligen Terminal ab. Auf einem Terminal mit 24-Zeilen-Anzeige enthält eine Seite 23 Zeilen plus eine Zeile mit dem Doppelpunkt (:). Besteht die Datei aus weniger als 23 Textzeilen, so enthält eine Seite die Zeilen in der Datei plus die Zeile mit dem Doppelpunkt (:).

Um den Inhalt einer Datei mit dem Kommando `pg` abzurufen, geben Sie eine Kommandozeile in folgendem Format ein:

`pg` *dateiname(n)*`<CR>`

Wenn Sie z.B. den Inhalt der Datei `outline` im Verzeichnis `draft` unseres Beispieldateisystems abrufen wollen, müssen Sie folgendes eingeben:

```
pg outline<CR>
```

Daraufhin erscheint die erste Seite der Datei auf dem Bildschirm. Da die Datei mehr Zeilen enthält, als auf eine Bildschirmseite paßt, erscheint am unteren Rand des Bildschirms ein Doppelpunkt (:). Dies zeigt Ihnen, daß noch weitere Zeilen vorhanden sind und Sie nach dem Durchlesen der aktuellen Bildschirmseite eine weitere Textseite abrufen können, indem Sie die Taste `RETURN` betätigen.

Die bisherige Beschreibung des Kommandos `pg` wird auf dem folgenden Bildschirm zusammengefaßt:

```
$ pg outline<CR>
After you analyze the subject for your
report, you must consider organizing and
arranging the material you want to use in
writing it.
      .
      .
      .
An outline is an effective method of
organizing the material.  The outline
is a type of blueprint or skeleton,
a framework for you the builder-writer
of the report; in a sense it is a recipe
:<CR>
```

Nach der Betätigung der Taste RETURN setzt pg die Bildschirmausgabe fort und zeigt den weiteren Inhalt der Datei auf dem Bildschirm an:

```
that contains the names of the
ingredients and the order in which
to use them.
      .
      .
      .
Your outline need not be elaborate or
overly detailed; it is simply a guide you
may consult as you write, to be varied,
if need be, when additional important
ideas are suggested in the actual writing.
(EOF):
```

Beachten Sie die Zeile am unteren Rand des Bildschirms. Die darin enthaltene Zeichenkette (EOF) : (end of file) zeigt Ihnen, daß Sie das Ende der Datei erreicht haben. Der Doppelpunkt (:) ist eine Aufforderung zur Eingabe eines weiteren Kommandos.

Betätigen Sie nun die Taste RETURN, wenn Sie die Ausgabe der Datei beenden wollen. Auf dem Terminal erscheint jetzt eine Eingabeaufforderung (dasselbe gilt, wenn Sie das Kommando q oder Q eingeben). Sie können jedoch auch nach Bedarf ein beliebiges pg- Kommando sowie in einer pg-Kommandozeile

verschiedene Optionen eingeben. Informationen hierzu finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `pg`(1).

Damit das Kommando `pg` ordnungsgemäß ausgeführt werden kann, muß der Typ des verwendeten Terminals angegeben werden. Das Programm `pg` ist so konzipiert, daß es auf einer Vielzahl unterschiedlicher Terminals ablauffähig ist. Das Format der erzeugten Ausgabe hängt vom verwendeten Terminal ab. Durch die Angabe des Terminal-Typs informieren Sie das Kommando `pg` darüber:

- Wieviele Zeilen eine Bildschirmseite enthält.
- Wieviele Spalten eine Bildschirmseite enthält.
- Wie der Bildschirms gelöscht wird.
- Wie Eingabeaufforderungen und andere Zeichen hervorgehoben werden.
- Wie die aktuelle Zeile gelöscht wird.

Der Typ des Terminals wird in der Datei `.profile` angegeben; ordnen Sie hier der Variablen `TERM` den Code für Ihr Terminal zu (weitere Angaben zu `.TERM` und `.profile` finden Sie im Kapitel 9. Die Einstellung der Variablen `TERM` wird in Anhang F beschrieben).

Bild 3-16 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `pg`.

Bild 3-16: `pg`**-Kurzübersicht**

Kurzbeschreibung		
`pg` – Inhalt einer Datei in großen Abschnitten bzw. Seiten anzeigen.		
Kommando	*Optionen*	*Argumente*
`pg`	vorhanden*	*dateiname(n)*
`Funktion:`	Das Kommando `pg` zeigt den Inhalt von ein oder mehreren in der Kommandozeile angegebenen Dateien seitenweise an.	
`Hinweise:`	Nach der Anzeige einer Textseite erwartet das Kommando `pg` von Ihnen weitere Informationen, z.B., ob Sie die Textausgabe fortsetzen, im Text nach einer Zeichenkette suchen oder den `pg`-Lesemodus verlassen möchten. Zusätzlich stehen eine Reihe von Optionen zur Verfügung. So können Sie z.B. den Dateiinhalt ab einer bestimmten Zeile oder ab einer Zeile mit einem bestimmten Muster abrufen sowie eine Textstelle erneut abrufen, die bereits zuvor angezeigt worden ist.	

* Einen Überblick über sämtliche verfügbaren Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `pg`(1).

Dateien für die Ausgabe aufbereiten: Das Kommando `pr`

Mit dem Kommando `pr` kann man den Inhalt einer Datei formatieren und ausgeben. Es weist Möglichkeiten zur paginierten Ausgabe mit Titel und Kopfzeilen, außerdem zur Ausgabe des Dateiinhalts in verschiedenen Seitenlängen und - breiten auf dem Terminal-Bildschirm auf.

Die Ausgabe kann auch über ein anderes Gerät wie z.B. einen Zeilendrucker vorgenommen werden (siehe Beschreibung des Kommandos `lp` in Kapitel 8, "Der Druck-Service LP"). Außerdem können Sie die Ausgabe des Kommandos `pr` in eine andere Datei umleiten (siehe die Abschnitte über die Ein- und

Ausgabeumleitung im Kapitel 9).

Wird keine dieser Optionen gewählt, gibt das Kommando `pr` den Text einspaltig, mit 66 Zeilen pro Seite und einem kurzen Kopfteil aus. Der Kopfteil besteht aus fünf Zeilen (zwei Leerzeilen, einer Zeile mit dem Datum, der Uhrzeit, dem Dateinamen, der Seitennummer sowie zwei weiteren Leerzeilen). Am Ende der formatierten Datei befindet sich ein aus fünf Leerzeilen bestehender Fußteil (ein vollständiger Satz von Textformatierungswerkzeugen wird auf UNIX-Systemen zur Verfügung gestellt, die mit der Software DOCUMENTER'S WORKBENCH ausgerüstet sind; erkundigen Sie sich danach bei Ihrem Systemverwalter).

Das Kommando `pr` wird oft mit dem Kommando `lp` kombiniert, um den Inhalt einer Textdatei auf Papier auszudrucken. Sie können den Text jedoch auch mit dem Kommando `pr` formatiert auf Ihrem Terminal abrufen. Mit dem folgenden Kommando können Sie, um bei unserem Beispiel zu bleiben, den Inhalt der Datei `johnson` abrufen:

```
$ pr johnson<CR>
```

Die Ausgabe dieses Kommandos kann beispielsweise folgendermaßen aussehen:

```
$ pr johnson<CR>


Mar  5 15:43 1986 johnson Page 1

March 5, 1986

Mr. Ron Johnson
Layton Printing
52 Hudson Street
New York, N.Y.


Dear Mr. Johnson:

I enjoyed speaking with you this morning
about your company's plans to automate
your business.
Enclosed please find
the material you requested
about AB&C's line of computers
and office automation software.

If I can be of further assistance to you,
please don't hesitate to call.
Yours truly,

John Howe



$
```

Bei den Leerzeilen nach der letzten Zeile der Datei handelt es sich um diejenigen Zeilen (in diesem Fall nur Leerzeilen), um die das Kommando `pr` die Ausgabe erweitert, damit jede Seite 66 Zeilen enthält. Auf einem Datensichtgerät mit 24- Zeilen-Anzeige werden die 66 Zeilen der formatierten Datei in einem Durchlauf, ohne Pause, angezeigt. D. h., die ersten 42 Zeilen der Datei jagen so schnell über den oberen Bildschirmrand hinaus, daß sie von Ihnen nicht mehr gelesen werden können - es sei denn, Sie können um ein- oder zwei Bildschirmseiten zurückblättern. Bei einer sehr langen Datei wird jedoch selbst dies nicht genügen, um den ganzen Inhalt zu lesen.

In einem solchen Fall können Sie die Bildschirmausgabe mit dem Steuerzeichen <^s> (Control-s) anhalten; über <^q> (Control-q) können Sie die Bildschirmausgabe dann wieder fortsetzen.

Bild 3-17 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `pr`.

Bild 3-17: `pr`-Kurzübersicht

Kurzbeschreibung `pr` – Inhalt einer Datei formatiert ausgeben.		
Kommando	*Optionen*	*Argumente*
`pr`	vorhanden*	*dateiname(n)*
`Funktion:`	Das Kommando `pr` zeigt Ihre Datei(en) formatiert auf dem Terminal-Bildschirm an, wenn nicht zusätzliche Optionen angegeben werden. Der Text der Datei(en) wird dabei auf Seiten zu je 66 Zeilen ausgegeben, mit einem fünfzeiligen Kopfteil am Anfang und 5 Leerzeilen am Ende jeder Seite. Der Kopfteil besteht aus zwei Leerzeilen, einer Zeile mit Datum, Uhrzeit, Dateiname und Seitenummer sowie zwei weiteren Leerzeilen.	
`Hinweise:`	Ist die angegebene Datei vorhanden, wird ihr Inhalt formatiert und angezeigt; andernfalls wird die Meldung `pr: can't open` *dateiname* angezeigt. Das Kommando `pr` wird oft mit dem Kommando `lp` kombiniert, um ein oder mehrere Dateien auf einem Drucker auszudrucken. Es kann auch zur Anzeige des Dateiinhalts auf einem Datensichtgerät benutzt werden. Um die Bildschirmausgabe einer Datei anzuhalten bzw. fortzusetzen, geben Sie <^s> bzw. <^q> ein.	

* Eine Übersicht über sämtliche verfügbaren Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `pr`(1).

Kopie einer Datei erstellen: Das Kommando `cp`

In einer UNIX-Sitzung wird es häufig vorkommen, daß Sie eine Datei kopieren möchten, z.B. um eine Datei zu überarbeiten und gleichzeitig die ursprüngliche Version der Datei zu erhalten. Das Kommando `cp` (für copy) kopiert den vollständigen Inhalt einer Datei in eine andere. Außerdem können Sie damit ein oder mehrere Dateien aus einem Verzeichnis in ein anderes kopieren, wobei die ursprünglichen Dateien in ihrem Verzeichnis erhalten bleiben.

Um beispielsweise innerhalb des Verzeichnisses `draft` die Datei `outline` in die Datei `new.outline` zu kopieren, müssen Sie die Kommandozeile `cp outline new.outline` eingeben und die Taste `RETURN` betätigen. Das System zeigt nach dem Ende des Kopiervorgangs die Eingabeaufforderung an. Um zu überprüfen, ob die neue Datei tatsächlich angelegt worden ist, geben Sie das Kommando `ls` ein und betätigen die Taste `RETURN`. Dieses Kommando zeigt die Namen aller Dateien und Verzeichnisse in Ihrem aktuellen Verzeichnis an. In unserem Beispiel werden die Namen und Verzeichnisse im Verzeichnis `draft` angezeigt. Die einzelnen Schritte sind auf dem folgenden Bildschirm aufgeführt:

```
$ cp outline new.outline<CR>
$ ls<CR>
new.outline
outline
table
$
```

Unter UNIX können in einem Verzeichnis in keinem Fall zwei Dateien mit gleichem Namen enthalten sein. In unserem Beispiel legte das System eine neue Datei mit dem Namen `new.outline` an, da keine Datei mit diesem Namen vorhanden war. Wäre eine Datei mit dem Namen `new.outline` bereits vorhanden gewesen, so wäre sie durch die Kopie der Datei `outline` ersetzt worden. Die ursprüngliche Version der Datei `new.outline` wäre dabei gelöscht worden.

Wenn Sie versuchen, in ein und demselben Verzeichnis die Datei `outline` in eine andere Datei mit dem Namen `outline` zu kopieren, teilt das System Ihnen mit, daß beide Dateinamen identisch sind und zeigt wieder seine Eingabeaufforderung an. Wenn Sie dann den Inhalt des Verzeichnisses abrufen, um einen Überblick über die Anzahl der Dateien mit dem Namen `outline` zu erhalten,

so sieht Ihr Bildschirm folgendermaßen aus:

```
$ cp outline outline<CR>
cp: outline and outline are identical
$ ls<CR>
outline
table
$
```

In unterschiedlichen Verzeichnissen können unter UNIX jedoch zwei Dateien gleichen Namens vorkommen. So könnten Sie z.B. die Datei `outline` im Verzeichnis `draft` in eine andere Datei namens `outline` kopieren, die im Verzeichnis `letters` enthalten ist. Zur Eingabe dieses Kommandos gibt es vier verschiedene Möglichkeiten. Bei den ersten beiden Möglichkeiten enthält die Kommandozeile den Namen der neuen Datei, die durch den Kopiervorgang angelegt werden soll:

- `cp outline /home/starship/letters/outline<CR>` (absoluter Pfadname)
- `cp outline ../letters/outline<CR>` (relativer Pfadname)

Es ist jedoch nicht in jedem Fall notwendig, den Namen der neuen Datei anzugeben. Wenn in der Kommandozeile kein Name für die neue Datei angegeben wird, ordnet das Kommando `cp` Ihrer neuen Datei standardmäßig den Namen der ursprünglichen Datei zu. Somit sind auch die zwei folgenden Kommandozeilen zulässig:

- `cp outline /home/starship/letters<CR>` (absoluter Pfadname)
- `cp outline ../letters<CR>` (relativer Pfadname)

Die vier Kommandozeilen haben dasselbe Ergebnis: das Kommando `cp` legt eine Kopie der Datei `outline` im Verzeichnis `letters` an und ordnet dieser Kopie ebenfalls den Namen `outline` zu.

Wenn Sie Ihrer neuen Datei jedoch einen neuen Namen zuordnen möchten, muß dieser selbstverständlich angegeben werden. Um beispielsweise die Datei `outline` im Verzeichnis `draft` in eine Datei mit dem Namen `outline.vers2` im Verzeichnis `letters` zu kopieren, muß eine der folgenden Kommandozeilen verwendet werden:

- `cp outline /home/starship/letters/outline.vers2<CR>` (absoluter Pfadname)
- `cp outline ../letters/outline.vers2<CR>` (relativer Pfadname)

Beachten Sie bei der Vergabe eines neuen Namens die Regeln, die im Abschnitt "Benennung von Verzeichnissen und Dateien" dieses Kapitels aufgeführt sind.

Bild 3-18 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `cp`.

Bild 3-18: `cp`-Kurzübersicht

Kurzbeschreibung cp – Kopie einer Datei anfertigen.		
Kommando	*Optionen*	*Argumente*
cp	vorhanden	*datei1 datei2* *datei(en) verzeichnis*
Funktion:	cp ermöglicht das Kopieren von *datei1* nach *datei2*, wobei *datei1* erhalten bleibt, sowie das Kopieren von ein oder mehreren Dateien in ein anderes Verzeichnis.	
Hinweise:	Wenn *datei1* nach *datei2* kopiert wird und *datei2* bereits vorhanden ist, überschreibt das Kommando cp die ursprüngliche Version von *datei2* durch eine Kopie von *datei1* und ordnet ihr den Namen *datei2* zu. Die ursprüngliche Version von *datei2* wird dabei gelöscht. Enthält die Kommandozeile von cp die Argumente *datei(en)* und *verzeichnis*, so kopiert cp die *datei(en)* in das *verzeichnis*. Verzeichnisse können nicht mit dem Kommando cp kopiert werden.	

Dateien versetzen und umbenennen: Das Kommando mv

Das Kommando mv (für move) ermöglicht das Umbenennen einer Datei im gleichen Verzeichnis oder das Versetzen einer Datei von einem Verzeichnis in ein anderes. Wird eine Datei in ein anderes Verzeichnis versetzt, kann sie umbenannt werden oder den gleichen Namen beibehalten.

Zum Umbenennen einer Datei innerhalb ein und desselben Verzeichnisses muß folgende Kommandozeile eingegeben werden:

mv *datei1 datei2*<CR>

Das Kommando mv ändert den Dateinamen von *datei1* in *datei2* und löscht *datei1*. *datei1* und *datei2* können beliebige zulässige Namen sein, auch Pfadnamen.

Wenn das Verzeichnis `draft` Ihr aktuelles Verzeichnis ist und Sie die Datei `table` in `new.table` umbenennen möchten, brauchen Sie nur das Kommando `mv table new.table` einzugeben und die Taste RETURN zu betätigen. Nach erfolgreicher Ausführung des Kommandos erhalten Sie erneut eine Eingabeaufforderung. Nun können Sie mit dem Kommando `ls` den Inhalt des Verzeichnisses abrufen und überprüfen, ob die Datei `new.table` vorhanden ist. Auf dem Bildschirm sieht Ihre Eingabe und die Ausgabe des Systems dann folgendermaßen aus:

```
$ mv table new.table<CR>
$ ls<CR>
new.table
outline
$
```

Mit dem Kommando mv kann auch eine Datei von einem Verzeichnis in ein anderes versetzt werden, wobei entweder der Name beibehalten oder ein neuer Name zugeordnet werden kann. Mit der folgenden Kommandozeile wird die Datei ohne Namensänderung versetzt:

mv *datei(en) verzeichnis*<CR>

Als Datei- und Verzeichnisnamen können beliebige zulässige Namen verwendet werden, auch Pfadnamen.

Angenommen, Sie möchten die Datei `table` aus dem aktuellen Verzeichnis `draft` (absoluter Pfadname: `/home/starship/draft`) in das Verzeichnis `letters` (relativer Pfadname von `draft` aus: `../letters`; absoluter Pfadname: `/home/starship/letters`) versetzen und den Namen beibehalten. Hierfür gibt es verschiedene Möglichkeiten, u.a. die folgenden Kommandozeilen:

```
mv table /home/starship/letters<CR>

mv table /home/starship/letters/table<CR>

mv table ../letters<CR>

mv table ../letters/table<CR>

mv /home/starship/draft/table /home/starship/letters/table<CR>
```

Soll die Datei `table` beim Versetzen in das Verzeichnis `letters` in `table2` umbenannt werden, kann dafür eine der folgenden Kommandozeilen benutzt werden:

```
mv table /home/starship/letters/table2<CR>

mv table ../letters/table2<CR>

mv /home/starship/draft/table2 /home/starship/letters/table2<CR>
```

Um zu überprüfen, ob das Kommando ordnungsgemäß ausgeführt worden ist, können Sie mit dem Kommando `ls` den Inhalt des Verzeichnisses abrufen.

Bild 3-19 enthält eine Kurzübersicht über die Syntax und Funktion des Kommandos `mv`.

Bild 3-19: `mv`**-Kurzübersicht**

Kurzbeschreibung `mv` – Dateien versetzen oder umbenennen.		
Kommando	*Optionen*	*Argumente*
`mv`	vorhanden	*datei1 datei2* *datei(en) verzeichnis*
`Funktion:`	`mv` ermöglicht das Umbenennen einer Datei oder das Versetzen einer oder mehrerer Dateien in ein anderes Verzeichnis.	
`Hinweise:`	Wenn Sie *datei1* in *datei2* umbenennen und bereits eine Datei mit dem Namen *datei2* vorhanden ist, überschreibt das Kommando `mv` die ursprüngliche Version von *datei2* mit *datei1* und ordnet ihr den Namen *datei2* zu; die ursprüngliche Version von *datei2* wird dabei gelöscht. Wenn beim Kommando die Argumente *datei(en)* und *verzeichnis* angegeben werden, versetzt `mv` die *datei(en)* nach *verzeichnis*.	

Datei löschen: Das Kommando `rm`

Wird eine Datei nicht mehr benötigt, kann sie mit dem Kommando `rm` (für remove) gelöscht werden. Dieses Kommando hat folgendes grundlegendes Format:

`rm` *datei(en)*<CR>

Es können mehrere Dateien gleichzeitig gelöscht werden. Dazu geben Sie in der Kommandozeile die Namen der zu löschenden Dateien, durch ein Leerzeichen voneinander getrennt, ein:

`rm` *datei1 datei2 datei3*`<CR>`

Das System erstellt von der gelöschten Datei keine Sicherungskopie. Nach der Ausführung des Kommandos ist die Datei unwiderruflich gelöscht.

Um zu überprüfen, ob das Kommando `rm` erfolgreich durchgeführt worden ist, können Sie mit dem Kommando `ls` die in Ihrem Verzeichnis verbliebenen Dateien abrufen.

Enthält Ihr Verzeichnis beispielsweise die beiden Dateien `outline` und `table`, so können Sie sie mit dem Kommando `rm` gleichzeitig löschen. Nach erfolgreicher Ausführung muß Ihr Verzeichnis leer sein; dies können Sie mit dem Kommando `ls` überprüfen:

```
$ rm outline table <CR>
$ ls
$
```

Die Eingabeaufforderung zeigt Ihnen, daß `outline` und `table` gelöscht worden sind.

Bild 3-20 enthält eine Kurzübersicht über die Syntax und Funktion des Kommandos `rm`.

Bild 3-20: `rm`-Kurzübersicht

Kurzbeschreibung `rm` – Datei löschen.		
Kommando	*Optionen*	*Argumente*
`rm`	vorhanden*	*datei(en)*
`Funktion:`	`rm` ermöglicht das Löschen von ein oder mehreren Dateien.	
`Hinweis:`	Die beim Kommando `rm` als Argumente angegebenen Dateien werden unwiederbringlich gelöscht.	

* Einen Überblick über sämtliche Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `rm`(1).

Anzahl der Zeilen, Wörter und Zeichen in einer Datei ausgeben: Das Kommando `wc`

Das Kommando `wc` (für word count) gibt für die Datei(en), die Sie in der Kommandozeile angeben, die Anzahl der Zeilen, Wörter und Zeichen aus. Bei der Angabe von zwei oder mehr Dateien ermittelt das Kommando `wc` die Anzahl der Zeilen, Wörter und Zeichen jeder einzelnen Datei und errechnet die Summen. Die Ausgabe des Kommandos `wc` können Sie auch selektiv auf die Anzahl der Zeilen, Wörter oder Zeichen beschränken; hierfür geben Sie in der Kommandozeile die Option `-l`, `-w` bzw. `-c` ein.

Damit die Anzahl der Zeilen, Wörter und Zeichen einer einzelnen Datei ausgegeben wird, muß die Kommandozeile folgendermaßen aussehen:

```
wc datei1<CR>
```

Die vom System generierte Ausgabe hat folgendes Format:

l w c datei1

Dabei steht:

- *l* (lines) für die Anzahl der Zeilen in *datei1*
- *w* (words) für die Anzahl der Wörter in *datei1*
- *c* (characters) für die Anzahl der Zeichen in *datei1*

Um beispielsweise die Anzahl der Zeilen, Wörter und Zeichen in der Datei `johnson` im aktuellen Verzeichnis `letters` zu ermitteln, muß folgende Kommandozeile eingegeben werden:

```
$ wc johnson<CR>
24      66      406 johnson
$
```

Diese Informationen zeigen Ihnen, daß die Datei `johnson` 24 Zeilen, 66 Wörter und 406 Zeichen enthält.

Um die Anzahl der Zeilen, Wörter und Zeichen von mehreren Dateien zu zählen, muß folgende Kommandozeile eingegeben werden:

`wc` *datei1 datei2*`<CR>`

Die vom System generierte Ausgabe hat folgendes Format:

```
l       w       c       datei1
l       w       c       datei2
l       w       c       total
```

Die Anzahl der Zeilen, Wörter und Zeichen werden für *datei1* und *datei2* in separaten Zeilen angezeigt. Die Summen aus den einzelnen Spalten stehen in der letzten Zeile neben dem Wort `total`.

Wird z.B. mit dem Kommando `wc` die Anzahl der Zeilen, Wörter und Zeichen der Dateien `johnson` und `sanders` im aktuellen Verzeichnis angefordert, sieht die Ausgabe folgendermaßen aus:

```
$ wc johnson sanders<CR>
    24      66     406 johnson
    28      92     559 sanders
    52     158     965 total
$
```

Die erste Zeile gibt an, daß die Datei `johnson` 24 Zeilen, 66 Wörter und 406 Zeichen enthält. Die zweite Zeile gibt an, daß die Datei `sanders` 28 Zeilen, 92 Wörter und 559 Zeichen enthält. Die letzte Zeile gibt an, daß die beiden Dateien zusammen 52 Zeilen, 158 Wörter und 965 Zeichen enthalten.

Um selektiv die Anzahl der Zeilen, der Wörter oder Zeichen abzurufen, geben Sie eine Kommandozeile in einem der folgenden Formate ein:

`wc`	`-l`	*datei1*`<CR>`	(Anzahl der Zeilen)
`wc`	`-w`	*datei1*`<CR>`	(Anzahl der Wörter)
`wc`	`-c`	*datei1*`<CR>`	(Anzahl der Zeichen)

Die Option `-l` bewirkt beispielsweise, daß nur die Anzahl der Zeilen in der Datei `sanders` ausgegeben wird.

```
$ wc -l sanders<CR>
      28 sanders
$
```

Mit den Optionen `-w` und `-c` gibt das System nur die Anzahl der Wörter bzw. Zeichen aus.

Bild 3-21 enthält eine Kurzübersicht über die Syntax und Funktion des Kommandos `wc`.

Bild 3-21: `wc`**-Kurzübersicht**

Kurzbeschreibung wc – Zeilen, Wörter und Zeichen in einer Datei zählen.		
Kommando	*Optionen*	*Argumente*
`wc`	`-l, -w, -c`	*datei(en)*
`Funktion:`	`wc` zählt die Zeilen, Wörter und Zeichen in der bzw. den angegebenen Datei(en) und summiert die einzelnen Zahlen auf, falls mehr als eine Datei angegeben wird.	
`Optionen`	`-l` Anzahl der Zeilen in der/den angegebenen Datei(en) `-w` Anzahl der Wörter in der/den angegebenen Datei(en). `-c` Anzahl der Zeichen in der/den angegebenen Datei(en).	
`Hinweise:`	Der in der Kommandozeile angegebene Dateiname wird zusammen mit der/den angeforderten Zahl/en ausgegeben.	

Zugriffsschutz für Dateien: Das Kommando `chmod`

Mit dem Kommando `chmod` (für change mode) können Sie angeben, wer Ihre Dateien lesen, Daten in sie schreiben und sie nutzen darf. Da UNIX ein Mehrbenutzersystem ist, sind Sie im Normalfall nicht der einzige Benutzer des Dateisystems. Die verschiedenen Systembenutzer können über Pfadnamen in zahlreiche Verzeichnisse gelangen und die dort enthaltenen Dateien anderer Benutzer lesen und benutzen - vorausgesetzt, sie besitzen die entsprechende Zugriffsberechtigung.

Sind Sie der Eigentümer einer Datei, so können Sie festlegen, wer sie lesen, Daten in sie schreiben (d.h. sie ändern) darf, oder - falls es sich um ein Programm handelt - sie ausführen darf. Mit dem Kommando `chmod` kann auch die Zugriffsberechtigung für Verzeichnisse festgelegt werden. Wenn Sie für ein Verzeichnis die Ausführungsberechtigung erteilen, ermöglichen Sie es den angegebenen Benutzern, mit `cd` in das Verzeichnis zu wechseln und seinen Inhalt mit dem Kommando `ls` abzurufen.

Die verschiedenen Zugriffsberechtigungen werden mit den folgenden drei Buchstaben angegeben:

- `r` (read) Die Systembenutzer können eine Datei lesen und ihren Inhalt kopieren.
- `w` (write) Die Systembenutzer können eine Datei (oder ihre Kopie) ändern.
- `x` (execute) Die Systembenutzer können eine ausführbare Datei ablaufen lassen.

Zur Angabe der Benutzer, denen diese drei Arten von Zugriffsberechtigungen erteilt bzw. entzogen werden, benutzen Sie die folgenden drei Buchstaben:

- `u` (user) Sie selbst als Eigentümer Ihrer Dateien und Verzeichnisse
- `g` (group) Die Benutzer Ihrer Gruppe (z.B. am selben Projekt beteiligte, alle Mitarbeiter einer Abteilung, oder eine Gruppe, der Sie beim Anlegen Ihres Benutzereintrags absichtlich zugeordnet worden sind).
- `o` (other) Alle anderen Systembenutzer.

Immer wenn Sie eine Datei oder ein Verzeichnis anlegen, werden Ihnen, den Mitgliedern Ihrer Gruppe oder den anderen Systembenutzern automatisch Zugriffsberechtigungen zugeteilt oder entzogen. Diesen Mechanismus können Sie durch eine Änderung Ihrer Umgebung (siehe Kapitel 9) modifizieren. Zusätzlich haben Sie, unabhängig von den Zugriffsberechtigungen, die beim Anlegen einer Datei gewährt werden, als Eigentümer der Datei oder des Verzeichnisses jederzeit die Möglichkeit, die Zugriffsberechtigungen nach Ihren Wünschen zu ändern. So können Sie Dateien beispielsweise ausschließlich zur eigenen Nutzung vorbehalten. Oder Sie möchten den anderen Benutzern Ihrer Gruppe und allen anderen Systembenutzern die Möglichkeit geben, Ihre Dateien zu lesen und Daten in sie zu schreiben. Sie können aber auch den Mitgliedern

Ihrer Gruppe die Ausführungsberechtigung für ein Programm geben, so daß sie es gemeinsam mit Ihnen nutzen können.

Abrufen der aktuellen Zugriffsberechtigungen

Mit dem Kommando `ls -l`, das den Inhalt eines Verzeichnisses im ausführlichen Format anzeigt, können Sie sich über die Zugriffsberechtigungen informieren, die für eine Datei oder ein Verzeichnis aktuell wirksam sind. Wenn Sie in unserem Beispieldateisystem im Verzeichnis `starship/bin` das Kommando `ls -l` eingeben und die Taste RETURN betätigen, wird folgende Ausgabe erzeugt:

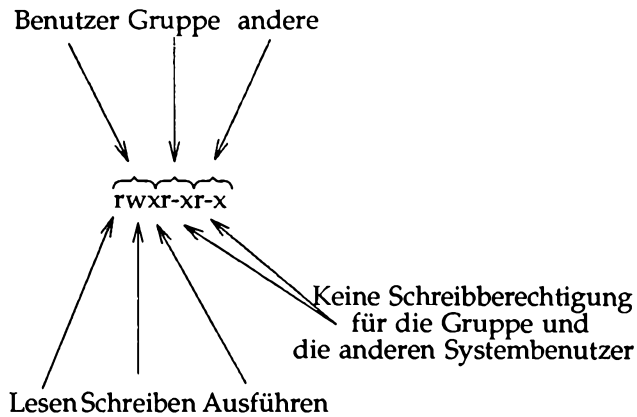
```
$ ls -l<CR>
total 35
-rwxr-xr-x   1 starship     project        9346 Nov 1  08:06  display
-rw-r--r--   1 starship     project        6428 Dec 2  10:24  list
drwx--x--x   2 starship     project          32 Nov 8  15:32  tools
$
```

Die Zugriffsberechtigungen für die Dateien `display` und `list` und das Verzeichnis `tools` werden auf der linken Bildschirmseite unter der Zeile `total 35` angezeigt und haben folgendes Format:

```
-rwxr-xr-x    (für die Datei display)
-rw-r--r--    (für die Datei list)
drwx--x--x    (für das Verzeichnis tools)
```

Das erste Zeichen kennzeichnet den Dateityp (z.B. ein Bindestrich (-) für eine normale Datei und `d` für ein Verzeichnis). Die nachfolgenden neun Zeichen geben die Zugriffsberechtigungen in drei Gruppen zu jeweils drei Zeichen an. Die erste Gruppe bezieht sich auf die Zugriffsberechtigungen für den Eigentümer, die zweite auf die für die Mitglieder einer Gruppe und die dritte auf die für alle anderen Systembenutzer. In jeder Gruppe zeigen die Zeichen `r`, `w` und `x` an, welche Zugriffsberechtigungen die jeweilige Benutzer-Kategorie aktuell hat. Wird anstelle von `r`, `w` oder `x` ein Bindestrich (-) angegeben, so ist die jeweilige Lese-, Schreib- oder Ausführungsberechtigung entzogen.

Das folgende Diagramm zeigt die Zugriffsberechtigungen für die Datei `display`:



Wie Sie sehen können, hat der Eigentümer die Zugriffsberechtigungen `r`, `w` und `x`, während die Mitglieder der Gruppe sowie die anderen Systembenutzer die Zugriffsberechtigungen `r` und `x` haben.

Für dieses Darstellungsschema gibt es zwei Ausnahmen. Es kommt gelegentlich vor, daß anstelle von `r`, `w` oder `x` die Buchstaben `s` oder `l` für die Zugriffsberechtigungen angegeben sind. Der Buchstabe `s` (für set user ID oder set group ID) steht für eine spezielle Ausführungsberechtigung für die Datei. Er erscheint anstelle eines `x` (oder `-`) in den Benutzer- oder Gruppenangaben (in der ersten und zweiten Gruppe der Zugriffsberechtigungen). Vom Standpunkt des Benutzers aus entspricht der Kleinbuchstabe `s` einem `x` an gleicher Stelle; er besagt, daß eine Ausführungsberechtigung vorhanden ist. Die spezielle Bedeutung von `s` ist nur für Programmierer und Systemverwalter von Interesse (siehe auch Informationen über das Setzen der Benutzer- oder Gruppennummer im *System Administrator's Guide*). Der Buchstabe `l` gibt an, daß bei einem Zugriff auf die Datei eine Sperre eintritt. Dies bedeutet allerdings nicht, daß die Datei gesperrt worden ist.

Ändern von Zugriffsberechtigungen

Nachdem Sie sich über die vorhandenen Zugriffsberechtigungen informiert haben, können Sie sie mit einem `chmod`-Kommando in folgendem Format ändern:

`chmod` *wer+zugriffsberechtigung datei(en)*<CR>

oder

`chmod` *wer=zugriffsberechtigung datei(en)*<CR>

Die Komponenten dieser Kommandozeile haben folgende Bedeutung:

`chmod`	Name des Programms
wer	Eine von drei Benutzergruppen (`u`, `g`, oder `o`) `u` = (user) Benutzer `g` = (group) Gruppe `o` = (others) Andere
+ oder −	Anweisung zur Erteilung (+) oder Verweigerung (−) der Berechtigung
Zugriffserechtigung	Eine beliebige Kombination aus den drei Zugriffsberechtigungen (`r`, `w`, and `x`) `r` = (read) Lesen `w` = (write) Schreiben `x` = (execute) Ausführen
datei(en)	Der bzw. die angegebene(n) Datei- oder Verzeichnisname(n); wenn keine absoluten Pfadnamen angegeben werden, wird davon ausgegangen, daß sie sich in der Verzeichnishierarchie im oder unter dem aktuellen Verzeichnis befinden.

Hinweis: In der `chmod`-Kommandozeile dürfen keine Leerzeichen zwischen der Angabe *wer*, der Anweisung zum Erteilen (+) oder Aufheben (−) sowie der *Zugriffsberechtigung* selbst stehen.

Die folgenden Beispiele zeigen einige Anwendungsmöglichkeiten für das Kommando chmod. Als Eigentümer der Datei display können Sie diese ausführbare Datei lesen, Daten in sie schreiben und sie ausführen. Um das Risiko auszuschalten, daß Sie die Datei versehentlich ändern, können Sie sich selbst die Schreibberechtigung (w) entziehen. Dazu geben Sie die folgende Kommandozeile ein:

```
chmod u-w display<CR>
```

Nachdem das System die Ausführung des Kommandos mit einer Eingabeaufforderung bestätigt hat, geben Sie das Kommando ls -l ein und betätigen die Taste RETURN. Damit können Sie überprüfen, ob sich die Zugriffsberechtigung wie auf dem folgenden Bildschirm geändert hat:

```
$ chmod u-w display<CR>
$ ls -l<CR>
total 35
-r-xr-xr-x   1 starship     project        9346 Nov 1  08:06  display
rw-r--r--    1 starship     project        6428 Dec 2  10:24  list
drwx--x--x   2 starship     project          32 Nov 8  15:32  tools
$
```

Sie haben also für die Datei keine Schreibberechtigung mehr. Sie können solange keine Änderungen an dieser Datei vornehmen, bis Sie sich selbst wieder die Schreibberechtigung erteilen.

Ein weiteres Beispiel: Wie obiger Bildschirm zeigt, haben die Mitglieder Ihrer Gruppe und die anderen Systembenutzer keine Schreibberechtigung für die Datei display. Da sie jedoch die Leseberechtigung haben, können sie die Datei in ihr eigenes Verzeichnis kopieren und dann Änderungen an ihr vornehmen. Um diese Möglichkeit sowohl für die Mitglieder ihrer Gruppe als auch die anderen Systembenutzer auszuschalten, müssen Sie ihnen die Leseberechtigung entziehen. Dafür geben Sie die folgende Kommandozeile ein:

```
chmod go-r display<CR>
```

g und o stehen für die Gruppe und alle anderen Systembenutzer; -r hebt die Leseberechtigung für diese Benutzer auf, so daß die Datei jetzt auch nicht mehr kopiert werden kann. Überprüfen Sie das Ergebnis mit dem Kommando ls -l:

```
$ chmod go-r display<CR>
$ ls -l<CR>
total 35
-rwx--x--x   1 starship     project        9346 Nov 1  08:06  display
rw-r--r--    1 starship     project        6428 Dec 2  10:24  list
drwx--x--x   2 starship     project          32 Nov 8  15:32  tools
$
```

Zugriffsberechtigungen und Verzeichnisse

Mit dem Kommando `chmod` können nicht nur für Dateien Zugriffsberechtigungen erteilt oder aufgehoben werden, sondern auch für Verzeichnisse; dafür muß in der Kommandozeile lediglich anstelle des Dateinamens der Verzeichnisname angegeben werden.

Allerdings muß auf die Auswirkungen aufmerksam gemacht werden, die eine Änderung der Zugriffsberechtigung für Verzeichnisse für die verschiedenen Systembenutzer mit sich bringt. Wenn Sie beispielsweise die Leseberechtigung für ein Verzeichnis sich selbst (`u`), den Mitgliedern Ihrer Gruppe (`g`) und den anderen Systembenutzern (`o`) erteilen, kann jeder Benutzer, der Zugriff auf das System hat, die Namen der Dateien in diesem Verzeichnis mit dem Kommando `ls -l` abrufen. Ebenso ermöglicht es die Schreibberechtigung für ein Verzeichnis den angegebenen Benutzern, in diesem Verzeichnis Dateien anzulegen und vorhandene Dateien zu löschen. Die Ausführungsberechtigung für ein Verzeichnis ermöglicht es den angegegebenen Benutzern, mit dem Kommando `cd` in dieses Verzeichnis zu wechseln und es somit zum aktuellen Verzeichnis zu machen.

Angabe der Zugriffsberechtigungen im Oktalformat

Das Kommando `chmod` kann in zwei verschiedenen Formaten eingegeben werden: Zum einen können die Zugriffsberechtigungen wie oben beschrieben über die Kleinbuchstaben `r`, `w` und `x` angegeben werden.

Bei der zweiten Möglichkeit geben Sie die Zugriffsberechtigungen über drei oktale Zahlen im Bereich von 0 bis 7 an (das oktale Zahlensystem unterscheidet sich vom üblicherweise verwendeten Dezimalsystem). Eine Einführung in dieses Format finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `chmod(1)`.

Bild 3-22 enthält eine Kurzübersicht über die Syntax und Funktion des Kommandos chmod.

Bild 3-22: chmod**-Kurzübersicht**

Kurzbeschreibung		
chmod – Zugriffsberechtigung für Dateien (und Verzeichnisse) ändern.		
Kommando	*Anweisung*	*Argumente*
chmod	wer + -	zugriffsberechtigung *verzeichnisname(n)*
Funktion:	chmod erteilt (+) die Lese-, Schreib- und Ausführungsberechtigung oder hebt sie auf (–). Die Systembenutzer werden dabei in drei Kategorien eingeteilt: user (Sie selbst), group (Mitglieder Ihrer Gruppe), und andere (alle anderen Benutzer, die auf das System, an dem Sie arbeiten, zugreifen können).	
Hinweise:	Die Art der Zugriffsberechtigung kann entweder mit einem Buchstaben oder einer Oktalzahl eingegeben werden.	

Kommandos für Fortgeschrittene

Sie sollten mit den bereits beschriebenen Kommandos anhand eigener Beispiele experimentieren, um sich mit dem Dateisystem vertraut zu machen.

Im folgenden werden drei weitere Kommandos zur Dateimanipulation und-verwaltung beschrieben: die Kommandos chown, id und groups. Wenn Sie der Eigentümer einer Datei sind, ist Ihr Benutzername als *eigentümer* eingetragen. Das Kommando chown ermöglicht es dem Eigentümer einer Datei, den Benutzernamen eines anderen Benutzers als Eigentümer einzusetzen. Ein

Beispiel: Bei der Eingabe der Kommandozeile

```
ls -l display <CR>
```

werden auf dem Bildschirm die folgenden Informationen ausgegeben:

```
-r-xr-xr-x  1 eigentümer  gruppe   9346 Nov 1  08:06  display
```

Um beispielsweise den Benutzer Sarah mit dem Benutzernamen sarah als Eigentümer Ihrer Datei einzusetzen, müssen Sie folgendes eingeben:

```
chown sarah display
```

Wenn Sie nun

```
ls -l display <CR>
```

eingeben, wird auf dem Bildschirm die folgende Meldung angezeigt:

```
-r-xr-xr-x  1 sara   gruppe   9346 Nov 1  08:06 display
```

Wenn nach der Eingabe des Kommandos `chown` eine Fehlermeldung auf dem Bildschirm angezeigt wird, so hat Ihr Systemverwalter diese Option beim Einrichten des Systems eingeschränkt. Nach der Eingabe von

```
id <CR>
```

zeigt das System Ihre Benutzernummer (uid) und effektive Gruppennummer (gid) an. Je nachdem, wie das System ursprünglich eingerichtet worden ist, können Sie auch zu zwei oder mehr Gruppen gehören. Über das folgende Kommando können Sie herausfinden, zu welchen Gruppen Sie gehören:

```
groups <CR>
```

Auf dem Bildschirm erscheint nun eine Auflistung aller Gruppen, deren Mitglied Sie sind. Sie haben auf alle Dateien Zugriff, deren Gruppennummer sich auf dieser zusätzlichen Gruppenliste befindet.

Mit wachsender Erfahrung im Umgang mit diesen Kommandos wird bei Ihnen das Bedürfnis nach komplexeren Datenmanipulationen zunehmen. Die drei Kommandos, die im folgenden beschrieben werden, ermöglichen Ihnen dafür den Einstieg.

`diff` Vergleich von zwei Dateien.

`grep`	Suchen nach einer Zeichenkette in einer Datei.
`sort`	Sortieren und Mischen von Dateien.

Ausführliche Informationen über diese Kommandos finden Sie im *Referenzhandbuch für Benutzer.*

Vergleich von Dateien: Das Kommando `diff`

Das Kommando `diff` vergleicht zwei Dateien und zeigt ihre Unterschiede auf dem Bildschirm an. Außerdem informiert es Sie darüber, wie die erste Datei geändert werden muß, um inhaltsgleich mit der zweiten zu werden. Das Kommando hat folgendes grundlegendes Format:

`diff` *datei1 datei2*`<CR>`

Wenn *datei1* und *datei2* identisch sind, gibt das System eine Eingabeaufforderung aus; andernfalls gibt das Kommando `diff` an, wie die erste Datei geändert werden muß, um mit der zweiten inhaltsgleich zu werden. Für diese Änderungen kann der Zeileneditor `ed` verwendet werden (weitere Informationen über den Zeileneditor finden Sie in Kapitel 6). Das UNIX-System markiert die zu ändernden Zeilen in *datei1* mit dem Kleiner-als-Zeichen (<) und die Zeilen der Datei *datei2* (der Vorlage) mit dem Größer-als-Zeichen (>).

Angenommen, Sie möchten die Dateien `johnson` und `mcdonough` mit dem Kommando `diff` miteinander vergleichen. Die Datei `mcdonough` enthalte den gleichen Brief wie die Datei `johnson`, jedoch mit den entsprechenden Änderungen für einen anderen Empfänger. Das Kommando `diff` zeigt diese Unterschiede folgendermaßen an:

```
3,6c3,6
< Mr. Ron Johnson
< Layton Printing
< 52 Hudson Street
< New York, N.Y.
---
> Mr. J.J. McDonough
> Ubu Press
> 37 Chico Place
> Springfield, N.J.
9c9
< Dear Mr. Johnson:
---
> Dear Mr. McDonough:
```

Die erste Ausgabezeile sieht folgendermaßen aus:

```
3,6c3,6
```

Daran sehen Sie folgendes: Wenn Sie den Inhalt der Datei `johnson` an den der Datei `mcdonough` anpasssen möchten, so müssen Sie die Zeilen 3 bis 6 in der Datei `johnson` so abändern, daß sie mit den Zeilen 3 bis 6 der Datei `mcdonough` übereinstimmen (der Buchstabe c steht hier für change). Das Kommando `diff` zeigt die beiden verglichenen Zeilenbereiche an.

Nach der Durchführung der Änderung (mit einem Texteditor wie `ed` oder `vi`) ist die Datei `johnson` mit der Datei `mcdonough` identisch. Beachten Sie, daß Sie mit dem Kommando `diff` lediglich Unterschiede zwischen Dateien ermitteln können. Um von einer vorhandenen Datei eine identische Kopie anzufertigen, benutzen Sie das Kommando `cp`.

Bild 3-23 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `diff`.

Bild 3-23: `diff`**-Kurzübersicht**

Kurzbeschreibung `diff` – Unterschiede zwischen zwei Dateien anzeigen		
Kommando	*Optionen*	*Argumente*
`diff`	vorhanden*	*datei1 datei2*
`Funktion:`	Das Kommando `diff` ermittelt die Zeilen, die in zwei Dateien Unterschiede aufweisen und informiert Sie darüber, wie Sie die erste Datei in inhaltliche Übereinstimmung mit der zweiten bringen können.	
`Hinweise:`	Die Änderungen können mit den folgenden Kommandos des Zeileneditors (`ed`) durchgeführt werden: `a` (append = anfügen), `c` (change = überschreiben) und `d` (delete = löschen). Die Zahlen, die bei `a`, `c` oder `d` angegeben werden, stehen für die laufenden Nummern der zu ändernden Zeilen. Das Kleiner-als-Zeichen (<) markiert die Zeilen der ersten Datei und das Größer-als-Zeichen (>) die Zeilen der zweiten Datei.	

* Einen Überblick über sämtliche Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `diff`(1).

Mustervergleich in Dateien: Das Kommando`grep`

Mit dem UNIX-Kommando `grep` (für globally search for a regular expression and print) können Sie in einer Datei nach einer Stelle suchen, die zu einem bestimmten Suchmuster (ein oder mehreren Wörtern oder einer Folge von Zeichen) paßt. Das Suchmuster geben Sie in Form von sogenannten regulären Ausdrücken an; bei einem regulären Ausdruck handelt es sich, grob ausgedrückt, um ein Zeichenmuster (ein Wort, ein Satz oder eine Gleichung),

das von Ihnen angegeben wird.

Das Kommando hat folgendes grundlegendes Format:

```
grep muster datei(en)<CR>
```

Wenn Sie beispielsweise in der Datei `johnson` nach allen Zeilen suchen möchten, die das Wort "automation" enthalten, müssen Sie folgendes eingeben:

```
grep automation johnson<CR>
```

Das System erzeugt die folgende Ausgabe:

```
$ grep automation johnson<CR>
and office automation software.
$
```

Die Ausgabe besteht aus allen Zeilen der Datei `johnson`, die das gesuchte Muster (automation) enthalten.

Wenn das Suchmuster aus mehreren Wörtern besteht und Zeichen enthält, die unter UNIX eine Sonderbedeutung haben (z.B. `$`, `|`, `*`, `?` usw.), so muß das gesamte Muster in Hochkommata (') eingeschlossen sein (Informationen zur Sonderbedeutung von diesen und anderen Zeichen finden Sie im Kapitel 9, Abschnitt "Metazeichen"). Wenn Sie beispielsweise nach der Zeichenkette `office automation` suchen möchten, sieht Ihre Eingabezeile und die Ausgabe des Systems folgendermaßen aus:

```
$ grep 'office automation' johnson<CR>
and office automation software.
$
```

Sie können sich z.B. nicht mehr erinnern, in welchem Brief `office automation` enthalten ist - im Brief an Mr. Johnson oder an Mrs. Sanders? Mit folgender Kommandozeile können Sie es herausfinden:

```
$ grep 'office automation' johnson sanders<CR>
johnson:and office automation software.
$
```

Diese Ausgabe zeigt Ihnen, daß die Zeichenkette `office automation` einmal in der Datei `johnson` vorhanden ist.

Von dem Kommando `grep` gibt es unter UNIX die Varianten `egrep` und `fgrep`; zusätzlich stehen Optionen zur Verfügung, die auch die Durchführung von komplexen Suchvorgängen ermöglichen. Weitere Informationen dazu finden Sie im *Referenzhandbuch für Benutzer* unter den Einträgen `grep`(1), `egrep`(1), und `fgrep`(1).

Bild 3-24 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `grep`.

Bild 3-24: `grep`-Kurzübersicht

Kurzbeschreibung `grep` – Mustervergleich in einer Datei.		
Kommando	*Optionen*	*Argumente*
`grep`	vorhanden*	*muster datei(en)*
`Funktion:`	Das Kommando `grep` durchsucht ein oder mehrere Dateien nach einem bestimmten Muster und zeigt die Zeilen an, die dieses Muster enthalten. Werden zwei oder mehr Dateien angegeben, wird zusätzlich der Name der Datei, in der das Muster gefunden worden ist, ausgegeben.	
`Hinweis:`	Wenn die angegebene Zeichenkette aus mehreren Wörtern oder Sonderzeichen besteht, muß das Suchmuster in der Kommandozeile zwischen Hochkommata (') stehen.	

* Einen Überblick über sämtliche Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `grep(1)`.

Dateien sortieren und mischen: Das Kommando `sort`

Das UNIX-System weist ein leistungsfähiges Werkzeug zum Sortieren und Mischen von Dateien auf. Dieses Werkzeug ist das Kommando `sort`, das folgendes Format hat:

`sort` *datei(en)*<CR>

Mit diesem Kommando werden die Zeilen in den angegebenen Dateien gemischt und ihr Inhalt in der folgenden Reihenfolge sortiert:

- Zeilen, die mit Ziffern beginnen, werden nach ihrer ersten Ziffer sortiert und vor die mit Buchstaben beginnenden Zeilen gesetzt.
- Zeilen, die mit Großbuchstaben beginnen, werden vor den Zeilen eingereiht, die mit Kleinbuchstaben beginnen.
- Zeilen, die mit Symbolen beginnen (z.B. `*`, `%` oder `@`), werden nach dem ASCII-Code für dieses Symbol sortiert.

Sollen beispielsweise die beiden Dateien `group1` und `group2`, in denen Namenslisten enthalten sind, gemischt und nach den Namen in alphabetische Reihenfolge gebracht werden, so rufen Sie zunächst den Inhalt der Dateien mit dem Kommando `cat` ab.

```
$ cat group1<CR>
Smith, Allyn
Jones, Barbara
Cook, Karen
Moore, Peter
Wolf, Robert
$ cat group2<CR>
Frank, M. Jay
Nelson, James
West, Donna
Hill, Charles
Morgan, Kristine
$
```

Sie können auch den Inhalt der beiden Dateien mit einer einzigen Kommandozeile (`cat group1 group2`) abrufen. Das System erzeugt dann dieselbe Ausgabe.

Mischen Sie nun mit dem Kommando `sort` den Inhalt der beiden Dateien und sortieren Sie sie. Die Ausgabe des Kommandos `sort` wird auf dem Bildschirm angezeigt, wenn nicht ausdrücklich eine andere Angabe gemacht wird.

```
$ sort group1 group2<CR>
Cook, Karen
Frank, M. Jay
Hill, Charles
Jones, Barbara
Moore, Peter
Morgan, Kristine
Nelson, James
Smith, Allyn
West, Donna
Wolf, Robert
$
```

Mit dem Kommando `sort` können Sie nicht nur einfache Listen wie im obigen Beispiel miteinander verknüpfen, sondern auch Zeilen oder Teile von Zeilen (sogenannte Felder) nach bestimmten Kriterien, die Sie in der Kommandozeile angeben, umordnen. Die Beschreibung dieser sehr komplexen Möglichkeiten würde jedoch den Rahmen dieses Kapitels sprengen. Eine vollständige Beschreibung der verfügbaren Optionen finden Sie im *Referenzhandbuch für Benutzer.*

Bild 3-25 enthält eine Kurzübersicht über Syntax und Funktion des Kommandos `sort`.

Bild 3-25: `sort`-Kurzübersicht

Kurzbeschreibung `sort` – Mischen und Sortieren von Dateien.		
Kommando	*Optionen*	*Argumente*
`sort`	vorhanden*	*datei(en)*
`Funktion:`	Das Kommando `sort` sortiert die Zeilen einer Datei und zeigt die Ausgabe auf Ihrem Bildschirm an; bei zwei oder mehr Dateien wird der Inhalt der Dateien gemischt und sortiert.	
`Anmerkungen:`	Wenn in der Kommandozeile keine Optionen angegeben werden, werden die Zeilen nach dem ASCII-Code des ersten Zeichens dieser Zeile sortiert.	

* Einen Überblick über sämtliche Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `sort`(1).

4 Überblick über die Einführungskurse

Einführung 4-1

Verwaltung des System-Office 4-2

Editieren von Text 4-3
Was ist ein Texteditor? 4-3
Zur Funktionsweise eines Texteditors 4-4
- Editor-Puffer 4-4
- Betriebsmodi 4-5

Der Zeileneditor 4-5
Der Bildschirmeditor 4-6

Die Shell 4-9
Unterstützung von Systemfunktionen 4-9
Anpassung Ihrer Rechnerumgebung 4-10
Programmieren in der Shell 4-11

Programmieren mit `awk` 4-14

Elektronische Kommunikation 4-15

Programmieren unter UNIX 4-16

Einführung

Dieses Kapitel leitet von den ersten drei Kapiteln mit dem allgemeinen Überblick über das UNIX-Betriebssystem zu den Einführungskursen in den nachfolgenden sieben Kapiteln über. Es enthält einen kurzen Überblick über die einzelnen Themen, die in diesen Einführungskursen behandelt werden: Die Benutzung einer neuen Systemschnittstelle zur vereinfachten Durchführung von häufig anfallenden Aufgaben, die Texteditoren, das Arbeiten mit der Shell, die Benutzung einer leistungsfähigen Programmiersprache zur Datenmanipulation und Informationsabfrage sowie die elektronische Kommunikation.

In Kapitel 5, "Einführung in FACE", wird eine neue UNIX-Einrichtung behandelt. Das Editieren von Texten ist Thema der Kapitel 6 "Der Zeileneditor `ed`", und Kapitel 7, "Der Bildschirmeditor `vi`". Kapitel 8, "Der Druck-Service `lp`", beschreibt das Ausdrucken von Dateien und dazugehörige Aufgaben. Das Arbeiten mit der Shell und die Shell-Programmierung ist Thema von Kapitel 9, "Die Shell". Die leistungsfähige Programmiersprache `awk` wird ausführlich in Kapitel 10, "Die Programmiersprache `awk` ", beschrieben; hier finden Sie auch eine Kurzübersicht über die `awk`-Kommandos. Die zahlreichen Möglichkeiten der elektronischen Kommunikation werden in Kapitel 11, "Die elektronische Post" und Kapitel 12, "Kommunikation mit fernen Systemen" behandelt.

Verwaltung des System-Office

Ihr UNIX-"Office" besteht - analog zu Ihrem realen Arbeitsbereich - aus Dateien, Ablagen, Kopien, Fächern zur Aufbewahrung von Daten sowie Möglichkeiten zum Versetzen, Reproduzieren und Vernichten der Daten. FACE (Framed Access Command Environment) ermöglicht Ihnen das die effiziente und benutzerfreundliche Organisation Ihres "Office". Mit FACE können Sie nicht nur Ihre eigenen Dateien manipulieren, sondern auch auf die Dateien anderer Benutzer zugreifen. Ebenso wie in Ihrem realen Arbeitsbereich haben Sie unter FACE zahlreiche Möglichkeiten zur Gestaltung Ihrer Umgebung in Form von Menüs. Die Bedienung der Menüs, der Ausschnitte, in denen sie angezeigt werden, sowie die Tasten zur Wahl von Optionen in den Menüs werden ausführlich in Kapitel 5 beschrieben.

Editieren von Text

Kenntnisse über die Bedienung des Dateisystems sind beim Arbeiten in einer UNIX-Umgebung unverzichtbar. Dieser Abschnitt enthält eine Einführung in das Erstellen und Ändern von Dateien mit einem sogenannten Texteditor. Zunächst erfahren Sie, was ein Texteditor ist und wie er funktioniert. Dann werden Ihnen zwei von UNIX unterstützte Texteditoren vorgestellt: der Zeileneditor `ed` sowie der Bildschirmeditor `vi` (für visual editor). Außerdem werden die beiden Editoren miteinander verglichen. Ausführliche Informationen über `ed` und `vi` finden Sie in Kapitel 6 bzw. 7.

Was ist ein Texteditor?

Bei jeder Überprüfung eines Briefs oder Berichts müssen Sie u.a. die folgenden Aufgaben durchführen: neuen oder zusätzlichen Text einfügen, nicht mehr benötigten Text löschen, Text versetzen (d.h. ausschneiden und wieder einfügen) und schließlich eine saubere, korrigierte Version erstellen. Ein Texteditor führt diese Funktionen auf Ihre Anweisung hin durch und ermöglicht Ihnen so ein wesentlich einfacheres und schnelleres Erstellen und Überarbeiten von Text, als wenn Sie dafür die Schreibmaschine oder Bleistift und Papier benutzen würden.

Die UNIX-Texteditoren sind wie die UNIX-Shell interaktive Programme, d.h., sie nehmen Ihre Kommandos entgegen und führen die geforderten Funktionen durch. Die Texteditoren sind für die Shell ausführbare Programme.

Ein wichtiger Unterschied zwischen einem Texteditor und der Shell liegt jedoch in den akzeptierten Kommandos. Alle Kommandos, die Sie bisher in diesem Leitfaden kennengelernt haben, gehören zu den Shell-Kommandos. Ein Texteditor verfügt über eigene Kommandos zum Erstellen, Versetzen, Einfügen und Löschen von Text in Dateien sowie zum Einbringen von Text aus anderen Dateien.

Zur Funktionsweise eines Texteditors

Zum Verständnis der Funktionsweise eines Texteditors benötigen Sie Kenntnisse über die Umgebung, die beim Arbeiten mit einem Editierprogramm angelegt wird, sowie über die Betriebsmodi, zwischen denen bei einem Texteditor unterschieden wird.

Editor-Puffer

Wenn Sie mit einem Texteditor eine neue Datei anlegen bzw. eine vorhandene ändern, müssen Sie zunächst die Shell anweisen, dem Editor die Kontrolle über Ihre Rechnersitzung zu geben. Sobald der Editor die Kontrolle übernimmt, belegt er im Speicher einen temporären Arbeitsbereich, der als Editor-Puffer bezeichnet wird. Alle Informationen, die Sie in einer Editor-Sitzung eingeben, werden in diesem Puffer gespeichert; hier können Sie die Informationen auch abändern.

Da der Puffer ein temporärer Arbeitsbereich ist, sind auch alle von Ihnen eingegebenen Textdaten und alle vorgenommenen Änderungen temporär. Der Puffer und sein Inhalt bleiben nur so lange bestehen, wie Sie sich in einer Editor-Sitzung befinden. Wenn Sie die Datei sichern möchten, müssen Sie den Texteditor anweisen, den Pufferinhalt in eine Datei einzubringen. Die Datei wird dann auf dem Festspeicher des Rechners abgespeichert. Wenn Sie die Datei nicht abspeichern, geht der Pufferinhalt beim Verlassen des Editierprogramms verloren. Um dies zu verhindern, gibt der Texteditor eine Warnung aus, wenn Sie eine Editor-Sitzung ohne vorheriges Abspeichern Ihrer Datei beenden möchten.

Hinweis: Wenn Sie einen schwerwiegenden Fehler gemacht haben oder mit der editierten Version nicht zufrieden sind, so können Sie den Editor auch ohne vorheriges Abspeichern der Datei beenden. Die ursprüngliche Datei bleibt dadurch unverändert; nur die editierte Kopie wird gelöscht.

Der Text im Puffer ist in Zeilen eingeteilt, unabhängig davon, ob Sie eine neue Datei erstellen oder eine vorhandene aktualisieren. Eine Textzeile ist einfach eine Reihe von Zeichen, die auf dem Bildschirm horizontal aufeinanderfolgen; sie endet, wenn Sie die Taste RETURN betätigen. Es kann vorkommen, daß eine Textzeile in der Breite nicht auf den Bildschirm des Terminals paßt. Auf einigen Terminals wird die Textzeile dann automatisch in der nächsten Zeile fortgesetzt.

Betriebsmodi

Bei Texteditoren wird zwischen zwei Betriebsmodi unterschieden: dem Kommandomodus und dem Eingabemodus. Unmittelbar nach dem Aufrufen des Editors befinden Sie sich automatisch im Kommandomodus. In diesem Modus können Sie an eine andere Stelle der Datei springen, in ihr nach Zeichenketten suchen oder vorhandenen Text austauschen. Die Erstellung von Text ist im Kommandomodus allerdings nicht möglich. Hierzu müssen Sie sich im Eingabemodus befinden. In diesem Modus werden die von Ihnen eingegebenen Zeichen in Ihre Textdatei im Puffer eingebracht. Wenn Sie keinen Text mehr eingeben und wieder Editierkommandos aufrufen möchten, so müssen Sie wieder in den Kommandomodus zurückschalten.

Da Sie während einer Editor-Sitzung in der Regel ständig zwischen diesen beiden Modi hin- und herschalten, kann es vorkommen, daß Sie Ihren aktuellen Modus nicht mehr kennen, d.h., daß Sie im Kommandomodus Text eingeben oder im Eingabemodus Kommandos aufrufen möchten. Selbst erfahrene Benutzer sind dagegen nicht vollkommen gefeit. Nachdem Sie die Einführungen in Kapitel 6 und 7 durchgearbeitet haben, werden Sie die Ursache und die Lösung des Problems schnell kennen.

Der Zeileneditor

Der Zeileneditor, der mit dem Kommando `ed` aufgerufen wird, ist ein schnelles, flexibles Programm zur Erstellung von Textdateien. Die Bezeichnung Zeileneditor kommt daher, daß der Text Zeile für Zeile manipuliert wird. Dies bedeutet, daß Sie die Zeile mit dem zu ändernden Text über ihre Zeilennummer angeben müssen. `ed` zeigt dann die Zeile auf dem Bildschirm an, so daß Sie sie ändern können.

Dieser Texteditor verfügt über Kommandos zum Ändern und Ausgeben von Zeilen, zum Einlesen und Abspeichern von Dateien sowie zum Eingeben von Text. Außerdem können Sie den Zeileneditor innerhalb eines Shell-Programms aufrufen; dies ist mit dem Bildschirmeditor nicht möglich (eine Einführung in die Shell-Programmierung finden Sie in Kapitel 9, "Die Shell").

Der Zeileneditor `ed` eignet sich sowohl für Datensichtgeräte als auch für Drucker-Terminals. Er ist auch auf Terminals mit niedriger Übertragungsgeschwindigkeit einsetzbar (der Bildschirmeditor `vi` kann ausschließlich auf Datensichtgeräten benutzt werden). Eine ausführliche

Einführung in die Bedienung dieses Editierprogramms finden Sie in Kapitel 6, "Der Zeileneditor ed". Anhang D enthält zusätzlich einen Überblick über die Kommandos des Zeileneditors.

Der Bildschirmeditor

Der Bildschirmeditor, der durch das Kommando `vi` aufgerufen wird, ist ein bildschirmorientiertes, interaktives Softwarewerkzeug. Bei diesem Editor wird die aktuell editierte Datei seitenweise auf dem Bildschirm angezeigt. Die höchste Effizienz erreichen Sie bei diesem Editor auf einem Terminal, das mit 1200 oder mehr Baud betrieben wird.

Das Editieren mit diesem Editor besteht im wesentlichen aus dem Ändern (d.h. dem Einfügen, Löschen oder Überschreiben) von Text; hierzu wird der Cursor zu der Stelle auf dem Bildschirm bewegt, die geändert werden soll, und dann die Änderung vorgenommen. Der Bildschirmeditor zeigt das Ergebnis einer durchgeführten Editier-Funktion sofort auf dem Bildschirm an; die Änderungen können also innerhalb des jeweiligen Kontexts
begutachtet werden. Aus diesem Grund gilt der Bildschirmeditor als komfortabler als der Zeileneditor.

Der Bildschirmeditor stellt außerdem eine Reihe von Kommandos zur Verfügung. So gibt es z.B. spezielle Kommandos, mit denen der Cursor an eine andere Stelle der Datei bewegt werden kann. Mit anderen Kommandos können Sie die Datei auf dem Bildschirm nach unten und oben verschieben ("rollen"). Auch zum Ändern von vorhandenem oder Eingeben von neuem Text gibt es spezielle Kommandos. Zusätzlich zu den Kommandos des Bildschirmeditors stehen während einer Editor-Sitzung auch die Zeilenditor-Kommandos zu Ihrer Verfügung.

Der Geschwindigkeit, ansprechenden Darstellung, Effizienz und Leistungsfähigkeit des Bildschirmeditors steht seine hohe CPU-Beanspruchung gegenüber. Bei jeder Änderung, und sei sie noch so einfach, muß `vi` die Bildschirmanzeige aktualisieren. Eine Einführung in die Bedienung dieses Editors finden Sie in Kapitel 7, "Der Bildschirmeditor vi ." Anhang E enthält eine Kurzübersicht über die Kommandos des Bildschirmeditors; in Tabelle 4-1 werden die Eigenschaften des Zeileneditors (`ed`) und des Bildschirmeditors (`vi`) einander gegenübergestellt.

Tabelle 4-1: Vergleich des Zeilen- und des Bildschirmeditors (`ed` und `vi`)

Eigenschaft	Zeileneditor (`ed`)	Bildschirmeditor (`vi`)
Empfohlener Terminal-Typ	Datensichtgerät oder Drucker-Terminal.	Datensichtgerät
Geschwindigkeit	Für hohe und niedrige Übertragungsgeschwindigkeiten geeignet.	Besonders bei Übertragungsleitungen mit hoher Übertragungsgeschwindigkeit empfehlenswert (1200 Baud und mehr).
Flexibilität	Kann sowohl innerhalb einer Shell-Prozedur als auch während einer Editor-Sitzung benutzt werden.	Kann nur interaktiv, während einer Editor-Sitzung, benutzt werden.
Leistungsfähigkeit	Schnelle Durchführung von Textänderungen. Relativ geringer Bedarf an CPU-Zeit.	Einfache Durchführung von Textänderungen. Allerdings hoher Verbrauch an CPU-Zeit.

Tabelle 4-1: Vergleich des Zeilen- und des Bildschirmeditors (`ed` und `vi`) (Fortsetzung)

Eigenschaft	Zeileneditor (`ed`)	Bildschirmeditor (`vi`)
Funktionsumfang	Kommandos für alle Editierfunktionen. Der Standard-UNIX-Texteditor.	Außer den speziellen vi-Kommandos stehen die Zeileneditor-Kommandos zur Verfügung.
Vorteile	Weniger Kommandos und dadurch kürzere Einarbeitungszeit für `ed`.	Unter `vi` können Sie die Auswirkungen Ihrer Änderungen sofort im Kontext einer Textseite begutachten (in einer `ed`-Sitzung sind die Durchführung der Änderungen und das Abrufen der Ergebnisse getrennte Schritte).

Die Shell

Bei jeder Anmeldung bei UNIX setzt ein Dialog mit der Shell ein, der erst bei Ihrer Abmeldung beendet ist. Allerdings wird Ihr Dialog mit der Shell während einer Texteditor-Sitzung unterbrochen; er wird wieder aufgenommen, sobald Sie den Editor verlassen.

Die Shell funktioniert im Großen und Ganzen wie andere Programme. Im Gegensatz zu Kommandos wie `cat` oder `ls` führt sie jedoch nicht nur eine bestimmte Funktion durch, sondern spielt bei Ihrem Dialog mit dem UNIX-Betriebssystem eine zentrale Rolle. In Ihrer wichtigsten Funktion ist die Shell ein Kommandointerpreter zwischen Ihnen und dem Rechner. Interpreter bedeutet, daß sie Ihre Anforderungen in eine Sprache umsetzt, die vom Rechner verstanden wird, die angeforderten Programme in den Speicher lädt und sie ausführt.

Unterstützung von Systemfunktionen

Die Shell weist verschiedene Funktionen und Einrichtungen auf, die Ihnen das Arbeiten mit den Systemfunktionen erleichtern. Außer zum Aufrufen von einzelnen Programmen können Sie die Shell benutzen, um:

- den abgekürzten Namen einer Datei oder eines Verzeichnisses zu interpretieren;
- den Eingabe- oder Ausgabedatenstrom Ihrer Programme umzuleiten;
- mehrere Programme gleichzeitig oder hintereinander in einer Pipeline auszuführen;
- Ihre Rechnerumgebung Ihren persönlichen Bedürfnissen anzupassen.

Die Shell ist nicht nur ein Kommandointerpreter, sondern auch eine Programmiersprache. Ausführliche Informationen über den Umgang mit der Shell als Kommandointerpreter und Programmiersprache finden Sie in Kapitel 9, "Die Shell".

Anpassung Ihrer Rechnerumgebung

Mit Hilfe der Shell können Sie Ihre Umgebung ändern. Bei Ihrer Anmeldung bei UNIX lädt die Shell automatisch eine Rechnerumgebung für Sie. Die von der Shell eingerichtete Standardumgebung enthält folgende Variablen:

`HOME`	Ihr Home-Verzeichnis
`LOGNAME`	Ihr Benutzername
`PATH`	Der Pfad, den die Shell nach ausführbaren Dateien oder Kommandos durchsucht (z.B. `PATH=:/usr/bin:/usr/usr/bin`).

Die Variable `PATH` teilt der Shell mit, wo sie die von einem Kommando aufgerufenen ausführbaren Programme suchen soll. Somit wird auf diese Variable bei jeder Eingabe eines Kommandos zugegriffen. Wenn Ihre ausführbaren Programme in mehr als einem Verzeichnis enthalten sind, können Sie alle Verzeichnisse von der Shell durchsuchen lassen, um sicherzustellen, daß jedes Kommandos ausfindig gemacht wird.

Sie können die von Ihrem System bereitgestellte Standardumgebung benutzen oder die Umgebung Ihren persönlichen Bedürfnissen anpassen. Zur Änderung Ihrer Umgebung gibt es zwei Möglichkeiten. Wenn die geänderte Umgebung nur für die Dauer Ihrer aktuellen Terminalsitzung gültig sein soll, können Sie Ihre Änderungen in einer Kommandozeile angeben. Wenn Sie mit dieser geänderten Umgebung (also nicht der Standardumgebung) jedoch zukünftig bei jeder Sitzung arbeiten möchten, können Sie Ihre Änderungen in eine Datei einbringen, die die gewünschte Umgebung bei jeder Anmeldung für Sie lädt. Diese Datei muß den Namen `.profile` haben und sich in Ihrem Home-Verzeichnis befinden.

In der Datei `.profile` sind normalerweise die folgenden Funktionen (bzw. ein Teil davon) festgelegt: Überprüfung, ob Post angekommen ist; Einstellung von Daten-Parametern, Terminal-Optionen und Tabulatorstopps; Festlegung eines Zeichens oder einer Zeichenkette für Ihre Eingabeaufforderung; Zuordnung des Zeichenlösch- und Zeilenlösch-Zeichen an bestimmte Tasten. Sie können in Ihrer `.profile` beliebig viele Funktionen festlegen. Sie können auch jederzeit Änderungen daran vornehmen (siehe auch Abschnitt "Änderung Ihrer Benutzerumgebung" in Kapitel 9).

Überprüfen Sie jetzt, ob es in Ihrem Dateisystem eine Datei namens `.profile` gibt. Wechseln Sie mit dem Kommando `cd` in Ihr Home-Verzeichnis, wenn Sie sich nicht bereits dort befinden. Rufen Sie mit dem folgenden Kommando Ihre Datei `.profile` auf dem Bildschirm ab:

```
cat .profile
```

Wenn es die Datei `.profile` in Ihrem Dateisystem gibt, erscheint ihr Inhalt nun auf dem Bildschirm; andernfalls kann sie mit einem der Texteditoren `ed` oder `vi` erstellt werden (siehe Kapitel 9, Abschnitt "Änderung Ihrer Benutzerumgebung").

Programmieren in der Shell

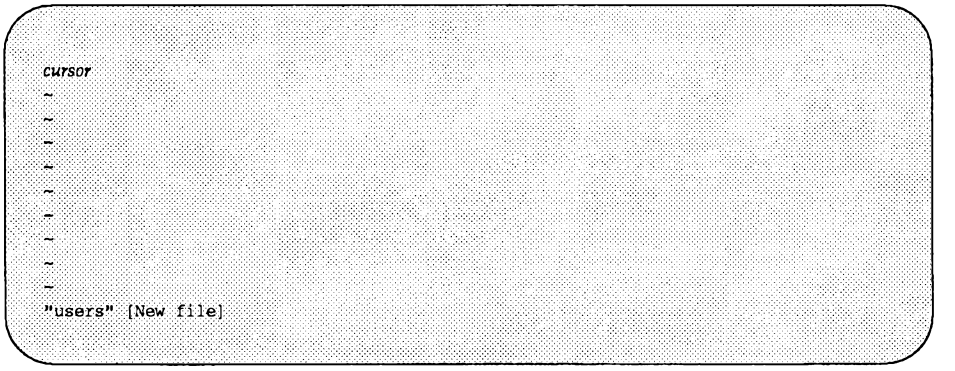
Die Shell ist nicht nur ein Kommandointerpreter, sondern auch eine Kommandosprache. D. h., Sie können die Shell nicht nur als Verbindung zwischen Ihnen und dem Rechner benutzen, sondern auch zur automatischen Wiederholung von Kommandofolgen anweisen. Zu diesem Zweck müssen Sie ausführbare Dateien erstellen, die mehrere Kommandos enthalten. Diese Dateien werden Shell-Prozeduren oder Shell-Skripts genannt. Nachdem Sie eine Shell-Prozedur für eine bestimmte Funktion erstellt haben, muß die Shell nur noch angewiesen werden, den Inhalt der Prozedur zu lesen und auszuführen.

Die Shell stellt Ihnen, ebenso wie andere Programmiersprachen, Einrichtungen wie Variablen, Strukturen zur Steuerung des Programmablaufs, Unterprogramme und Parameterübergabe zur Verfügung. Dies gibt Ihnen die Möglichkeit, Ihre eigenen Werkzeuge zu entwickeln, indem Sie einfach Systemkommandos miteinander verknüpfen.

So können Sie z.B. die drei UNIX-Kommandos `date`, `who` und `wc` zu einer einfachen Shell-Prozedur namens `users` verknüpfen, die das aktuelle Datum, die Uhrzeit und die Anzahl der aktuell angemeldeten Benutzer ausgibt. Wenn Sie Ihre Shell-Prozedur mit dem Editor `vi` (siehe Kapitel 7) erstellen, können Sie das folgende Beispiel nachvollziehen. Legen Sie zunächst mit dem Editor die Datei `users` an:

```
vi users<CR>
```

Der Editor zeigt eine leere Bildschirmseite an und wartet darauf, daß Sie nach der Eingabe von "`i`" für einfügen (insert) oder "`a`" für anfügen (append) einen Text eingeben.



Geben Sie die drei UNIX-Kommandos in einer Zeile ein:

```
date; who | wc -l
```

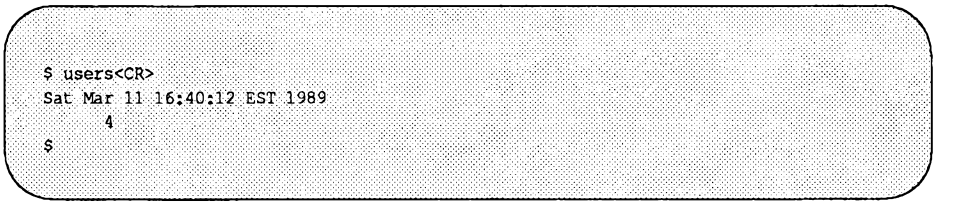
Betätigen Sie dann die Taste ESCAPE; speichern Sie die Datei abschließend ab und verlassen Sie den Editor:

```
:wq
```

Machen Sie die Datei `users` durch die Zuteilung der Ausführungsberechtigung zu einer ausführbaren Datei. Dafür geben Sie das folgende `chmod`-Kommando ein:

```
chmod ug+x users<CR>
```

Versuchen Sie jetzt, Ihr neues Kommando aufzurufen. Der folgende Bildschirm zeigt die Ausgabe des Kommandos:



Die Ausgabe zeigt Ihnen, daß zum Zeitpunkt der Kommandoeingabe (Saturday, March 11, 16:40) vier Benutzer angemeldet waren.

Eine ausführliche Einführung in die Erstellung von Shell-Prozeduren sowie in komplexere Shell-Programmstrukturen finden Sie in Kapitel 9, "Die Shell".

Programmieren mit awk

awk ist sowohl eine effiziente Programmiersprache als auch ein ausgereiftes Werkzeug zur Ausführung von Shell- Prozeduren, das Ihnen die einfache Realisierung von Datenmanipulationen und Informationsabfragen ermöglicht. Eine Einführung in die Programmiersprache awk und den Aufbau eines awk- Programms finden Sie in Kapitel 10, "Die Programmiersprache awk".

Ein awk- Programm besteht im Normalfall aus einem einzigen Muster-Aktions-Paar. Die grundlegende Funktion eines awk- Programms besteht darin, jede Eingabezeile mit jedem Muster zu vergleichen. Für jedes passende Muster wird die zugehörige Aktion ausgeführt. Die Mustervergleiche werden so lange ausgeführt, bis die gesamte Eingabe eingelesen worden ist.

Wegen der breitgefächterten Anwendungsmöglichkeiten und ihres Schwerpunkts auf der Konzeption von awk sollten Sie die ausführliche Beschreibung von awk erst durcharbeiten, wenn Sie etwas Erfahrung mit der Shell und deren Möglichkeiten haben. Bis zu diesem Zeitpunkt reicht die Kurzübersicht am Ende von Kapitel 10 als erste Einführung in die verschiedenen Aspekte der Programmiersprache awk aus.

Elektronische Kommunikation

Als UNIX-Benutzer können Sie an andere Benutzer, die an Ihrem oder einem anderen UNIX-System arbeiten, Nachrichten oder Informationen schicken, die in Dateien enthalten sind. Sie können Nachrichten senden und empfangen, Dateien austauschen und Netzwerke mit anderen UNIX-Systemen bilden. Zu diesem Zweck müssen Sie an einem UNIX-System angemeldet sein, das mit dem UNIX-System, an das Sie Informationen senden wollen, kommunizieren kann. Für die Datenübertragung gibt es verschiedene Kommandos für unterschiedliche Zwecke. In diesem Leitfaden werden Ihnen die Übertragungsprogramme und die zugehörigen Kommandos vorgestellt. In Kapitel 11, "Die elektronische Post", und Kapitel 12, "Kommunikation mit fernen Systemen" finden Sie außer ausführlichen Informationen zu den UNIX-Übertragungskommandos zusätzlich Übungen und Kurzübersichten.

Programmieren unter UNIX

UNIX stellt dem Programmierer und Software-Entwickler eine leistungsfähige und komfortable Umgebung zur Verfügung. Neben zahlreichen Programmiersprachen weist UNIX einige ausgereifte Werkzeuge auf, die die Software-Entwicklung erleichtern und die systematische Lösung von Programmieraufgaben ermöglichen.

Informationen über die verfügbaren UNIX-Programmiersprachen finden Sie in *Product Overview* oder *Documentation Roadmap.*

Allgemeine Informationen zur Programmierung unter UNIX finden Sie im *Leitfaden für Programmierer,* in dem außer Beschreibungen der Programmiersprachen auch Einführungen in verschiedene Werkzeuge zur Software-Entwicklung enthalten sind.

5 Einführung in FACE

Was ist FACE? 5-1
Zur Einteilung dieses Kapitels 5-1

Einstieg 5-3
Anmelden 5-3
Die Funktionsbereiche des FACE-Bildschirms 5-5
Der Bildschirm unmittelbar nach Ihrer Anmeldung 5-7
- Ersatztasten 5-10

Benutzung eines Menüs 5-13
- Funktionstasten-Titel in Menüs 5-13
- Steuerung des Cursors in einem Menü 5-14
- Optionswahl in einem Menü 5-18

Arbeiten mit Schablonen 5-21
- Funktionstasten-Titel in Schablonen 5-22
- Steuerung des Cursors in und Editieren einer Schablone 5-23

Umschalten zwischen Ausschnitten 5-29
- Umschalten zwischen Ausschnitten von der Kommandozeile aus 5-29
- Umschalten zwischen Ausschnitten über Funktionstasten 5-30
- Umschalten zwischen Ausschnitten über das Kommando `frm-mgmt` 5-31

Aussehen Ihres FACE-Office festlegen 5-32
- Versetzen eines Ausschnitts 5-33
- Größe eines Ausschnitts ändern 5-34

Das Command Menu 5-35
Zugriff auf das Hilfesystem 5-38
- Informationen über FACE-Kommandos, Menüs und Schablonen-Felder 5-38
- Allgemeine Informationen zu FACE abfragen 5-39

Beenden von FACE 5-42

Benutzung des FACE-Office 5-44
Ihr Dateisystem 5-44
- Was sind Dateien und Kataloge? 5-44
- Anlegen von Dateien und Katalogen 5-48
- Kopieren und Versetzen von Dateien und Katalogen 5-52
- Umbenennen einer Datei bzw. eines Katalogs 5-54
- Neue Beschreibung für eine Datei oder einen Katalog 5-55
- Umorganisation des Inhalts eines einzelnen Katalogs 5-57
- Löschen einer Datei bzw. eines Katalogs 5-61
- Löschen einer Datei bzw. eines Katalogs rückgängig machen 5-62
- Suchen von Dateien und Katalogen 5-64
- Anzeige des Dateiinhalts 5-68
- Zugriffsschutz für eine vorhandene Datei 5-68

Zugriff auf die Dateisysteme der anderen Benutzer 5-72
Einstellung Ihrer Office-Parameter 5-72
- Change Password 5-74
- Color Attributes 5-75
- Anzeige mehrerer Menüs nach der Anmeldung 5-76
- File Permissions 5-77
- Office Functions 5-78

Programs Administration 5-80
Wastebasket 5-81

Weitere FACE-Funktionen 5-82
Printer Operations 5-82
Ausdrucken einer Datei 5-84
Programs 5-85
- Mail Services 5-86
- Spell Checker 5-87
- Arbeiten mit anderen Programmen 5-88

Programs Administration 5-89
- Benutzer-eigene Programme 5-89

FACE Administration 5-93
- FACE User Administration 5-94
- Verwaltung von globalen Programmen 5-98

Das UNIX-System 5-103
- Aufrufen von UNIX-Kommandos im FACE-Office 5-103

Arbeiten mit ausführbaren Dateien 5-106
Aufrufen einer Shell-Prozedur 5-107
Vorübergehendes Verlassen von und Rückkehr zu Dateien 5-109
- Vorgehensweise 5-110
- Rückkehr zu vorübergehend verlassenen Dateien 5-111

Was ist FACE?

FACE (Framed Access Command Environment) ist eine bedienungsfreundliche Schnittstelle zum UNIX-System; diese Schnittstelle stellt Ihnen einen elektronischen Arbeitsbereich zur Verfügung, in dem Sie eine Vielzahl von häufig anfallenden Aufgaben wie die Organisation Ihres Dateisystems, die gemeinsame Bearbeitung von Projekten oder die Durchführung von verschiedenen Funktionen gleichzeitig erledigen können. Außerdem können Sie damit auf die angeschlossenen Drucker, die installierten Programme, die Systemverwaltungs-Funktionen sowie das Betriebssystem UNIX zugreifen.

Zur Einteilung dieses Kapitels

Die Informationen in diesem Kapitel sind folgendermaßen gegliedert:

- Im Vorwort "Was ist FACE", das Sie gerade lesen, finden Sie eine Einführung in AT&T FACE sowie Informationen über die Gliederung dieses Kapitels.
- Der Abschnitt "Einstieg" enthält eine Einführung in Ihr "FACE-Office". Im einzelnen werden das An- und Abmelden, die benannten Tasten und ihre Ersatztasten, das Umschalten zu Menüs, die Steuerung des Cursors innerhalb von Menüs, Schablonen und Textausschnitten, das Versetzen und Verkleinern/Vergrößern von Ausschnitten, das Command Menu mit den verfügbaren FACE-Kommandos sowie der Zugriff auf das Hilfesystem beschrieben.
- Im Abschnitt "Benutzung des FACE-Office" finden Sie Informationen über Dateien, Kataloge sowie die wichtigsten FACE-Kommandos zur Manipulation von Dateien und Katalogen. Außerdem wird in diesem Abschnitt die Konfiguration Ihres AT&T FACE-Office, der Schutz von Dateien sowie die Benutzung des "Abfalls" beschrieben.
- Im Abschnitt "Weitere FACE-Funktionen" werden die Funktion Printer Operations und das Kommando `print` zum Ausdrucken von Dateien beschrieben, die Verwaltung von FACE nach der Wahl der Funktion System Administration, die Bedienung des Menüs Programs (mit den Funktionen Mail Services and Spell Checker), der Zugriff auf die Funktion Programs Administration, der Zugriff auf das UNIX-Betriebssystem selbst, die globale Verwaltung weiterer Programme unter FACE, das Aufrufen von ausführbaren Dateien, das Aufrufen von UNIX-Shell-Prozeduren unter FACE sowie das vorübergehende Verlassen von Dateien.

Als Zusatz zu diesem Kapitel werden in Anhang C, *FACE-Kurzübersicht*, die Funktionen aufgeführt und kurz erläutert, die Ihnen unter FACE zur Verfügung stehen.

Einstieg

Dieser Abschnitt enthält eine Einführung in das Arbeiten mit Ihrer FACE-Umgebung von AT&T. Sie finden hier Informationen darüber, wie Sie sich anmelden, was man unter Menüs, Schablonen und Textausschnitten versteht und wie man sie benutzt, wie Sie die Ausschnitte auf dem Bildschirm verschieben, auf das Hilfesystem zugreifen und sich abmelden können.

Außerdem wird das Umschalten zwischen Ausschnitten und zwischen den Katalogen innerhalb eines Ausschnitts beschrieben. Ein weiteres Thema ist das Aufrufen von FACE-Kommandos über die Funktionstasten (bzw. Ersatztasten), das Command Menu sowie die Kommandozeile.

Anmelden

Bevor Sie mit FACE arbeiten können, müssen Sie sich am Rechner anmelden. Das Anmelden entspricht in etwa dem Aufschließen Ihrer Büroür. Ähnlich wie andere Personen Ihre Bürotür nicht ohne Schlüssel öffnen können, kann niemand sich ohne den Schlüssel zu Ihrem Benutzernamen an Ihrem Rechner anmelden. Dem Schlüssel entspricht in diesem Fall Ihr Paßwort.

Hinweis: Sie sollten einen Benutzernamen in keinem Fall gemeinsam mit einer anderen Person benutzen. Außer zu Sicherheitsrisiken für Ihr Office kann dies zu einer gestörten FACE-Bildschirmanzeige führen (z.B. zu nicht ordnungsgemäß dargestellten Ausschnitten oder Problemen mit der Cursor-Steuerung).

Zur Anmeldung bei FACE benötigen Sie die nachfolgend aufgeführten Informationen, die teilweise vom Typ des benutzten Rechners und Terminals abhängen:

- Wie wird Ihr Terminal eingeschaltet?
- Wie wird die Verbindung zum Rechner hergestellt?
- Wie lautet Ihr Benutzername?
- Wie lautet Ihr Paßwort?
- Wie heißt der Typ Ihres Terminals?

Mit diesen Informationen können Sie anfangen.

1. Schalten Sie Ihr Terminal ein, falls es ausgeschaltet ist.
2. Stellen Sie die Verbindung zum Rechner her (falls notwendig).
3. Bei der Eingabeaufforderung `Login:` geben Sie Ihren Benutzernamen ein und betätigen die Taste [ENTER].

   Wenn Sie bei der Eingabe Ihres Benutzernamens einen Fehler machen, fängt die ganze Prozedur wieder von vorn an, d.h., auf dem Bildschirm wird wieder die Eingabeaufforderung `Login:` angezeigt.

   Bei der Eingabeaufforderung `Password:` geben Sie Ihr Paßwort ein und betätigen die Taste [ENTER]. Ihr Paßwort wird bei der Eingabe aus Sicherheitsgründen nicht auf dem Bildschirm angezeigt.
4. Bei der Aufforderung zur Eingabe des Terminal-Typs geben Sie die entsprechenden Informationen ein und betätigen die Taste [ENTER].

   Wenn Sie an einem AT&T-Terminal unbekannten Typs arbeiten, so müssen Sie lediglich die Taste [ENTER] betätigen; FACE versucht dann den Typ Ihres Terminals herauszufinden.

Hinweis: FACE kann allerdings nicht alle Terminal-Typen automatisch ermitteln; in einem solchen Fall wird erneut die Eingabeaufforderung `TERM=` angezeigt; Sie müssen den Terminal-Typ dann selbst ermitteln und eingeben.

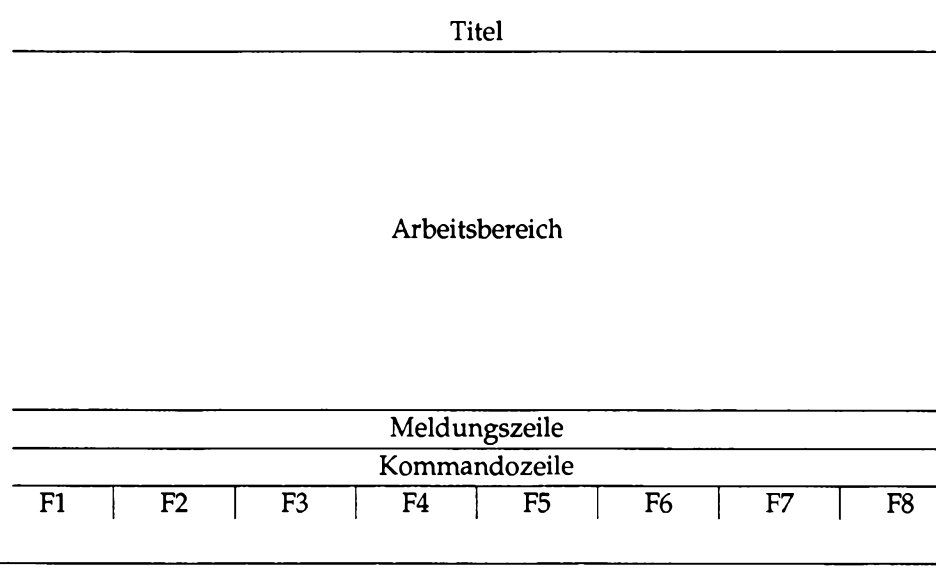
Mehr ist zum Anmelden nicht notwendig. Bevor Sie jedoch mit dem AT&T-FACE-Office zu arbeiten beginnen, sollten Sie sich mit den verschiedenen Funktionsbereichen des FACE-Bildschirms befassen.

Hinweis: Wenn FACE bei Ihrer Anmeldung nicht automatisch aufgerufen wird, müssen Sie es möglicherweise starten, indem Sie bei der UNIX-Eingabeaufforderung das Kommando `face` eingeben (ausführliche Informationen hierzu finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `face(1)`).

Die Funktionsbereiche des FACE-Bildschirms

Der FACE-Bildschirm ist in fünf logische Funktionsbereiche unterteilt (siehe Bild 5-1). Beim Durcharbeiten der Erläuterung dieser Funktionsbereiche im Anschluß an das folgende Bild können Sie Bild 5-1 als Referenz benutzen.

Bild 5-1: Die Funktionsbereiche des FACE-Bildschirms



Titel

In der Titelzeile werden Informationen über den Status des Systems angezeigt. In der Mitte dieser Zeile stehen in jedem Fall die Angaben `AT&T FACE`, das Datum sowie die Uhrzeit. Wenn Sie an die elektronische Post angeschlossen sind, wird am Anfang der Zeile `MAIL` angezeigt, wenn Post für Sie angekommen ist. Wenn FACE gerade eine Funktion durchführt, ist ganz am Ende dieser Zeile die Meldung `working ...` zu sehen. Wenn diese Meldung erscheint, können Sie erst dann mit dem FACE-Office weiterarbeiten, nachdem FACE die Meldung wieder gelöscht hat.

Hinweis: Wenn Sie Ihre Post lesen möchten, so wählen Sie zunächst im Menü AT&T FACE die Option `Programs` und dann `Mail Services` . Dadurch wird das Kommando `mailx` aufgerufen (weitere Informationen darüber, wie Sie beim Lesen Ihrer Post vorgehen, finden Sie im Abschnitt "Programs").

Arbeitsbereich

Der Arbeitsbereich ist der zentrale Teil Ihres Bildschirms. Hier werden die Schablonen, Menüs, Dateien und Kataloge, mit denen Sie aktuell arbeiten, in Ausschnitten angezeigt. Ein Beispiel für einen Ausschnitt ist das Menü AT&T FACE, das unmittelbar nach Ihrer Anmeldung bei FACE auf dem Bildschirm angezeigt wird. Die von Ihnen eröffneten (ausgewählten) Ausschnitte erscheinen auf jeden Fall im Arbeitsbereich. Wenn mehrere Ausschnitte gleichzeitig eröffnet sind, kann ein Ausschnitt vorübergehend vollständig oder teilweise durch einen anderen Ausschnitt überlagert werden.

Meldungszeile

Die Meldungszeile befindet sich in der zweiten Zeile von unten, oberhalb der Bildschirm-Titel für die Funktionstasten. Hier werden während bzw. nach der Durchführung einer Funktion (Fehler-)Meldungen angezeigt. Außerdem werden Sie hier zur Eingabe weiterer Informationen aufgefordert, die für die angeforderte Funktion gegebenenfalls erforderlich sind. Unmittelbar nach Ihrer Anmeldung wird die folgende Meldung angezeigt: `Move to an item with arrow keys and press the ENTER key to select the item.`

Kommandozeile

In der Kommandozeile werden die Kommandos angezeigt, die Sie eingeben. Um ein Kommando einzugeben, betätigen Sie die Taste [CTRL-]] oder [CTRL-f] [c] (die Kommandozeile ist unmittelbar nach Ihrer Anmeldung wahrscheinlich leer). Der Umgang mit der Kommandozeile wird im Abschnitt "Umschalten zwischen Ausschnitten" beschrieben.

Bildschirm-Titel für Funktionstasten

In der letzten Zeile Ihres FACE-Bildschirms befinden sich acht Bildschirm-Titel für die Funktionstasten. In Bild 5-1 werden sie mit F1 bis F8 angegeben. Wenn auf Ihrer Tastatur Funktionstasten vorhanden sind, so entsprechen diese Titel den Funktionstasten [F1] bis [F8] Ihrer Tastatur. So wurde z.B. der Funktionstaste [F4] in Bild 5-2 der Titel [PREV-FRM] zugeordnet.

Die Funktionstasten sind, abhängig vom Typ des aktiven Ausschnitts, unterschiedlich belegt. Mit dem Umschalten zu einer anderen Funktion ändern sich auch die Titel auf Ihrem Bildschirm. So hat ein Menü-Ausschnitt andere Funktionstasten als ein Schablonen-Ausschnitt, da Sie in Menüs andere Funktionen durchführen als in einer Schablone.

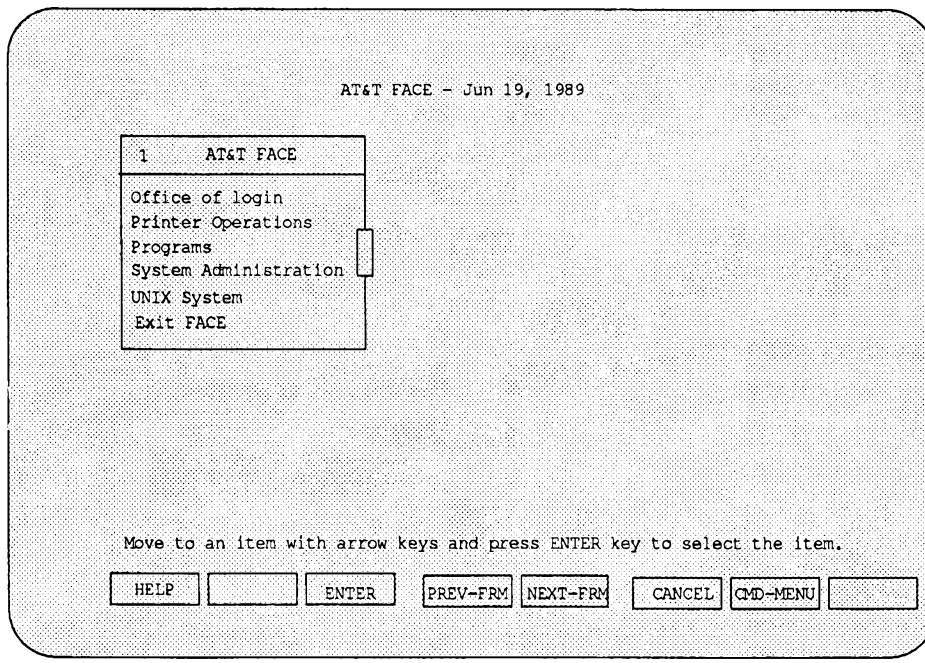
Hinweis: Wenn es auf Ihrem Terminal keine Funktionstasten gibt oder die Funktionstasten nicht korrekt funktionieren, so können Sie auf die Funktionen, die durch diese Titel angegeben werden, über spezielle Ersatztasten zugreifen (siehe Bild 5-3).

Die Funktionsfähigkeit der Funktionstasten kann davon abhängen, ob Sie während der Anmeldung bei der Eingabeaufforderung `TERM=` den korrekten Terminal-Typ angegeben haben.

Der Bildschirm unmittelbar nach Ihrer Anmeldung

Nachdem Sie sich erfolgreich angemeldet haben, wird der Bildschirm gelöscht und der erste Ausschnitt (das Menü AT&T FACE) angezeigt. Dieser Ausschnitt ist in jedem Fall eröffnet und auf Ihrem Bildschirm vorhanden. Eröffnete Ausschnitte können entweder aktiv oder inaktiv sein; auf einen eröffneten Ausschnitt können Sie jederzeit innerhalb Ihres Arbeitsbereichs zugreifen. Selbst wenn noch weitere Ausschnitte eröffnet sind, ist dem Menü AT&T FACE in jedem Fall die Nummer `1` zugeordnet (diese Nummer steht vor dem Titel, der stets AT&T FACE lautet). Von diesem Ausschnitt aus können Sie auf das gesamte FACE-System zugreifen. Das Menü AT&T FACE wird in Bild 5-2 gezeigt.

Bild 5-2: Das Menü AT&T FACE



Ihr Bildschirm sollte jetzt in etwa wie der in Bild 5-2 aussehen. Der wichtigste Teil des FACE-Bildschirms ist hier das Menü AT&T FACE mit den folgenden Optionen:

Office of login

Bei der Wahl dieser Option wird im Arbeitsbereich ein neuer Ausschnitt mit dem Titel `Office of login` eröffnet, in dem ein Menü mit weiteren verfügbaren FACE-Funktionen aufgerufen wird. Diese Option ist *Ihrem* FACE-Office zugeordnet. Wenn Sie beispielsweise den Benutzernamen `tom` haben, so sieht diese Option bei Ihrer Anmeldung folgendermaßen aus: `Office of tom`. In diesem Kapitel wird auf diese Option der Einfachheit halber immer mit `Office of login` Bezug genommen - tatsächlich steht an dieser Stelle selbstverständlich

	stets Ihr eigener `Benutzername`.
Printer Operations	Über diese Option können Sie Informationen über die Drucker abrufen, die an Ihren Rechner angeschlossen sind, und das Druck-Kommando auf Ihre Anforderungen zuschneiden. Weitere Informationen hierzu finden Sie im Abschnitt "Printer Operations".
Programs	Über diese Option können Sie ein Menü aufrufen, in dem sämtliche auf Ihrem Rechner installierten Programme enthalten sind, auf die Sie zugreifen können. Ein Programm können Sie aufrufen, indem Sie seinen Namen wählen. Sind noch keine Programme installiert, die unter FACE aufgerufen werden können, so ist diese Option nicht in Ihrem Menü AT&T FACE aufgeführt (ausführliche Informationen über diese Option finden Sie im Abschnitt "Weitere FACE-Funktionen").
System Administration	Über diese Option kann eine Person, die mit den Privilegien des Systemverwalters ausgestattet ist, FACE verwalten, Benutzer eintragen, Programme installieren, Drucker oder Modems einrichten, Benutzer-Informationen und die Systemkonfiguration abrufen, Disketten kopieren und eine Vielzahl weiterer Verwaltungsaufgaben durchführen. Wenn Sie nicht mit den Privilegien des Systemverwalters ausgestattet sind, ist diese Option in Ihrem Menü AT&T FACE nicht aufgeführt. Weitere Informationen zu diesem Thema finden Sie im Abschnitt "System Administration".
UNIX System	Über diese Option können Sie direkt auf das UNIX-System zugreifen. Wenn Sie die Zugriffsberechtigung für UNIX nicht haben, ist diese Option in Ihrem Menü AT&T FACE nicht aufgeführt. Weitere Informationen zu diesem Thema finden Sie im Abschnitt "Weitere FACE-Funktionen".

Exit FACE	Über diese Option können Sie sich von FACE abmelden. Weitere Informationen zu diesem Thema finden Sie im Abschnitt "Beenden von FACE".

Auf dem FACE-Bildschirm werden möglicherweise noch weitere Ausschnitte angezeigt, die von Ihnen zuvor festgelegt worden sind. Ignorieren Sie vorerst diese Ausschnitte und fahren Sie mit dem Kapitel fort.

Hinweis: Sind auf Ihrem Bildschirm das Menü AT&T FACE und andere Ausschnitte eröffnet, so werden Sie in der Meldungszeile möglicherweise darüber informiert, daß in Ihrem Abfall (wastebasket) Kataloge und Dateien enthalten sind, die demnächst gelöscht werden. Außerdem wird ein Menü mit diesen Dateien bzw. Katalogen angezeigt. Sie können jetzt über die Funktionstaste [MARK] die Dateien bzw. Kataloge kennzeichnen, die Sie abspeichern möchten, und die übrigen über die Taste [ENTER] löschen. Sie können aber auch angeben, daß überhaupt keine Dateien oder Kataloge gelöscht werden sollen; hierzu betätigen Sie die Taste [CANCEL] bzw. die Ersatztaste [CTRL-f]6 (ausführliche Informationen über die Wahl von Optionen in Menüausschnitten finden Sie im Abschnitt "Benutzung eines Menüs"). Vorläufig betätigen Sie in diesem Menü die Taste [CANCEL].

Die Benutzung des Abfalls wird im Abschnitt "Benutzung des FACE-Office" ausführlich beschrieben.

Ersatztasten

Es ist sehr unwahrscheinlich, daß sämtliche in diesem Kapitel angegebenen Tasten auf Ihrer Tastatur vorhanden sind. So fehlen häufig die Taste [BACKTAB] oder die Funktionstasten [F1] bis [F8]. Wenn auf Ihrer Tastatur einige der angegebenen Tasten nicht vorhanden sind (bzw. zwar vorhanden sind, aber nicht wie angegeben funktionieren), so können Sie stattdessen bestimmte Ersatztasten benutzen.

Bild 5-3 zeigt Kombinationen aus zwei oder drei Ersatztasten, die dieselbe Funktion wie die angegebene Funktionstaste haben. Bei diesen Ersatztasten steht [CTRL] für die CONTROL-Taste. Zur Betätigung einer Ersatztaste betätigen Sie die CONTROL-Taste, so wie Sie die SHIFT-Taste zur Eingabe eines Großbuchstabens betätigen. Wenn es auf Ihrer Tastatur beispielsweise die Funktionstaste [F1] nicht gibt, so betätigen Sie [CTRL-f] [1]. Zur Eingabe dieser Steuersequenz halten Sie die CONTROL-Taste gedrückt und geben das erste Zeichen ein, der auf den Bindestrich folgt (in diesem Fall den Buchstaben "f");

vor der Eingabe des zweiten Zeichens (in diesem Fall "1") lassen Sie die CONTROL-Taste jedoch wieder los. Ein weiteres Beispiel: Bei der Anweisung "...betätigen Sie die Taste →" geben Sie die Sequenz CTRL-r ein, wenn es auf Ihrer Tastatur keine Taste gibt, die mit einem nach rechts zeigenden Pfeil beschriftet ist (bzw. diese Taste nicht funktioniert).

Bild 5-3: Benannte Tasten und ihre Ersatztasten

Benannte Taste	Ersatztaste	Benannte Taste	Ersatztaste
BACKSPACE	CTRL-h	INSERT-CHAR	CTRL-a
BACKTAB	CTRL-t	INSERT-LINE	CTRL-o
BEG	CTRL-b	←	CTRL-l
CLEAR	CTRL-y	NEXT	CTRL-n
CLEAR-EOL oder CLEAR-LINE	CTRL-f y	NEXTPAGE oder PAGE-DOWN	CTRL-w
CMD LINE	CTRL-j oder CTRL-f c	PAGE-UP oder PREVPAGE	CTRL-v
DEL oder DEL-CHAR	CTRL-x	PREV	CTRL-p
DEL-LINE	CTRL-k	↑	CTRL-u
↓	CTRL-d	RESET	CTRL-f r
END	CTRL-e	→	CTRL-r
ENTER	CTRL-m	SCROLL-DOWN	CTRL-f d
F1 bis F8	CTRL-f 1 bis CTRL-f 8	SCROLL-UP	CTRL-f u
HOME	CTRL-f b	SPACEBAR	Nicht vorhanden
HOME-DOWN	CTRL-f e	TAB	CTRL-i

Hinweis: Auf manchen Tastaturen ist die Wagenrücklauf-Taste auch mit ENTER oder RETURN beschriftet. In diesem Kapitel wird mit ENTER auf die Wagenrücklauf-Taste Bezug genommen. ENTER kann auch der Funktionstaste mit dem Bildschirm-Titel F3 zugeordnet sein. Wenn es auf Ihrer Tastatur jedoch eine Taste RETURN gibt, so sollten Sie stattdessen sie oder CTRL-m benutzen.

Benutzung eines Menüs

Funktionstasten-Titel in Menüs

Die Bildschirm-Titel in der untersten Zeile des FACE-Bildschirms sind den acht Funktionstasten [F1] bis [F8] auf Ihrer Tastatur zugeordnet. Ist der aktuell aktive Ausschnitt (wie in unserem Fall) ein Menü, so werden in der untersten Zeile des Bildschirms wie in Bild 5-4 die Funktionstasten auf der höchsten Ebene angezeigt. Handelt es sich beim aktuell aktiven Ausschnitt dagegen um einen Katalog, so wechselt der Titel für [F8] von keiner Beschriftung zu [CHG-KEYS]; Sie können jetzt auf die Funktionstasten auf der zweiten Ebene (die in Bild 5-4 ebenfalls gezeigt wird) zugreifen (weitere Beispiele hierzu finden Sie im Abschnitt "Benutzung des FACE-Office" an einer späteren Stelle dieses Kapitels).

Hinweis: Wenn Sie an dem Terminal AT&T 5620 oder AT&T 630 mit Shell-Fenstern arbeiten, so werden die Funktionen, mit denen die Funktionstasten belegt sind, nicht geladen; in diesem Fall müssen die Ersatztasten benutzt werden (siehe "Ersatztasten" in diesem Abschnitt). Auf einigen Terminals gibt es programmierbare Funktionstasten, die Sie mit FACE-Kommandos belegen können. Schlagen Sie die entsprechenden Informationen in der Dokumentation Ihres Terminals nach.

Bild 5-4: Funktionstasten in einem Menü

Erste Ebene		Zweite Ebene	
[F1]	[HELP]	[F1]	[HELP]
[F2]	Ohne Titel	[F2]	[COPY]
[F3]	[ENTER]	[F3]	[MOVE]
[F4]	[PREV-FRM]	[F4]	[DELETE]
[F5]	[NEXT-FRM]	[F5]	[RENAME]
[F6]	[CANCEL]	[F6]	[CREATE]
[F7]	[CMD-MENU]	[F7]	[SECURITY]
[F8]	Ohne Titel*	[F8]	Ohne Titel*

* Die Funktionstaste F8 ist mit CHG-KEYS beschriftet, wenn in einem Menü der Inhalt eines Katalogs angezeigt wird.

Steuerung des Cursors in einem Menü

Unter FACE gibt es zwei Einrichtungen, mit denen die aktuelle Option innerhalb eines Menüs gekennzeichnet wird. Zum einen wird links neben der Menüoption ein Symbol dargestellt. Auf einigen Terminals ist das Symbol zur Positionsangabe ein Größer-als-Zeichen, (> auf anderen ein Unterstrich (_). Es gibt aber auch Terminals, auf denen andere Symbole benutzt werden.

Die zweite Methode ist nur dann sichtbar, wenn Ihr Terminal die Umkehranzeige unterstützt. Als Umkehranzeige wird die Darstellungsmöglichkeit bezeichnet, mit der auf Terminals mit Monochrom-Darstellung Teile des Bildschirms in der jeweils anderen Farbkombination angezeigt werden (werden die Zeichen z.B. normalerweise in Weiß auf schwarzem Grund dargestellt, so erscheinen die Zeichen in einigen Bereichen in Schwarz auf weißem Grund). Man spricht auch von markierten Bereichen. Ist diese Möglichkeit auf Ihrem Terminal vorhanden, so wird die aktuell markierte Option in Umkehranzeige dargestellt.

Wie wird die aktuelle Option im Menü AT&T FACE auf Ihrem Bildschirm gekennzeichnet (die Kennzeichnung wird auch "Cursor" genannt)? In den Abbildungen dieses Kapitels, in denen Bildschirmanzeigen dargestellt werden, ist die aktuelle Position des Cursors auf dem Bildschirm durch ein Größer-als-Zeichen (> gekennzeichnet.

Die Steuerung des Cursors innerhalb eines Ausschnitts erfolgt in allen FACE-Menüs über dieselben Tasten.

Hinweis: Wenn Ihre Funktionstasten und beschrifteten Tasten nicht wie angegeben funktionieren, so benutzen Sie stattdessen die Ersatztasten.

Bild 5-5: Steuerung des Cursors in einem Menü

Benannte Taste	Ersatz-Taste	Funktion in einem Menü
↓	CTRL-d	Der Cursor wird um eine Option nach unten bewegt. Wenn er in der aktuellen Spalte bereits die letzte Option erreicht hat, springt er zur obersten Option in der nächsten Spalte. Wenn es lediglich eine Spalte gibt oder der Cursor sich bereits in der letzten Spalte befindet, so springt er zur obersten Option in der ersten Spalte.
↑	CTRL-u	Der Cursor wird um eine Option nach oben bewegt. Wenn er in der aktuellen Spalte bereits die oberste Option erreicht hat, springt er zur untersten Option der in vorangegangenen Spalte. Befindet sich der Cursor bei der ersten Menüoption, so springt er zur letzten Option in der letzten Spalte.
→ oder LEERTASTE	CTRL-r	In einem einspaltigen Menü wird der Cursor um eine Option nach unten bewegt, in einem mehrspaltigen Menü um eine Option nach rechts. In einem mehrspaltigen Menü springt er nicht von der äußerst rechten Option zur äußerst linken.
← oder BACKSPACE	CTRL-l	In einem einspaltigen Menü wird der Cursor um eine Option nach oben bewegt, in einem mehrspaltigen Menü um eine Option nach links. In mehrspaltigen Menüs springt er nicht von der äußerst linken Option zur äußerst rechten.

Bild 5-5: Steuerung des Cursors in einem Menü (Fortsetzung)

NEXT	CTRL-n	Die Funktion dieser Taste entspricht derjenigen von →, nur springt der Cursor zur ersten Option innerhalb der jeweiligen Zeile bzw. Spalte, wenn er die letzte Option erreicht hat.

Benannte Taste	Ersatztaste	Funktion in einem Menü
PREV	CTRL-p	Die Funktion dieser Taste entspricht der von ←, nur springt der Cursor zur letzten Option innerhalb der jeweiligen Zeile bzw. Spalte, wenn er die erste Option erreicht hat.
HOME	CTRL-f b	Der Cursor springt zur ersten der Optionen, die aktuell im Menü sichtbar sind.
HOME-DOWN	CTRL-f e	Der Cursor springt zur letzten Option in der ersten Spalte bzw. auf der ersten Seite des Menüs.

Passen nicht alle Optionen eines Menüs auf den Bildschirm, so kann der Ausschnitt verschoben werden. Ein verschiebbarer Ausschnitt ist an der Verschiebeleiste am rechten Bildschirmrand zu erkennen. An den Verschiebesymbolen in der Verschiebeleiste können Sie die Richtung erkennen, in der der Ausschnitt verschoben werden kann.

Hinweis: Die Verschiebesymbole können auf den unterschiedlichen Terminals unterschiedlich aussehen. Auf einigen Terminals wird ein Ausschnitt, der nach oben verschoben werden kann, durch ein Zirkumflex (`^`) kenntlich gemacht, ein Ausschnitt, der nach unten verschoben werden kann, durch den Kleinbuchstaben `v`.

In einem Menü, das verschoben werden kann, können zusätzlich die folgenden benannten Tasten benutzt werden:

Bild 5-6: Zusätzliche Tasten zur Steuerung des Cursors in einem verschiebbaren Menü

Benannte Taste	Ersatztaste	Funktion in einem Menü
PAGE-DOWN	CTRL-w	Der Cursor wird zur ersten Option auf der nächsten Options-Seite bewegt und zeigt diese Options-Seite an, vorausgesetzt, es gibt noch eine volle Seite mit Menüoptionen.
PAGE-UP	CTRL-v	Der Cursor wird zur letzten Option auf der vorhergehenden Options-Seite bewegt und zeigt diese Options-Seite an, vorausgesetzt, es gibt noch eine volle Seite mit Menüoptionen.
BEG	CTRL-b	Der Cursor wird zur ersten Option innerhalb des Menüs bewegt und zeigt die erste Seite an, unabhängig davon, ob die erste Option aktuell sichtbar ist oder nicht.
END	CTRL-e	Der Cursor wird zur letzten Option innerhalb des Menüs bewegt und zeigt die letzte Seite an, unabhängig davon, ob die letzte Option aktuell sichtbar ist oder nicht,.
SCROLL-DOWN	CTRL-f d	Der Menü-Inhalt wird um eine Zeile nach unten verschoben.
SCROLL-UP	CTRL-f u	Der Menü-Inhalt wird um eine Zeile nach oben verschoben.

Da es im Menü AT&T FACE lediglich eine Optionsspalte gibt, können viele dieser Tasten im Augenblick nicht an Beispielen erläutert werden; Sie können aber versuchen, den Cursor über die Cursor (bzw. Ersatz)-Tasten zu verschiedenen Optionen innerhalb der Liste zu bewegen.

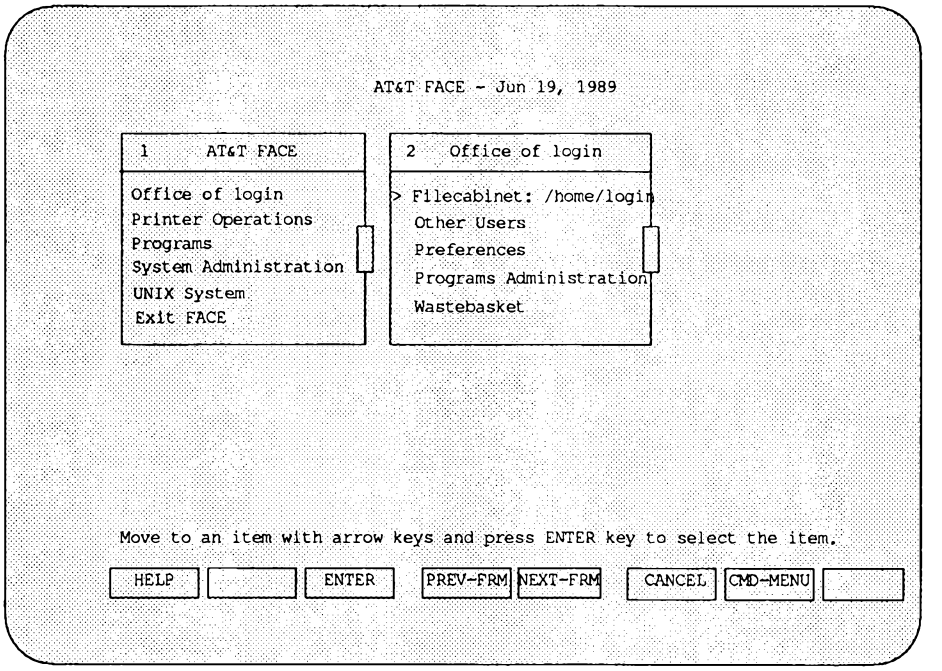
Optionswahl in einem Menü

Im folgenden werden Ihnen anhand kurzer Übungen zwei Methoden zur Optionswahl in einem Menü vorgestellt. Probieren Sie beide Methoden aus; suchen Sie sich diejenige aus, die Ihnen besser liegt und benutzen Sie sie beim Nachvollziehen der Übungen und beim Arbeiten mit FACE.

Cursor zu der Option bewegen und Taste ENTER betätigen

Zur Wahl einer Menüoption bewegen Sie einfach den Cursor zu der betreffenden Option und betätigen die Taste [ENTER].

1. Betätigen Sie so oft die Taste [↓] (bzw. [CTRL-d]), bis sich der Cursor bei der Menüoption `Office of login` befindet (statt `login` steht bei Ihnen Ihr Benutzername).
2. Betätigen Sie die Taste [ENTER], um diese Option auszuwählen (aufzurufen). Auf dem Bildschirm wird jetzt der Ausschnitt Office of login angezeigt.



Hinweis: Statt /home kann bei Ihnen ein anderer Name angezeigt werden, wenn Ihr Systemverwalter andere Verzeichnisse als Home-Verzeichnisse eingerichtet hat.

3. Betätigen Sie so oft die Taste ↓ (bzw. CTRL-d), bis sich der Cursor bei der Option `Filecabinet` befindet; dann betätigen Sie die Taste ENTER, um die Option auszuwählen (aufzurufen).

   Nach der Wahl von `Filecabinet` erscheint im Arbeitsbereich der Ausschnitt /home/login .

4. Betätigen Sie die Taste CANCEL (bzw. CTRL-f 6), um den Ausschnitt /home/login zu schließen.

   Dadurch wird der Ausschnitt vom Bildschirm gelöscht, und Sie kehren wieder in das Menü Office of login zurück.

Anfangsbuchstaben des Optionsnamens eingeben

Bei der zweiten Möglichkeit zur Auswahl einer Option in einem Menü geben Sie einfach deren Namen ein.

1. Geben Sie den Buchstaben `o` ein (auch die Großschreibweise ist möglich).

   Der Cursor wird sofort zur Option `Other Users` bewegt.

2. Geben Sie jetzt den Buchstaben `w` ein.

   Der Cursor springt nicht zur Option `Wastebasket`. FACE kann nicht erkennen, daß Sie jetzt nach einer anderen Option suchen - da es den von Ihnen eingegebenen Buchstaben `o` zwischengespeichert hat, versucht es nun, eine Menüoption zu finden, die mit den Buchstaben `ow` beginnt. Wenn es keine derartige Menüoption gibt, zeigt es in der Meldungszeile eine entsprechende Meldung an.

3. Über die Taste [↓] (bzw. [CTRL-d]) oder eine der übrigen Cursor-Tasten können Sie einen oder mehrere der eingegebenen Buchstaben löschen.

   Wenn Sie bei der Eingabe einen Fehler machen oder sich entschlossen haben, nach einer anderen Option zu suchen, so müssen Sie die bereits eingegebenen Buchstaben über eine Cursor-Taste löschen; erst dann können Sie nach dieser Methode eine andere Menüoption auswählen.

4. Geben Sie erneut `o` ein und betätigen Sie die Taste [ENTER]. Diesmal haben Sie im Menü Office of login die Option `Other Users` gewählt; auf dem Bildschirm wird daher das Menü Other Users als aktiver Ausschnitt angezeigt. Im Menü Other Users sind die Benutzernamen anderer Benutzer aufgeführt, die ebenfalls zum Zugriff auf Ihr Rechnersystem berechtigt sind.

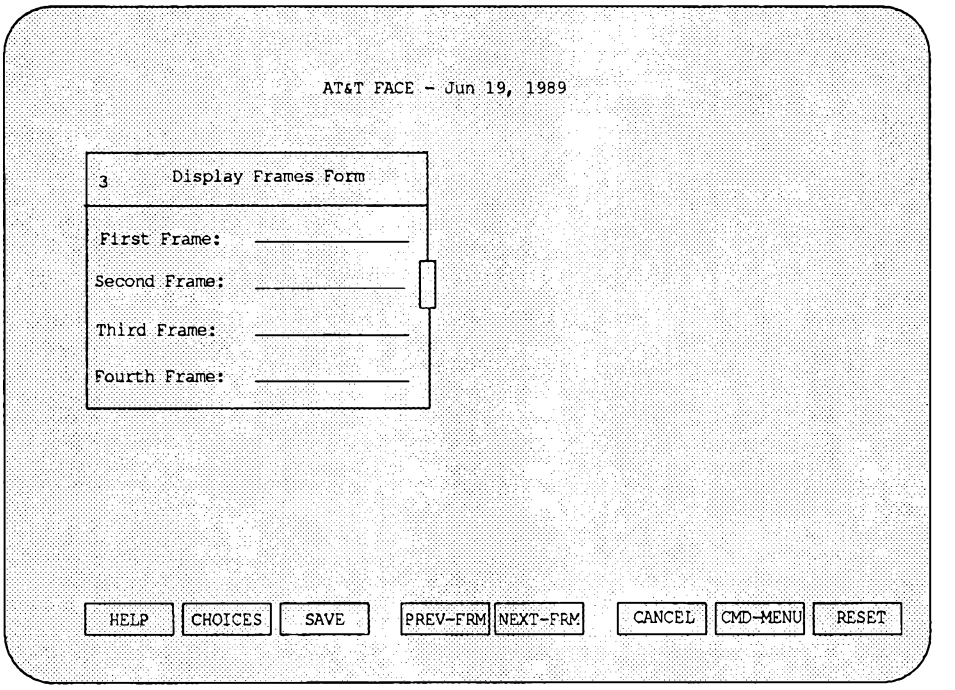
5. Betätigen Sie die Taste [CANCEL], um das Menü Other Users zu schließen.

Wenn Sie in den übrigen Abschnitten dieser Einführung zur Wahl einer Option aufgefordert werden, so verwenden Sie die Methode, die Ihnen am sympathischsten ist.

Arbeiten mit Schablonen

Der zweite Ausschnitt-Typ, mit dem Sie es unter FACE zu tun haben, ist die sogenannte Schablone (form). Eine Schablone sieht wie ein leerer Fragebogen aus. Bild 5-7 zeigt eine der Schablonen von FACE, die Schablone Display Frames Form

Bild 5-7: Die Schablone Display Frames Form



Anhand der Informationen, die Sie in die Felder einer Schablone eintragen, wird von FACE das Aussehen Ihres FACE-Office festgelegt bzw. die Funktionsweise eines FACE-Kommandos geändert. In Bild 5-7 können Sie folgende Felder ausfüllen: `First Frame`, `Second Frame`, `Third Frame` und `Fourth Frame` (das Ausfüllen dieser Schablone wird im Abschnitt "Ändern von weiteren Standardparametern" beschrieben.)

Bei Schablonen ist außerdem zu beachten, daß die meisten Felder bereits Standardwerte enthalten (ein Standardwert wird von FACE automatisch dem betreffenden Feld zugeordnet, es sei denn, *Sie* haben einen anderen Wert dafür angegeben.)

Funktionstasten-Titel in Schablonen

In Schablonen gibt es andere Funktionstasten als in Menüs (siehe Bild 5-8). Da in den Schablonen lediglich acht Funktionen aufgerufen werden können, muß FACE die Funktionstaste [F8] nicht wie in den Menüs zum Umschalten zwischen den verschiedenen Funktionstasten-Ebenen reservieren. Die Änderung der Funktionstasten-Titel werden Sie feststellen können, wenn Sie später in diesem Abschnitt die Cursor-Steuerung in und das Editieren von einer Schablone üben werden.

Bild 5-8: Funktionstasten in einer Schablone

Funktionstaste	
[F1]	[HELP]
[F2]	[CHOICES]
[F3]	[SAVE]
[F4]	[PREV-FRM]
[F5]	[NEXT-FRM]
[F6]	[CANCEL]
[F7]	[CMD-MENU]
[F8]	[RESET]

Bei den drei Funktionstasten, die in Schablonen anders belegt sind als in Menüs, handelt es sich um [CHOICES], [SAVE] und [RESET]. Über die Funktionstaste [CHOICES] können Sie die Werte abrufen, die für das aktuelle Feld zur Wahl stehen (die Meldung `No choices available` in der Meldungszeile signalisiert Ihnen, daß Sie einen gültigen Wert in das Feld eintragen müssen). Über die Funktionstaste [RESET] wird das aktuelle Feld wieder auf den Wert zurückgesetzt, der in ihm vor der letzten Änderung enthalten war. Über die Funktionstaste [SAVE] können Sie die Werte abspeichern, die Sie in alle Felder eingetragen haben; gleichzeitig wird die Schablone geschlossen.

Steuerung des Cursors in und Editieren einer Schablone

Da Sie in einer Schablone sowohl neue Informationen eingeben bzw. bereits vorhandene Informationen abändern als auch den Cursor in die verschiedenen Felder bewegen können, benötigen Sie Möglichkeiten zum Editieren und zur Steuerung des Cursors. Daher sind einige der benannten Tasten in einer Schablone anders belegt als in einem Menü. Bild 5-9 zeigt die Belegung der Tasten in einer Schablone.

Wie bereits erwähnt, können Sie auf Ihrer Tastatur stattdessen auch Ersatztasten benutzen, wenn die angegebenen Tasten nicht vorhanden sind bzw. nicht funktionieren.

Bild 5-9: Steuerung des Cursors in einer Schablone

Benannte Taste	Ersatz-Tasten	Funktion in einer Schablone
↓	CTRL-d	Der Cursor wird um ein Feld nach unten bewegt. Vom letzten Feld aus springt er ins oberste.
↑	CTRL-u	Der Cursor wird um ein Feld nach oben bewegt. Vom obersten Feld aus springt er ins letzte.
→	CTRL-r	Der Cursor wird innerhalb eines Felds um ein Zeichen nach rechts bewegt (das Zeichen wird nicht gelöscht). Von der letzten Zeichenposition aus springt er nicht in das nächste Feld.
←	CTRL-l	Der Cursor wird innerhalb eines Felds um ein Zeichen nach links bewegt (das Zeichen wird nicht gelöscht). Von der ersten Zeichenposition aus springt er nicht in das vorhergehende Feld.
TAB	CTRL-i	Der Cursor wird in das nächste Feld der Schablone bewegt. Im letzten Feld verhält er sich wie bei der Betätigung der Taste DA.
BACKTAB	CTRL-t	Der Cursor wird zum vorhergehenden Ausschnitt bewegt. Im obersten Feld verhält er sich wie bei der Betätigung der Taste UA.

Bild 3-9: Steuerung des Cursors in einer Schablone (Fortsetzung)

Benannte Taste	Ersatz-tasten	Funktion in einer Schablone
HOME BEG	CTRL-f b CTRL-b	Der Cursor wird zum zum ersten Zeichen des aktuellen Felds bewegt.
HOME-DOWN END	CTRL-f e CTRL-e	Der Cursor wird zum letzten Zeichen des aktuellen Felds bewegt.
BACKSPACE	CTRL-h	Der Cursor wird nach links bewegt; das Zeichen an dieser Zeichenposition wird gelöscht.
LEERTASTE	Nicht vorhanden	Das Zeichen an der aktuellen Cursor-Position wird durch ein Leerzeichen überschrieben; der Cursor wird um ein Zeichen nach rechts bewegt.
DEL oder DELETE-CHAR	CTRL-x	Das Zeichen an der aktuellen Cursor-Position wird gelöscht; die entstandene Lücke wird geschlossen.
DELETE-LINE	CTRL-k	Die aktuelle Zeile eines Felds wird gelöscht; die entstandene Lücke wird geschlossen. In einem einzeiligen Feld hat diese Taste dieselbe Funktion wie CLEAR-LINE.

Bild 3-9: Steuerung des Cursors in einer Schablone (Fortsetzung)

Benannte Taste	Ersatz-tasten	Funktion in einer Schablone
RESET	CTRL-f r	Ein Feld wird auf seinen ursprünglichen Wert zurückgesetzt.
CLEAR-EOL	CTRL-f y	Die Zeichen zwischen der aktuellen Cursor-Position und dem Zeilenende werden gelöscht.
CLEAR CLEAR-LINE	CTRL-y	Die aktuelle Zeile des aktuellen Felds wird gelöscht.

Die folgende Übung zeigt Ihnen, wie Sie die Schablone Office Functions aufrufen und Feldwerte darin ändern können. Beachten Sie stets, daß Sie bei der Anweisung "Wählen Sie ..." den Cursor zu einer Menüoption bewegen und die Taste ENTER betätigen müssen.

Hinweis: Hat der Cursor auf Ihrem Terminal die Form eines Unterstrichs (_), so werden Sie ihn beim Arbeiten in einer Schablone nur schwer erkennen können, da die Felder standardmäßig unterstrichen sind.

Hinweis: Wenn Sie einen neuen Wert in ein Feld eintragen möchten, so wird das gesamte Feld bei der Eingabe des ersten Zeichens gelöscht. Handelt es sich beim ersten Zeichen um ein Leerzeichen, so scheint das Feld leer zu sein - das sieht aber nur so aus. In Wirklichkeit enthält die Zeile an ihrer ersten Position ein Leerzeichen, und der Cursor wird zur zweiten Zeichenposition dieser Zeile bewegt. Wenn Sie dieses erste Leerzeichen versehentlich nicht löschen, so sieht Ihr Eintrag äußerlich zwar korrekt aus, funktioniert aber in den meisten Fällen nicht wie erwartet.

1. Wählen Sie im Menü Office of login die Option `Preferences`.

   Auf dem Bildschirm wird das Menü Preferences mit seinen vier Optionen angezeigt (auf diese Optionen kommen wir an einer späteren Stelle dieses Kapitels noch ausführlich zurück).

2. Wählen Sie im Menü Preferences die Option `Office Functions`.

   Auf dem Bildschirm wird die Schablone Office Functions mit ihren sieben Feldern angezeigt; der Cursor befindet sich im (ersten) Feld `Delete objects in my Wastebasket after (# of days)`, das standardmäßig den Wert `1` hat.

   Beachten Sie, daß in der untersten Zeile des Bildschirms jetzt andere Funktionstasten-Titel angezeigt werden. Der Meldungszeile können Sie die Werte entnehmen, die für dieses Feld zulässig sind.

3. Tragen Sie in dieses Feld den Wert `32` ein und betätigen Sie die Taste [ENTER].

   Beachten Sie, daß in der Meldungszeile jetzt `Input is not valid` angezeigt wird und der Cursor nicht in das nächste Feld bewegt worden ist. Wenn FACE erkennen kann, daß Sie in einem Feld einen ungültigen Wert (in diesem Fall `32`) einzutragen versuchen, so können Sie das Feld erst dann verlassen, nachdem Sie einen zulässigen Wert eingetragen haben.

   Probieren Sie an diesem Feld einige der Editier-Tasten für Schablonen aus, die in Bild 5-10 aufgeführt sind (z.B. [BACKSPACE] oder [DELETE-CHAR]).

4. Betätigen Sie die Taste [RESET] (bzw. [CTRL-f] [8]), um den ursprünglichen Wert dieses Felds wiederherzustellen. Die Betätigung der Taste

   [RESET] wirkt sich nur auf das aktuelle Feld aus.

5. Bewegen Sie den Cursor zum Feld `Prompt before deleting ...` (hier wird der Standardwert `yes` angezeigt), und betätigen Sie die Taste [CHOICES] (bzw. [CTRL-f] [2]). Im Feld wird jetzt der Wert `no` angezeigt.

6. Betätigen Sie erneut die Taste [CHOICES]; das Feld enthält jetzt wieder den Wert `yes`.

7. Betätigen Sie die Taste [↓], um den Cursor zum Feld `Folder Display Format:` zu bewegen. Dieses Feld enthält den Standardwert `Name only`.

9. Betätigen Sie erneut die Taste CHOICES (bzw. CTRL-f 2).

   Auf dem Bildschirm wird jetzt das Menü Choices angezeigt. Bei mehrmaliger Betätigung der Taste CHOICES (bzw. CTRL-f 2) können Sie sämtliche zulässigen Werte innerhalb des Felds hintereinander abrufen, vorausgesetzt, es gibt weniger als vier zulässige Werte. Vier oder mehr zulässige Werte werden im Menü Choices angezeigt.

   Das Menü Choices unterscheidet sich von anderen Menüs in einer Reihe von Punkten. Zum einen gibt es nur zwei Funktionstasten, nämlich ENTER und CANCEL. Zum anderen wird bei der Wahl einer Option im Menü Choices nichts aufgerufen; die gewählte Option wird einfach in das Feld der Schablone eingetragen.

10. Betätigen Sie die Funktionstaste ENTER (bzw. CTRL-f 3), um im Menü Choices eine beliebige Option auszuwählen (eine andere Möglichkeit zur Wahl einer Option im Menü Choices ist die Taste RETURN bzw. CTRL-m).

    Das Menü Choices wird ausgeblendet, und die ausgewählte Option wird in das Feld `Folder Display Format:` eingetragen.

11. Betätigen Sie die Taste CANCEL (bzw. CTRL-f 6), um sämtliche Änderungen, die Sie in *allen* Feldern der Schablone Office Functions vorgenommen haben, zu "stornieren"; gleichzeitig wird die Schablone geschlossen und wieder das Menü Preferences aufgerufen.

12. Betätigen Sie die Taste CTRL-j, um den Cursor in die Kommandozeile zu bewegen, geben Sie `cancel` ein und betätigen Sie die Taste ENTER.

    Dadurch wird das Menü Preferences geschlossen und wieder das Menü Office of login dargestellt.

    Die Eingabe des Kommandos `cancel` in der Kommandozeile hat dieselbe Auswirkung wie die Betätigung der Funktionstaste CANCEL (bzw. CTRL-f 6). Tatsächlich führt FACE bei Ihrer Betätigung der Taste CANCEL, einfach das Kommando `cancel` für Sie aus. D. h., es schließt Ihren aktuellen Ausschnitt und blendet ihn aus dem Arbeitsbereich aus.

13. Betätigen Sie die Taste CANCEL (bzw. CTRL-f 6), um den Ausschnitt Office of login zu schließen und zum Menü AT&T FACE zurückzukehren.

Umschalten zwischen Ausschnitten

Um von einem Ausschnitt zu einem anderen umzuschalten, gibt es zwei Möglichkeiten:

- Wenn Sie eine Option auswählen, so schalten Sie automatisch zu dem Ausschnitt um, in dem diese Option enthalten ist (es sei denn, Sie befinden sich im Menü Choices).
- Beim Schließen eines Ausschnitts wird der zuvor aktive Ausschnitt wieder aktiviert (es sei denn, Sie befinden sich im Menü Choices).

Damit Sie das Umschalten zwischen verschiedenen Ausschnitten ohne gleichzeitiges Schließen des aktuellen Ausschnitts üben können, müssen Sie mehrere Ausschnitte auf Ihrem FACE-Bildschirm eröffnen. Im folgenden finden Sie eine Anleitung zum Eröffnen der Ausschnitte Office of login, /home/login und Wastebasket. Nachdem Sie die Ausschnitte eröffnet haben, können Sie zwischen den verschiedenen Ausschnitten umschalten, ohne einen der Ausschnitte zu schließen.

1. Wählen Sie im Menü AT&T FACE die Option `Office of login`.
2. Wählen Sie im Menü Office of login die Option `Filecabinet`.
3. Wählen Sie im Menü /home/login die Option `WASTEBASKET`.

Es gibt die unterschiedlichsten Situationen, in denen das Umschalten in einen anderen Ausschnitt ohne vorheriges Schließen des aktuell aktiven Ausschnitts sinnvoll ist. Vielleicht möchten Sie beispielsweise das Menü Other Users abrufen, um sich einen Überblick über die Benutzernamen zu verschaffen. Zur Eröffnung des Menüs Other Users müssen Sie eigentlich wieder zum Menü Office of login zurückschalten - eventuell möchten Sie aber den Ausschnitt WASTEBASKET nicht gleichzeitig schließen.

Umschalten zwischen Ausschnitten von der Kommandozeile aus

Bei dieser Methode zum Umschalten zwischen Ausschnitten werden die Ausschnitt-Nummern benutzt, die links neben dem Titel jedes eröffneten Ausschnitts angezeigt werden. Jeder eröffnete Ausschnitt hat eine eigene Nummer; das Menü AT&T FACE hat stets die Nummer `1`.

1. Betätigen Sie die Taste CTRL-j, um den Cursor in die Kommandozeile zu bewegen.
2. Geben Sie die Zahl `2` ein und betätigen Sie die Taste ENTER, um zu Ausschnitt 2, also zum Menü Office of login, zurückzuschalten.
3. Wählen Sie die Option `Other Users`.

   Auf dem Bildschirm wird das Menü Other Users eröffnet, in dem Sie beispielsweise nach einem bestimmten Benutzername suchen können. Auch im Menü Other Users steht zur Linken des Titels seine Nummer.
4. Wenn Sie das Menü Other Users nicht mehr benötigen, betätigen Sie die Taste CTRL-j, geben `cancel` ein und betätigen dann die Taste ENTER, um das Menü zu schließen.

   Der Cursor befindet sich jetzt wieder im ursprünglich aktiven Ausschnitt, dem Menü Office of login.

Mit dem Kommando `goto` können Sie ebenfalls zu einem Ausschnitt umschalten, indem Sie dessen Ausschnitt-Nummer angeben (nähere Informationen hierzu finden Sie in Anhang C).

Umschalten zwischen Ausschnitten über Funktionstasten

In den meisten Ausschnitten, die Sie auf dem FACE-Bildschirm abrufen können, sind die Funktionstasten F4 und F5 mit PREV-FRM bzw. NEXT-FRM belegt. Wenn Sie diese Funktionstasten betätigen, springt der Cursor von einem Ausschnitt zum anderen. Der Ausschnitt, in den Sie mit dem Cursor springen, wird der aktive Ausschnitt; der ursprüngliche Ausschnitt wird inaktiv.

Im Augenblick sollte das Menü Office of login bei Ihnen der aktive Ausschnitt sein.

1. Schalten Sie über die Taste PREV-FRM (bzw. CTRL-f 4) zu dem Ausschnitt zurück, der vor dem Aufrufen des Menüs Office of login aktiv war.

   Der Cursor springt zum Menü AT&T FACE zurück.
2. Betätigen Sie die Taste NEXT-FRM (bzw. CTRL-f 5), um zum Menü Office of login zurückzuspringen.

3. Betätigen Sie die Taste CANCEL (bzw. CTRL-f 6), um das Menü Office of login zu schließen.

   Das Menü AT&T FACE sollte jetzt wieder der aktive Ausschnitt sein.

Umschalten zwischen Ausschnitten über das Kommando `frm-mgmt`

Obwohl das Command Menu im nächsten Abschnitt ausführlich behandelt wird, wird bereits an dieser Stelle eines der hier enthaltenen Kommandos beschrieben, das Ihnen das Umschalten zwischen verschiedenen Ausschnitten ohne gleichzeitiges Schließen des aktuell aktiven Ausschnitten ermöglicht. Vollziehen Sie die Übung vorerst genau wie angegeben nach.

1. Wählen Sie im Menü AT&T FACE die Option `Office of login`.
2. Betätigen Sie die Taste CMD-MENU (bzw. CTRL-f 7), um das Command Menu aufzurufen.

   Bei der Optionswahl im Command Menu gehen Sie wie in allen anderen Menüs vor. Allerdings enthält das Command Menu keine Optionen zum Aufrufen von Menüs oder Schablonen, sondern FACE-Kommandos. Zum Umschalten zwischen Ausschnitten gibt es das Kommando `frm-mgmt`.
3. Wählen Sie im Command Menu die Option `frm-mgmt`.

   Das Command Menu wird ausgeblendet und durch das Menü Frame Management abgelöst, in dem die Option `list` enthalten ist.
4. Wählen Sie die Option `list`. Auf dem Bildschirm wird das Menü Open Frames mit einer Übersicht über sämtliche aktuell eröffneten Ausschnitte angezeigt.
5. Wählen Sie die Option `Office of login`.

   Das Menü Open Frames wird ausgeblendet, und der Ausschnitt Office of login wird aktiv.
6. Betätigen Sie die Taste CANCEL (bzw. CTRL-f 6), um den Ausschnitt Office of login zu schließen.

   Das Menü AT&T FACE ist jetzt wieder der aktive Ausschnitt.

Bestimmt ist Ihnen in der bisherigen Beschreibung ein wichtiger Unterschied zwischen dem Schließen eines Ausschnitts über die Taste CANCEL und dem Umschalten von einem Ausschnitt zum anderen nach den oben beschriebenen Methoden aufgefallen. Wenn Sie einen Ausschnitt über die Taste CANCEL schließen, wird er ausgeblendet; Sie schalten dann automatisch wieder zum zuvor aktiven Ausschnitt zurück. Auf den so geschlossenen Ausschnitt können Sie nicht mehr zugreifen, es sei denn, Sie eröffnen ihn wieder (siehe Anleitung am Anfang dieses Abschnitts). Wenn Sie dagegen nach einer der oben beschriebenen Methoden von einem eröffneten Ausschnitt zu einem anderen umschalten, wird der Ausschnitt nicht ausgeblendet, sondern lediglich inaktiv. Somit können Sie ihn jederzeit wieder reaktivieren, um darin zu arbeiten.

Aussehen Ihres FACE-Office festlegen

FACE ermittelt für jeden Ausschnitt, den Sie eröffnen, automatisch seine Größe und Position auf dem Bildschirm. Die Ausschnitte werden von FACE so angeordnet, daß sie sich nicht überlagern, wenn es nicht unbedingt notwendig ist. Außerdem paßt FACE die Größe jedes Ausschnitts an die Anzahl der aufgeführten Dateien und Kataloge an.

Mit dem Kommando `frm-mgmt` können Sie die Position und Größe eines offenen Ausschnitts ändern. Wie im letzten Abschnitt beschrieben wurde, können Sie damit auch eine Übersicht über sämtliche eröffneten Ausschnitte abrufen
und einen beliebigen Ausschnitt aktivieren (siehe Abschnitt "Umschalten zwischen Ausschnitten über das Kommando " `frm-mgmt`").

Nachdem Sie die Größe eines Menüs oder Textausschnitts geändert haben, bleibt die Änderung wirksam, *solange sich am Inhalt des Ausschnitts nichts ändert.* Nachdem Sie einen versetzten, aktualisierten oder vergrößerten/ verkleinerten Ausschnitt geschlossen oder sich abgemeldet haben, wird der Ausschnitt von FACE bei seiner nächsten Eröffnung automatisch neu positioniert und seine Größe geändert. Dabei versucht es, den Ausschnitt wieder an seine ursprüngliche Position zu setzen, es sei denn, in der Zwischenzeit sind andere Ausschnitte eröffnet worden, die dem entgegenstehen.

Versetzen eines Ausschnitts

Einige Benutzer haben für die Ausschnitte bevorzugte Positionen, z.B. die obere rechte Ecke des Arbeitsbereichs für den Ausschnitt /home/login. Bei anderen sollen die Ausschnitte so schmal sein, daß mehrere Ausschnitte gleichzeitig im Arbeitsbereich Platz finden.

In der folgenden Übung wird der Ausschnitt /home/login mit dem Kommando `frm-mgmt` im Command Menu an eine neue Position innerhalb des Arbeitsbereichs versetzt.

1. Wählen Sie im Menü AT&T FACE die Option `Office of login`.
2. Wählen Sie im Menü Office of login die Option `Filecabinet`.

   Der zu versetzende Ausschnitt (in diesem Fall /home/login) muß *vor* dem Aufrufen von `frm-mgmt` aktiviert sein.
3. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]), um das Command Menu aufzurufen.
4. Wählen Sie die Option `frm-mgmt`.
5. Wählen Sie im Menü Frame Management die Option `move`.

   Beachten Sie, daß der Cursor sich auf der oberen linken Ecke des Ausschnitts /home/login befindet, die übrigen drei Ecken blinken und in der Meldungszeile die Aufforderung zum Verschieben der Ecke an die neue gewünschte Position dargestellt wird.
6. Zeigen Sie die neue Position des oberen linken Eckpunkts über die Cursor-Tasten an und betätigen Sie die Taste [ENTER].

   Der Ausschnitt /home/login befindet sich auf Ihrem Bildschirm jetzt an einer neuen Position.
7. Schalten Sie wieder zum Menü Office of login zurück, ohne jedoch den Ausschnitt /home/login zu schließen (betätigen Sie also [PREV-FRM] bzw. [CTRL-f] [4]).

   Beachten Sie, daß der Ausschnitt /home/login an seiner neuen Position bleibt.
8. Schalten Sie über die Taste [NEXT-FRM] (bzw. [CTRL-f] [5]) zum Ausschnitt /home/login zurück und schließen Sie ihn dann über die Taste [CANCEL] (bzw. [CTRL-f] [6]).

9. Wählen Sie im Menü Office of login erneut die Option `Filecabinet`.

   Der Ausschnitt /home/login befindet sich jetzt wieder an seiner ursprünglichen Position. Wie bereits erwähnt, kann ein Ausschnitt, den Sie mit `frm-mgmt` versetzt und dann geschlossen haben, sich nach seiner erneuten Eröffnung wieder an seiner ursprünglichen Position befinden.

Größe eines Ausschnitts ändern

In dieser Übung wird die Größe des Ausschnitts /home/login mit dem Kommando `frm-mgmt` geändert, das Sie in der Kommandozeile aufrufen.

1. Sie sollten sich noch im Ausschnitt /home/login befinden; andernfalls sollten Sie zu diesem Ausschnitt umschalten.
2. Betätigen Sie die Taste CTRL-j, um den Cursor in die Kommandozeile zu bewegen. Geben Sie `frm-mgmt` ein und betätigen Sie die Taste ENTER.
3. Wählen Sie im Menü Frame Management die Option `Reshape`.
4. Wenn Sie zur Angabe der neuen Position der oberen linken Ecke aufgefordert werden, so geben Sie die gewünschte Position über die Cursor-Tasten an.
5. Betätigen Sie die Taste ENTER.

   Jetzt werden Sie zur Angabe der neuen Position der unteren rechten Ecke aufgefordert.
6. Vergrößern Sie den Ausschnitt über die Cursor-Tasten und betätigen Sie die Taste ENTER.

   Wie bereits erwähnt, wird ein Ausschnitt, den Sie vergrößert bzw. verkleinert und dann geschlossen haben, wieder an die Position zurückgesetzt, die ihm von FACE automatisch zugewiesen wird (dasselbe gilt auch, wenn Sie FACE nach der Verkleinerung bzw. Vergrößerung des Ausschnitts beendet haben). Enthält

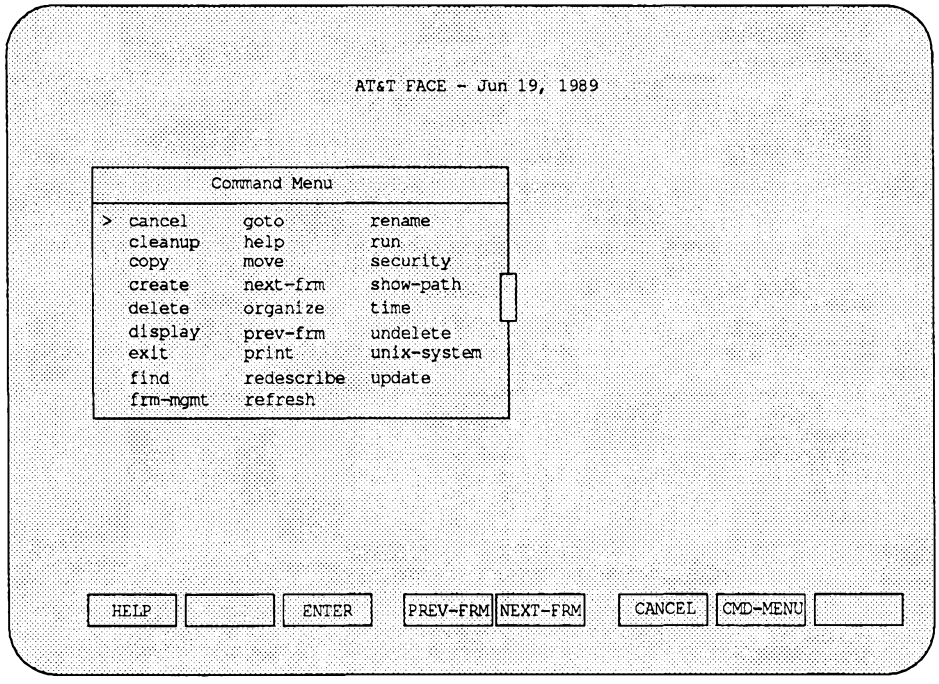
der Ausschnitt ein Katalog-Menü, so wird das Menü durch jedes Kommando zur Aktualisierung dieses Menüs (bzw. durch ein Kommando zum Anlegen einer neuen Datei) in seiner ursprünglichen Größe wiederhergestellt und an seine ursprüngliche Position zurückgesetzt.

Hinweis: Eine Verkleinerung bzw. Vergrößerung ist nur bei Menüs und Textausschnitten möglich, nicht aber bei Schablonen.

Das Command Menu

Das Command Menu (siehe Bild 5-11) enthält sämtliche FACE-Kommandos, die von Ihnen aufgerufen werden können. Um dieses Menü aufzurufen, betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]) (sämtliche Kommandos können auch direkt von der Kommandozeile aus aufgerufen werden, indem die Taste [CTRL-j] betätigt, der Kommandoname eingegeben und dann die Taste [ENTER] betätigt wird. Das Command Menu stellt lediglich eine Art Hilfestellung dar).

Bild 5-10: Das Command Menu



In dieser Übung werden Sie lediglich mit einem kleinen Teil der im Command Menu enthaltenen Kommandos arbeiten. Informationen über jedes einzelne Kommando im Command Menu finden Sie in Anhang C. Für jedes Kommando können Sie auf dem Bildschirm auch Hilfe abrufen; der Zugriff auf das Hilfesystem wird im Abschnitt "Zugriff auf das Hilfesystem" behandelt.

1. Wählen Sie im Menü AT&T FACE die Option `Office of login`.
2. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]), um das Command Menu aufzurufen.

   Jetzt können Sie einige der Cursor-Tasten für mehrspaltige Menüs ausprobieren. Beachten Sie, daß die Tasten [→] (bzw. [CTRL-r]) und [←] (bzw. [CTRL-l]) in einem mehrspaltigen Menü eine andere Funktion haben als in einem einspaltigen.

Die Vorgehensweise bei der Wahl eines Kommandos ist Ihnen bereits bekannt: Bewegen Sie den Cursor zu dem betreffenden Kommando und betätigen Sie die Taste [ENTER]. Ist bei dem Kommando die Eingabe weiterer Informationen erforderlich, so werden Sie dazu aufgefordert.

Auch die Taste [CTRL-j], mit der Sie auf die Kommandozeile zugreifen können, ist Ihnen bereits bekannt. Wenn Sie diese Taste im Command Menu betätigen, wird die aktuell durch den Cursor markierte Option hinter die Eingabeaufforderung in der Kommandozeile kopiert.

3. Markieren Sie das Kommando `time` und betätigen Sie die Taste [CTRL-j]. Das Kommando erscheint jetzt hinter der Eingabeaufforderung `-->` in der Kommandozeile. Das Command Menu wird ausgeblendet; in der Funktionstasten-Reihe sind jetzt nur noch die Funktionstasten-Titel [CANCEL] und [HELP] zu sehen. Diese Funktionstasten können in der Kommandozeile benutzt werden.

4. Betätigen Sie die Taste [CANCEL] (bzw. [CTRL-f] [6]). Die Eingabeaufforderung in der Kommandozeile wird ausgeblendet, da [CANCEL] im Augenblick nur in der Kommandozeile gültig ist. Letztendlich haben Sie das Kommando `time` abgebrochen, und der zuletzt aktive Ausschnitt, das Menü Office, wird wieder aktiv.

5. Betätigen Sie erneut die Taste [CMD-MENU] (bzw. [CTRL-f] [7]), markieren Sie das Kommando `cancel` und betätigen Sie die Taste [CTRL-j], um auf die Kommandozeile zuzugreifen.

   Das Command Menu wird ausgeblendet (geschlossen), und `cancel` wird in der Kommandozeile angezeigt.

6. Betätigen Sie die Taste [ENTER]. Über diese Taste haben Sie jetzt das Kommando `cancel` aufgerufen. In diesem Fall wirkte sich das Kommando `cancel` nur auf den aktuell aktiven Ausschnitt, das Menü Office of login, aus. Das Menü Office of login wird ausgeblendet, und das Menü AT&T FACE wird zum aktiven Ausschnitt.

Zugriff auf das Hilfesystem

Über das eingebaute Hilfesystem kann der Benutzer Informationen über das gesamte FACE-System abrufen. Im folgenden Abschnitt wird beschrieben, wie Sie Informationen über bestimmte Kommandos, Menüs, und Schablonen-Felder abrufen können. Im nachfolgenden Abschnitt geht es dann um das Abrufen von Informationen zu FACE allgemeiner Art.

Informationen über FACE-Kommandos, Menüs und Schablonen-Felder

Über das aktuell aktive Kommando, Menü oder Schablonen-Feld können Sie sehr einfach Informationen abrufen, da das Kommando `help` ständig auf die Funktionstaste [F1] (bzw. [CTRL-f] [1]) gelegt ist (vorausgesetzt, das Hilfesystem enthält Informationen zu dem betreffenden Thema).

- Ist das Command Menu der aktuelle Ausschnitt, so können Sie über die Taste [HELP] einen Ausschnitt mit Informationen über den Umgang mit dem aktuell markierten Kommando abrufen.
- Ist der aktive Ausschnitt ein Menü, so können Sie über die Taste [HELP] einen Ausschnitt mit Informationen über jede Option in diesem Menü abrufen.
- Ist der aktive Ausschnitt eine Schablone, so können Sie über die Taste [HELP] einen Ausschnitt mit Informationen über das Ausfüllen des aktuellen Felds abrufen.
- Von der Kommandozeile aus können Sie über die Taste [HELP] ein Menü mit den Kommandos sowie Informationen über das gewählte Kommando abrufen.

Die nächste Übung zeigt Ihnen, wie Sie über ein FACE-Kommando spezielle Informationen abrufen können. Nach derselben Methode können Sie auch in einem Menü oder Schablonen-Feld verfahren. Das Hilfesystem liefert Ihnen Informationen über das aktuell aktive FACE-Kommando bzw. den aktuell aktiven Ausschnitt. In dieser Übung wird erklärt, wie Sie Informationen über den Umgang mit einem bestimmten Kommando abrufen können.

1. Schalten Sie zum Ausschnitt Office of login um.
2. Betätigen Sie die Taste [CMD-MENU] bzw. [CTRL-f] [7], um das Command Menu zu eröffnen. Der Cursor wird zum Kommando `cancel` bewegt. Angenommen, Sie benötigen über die Funktion dieses Kommandos einige Informationen.
3. Betätigen Sie die Taste [HELP] (bzw. [CTRL-f] [1]), um den Help-Ausschnitt für das Kommando `cancel` zu eröffnen.

   Das Command Menu wird ausgeblendet, und der Ausschnitt Help on cancel mit einer kurzen Funktionsübersicht und Benutzerhinweisen zum Kommando `cancel` wird eröffnet.

   Über die Funktionstasten [F2] und [F3], die jetzt mit [PREVPAGE] und [NEXTPAGE] beschriftet sind, können Sie die vorhergehende bzw. nächste Seite eines Help-Ausschnitts abrufen. Die Verschiebeleiste am rechten Rand des Ausschnitts signalisiert Ihnen mit dem Symbol `v`, daß auf diese Textseite noch weitere folgen.
4. Betätigen Sie die Taste [NEXTPAGE] (bzw. [CTRL-f] [3]).

   In der Verschiebeleiste erscheint ein zweites Verschiebesymbol, nämlich der Zirkumflex (`^`). Daran können Sie erkennen, daß es vor der aktuellen Seite eine weitere gibt, auf die Sie über die Taste [PREVPAGE] (bzw. [CTRL-f] [2]) zugreifen können)
   (eine vollständige Übersicht über die Cursor-Tasten in Help- und anderen Text-Ausschnitten finden Sie in Bild 5-12).
5. Betätigen Sie die Taste [CANCEL] (bzw. [CTRL-f] [6]), um den Ausschnitt Help on cancel zu schließen.

Allgemeine Informationen zu FACE abfragen

Allgemeine Informationen zu FACE können Sie über die Taste [HELP] abrufen. Diese Funktionstaste steht im Menü Overview of Contents zur Verfügung, das wiederum über die Funktionstaste [CONTENTS] (bzw. [CTRL-f] [8]) aufgerufen wird (diese Funktionstaste ist nur dann verfügbar, wenn ein Help-Ausschnitt auf dem Bildschirm eröffnet ist). Über die Funktionstasten auf der höchsten Ebene können keine allgemeinen Informationen abgerufen werden, da diese weniger in Anspruch genommen werden als spezielle Informationen zu Kommandos oder Ausschnitten.

Die folgende Übung können Sie von einem beliebigen Ausschnitt aus starten (wenn Sie soeben den letzten Abschnitt "(Informationen über FACE-Kommandos)" durchgearbeitet haben, ist aktuell das Menü Office of login aktiv).

1. Betätigen Sie die Taste [HELP] (bzw. [CTRL-f] [1]) auf dem Bildschirm werden in einem Ausschnitt Informationen über den aktuell aktiven Ausschnitt angezeigt.

   Beachten Sie, daß die Funktionstaste [F8] jetzt den Titel [CONTENTS] hat.

2. Betätigen Sie die Taste [CONTENTS] (bzw. [CTRL-f] [8]).

   Bei Betätigung der Funktionstaste [F8] wird in jedem Fall dieser Bildschirm-Titel angezeigt, wenn Sie sich in einem Help-Ausschnitt befinden. Das jetzt aufgerufene Menü Contents enthält eine thematische Inhaltsübersicht über die allgemeinen Informationen, die Sie zu FACE abfragen können.

3. Wählen Sie im Menü Contents die Option `Frames and Function Keys`.

   Auf dem Bildschirm wird ein Ausschnitt mit dem Titel Help Facility: Frames and Function Keys eröffnet. Das Verschiebesymbol in der Verschiebeleiste zeigt Ihnen, daß auf die aktuelle Seite noch weitere Textseiten folgen.

4. Betätigen Sie die Taste [NEXTPAGE] (bzw. [CTRL-f] [3]). In der Verschiebeleiste sehen Sie jetzt an einem zweiten Verschiebesymbol, daß der aktuellen Textseite eine weitere vorausgeht. Die vorhergehende Seite können Sie über die Taste [PREVPAGE] (bzw. [CTRL-f] [2]) abrufen

   (eine Übersicht über weitere Cursor-Tasten für Help- und andere Text-Ausschnitte finden Sie in Bild 5-12 im Anschluß an diese Übung).

5. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]) und wählen Sie das Kommando `cleanup`.

   Durch das Kommando `cleanup` werden der aktuelle Ausschnitt sowie alle anderen gegebenenfalls eröffneten Ausschnitte mit Ausnahme des Ausschnitts AT&T FACE geschlossen (zum Verlassen eines Help-Ausschnitts können Sie auch das Kommando [CANCEL] benutzen).

In einem Help- oder anderen Text-Ausschnitt stehen Ihnen außer

PREVPAGE und NEXTPAGE zusätzlich die Tasten zur Steuerung des Cursors zur Verfügung, die in Bild 5-12 aufgeführt sind. Wenn einige dieser benannten Tasten auf Ihrer Tastatur nicht vorhanden sind (bzw. nicht funktionieren), so können Sie stattdessen die Ersatztasten benutzen.

Bild 5-11: Steuerung des Cursors in Help- und anderen Text-Ausschnitten

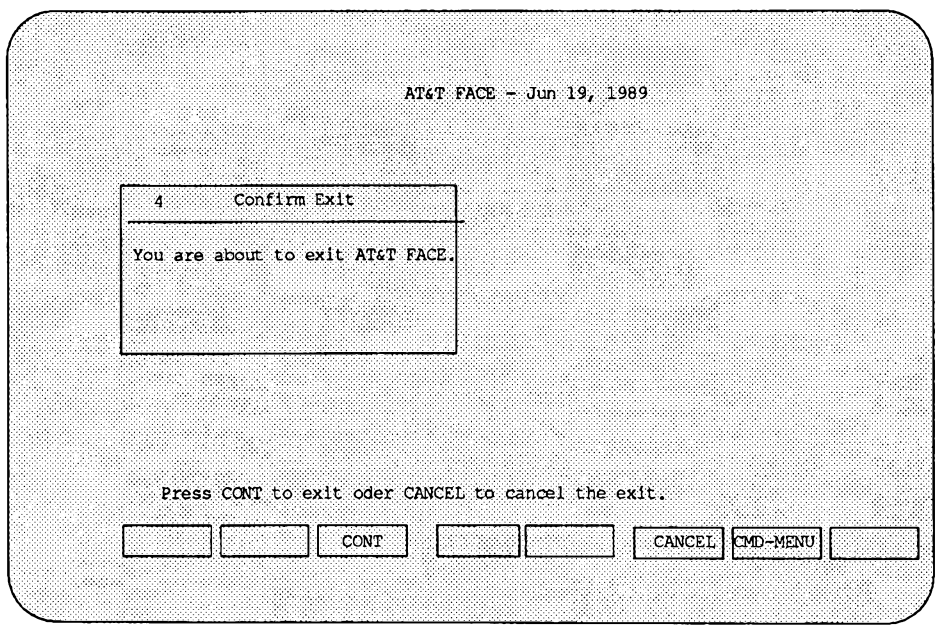
Benannte Taste	Ersatztaste	Funktion in einem Ausschnitt
↑	CTRL-u	Cursor um eine Zeile nach oben bewegen.
↓	CTRL-d	Cursor um eine Zeile nach unten bewegen.
SCROLL-DOWN	CTRL-f d	Text um eine Zeile nach unten verschieben.
SCROLL-UP	CTRL-f u	Text um eine Zeile nach oben verschieben.
PAGE-DOWN	CTRL-w	Nächsten Text-Ausschnitt anzeigen, wobei zwei Zeilen aus dem aktuellen Ausschnitt übernommen werden.
PAGE-UP	CTRL-v	Vorhergehenden Text-Ausschnitt anzeigen, wobei zwei Zeilen aus dem aktuellen Ausschnitt übernommen werden.
BEG	T{ CTRL-b	Ersten vollständig gefüllten Text-Ausschnitt anzeigen.
END	CTRL-e	Letzten vollständig gefüllten Text-Ausschnitt anzeigen.

Beenden von FACE

Die einführenden Informationen in den letzten Abschnitten ermöglichten Ihnen den Einstieg in das Arbeiten mit FACE. Um FACE zu beenden, rufen Sie im Menü AT&T FACE die Option `Exit FACE` auf. Dabei gehen Sie folgendermaßen vor:

1. Zur Beendigung von FACE ist es nicht notwendig, alle eröffneten Ausschnitte zu schließen. Benutzen Sie zum Umschalten in das Menü AT&T FACE die Methode, die Ihnen am meisten liegt.
2. Wählen Sie die Option `Exit FACE`.

   In der Meldungszeile werden Sie zur Eingabe einer Bestätigung aufgefordert, daß Sie FACE tatsächlich beenden möchten.



3. Betätigen Sie die Taste **CONT**, um Ihre Absicht, FACE zu verlassen, zu bestätigen; wenn Sie AT&T FACE doch nicht beenden möchten, so betätigen Sie die Taste **CANCEL**, um die Prozedur zu beenden.

Hinweis: Wenn FACE bei Ihrer Anmeldung automatisch aufgerufen worden ist, werden Sie vom Rechner abgemeldet; andernfalls kehren Sie zur UNIX-Shell zurück (auf dem Bildschirm wird dann die Eingabeaufforderung angezeigt). Diesen Ablauf können Sie über die Option `Office Functions` im Menü `Preferences` ändern.

Vergessen Sie auf keinen Fall, FACE zu beenden und sich an Ihrem Rechner abzumelden, wenn Sie (auch vorübergehend) nicht mehr am Rechner arbeiten möchten, um die Sicherheit sowohl Ihrer Dateien als auch Ihres gesamten Rechnersystems zu gewährleisten.

Benutzung des FACE-Office

Ihr Dateisystem

Was sind Dateien und Kataloge?

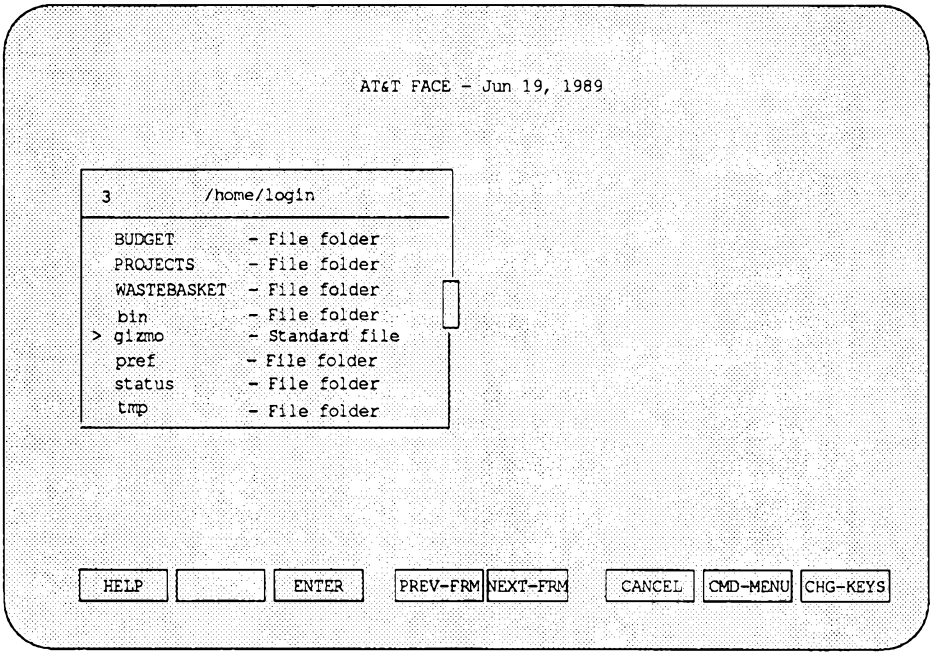
Dateien und Kataloge dienen in Ihrem FACE-Office zur Speicherung von Informationen. Die Dateien und Kataloge legen Sie im Normalfall in Ihrem eigenen Dateisystem an und bewahren sie hier auch auf.

Datei — Eine Datei ist in Ihrem FACE-Office die kleinste Einheit zur Informationsspeicherung. D. h., in einer Datei können Sie weder weitere Dateien noch Kataloge ablegen. Eine Datei ist mit einem flachen Blatt Papier vergleichbar: Auf einem Blatt Papier können Wörter, Bilder und Informationen "gespeichert" werden, nicht aber ein weiteres Blatt Papier oder ein Katalog. Eine Datei kann eine Standard-Datei sein (in die Sie Text oder Daten eingeben), eine Daten-Datei (die Binärdaten enthält), ein ausführbares Programm oder eine Datei unbekannten Formats (die unter einem anderen Betriebssystem erstellt worden ist). Dateien sind immer in Katalogen enthalten. So kann in Ihrem Dateisystem beispielsweise ein Katalog namens `NEW-PRODUCTS`, enthalten sein und eine Datei namens `gizmo` mit einem Kurzbericht über ein neues Produkt in `NEW-PRODUCTS`.

Katalog — Ein Katalog ist eine Einheit, die Dateien oder weitere Kataloge enthält. Innerhalb des Ausschnitts /home/login, der selbst ein Katalog ist, sind Ihre Informationen in weiteren Dateien und Katalogen enthalten. Zwischen den Dateien in einem Katalog besteht im Normalfall ein thematischer Zusammenhang. So kann ein Katalog namens `BUDGET` 12 weitere Kataloge namens `JAN`, `FEB` etc. enthalten, die wiederum die Dateien `food`, `transportation`, `entertainment` etc. enthalten könnten.

Die Kataloge und Dateien werden unter FACE in der alphabetischen Reihenfolge ihrer Namen aufgeführt. Bild 5-13 zeigt, wie die Datei `gizmo` und der Katalog `BUDGET` standardmäßig in Ihrem Dateisystem eingereiht werden. Beachten Sie, daß großgeschriebene Namen bei alphabetischer Reihenfolge vor den kleingeschriebenen stehen.

Bild 5-12: Standard-Format eines Menüs mit Dateien und Katalogen



Die Dateien und Kataloge können jedoch auch nach anderen Kriterien (z.B. in chronologischer Reihenfolge ihrer Erstellung oder Änderung) sortiert sein. Die Angabe von anderen Sortierkriterien für die Anzeige der Dateien in Ihrem Dateisystem wird im Abschnitt "Umorganisation des Inhalts eines einzelnen Katalogs" beschrieben.

Richtlinien zur Benennung von Dateien und Katalogen

Dateien und Kataloge können in einem Menü mit bis zu vier separaten Bezeichnern aufgeführt sein: Einem Namen, einer Beschreibung, einer Typangabe sowie Datum und Uhrzeit der letzten Änderung. Die Namen ordnen Sie den Dateien und Katalogen in Ihrem Dateisystem beim Anlegen zu. FACE fügt eine Standard-Beschreibung hinzu, die vom Typ der angelegten Datei bzw. des angelegten Katalogs abhängt. So ist jedem Eintrag im Ausschnitt /home/login in Bild 5-13 eine der Beschreibungen File Folder (Katalog) oder Standard File (Standard-Datei) zugeordnet, je nachdem, wie sie vom Benutzer beim Anlegen

festgelegt worden ist.

Für die Benennung bzw. Umbenennung eines Katalogs oder einer Datei gelten folgende Richtlinien:

- Länge: Der Name einer Datei oder eines Katalogs, die/den Sie in Ihrem FACE-Office anlegen, kann in einem Dateisystem unter UNIX System V Release 4.0 1 bis 14 Zeichen enthalten.

  Hinweis: Wenn Sie mit einem anderen, unterstützten Dateisystem arbeiten, so dürfen die Namen der Dateien und Kataloge maximal 255 Zeichen lang sein. Allerdings kann es in einigen Dateisystemen Probleme geben, wenn Sie den Editor `vi` mit einer Datei aufrufen, deren absoluter Pfadname mehr als 128 Zeichen enthält: Der Name der angelegten Datei enthält dann nur die ersten 128 Zeichen, und Sie werden in einer Meldung darüber informiert, daß die Datei nicht Ihren Angaben entsprechend angelegt worden ist. Text Text Text Text

- Buchstaben/Zeichen: Die Namen dürfen Zahlen und Buchstaben in beliebiger Kombination enthalten.
- Einschränkungen hinsichtlich des ersten Zeichens: Ein Datei- oder Katalogname darf nicht mit einem Punkt (.) oder einem Steuerzeichen (z.B. CTRL oder ESC) beginnen.
- Einige Zeichen sind in einem Datei- oder Katalognamen überhaupt nicht zulässig: Das Leerzeichen, das Tabulatorzeichen, nicht-darstellbare Zeichen (z.B. ESC oder CTRL) sowie die folgenden Zeichen:
  & ! | < >

Warnung: Vor der Verwendung dieser Zeichen in einem Datei- oder Katalognamen kann nicht oft genug gewarnt werden - diese Zeichen haben unter UNIX und FACE eine spezielle Bedeutung. Wenn ein Datei- oder Katalogname eines dieser Zeichen enthält, so können Sie die Datei bzw. den Katalog nicht eröffnen oder benutzen.

- Zwei Dateien bzw. ein Katalog und eine Datei können denselben Namen haben, vorausgesetzt, sie sind nicht in ein und demselben Katalog enthalten. So kann eine Datei namens `status` sowohl in Ihrem Home-Verzeichnis (/home/login) enthalten sein, als auch in einem Katalog

namens PROJECTS (/home/login/PROJECTS). In ein und demselben Katalog dürfen jedoch nicht eine Datei und ein Katalog namens status vorkommen.

FACE gibt eine Fehlermeldung aus und verhindert, daß Sie einer Datei bzw. einem Katalog einen Namen zuordnen, den es in diesem Katalog bereits gibt.

Pfadnamen in FACE-Kommandos

In der Bezugnahme auf einen Katalog oder eine Datei innerhalb eines FACE-Kommandos müssen Sie FACE den Zugriffspfad (Pfadnamen) mitteilen. Befindet sich der Katalog bzw. die Datei im aktuellen Katalog, so müssen Sie lediglich den Namen angeben; andernfalls müssen Sie den Zugriffspfad zu dem Katalog angeben. Gibt es im Menü /home/login beispielsweise einen Katalog namens BUDGET, so hat dieser Katalog den Pfadnamen /home/login/BUDGET.

Jeder Benutzer, dem auf dem Rechner ein Benutzername zugeordnet worden ist, hat sein eigenes Menü /home/login bzw. $HOME. $HOME ist dabei eine Umgebungsvariable, der die Stelle des Dateisystems zugeordnet ist, an die Sie nach Ihrer Anmeldung automatisch versetzt werden. Unter UNIX befindet sich diese Stelle im Katalog /home/*benutzer_name*, wobei *benutzer_name* Ihr eigener Benutzername ist; in diesem Katalog befinden Sie sich unmittelbar nach Ihrer Anmeldung bei FACE. Die Abkürzung $HOME ist sowohl unter UNIX als auch unter FACE eine gültige Angabe. Ausführliche Informationen über $HOME sowie absolute und relative Pfadnamen finden Sie im Abschnitt "Ihre Position im UNIX-System" in Kapitel 1 dieses Leitfadens.

Hinweis: Möglicherweise beginnt das Verzeichnis, das auf Ihrem System der Umgebungsvariablen $HOME zugeordnet ist, nicht mit dem Standardnamen /home.

Wenn der aktuelle Pfadname eines Katalogs aus 255 Zeichen oder weniger besteht (aus maximal 1023 in einem BSD-Dateisystem), darf der Name einer neuen Datei bzw. eines neuen Katalogs bis zu 14 Zeichen enthalten. Allerdings kann es zu Problemen kommen, wenn Sie den Editor vi mit einer Datei aufrufen, deren absoluter Pfadname über 128 Zeichen enthält; der Dateiname der angelegten Datei wird auf 128 Zeichen gekürzt, und Sie werden in einer Meldung darüber informiert, daß die Datei nicht wie gewünscht angelegt worden ist (die Schrägstriche (/) zur Unterteilung des Pfadnamens werden mitgezählt). Auf die Länge des absoluten Pfadnamens müssen Sie nicht nur

beim Anlegen einer neuen Datei bzw. eines neuen Katalogs achten, sondern auch beim Kopieren oder Versetzen einer Datei bzw. eines Katalogs in einen anderen Katalog; andernfalls gibt FACE eine entsprechende Fehlermeldung aus.

In einer Bezugnahme auf eine Datei über ihren absoluten Pfadnamen müssen Sie in einem FACE-Kommando jeden Teil des Pfadnamens eingeben. Allerdings werden sehr lange Pfadnamen, deren Länge die Breite der Bildschirmzeile übersteigt, von FACE gekürzt, wenn sie in einer Kommando-Ausgabe angezeigt werden (an einigen Stellen des Pfadnamens können dann Punkte stehen).

Hinweis: Mit dem Kommando `showpath` können Sie den vollständigen Pfadnamen der Datei bzw. des Katalogs jederzeit abrufen.

Legt eine Person mit dem Benutzernamen `det` z.B. die Datei `footnotes` mit dem absoluten Pfadnamen

```
/home/det/work/newbook/src_text/chapter1/footnotes
```

an, so kann dieser Pfadname von FACE bei der Anzeige innerhalb eines Ausschnitts folgendermaßen gekürzt werden:

```
/home/det/.../chapter1/footnotes
```

Anlegen von Dateien und Katalogen

Anhand der folgenden Übungen können Sie etwas Praxis im Anlegen von Dateien und Katalogen erwerben. Die einzusetzenden Namen werden angegeben. Wenn Sie mit eigenen Namen arbeiten möchten, so achten Sie darauf, daß sie keines der im Abschnitt "Richtlinien zur Benennung von Dateien und Katalogen" aufgeführten Sonderzeichen enthalten.

Anlegen eines Katalogs

In dieser Übung werden Sie die beiden Kataloge `alpha` und `beta` im Ausschnitt /home/login anlegen.

1. Wählen Sie im Menü Office of login die Option `Filecabinet`.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]), um auf die zweite Funktionstasten-Ebene (für Menüs) zugreifen zu können.

   Über die meisten Funktionstasten auf der zweiten Ebene werden Datei- und Katalog-Manipulationen durchgeführt. Beachten Sie, daß die Funktionstaste [F6] jetzt mit [CREATE] beschriftet ist. D. h., die

Funktionstaste [F6] ist jetzt mit dem FACE-Kommando `create` belegt.

3. Betätigen Sie die Taste [CREATE] (bzw. [CTRL-f] [6]) um `create` aufzurufen.

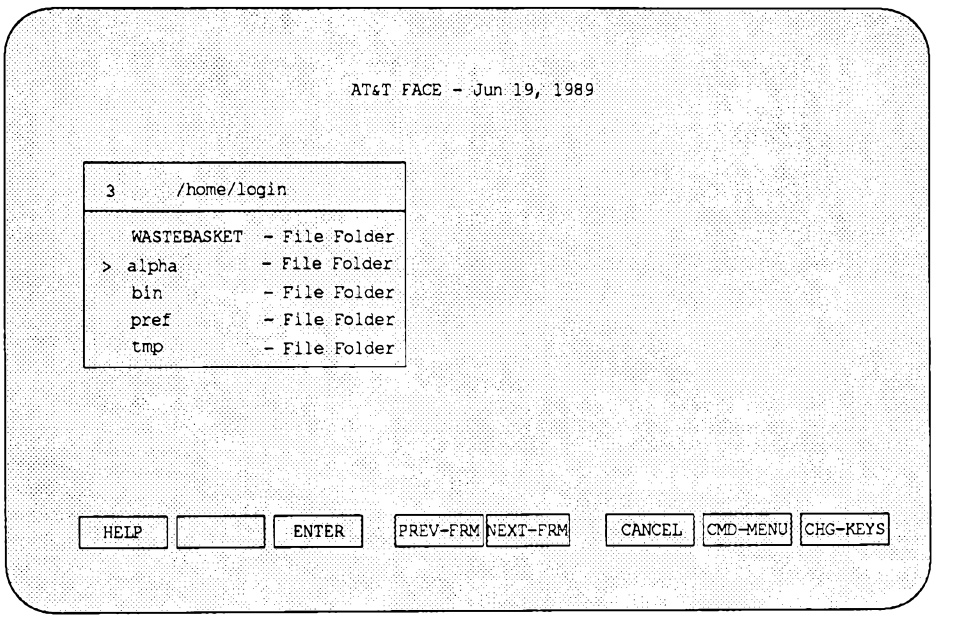
   In der Kommandozeile wird die Eingabeaufforderung `Enter the new object name:` angezeigt.

4. Geben Sie `alpha` ein und betätigen Sie die Taste [ENTER], um den Namen dem neuen Katalog zuzuordnen.

   Damit haben Sie FACE zwar den Namen des neuen Katalogs mitgeteilt, nicht aber, daß es ein Katalog sein soll. Auf dem Bildschirm erscheint ein Menü mit dem Titel `Create alpha` mit mindestens zwei zur Wahl stehenden Einträgen: `File Folder` und `Standard File`.

5. Wählen Sie im Menü Choices die Option `File Folder`.

   Im Ausschnitt /home/login wird jetzt der Inhalt des Katalogs `alpha` aufgelistet. Wird auch der Typ angezeigt, sollte der Ausschnitt /home/login in etwa wie folgt aussehen:



In der Meldungszeile wird auch der Name und der Zugriffspfad des neuen Katalogs angezeigt.

6. Legen Sie im Ausschnitt /home/login einen weiteren Katalog namens `beta` an.

Ineinandergeschachtelte Kataloge Ein Katalog, der in einem anderen Katalog angelegt wird, heißt "ineinandergeschachtelter Katalog." Vielleicht kennen Sie die russische "Matrioschka-Puppe", die aus mehreren ineinandergesteckten Puppen besteht: Wenn man eine Puppe aufmacht, kommt eine kleinere zum Vorschein, in der dann wiederum eine noch kleinere steckt usw. Auf dieselbe Weise kann auch ein Katalog einen weiteren Katalog enthalten, der dann wiederum einen weiteren Katalog enthält usw. (allerdings muß der Katalog, der in einem Katalog angelegt wird, nicht unbedingt kleiner sein als der übergeordnete Katalog). In dieser Übung wird in einem Katalog ein neuer Katalog angelegt.

1. Wählen Sie im Ausschnitt /home/login die Option `alpha`.

   In Ihrem Arbeitsbereich wird jetzt ein neuer Ausschnitt mit dem Titel alpha angelegt, der zum aktiven Ausschnitt wird.

2. Legen Sie im Ausschnitt alpha einen Katalog namens `alpha2` an.

Anlegen einer Datei

In dieser Übung werden zwei Dateien nach derselben Methode angelegt, nach der Sie bereits Kataloge angelegt haben.

1. Betätigen Sie die Taste [PREV-FRM] (bzw. [CTRL-f] [4]), um zum Ausschnitt /home/login zurückzuschalten.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]) und betätigen Sie dann die Taste [CREATE] (bzw. [CTRL-f] [6]).

   In der Kommandozeile erscheint die Eingabeaufforderung `Enter the new object name:`.

3. Geben Sie den Namen der neuen Datei (`file1`) ein und betätigen Sie die Taste [ENTER], um der neuen Datei einen Namen zuzuordnen.
4. Wählen Sie im Menü Choices die Option `Standard File`

   Bei der Wahl von `Standard File` legt `create` eine neue Datei an, der FACE-Bildschirm wird vollständig gelöscht, und ein Editor (Standard

unter FACE: Der UNIX-Editor `vi`) wird automatisch für Sie aufgerufen, mit dem Sie Daten in die Datei einbringen können. In einer Zeile am unteren Bildschirmrand werden Sie über den absoluten Pfadnamen Ihrer neuen Datei informiert.

Hinweis: Sieht Ihr Bildschirm nach dem Aufrufen der Datei anders aus, so ist `vi` möglicherweise nicht Ihr Standard-Editor. In diesem Fall können Sie die Anweisungen zu `vi` in dieser Übung auslassen. Eine Einführung in den Editor `ed` finden Sie in Kapitel 6 dieses Leitfadens.

5. Weitere Informationen über den Editor `vi` finden Sie in Kapitel 7 dieses Leitfadens. Vorerst sollten Sie die folgende Übung genau wie angegeben nachvollziehen.
6. Betätigen Sie die Taste `i` (für insert), um den Einfügen-Modus von `vi` aufzurufen.
7. Geben Sie einige Textzeilen ein.
8. Betätigen Sie die Taste ESC, um den Einfügen-Modus von `vi` zu verlassen (da Sie keinen Text mehr in die Datei einfügen möchten).
9. Geben Sie `:wq` ein und betätigen Sie die Taste ENTER.

   Über den Doppelpunkt `(:)` schalten Sie in den `vi`- Kommandomodus um, über das Kommando `w` (für write) speichern Sie die Datei ab, und über das Kommando `q` (für quit) verlassen Sie den Editor und kehren zu Ihrem FACE-Office zurück. Im Menü /home/login ist jetzt der Eintrag `file1` enthalten.

Hinweis: In einem Katalog-Menü können maximal 18 Optionen gleichzeitig dargestellt werden. Sind in Ihrem Katalog /home/login mehr als 18 Einträge enthalten, so müssen Sie die Bildschirmanzeige möglicherweise nach oben oder unten verschieben, um den Eintrag `file1` zur Darstellung zu bringen.

Im nächsten Teil dieser Übung werden Sie eine weitere Datei anlegen, diesmal aber im Katalog alpha.

1. Wählen Sie im Menü /home/login die Option `alpha`.

   `alpha` ist jetzt der aktive Ausschnitt; alle Dateien, die Sie jetzt anlegen, werden in diesem Ausschnitt angelegt.

2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]), dann [CREATE] (bzw. [CTRL-f] [6]).

3. Geben Sie bei der Eingabeaufforderung in der Kommandozeile `file2` ein und betätigen Sie die Taste [ENTER].

4. Wählen Sie im Menü Choices die Option `Standard File`.

5. Auch in diesem Fall wird eine neue Datei eröffnet. Geben Sie mit Ihrem Editor etwas Text ein.

6. Speichern Sie `file2` ab und verlassen Sie den Editor; in Ihrem Arbeitsbereich werden wieder das Menü Office of login sowie die übrigen eröffneten Ausschnitte angezeigt.

   Beachten Sie, daß der Eintrag `file2` jetzt im Ausschnitt alpha enthalten und als Standard File (Standard-Datei) gekennzeichnet ist.

Kopieren und Versetzen von Dateien und Katalogen

Mit der Zeit möchten Sie vielleicht die Anordnung der Dateien oder Kataloge in Ihrem Dateisystem ändern, indem Sie z.B. eine Datei oder einen Katalog an eine andere Stelle innerhalb des Dateisystems versetzen. Für die Benutzung der Kommandos `copy` und `move` gibt es eine Reihe von Gründen:

- Als Absicherung gegen das Verlieren oder Zerstören von wichtigen Daten.
- Sie möchten Teile einer Datei als Grundlage für eine zweite Datei benutzen, oder Sie möchten einem anderen Benutzer die Kopie einer Datei zur Verfügung stellen.

Hinweis: Wenn Sie eine Datei bzw. einen Katalog aus dem UNIX-System in Ihr FACE-Office kopieren, so bleiben die UNIX-Zugriffsberechtigungen erhalten. Wenn Sie Ihren Dateien und Katalogen unter FACE andere Zugriffsberechtigungen zuteilen als unter UNIX, müssen Sie sie möglicherweise vor der Bearbeitung oder sonstigen Verwendung einer kopierten Datei ändern (siehe Abschnitt "Zugriffsschutz für eine vorhandene Datei.")

Die Kommandos `move` und `copy` laufen im wesentlichen gleich ab; beim

Ergebnis besteht jedoch der folgende entscheidende Unterschied:

- Wenn Sie eine Datei bzw. einen Katalog mit dem Kommando `copy` kopiert haben, ist sie bzw. er an zwei verschiedenen Stellen vorhanden, nämlich der ursprünglichen und der neuen Stelle.
- Nachdem Sie eine Datei bzw. einen Katalog mit dem Kommando `move` versetzt haben, ist sie bzw. er nur an einer einzigen Stelle vorhanden, nämlich der neuen Stelle.

In der folgenden Übung wird das Kopieren einer Datei beschrieben. Das Versetzen wird hier nicht geübt; beachten Sie aber, daß die Vorgehensweise beim Kopieren von Dateien und Katalogen mit der beim Versetzen identisch ist.

Kopieren einer Datei

Im letzten Abschnitt haben Sie im Ausschnitt /home/login die Datei `file1` angelegt. Wenn Sie jetzt im Ausschnitt /home/login eine zweite Kopie von `file1` anlegen möchten, so führen Sie die folgenden Schritte durch:

1. Schalten Sie zum Ausschnitt /home/login um (falls erforderlich).
2. Bewegen Sie den Cursor zum Eintrag `file1`.
3. Rufen Sie über die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]) die zweite Funktionstasten-Ebene auf und betätigen Sie dann die Taste [COPY] (bzw. [CTRL-f] [2]).

   Die Funktionstasten-Titel haben sich erneut geändert; [F8] hat jetzt den Titel [SELECT]. Sie werden in einer Meldungszeile zur Eröffnung des Ziel-Katalogs (d.h. zum Umschalten zu diesem Katalog) aufgefordert; betätigen Sie jetzt die Taste [SELECT]. Als Ziel-Katalog wird der Katalog bezeichnet, in den Sie die Datei kopieren möchten.
4. In diesem Fall ist der Katalog /home/login der Ziel-Katalog. Da Sie sich bereits in diesem Katalog befinden, betätigen Sie lediglich die Taste [SELECT] (bzw. [CTRL-f] [8]).
5. The Meldungszeile informiert Sie darüber, daß eine Datei namens `file1` bereits im Katalog /home/login enthalten ist; Sie müssen der Kopie also einen neuen Namen zuordnen.

   Geben Sie `file2` ein und betätigen Sie die Taste [ENTER].

   Das Menü im Ausschnitt /home/login wird sofort aktualisiert und mit dem neuen Eintrag `file2` angezeigt. In der Meldungszeile werden Sie

auch darüber informiert, wohin die Kopie gesetzt worden ist, und daß sie den Namen `file2` erhalten hat.

Mach dieser Methode gehen Sie auch beim Kopieren und Versetzen eines Katalogs vor.

Hinweis: Bei der Umstrukturierung des Dateisystems ist zu beachten, daß beim Kopieren oder Versetzen eines Katalogs sämtliche darin enthaltenen Kataloge und Dateien ebenfalls kopiert oder versetzt werden.

Umbenennen einer Datei bzw. eines Katalogs

Um einer Datei bzw. einem Katalog einen neuen Namen zuzuordnen, benutzen Sie das Kommando `rename`. Damit können Sie z.B. den Katalog `newwork` in `oldwork` umbenennen, wenn er nicht mehr auf dem aktuellsten Stand ist.

Beim Nachvollziehen der folgenden Übung sollten Sie mit den vorgegebenen Dateinamen arbeiten; bei der Wahl eigener Dateinamen sollten Sie sicherstellen, daß sie keines der im Abschnitt "Richtlinien zur Benennung von Dateien und Katalogen" angegebenen Sonderzeichen enthalten.

Angenommen, die Datei `file2`, die Sie in einer früheren Übung im Katalog /home/login angelegt haben, enthält einen Arbeitsplan. Da in Ihrem Dateisystem inzwischen mehrere Dateien enthalten sind, möchten Sie `file2` umbenennen, damit ihr Inhalt besser aus ihrem Namen hervorgeht. Dabei gehen Sie folgendermaßen vor:

1. Schalten Sie zum Ausschnitt /home/login um.
2. Bewegen Sie den Cursor zur Menüoption `file2` im Menü /home/login.
3. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]) und betätigen Sie dann die Taste [RENAME] (bzw. [CTRL-f] [5]).
4. Bei der Eingabeaufforderung für einen neuen Dateinamen geben Sie `workplans` ein und betätigen Sie die Taste [ENTER].

   Im Ausschnitt /home/login ist als Menüoption jetzt `workplans` und nicht mehr `file2` aufgeführt. Die Datei hat zwar denselben Inhalt, aber einen neuen Namen.

Neue Beschreibung für eine Datei oder einen Katalog

Beim Anlegen einer neuen Datei bzw. eines neuen Katalogs können Sie außer einem Namen auch eine Beschreibung angeben (die Beschreibungen werden von FACE in einem Menü angezeigt, wenn Sie sie mit dem entsprechenden Standard-Parameter in der Schablone Preferences oder mit dem Kommando `organize` angefordert haben. Die Schablone Preferences und das Kommando `organize` werden in einem späteren Abschnitt beschrieben). Beim Anlegen einer neuen Datei bzw. eines neuen Katalogs entspricht ihre/seine Beschreibung standardmäßig der Typangabe. D. h., FACE benutzt den Typ der Option (in den meisten Fällen File Folder oder Standard File) als anzuzeigende Beschreibung. Da FACE den Typ der Datei bzw. des Katalogs in jedem Fall abspeichert, können Sie die Beschreibung ohne weiteres etwas aussagekräftiger gestalten. Die Beschreibung einer Menüoption können Sie mit dem Kommando `redescribe` angeben bzw. ändern; den Typ dagegen können Sie in keinem Fall ändern.

Hinweis: Machen Sie sich den Unterschied zwischen der Beschreibung und dem Typ einer Datei klar. FACE kennt lediglich wenige Typen von Dateien und Katalogen und kann sie auf der Grundlage einer Typangabe suchen. FACE erkennt die folgenden Datei- bzw. Katalog-Typen: Standard File (Standard-Datei), File Folder (Katalog), Foreign file (Unbekanntes Dateiformat), Data file (Daten-Datei), Executable file (Ausführbare Datei), Text file (Textdatei), Form (Schablone) und Menu (Menü). Die Beschreibung einer Datei bzw. eines Katalogs dagegen können Sie beliebig festlegen. FACE benutzt die Beschreibung lediglich zu Ihrer Information und ermöglicht Ihnen jederzeit ihre Änderung.

Der Eintrag einer Datei bzw. eines Katalogs innerhalb eines Menüs besteht aus vier Bezeichnern: Dem Namen, der Beschreibung, dem Typ sowie Datum und Uhrzeit der Erstellung bzw. der letzten Änderung. Diese vier Bezeichner werden nur unter den folgenden Bedingungen vollständig angezeigt: Wenn das Feld `Folder Display Format` in der Schablone Office Functions auf den Wert `long form` gesetzt ist und Sie zumindest einem der im Menü aufgeführten Datei- bzw. Katalog-Einträge eine neue Beschreibung zugeordnet haben.

Je mehr Dateien und Kataloge in Ihrem Dateisystem enthalten sind, desto empfehlenswerter ist die Anpassung des Beschreibungs-Felds an den Dateiinhalt, so daß Sie die verschiedenen Dateien leichter voneinander unterscheiden können. In der folgenden Übung wird die Beschreibung eines Felds bzw. Katalogs mit dem Kommando `redescribe` geändert.

Format der Beschreibungen

Bei der Zuordnung einer neuen Beschreibung für eine Datei oder einen Katalog ist folgendes zu beachten:

- Länge: Eine Beschreibung darf maximal 23 Zeichen enthalten (bei ausführlichen Dateieinträgen (long form) werden nur 19 angezeigt).
- Buchstaben/Zeichen: Sie können jede beliebige Kombination aus Zahlen, Buchstaben und Leerzeichen benutzen.
- Unzulässige Sonderzeichen: Nicht benutzt werden dürfen das Pipe-Symbol (|) sowie nicht-darstellbare Zeichen wie ESC oder CTRL.

In der folgenden Übung wird die Beschreibung der Datei `workplans` mit dem Kommando `redescribe` geändert.

1. Bewegen Sie den Cursor zum Eintrag `workplans` im Ausschnitt /home/login.
2. Betätigen Sie die Taste CMD-MENU (bzw. CTRL-f 7), und wählen Sie `redescribe`.

   (Beachten Sie, daß das Kommando `redescribe` nicht auf eine Ihrer Funktionstasten gelegt werden kann).
3. Geben Sie bei der Eingabeaufforderung in der Kommandozeile eine neue Beschreibung ein und betätigen Sie die Taste ENTER.

   Verwenden Sie eine möglichst aussagekräftige Beschreibung wie z.B. "plans for 1990". Beachten Sie dabei die Richtlinien unter "Format der Beschreibungen" zu Beginn dieser Übung.

   Sie müssen auf jeden Fall eine Beschreibung eingeben; wenn Sie also einfach die Taste ENTER betätigen, stellt FACE erneut eine Aufforderung zur Eingabe einer Beschreibung dar.

   Die neue Beschreibung wird jetzt im Eintrag der Menüoption `workplans` angezeigt.

Nach exakt derselben Methode können Sie mit dem Kommando `redescribe` auch die Beschreibung eines Katalogs ändern.

Umorganisation des Inhalts eines einzelnen Katalogs

Die Anzahl der Bezeichner-Felder, die in einem Katalog für jede Option angezeigt werden, kann ebenso wie die Reihenfolge der Optionen von Ihnen festgelegt werden. Das Kommando `organize` wirkt sich ausschließlich auf die Dateien und Kataloge im aktuellen Katalog aus.

In der folgenden Übung wird mit dem Kommando `organize` die Reihenfolge der Dateien und Kataloge im Katalog `alpha` sowie die Anzahl der angezeigten Bezeichner-Felder geändert.

1. Wählen Sie im Menü /home/login die Option `alpha`.
2. Betätigen Sie die Taste CMD-MENU (bzw. CTRL-f 7) und wählen Sie das Kommando `organize`.

   Auf dem Bildschirm erscheint eine Schablone mit dem Titel `Organize alpha` und drei Eingabefeldern.

   Im Feld `Default Oganization` wird der Standard-Wert `no` angezeigt. Da Sie den Inhalt dieses Katalogs auf eine bestimmte Weise umordnen möchten, lassen Sie den Wert `no` unverändert.

   Wenn Sie den Wert dieses Felds in `yes` ändern und diese Schablone abspeichern würden, so würden die Eigenschaften dieses Katalogs wieder auf die Standardwerte zurückgesetzt, wie sie in der Schablone Preferences festgelegt worden sind (das Ändern der Standard-Parameter für Ihr gesamtes FACE-Office wird im Abschnitt "Ändern von Standard-parametern" an einer späteren Stelle dieses Kapitels beschrieben).
3. Bewegen Sie den Cursor zum Feld `Folder Display Format:`

   Über dieses Feld steuern Sie die Anzahl der Bezeichner-Felder, die im Katalog für jede Datei angezeigt werden. Dieses Feld kann folgendermaßen belegt werden:

   `long form` — Über diesen Wert geben Sie an, daß Name, Beschreibung, Datei-Typ sowie Datum und Uhrzeit (in dieser Reihenfolge) angezeigt werden sollen.

   `name und description` — Der Standard-Wert für dieses Feld; für jede Option wird der Dateiname sowie die Beschreibung (entweder die Standard-

Beschreibung, die dem Datei-Typ entspricht, oder die beim Kommando `redescribe` festgelegte Beschreibung) angezeigt.

`name and marks` — Mit diesem Wert wird nach dem Dateinamen ein Symbol angezeigt, das für den Typ der Datei steht (der Stern (*) für eine ausführbare Datei, der Schrägstrich (/) für einen Katalog, keine Markierung für eine Standard-Datei).

`name only` — Mit diesem Wert wird nur der Name der Datei angezeigt.

Hinweis: Organisieren Sie gerade den Katalog WASTEBASKET um, so steht als weitere Einstellung `wastebasket` zur Auswahl; dies entspricht dem Wert `name and description` plus dem Zugriffspfad zur ursprünglichen Position der Objekte, die aktuell im Katalog WASTEBASKET enthalten sind.

In diesem Feld sollte aktuell der Eintrag `name and description` enthalten sein, d.h., sämtliche Datei- und Katalog-Einträge im Katalog alpha sollten bisher aus dem Namen sowie einer kurzen Beschreibung bestehen.

4. Betätigen Sie die Taste [CHOICES] (bzw. [CTRL-f] [2]).

   Auf dem Bildschirm wird das Menü Choices mit den vier oben beschriebenen Werten angezeigt.

5. Bewegen Sie den Cursor zum Eintrag `long form` und betätigen Sie die Taste [ENTER] (bzw. [CTRL-f] [3]), um diesen neuen Wert in das Feld `Folder Display Format` einzutragen.

6. Speichern Sie den neuen Wert über die Taste [SAVE] (bzw. [CTRL-f] [3]) ab.

   Beachten Sie die Änderungen an den Menüoptionen im Katalog alpha. Von jeder Option werden nach wie vor Name und Beschreibung angezeigt, nun aber ergänzt durch Datum und Uhrzeit der Erstellung bzw. der letzten Änderung.

Hinweis: Die Kommandos `organize` und `redescribe` wirken sich nicht auf das Datum und die Uhrzeit aus, da sie an der Datei bzw. am Katalog selbst nichts ändern. Über diese Kommandos wird lediglich der Umfang der angezeigten Informationen geändert.

Zwischen dem Namen einer Datei und dem Bezeichner für Datum und Uhrzeit können ein oder zwei weitere Bezeichner stehen. Wenn Sie die Beschreibung einer Option nicht über das Kommando `redescribe` geändert haben, so folgt auf ihren Namen noch die Beschreibung, die FACE ihr beim Anlegen zugeordnet hat (die Standard-Beschreibung). Da diese Standard-Beschreibung dem Datei-Typ entspricht (siehe Abschnitt "Anlegen von Dateien und Katalogen"), stellt FACE den Typ nicht dar. Haben Sie einer Datei jedoch eine neue Beschreibung zugeordnet, so folgt auf den Namen diese Beschreibung, auf die wiederum der Datei-Typ folgt. Der Bezeichner für Datum und Uhrzeit steht bei einer ausführlichen Anzeige (long form) stets an letzter Stelle.

Probieren Sie das Kommando `redescribe` am Katalog `alpha` aus.

Über das Feld `Folder Display Order` steuern Sie die Reihenfolge, in der die Dateien und Kataloge aufgelistet sind. Für dieses Feld stehen folgende Werte zur Verfügung:

`alphabetical` Die Datei- und Katalog-Einträge sind nach dem Alphabet sortiert und in folgender Reihenfolge aufgelistet:

- Namen, die mit Zahlen beginnen.
- Namen, die mit Großbuchstaben beginnen.
- Namen, die mit Kleinbuchstaben beginnen.

`alphabetical by description`
Dateien und Kataloge werden zunächst nach ihrer Beschreibung und dann - wenn es zwei oder mehr Dateien eines bestimmten Typs gibt - nach ihrem Namen in alphabetische Reihenfolge gebracht.

`least recent first`
Die Dateien und Kataloge werden in umgekehrter chronologischer Reihenfolge aufgeführt (die Einträge mit dem "jüngsten" Erstellungs- oder Änderungsdatum stehen ganz oben).

`most recent first`
: Die Dateien und Kataloge werden in chronologischer Reihenfolge aufgeführt (die Einträge mit dem "ältesten" Erstellungs- oder Änderungsdatum stehen ganz oben).

Das Feld `Folder Display Order` ist standardmäßig mit dem Wert `alphabetical` belegt. Die folgende Übung zeigt Ihnen, wie Sie den Inhalt dieses Felds ändern können.

1. Da der Katalog alpha bereits eröffnet und der aktive Ausschnitt in Ihrem Arbeitsbereich sein sollte, wählen Sie das Kommando `organize`, um die Schablone `Organize alpha` erneut aufzurufen.
2. Bewegen Sie den Cursor zum Feld `Folder Display Order`.
3. Betätigen Sie die Taste [CHOICES] (bzw. [CTRL-f] [2]) auf dem Bildschirm erscheint einer neuer Ausschnitt mit dem Titel Choices.
4. Wählen Sie im Menü Choices die Option `most recent first`.

   Dadurch wird das Menü Choices geschlossen, und der Ausschnitt Organize alpha wird wieder aktiv. Im Feld `Folder Display Order` erscheint wieder der Wert `most recent first`.
5. Betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um diese Werte abzuspeichern.

   Die Schablone wird wieder ausgeblendet, und auf dem Bildschirm erscheint unmittelbar darauf der Katalog alpha in seiner neuen Anordnung (in umgekehrter chronologischer Reihenfolge).

Hinweis: Die Schablone `Organize` wird nicht angezeigt, wenn Sie für den aktuellen Katalog weder die Schreibberechtigung haben noch sein Eigentümer sind. Dies kann beispielsweise vorkommen, wenn Sie im Dateisystem eines anderen Benutzers arbeiten (Informationen zu den Zugriffsberechtigungen finden Sie im Abschnitt "Zugriffsschutz für eine vorhandene Datei").

Löschen einer Datei bzw. eines Katalogs

Die Kataloge und Dateien, die Sie nicht mehr benötigen, sollten Sie löschen. Wenn Sie eine Datei oder einen Katalog löschen, wird sie bzw. er in den Abfall gesetzt. Hier bleibt die Datei bzw. der Katalog für den Zeitraum (Anzahl der Tage), die Sie in der Schablone Office Functions angegeben haben; danach wird sie bzw. es permanent aus dem System gelöscht.

Hinweis: Eine Datei bzw. einen Katalog können Sie nicht löschen, wenn es bereits eine Datei bzw. einen Katalog dieses Namens in Ihrem Abfall gibt. Wenn Sie z.B. vormittags bereits die Datei `memo1` aus dem Ausschnitt /home/login gelöscht haben, am Nachmittag desselben Tags eine neue Datei namens `memo1` angelegt haben und jetzt auch diese Datei löschen möchten, so werden Sie zur Umbenennung der zweiten Datei mit dem Namen `memo1` aufgefordert. Nach der Umbenennung der Datei wird die Datei aus Ihrem Katalog gelöscht und unter dem neuen Namen in den Abfall gesetzt.

Einen eröffneten Katalog können Sie, ebenso wie darin enthaltene eröffnete Dateien und Kataloge, nicht löschen. Zu beachten ist außerdem, daß der zu löschende Katalog leer sein muß, d.h., alle enthaltenen Dateien und Kataloge gegebenenfalls zuvor gelöscht werden müssen.

Da Sie den Katalog alpha nicht mehr benötigen, können Sie daran in der folgenden Übung das Kommando `delete` ausprobieren.

1. Schließen Sie den Katalog `alpha` (mit [CANCEL] oder [CTRL-f] [6]).
2. Bewegen Sie den Cursor zur Menüoption `alpha` im Menü /home/login.
3. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]).
4. Betätigen Sie die Taste [DELETE] (bzw. [CTRL-f] [4]), um das Kommando `delete` aufzurufen.

   In der Meldungszeile erscheint nun eine Warnung, daß der Katalog `alpha` nicht leer ist.
5. In einer Kommandozeile werden Sie zur Betätigung der Taste [ENTER] aufgefordert, um `alpha` zu löschen.

   Danach werden Sie in einer Meldung darüber informiert, daß `alpha` in den Katalog /home/login/WASTEBASKET versetzt worden ist.

   Das Menü /home/login `alpha` ist jetzt nicht mehr als Menüoption

vorhanden.

Jetzt können Sie überprüfen, ob der Katalog `alpha` tatsächlich im Katalog /home/login/WASTEBASKET darauf wartet, permanent gelöscht zu werden (wenn Sie Ihren Katalog /home/login/WASTEBASKET so umorganisiert haben, daß das Feld `Folder Display Format:` auf `wastebasket` gesetzt ist, so zeigen die Optionen im Ausschnitt /home/login/WASTEBASKET jetzt den ursprünglichen Zugriffspfad jeder Option an, außerdem ihre Namen und Beschreibungen).

1. Schalten Sie zum Ausschnitt Office of login um und wählen Sie die Option `Wastebasket`.

   Auf Ihrem Bildschirm erscheint jetzt ein neuer Ausschnitt mit dem Titel /home/login/WASTEBASKET. In diesem Ausschnitt sollte jetzt der Katalog `alpha` enthalten sein.

2. Wählen Sie im Menü /home/login/WASTEBASKET die Menüoption `alpha`. Achten Sie auf seinen Inhalt! Beim Löschen eines Katalogs wird sein gesamter Inhalt ebenfalls gelöscht.

3. Schließen Sie den Ausschnitt /home/login/WASTEBASKET/alpha.

Um eine Datei bzw. einen Katalog sofort permanent zu löschen, können Sie Optionen mit dem Kommando `delete` aus dem Abfall selbst entfernen. Hierbei können Sie alle enthaltenen Dateien bzw. Kataloge oder auch nur einen bestimmten Katalog oder eine bestimmte Datei angeben. So können Sie mit dem Kommando `delete wastebasket` sämtliche Dateien bzw. Kataloge löschen, die aktuell in Ihrem Abfall enthalten sind (zuvor werden Sie jedoch zur Eingabe einer Bestätigung aufgefordert).

Löschen einer Datei bzw. eines Katalogs rückgängig machen

Die gelöschten Dateien bzw. Kataloge bleiben standardmäßig einen Tag lang im Abfall. Als zusätzliche Absicherung gegen das versehentliche Löschen einer Datei bzw. eines Katalogs gibt es unter FACE die Möglichkeit, das endgültige Löschen zu stornieren. Diese "zweite Chance" haben Sie bei der Standard-Konfiguration des Abfalls (die Änderung der Standard-Konfiguration des Abfalls wird im Abschnitt "Ändern von weiteren Standardparametern" beschrieben).

Solange eine Datei bzw. ein Katalog sich noch im Abfall befindet, können Sie sie/ihn jederzeit wieder laden (nähere Informationen darüber, wie Sie die Verweildauer einer Datei bzw. eines Katalogs im Abfall festlegen können, finden Sie im Abschnitt "Ändern von Standardparametern"). Das Kommando `undelete` rufen Sie immer dann auf, wenn Sie etwas gelöscht haben und feststellen, daß Sie es doch noch brauchen. In dieser Übung wird das Löschen des Katalogs `alpha` rückgängig gemacht.

Nachdem Sie den Ausschnitt /home/login/WASTEBASKET/alpha in der letzten Übung geschlossen haben, kehrten Sie zum Menü /home/login/WASTEBASKET zurück.

1. Bewegen Sie den Cursor im Menü /home/login/WASTEBASKET zur Option `alpha`.

   Beachten Sie, daß vom Katalog `alpha` außer seiner Beschreibung und seinem Namen auch sein absoluter Pfadname angezeigt wird. Daran können Sie erkennen, an welche Position innerhalb des Dateisystems die Datei bzw. der Katalog nach dem Aufheben des Löschvorgangs gesetzt wird.

2. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]), und wählen Sie `undelete`.

   Das Kommando `undelete` ist nicht auf eine Funktionstaste gelegt worden; daher müssen Sie es in die Kommandozeile eintragen oder es (wie in diesem Fall) im Command Menu wählen.

   Achten Sie auf den Inhalt des Menüs /home/login/WASTEBASKET. Der Katalog `alpha` wird wieder ins Menü /home/login zurückgesetzt, und im Menü /home/login/WASTEBASKET erscheint die Meldung `Empty Folder`.

3. Schließen Sie den Ausschnitt /home/login/WASTEBASKET.

Es kommt nicht selten vor, daß bereits gelöschte Dateien und Kataloge wiederhergestellt werden sollen. Nach der Durchführung des Kommandos `undelete` werden Dateien bzw. Kataloge automatisch wieder in ihre ursprünglichen Kataloge zurückgesetzt. Mit der Wiederherstellung eines Katalogs werden auch alle zugehörigen Optionen wiederhergestellt (sollen die Dateien bzw. Kataloge nach ihrer Wiederherstellung nicht mehr im ursprünglichen Katalog enthalten sein, so benutzen Sie statt des Kommandos `undelete` das Kommando `copy` oder `move`).

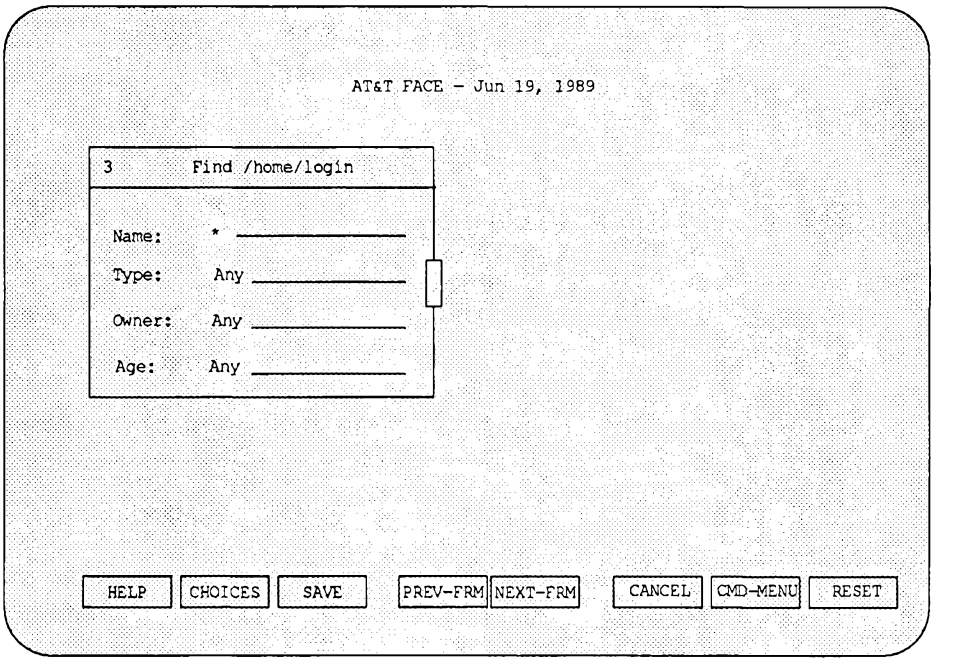
Hinweis: Wenn FACE nach dem Kommando `undelete` eine Meldung anzeigt, daß es einen Katalog nicht eröffnen oder auf eine Option nicht zugreifen kann, so haben Sie möglicherweise den Katalog, in dem die Option zuvor enthalten war, gelöscht oder umbenannt. In einem solchen Fall müssen Sie die Option mit dem Kommando `move` oder `copy` aus dem Abfall zurückholen.

Suchen von Dateien und Katalogen

Einer der wichtigsten Vorteile der Benutzung eines Rechners ist, daß er eine rationelle Arbeitsweise ermöglicht. Unter FACE können Sie die zu bearbeitenden Dateien bzw. Kataloge schnell und einfach mit dem Kommando `find` ausfindig machen. Das Kommando `find` ist vor allem dann hilfreich, wenn Ihr Dateisystem aus mehreren Hierarchiestufen besteht oder Sie sich nicht mehr an die Position einer Datei erinnern können.

Nachdem Sie das Kommando `find` aufgerufen haben, erscheint in Ihrem Arbeitsbereich eine Find-Schablone mit vier Feldern, in die Sie Suchkriterien eintragen können. Bild 5-14 zeigt ein Beispiel für eine Find-Schablone, die bei aktivem Ausschnitt /home/login eröffnet worden ist. Die Schablone enthält vier Felder, in die Sie die Kriterien für den Suchvorgang eintragen können: `Name`, `Type`, `Owner` und `Age`.

Bild 5-13: Die Find-Schablone



Im Gegensatz zu den meisten Schablonen, die bei der Betätigung der Taste (SAVE) automatisch geschlossen werden, bleibt diese Schablone eröffnet, bis Sie sie explizit schließen. Dies hat für Sie den Vorteil, daß Sie Ihre Suchkriterien nochmals überprüfen oder andere Suchkriterien eingeben können, wenn eine Option nicht gefunden werden kann. In die vier nachfolgend aufgeführten Felder können Sie die folgenden Suchkriterien eintragen:

`Name:` Das Feld `Name` enthält den Namen der gesuchten Option; der Name kann vollständig oder - mit Hilfe von Sonderzeichen- teilweise eingegeben werden (in der nachfolgenden Übung erfahren Sie, wie Sie über die Funktionstaste (HELP) Informationen zu den Sonderzeichen abrufen können, die sich für dieses Feld eignen).

`Type:` Das Feld `Type` enthält den Typ der Option (z.B. `File Folder` oder `Standard file`). Sie können über die Funktionstaste [CHOICES] (bzw. [CTRL-f] [2]) eine vollständige Übersicht über die Optionstypen abrufen, die Ihnen in Ihrem FACE-Office zur Wahl stehen. Beachten Sie, daß FACE eine Datei von einem Katalog unterscheiden kann.

`Owner:` Das Feld `Owner` enthält den Benutzernamen des Eigentümers der Datei bzw. des Katalogs. Über die Funktionstaste [CHOICES] (bzw. [CTRL-f] [2]) können Sie in diesem Feld eine Auflistung aller beim System eingetragenen Benutzernamen abrufen. Das Kommando `find` sucht nach den Dateien des angegebenen Eigentümers *unterhalb* des aktuellen Katalogs.

`Age:` Im Feld `Age` kann als Kriterium das Alter der Datei bzw. des Katalogs (Anzahl der Tage seit dem Anlegen bzw. Ändern) angegeben werden. Über die Taste [CHOICES] (bzw. [CTRL-f] [2]) können Sie in diesem Feld eine kurze Meldung abrufen, in der die zulässigen Werte erklärt werden. Beachten Sie, daß das Alter einer Datei bei einer Umbenennung oder einer Änderung ihrer Beschreibung nicht geändert wird.

Hinweis: Das Kommando `find` startet einen Suchvorgang standardmäßig in Ihrem Katalog /home/login, wenn Sie sich beim Aufrufen des Kommandos nicht in einem Katalog befinden (und Sie das Kommando `find` beispielsweise in einer Schablone oder einem Textausschnitt aufrufen).

Wenn das Kommando `find` Dateien bzw. Kataloge ausfindig machen kann, die zu Ihren Kriterien passen, so stellt es sie in einem Abschnitt namens Objects Found dar. Die Funktionen, die Sie für Optionen im Ausschnitt Objects Found durchführen (z.B. das Kommando `copy`, `move` oder `delete`) haben dieselbe Auswirkung, als würden Sie sie in dem Katalog aufrufen, in dem sich die Option tatsächlich befindet.

Hinweis: Das Menü Objects Found wird nicht automatisch aktualisiert; daher müssen Sie nach der Durchführung einer der oben aufgeführten Funktionen in jedem Fall das Kommando `update` im Command Menu oder in der Kommandozeile aufrufen. Andernfalls spiegelt das Menü die durchgeführten Änderungen nicht wider.

Wenn Ihnen z.B. das Verzeichnis entfallen ist, in dem die in einer früheren Übung angelegte Datei `file2` enthalten ist, so gehen Sie folgendermaßen vor:

1. Schalten Sie zum Menü /home/login um. Das Kommando `find` durchsucht sämtliche Kataloge unterhalb des aktiven Ausschnitts, ob eröffnet oder nicht. Da es den Suchvorgang im Menü /home/login startet, durchsucht `find` zwar Ihr gesamtes Dateisystem, nicht jedoch den Abfall.

2. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]), und wählen Sie `find`.

   Auf dem Bildschirm erscheint eine Find-Schablone. Beachten Sie, daß diese Schablone den Titel Find /home/login hat. Aus dem Titel einer Find-Schablone geht die Stelle innerhalb des Dateisystems hervor, an dem der Suchvorgang gestartet wird.

3. Geben Sie im Feld `Name` den Namen `file2` ein und betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]).

   Wenn Sie das Kommando `find` in einer anderen Situation eingeben, können Sie (wie in diesem Fall) den Dateinamen als einziges Suchkriterium eingeben oder auch in die übrigen Felder Informationen eintragen, um die Suchkriterien enger einzugrenzen.

4. Im Ausschnitt /home/login wird jetzt das Menü Objects Found mit dem Pfadnamen der Datei bzw. des Katalogs angezeigt, also `alpha/file2`. Dies zeigt Ihnen, daß `file2` im Katalog alpha enthalten ist. Da alpha der einzige Katalog-Name in diesem Pfadnamen ist, wissen Sie auch, daß alpha sich direkt im Katalog /home/login file befindet.

Sie können im Menü Objects Found des Ausschnitts /home/login die Datei `alpha/file2` auswählen und alle Manipulationen damit durchführen, als würden Sie sich im Katalog alpha selbst befinden. In der nächsten Übung lernen Sie das Kommando `display` kennen; Sie werden damit im Menü Objects Found des gerade eröffneten Ausschnitts /home/login den Inhalt von `file2` abrufen.

Anzeige des Dateiinhalts

Nachdem Sie die Datei `file2` ausfindig gemacht haben, lernen Sie in dieser Übung, wie Sie ihren Inhalt abrufen können, ohne sie zum Editieren zu eröffnen. Zum Abrufen des Inhalts einer Datei innerhalb eines Ausschnitts rufen Sie einfach das Kommando `display` auf.

1. Bewegen Sie den Cursor im Menü Objects Found des Ausschnitts /home/login zum Eintrag `alpha/file2`.
2. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]), und wählen Sie `display`.

   Im Arbeitsbereich wird jetzt ein neuer Ausschnitt mit dem Titel "`file2 - Standard file`" eröffnet.

   Über die folgenden Tasten können Sie diesen Textausschnitt nach oben und unten verschieben:

   [↑] (bzw. [CTRL-u])
   [↓] (bzw. [CTRL-d])
   [SCROLL-UP] (bzw. [CTRL-f] [u])
   [SCROLL-DOWN] (bzw. [CTRL-f] [d])
   [BEG] (bzw. [CTRL-b])
   [END] (bzw. [CTRL-e])
3. Nachdem Sie die Datei gelesen haben, betätigen Sie die Taste [CANCEL] (bzw. [CTRL-f] [6]).

Der Dateiinhalt, der bei `display` angezeigt wird, ist lediglich ein temporäres Abbild von `file2`. Somit können Sie keine Änderungen an einer Datei vornehmen, deren Inhalt Sie mit `display` abrufen.

Nach dieser Methode können Sie nur bei Textdateien vorgehen. Den Inhalt von Katalogen können Sie nicht auf diese Weise abrufen.

Zugriffsschutz für eine vorhandene Datei

Mit dem Kommando `security` können Sie festlegen, in welchem Umfang die übrigen Benutzer auf Ihre Dateien und Kataloge zugreifen können. Ein Teil Ihrer Dateien sollte vielleicht nur von Ihnen genutzt werden können, während andere auch Mitarbeitern, Vorgesetzten oder Lektoren zur Verfügung stehen sollten. Sie können sich selbst, Ihrer Gruppe sowie den anderen Systembenutzern unterschiedliche Zugriffsberechtigungen zum Ändern (Editieren,

Versetzen, Löschen) oder Ausführen/Suchen (Öffnen von Katalogen, Abrufen einer Inhaltsübersicht, Ausführen von ausführbaren Dateien oder Shell-Prozeduren) erteilen. Dazu haben Sie beim Kommando `security` zwei Möglichkeiten:

- Sie können die Master-Kopie einer Datei schützen, indem Sie ihre Zugriffsberechtigungen so festlegen, daß sie von niemandem - auch nicht von Ihnen - geändert werden kann. Ist eine Aktualisierung der Master-Datei notwendig, so können Sie die Zugriffsberechtigungen so ändern, daß Sie die Datei editieren können (Schreibberechtigung). Später können Sie dann wieder den weitergehenden Schutz einrichten.
- Normalerweise sollten sämtliche Dateien oder Kataloge in Ihrem Dateisystem so geschützt werden, daß kein anderer Benutzer Ihre Dateien lesen kann. Vielleicht möchten Sie aber Ihrer Gruppe die Möglichkeit geben, einen bestimmten Text (z.B. eine Hausmitteilung) zu lesen. Andererseits können Sie ein ziemlich "offenes" Dateisystem haben, in dem Sie eine bestimmte Datei (z.B. mit der Gehaltsübersicht) schützen möchten. Die Zugriffsberechtigungen einer derartigen Datei können Sie so abändern, daß sie von niemandem gelesen oder geändert werden kann.

Wenn Sie das Kommando `security` für eine Datei im Dateisystem eines anderen Benutzers ausführen möchten, so müssen Sie den betreffenden Benutzer zunächst im Menü Other Users auswählen und seinen Katalog /home/login eröffnen.

Bei der Änderung von Zugriffsberechtigungen sollten Sie stets diejenigen des jeweils übergeordneten Katalogs beachten. Die Zugriffsberechtigungen für eine Datei bzw. einen Katalog hängen auch mit den Zugriffsberechtigungen für den Katalog zusammen, in dem sie bzw. er enthalten ist. Wenn Sie beispielsweise Ihrer Gruppe die Berechtigung zum Ändern einer Datei erteilen, so sollten dem Katalog, in dem diese Datei enthalten ist, dieselben Zugriffsberechtigungen zugeordnet sein. Die Mitglieder Ihrer Gruppe können die Datei nur ändern, wenn sie auch auf den Katalog zugreifen können.

Wenn z.B. ein Mitglied Ihrer Gruppe eine Datei in Ihrem Dateisystem editieren möchte, müssen Sie zunächst die Zugriffsberechtigungen für diese Datei ändern. Die folgenden Anweisungen zeigen Ihnen, wie Sie die Zugriffsberechtigungen der Datei `workplans` ändern können, die Sie an einer früheren Stelle im Katalog /home/login angelegt haben.

1. Schalten Sie zum Katalog /home/login um und bewegen Sie den Cursor zum Eintrag `workplans`.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]) und betätigen Sie dann die Taste [SECURITY] (bzw. [CTRL-f] [7]).

   Auf dem Bildschirm wird die Security-Schablone angezeigt. Aus dem Titel dieser Schablone geht hervor, daß Sie im Augenblick die aktuellen Zugriffsberechtigungen für die Datei `workplans` vor sich haben. Der Titel der Security-Schablone informiert Sie auch stets darüber, auf welche Datei bzw. welchen Katalog Ihre Änderungen sich auswirken werden.

   Diese Schablone besteht aus zehn Feldern, die die Standard-Zugriffsberechtigungen für Ihr Menü /home/login enthalten. Anhand dieser Felder können Sie sich darüber informieren, wie die drei Zugriffsberechtigungen für Sie (den Eigentümer), die Mitglieder Ihrer Gruppe und alle anderen Benutzer an Ihrem Rechner eingestellt sind. Die zehn Felder sind in vier Abschnitte eingeteilt.

   `Owner:`
   Hier wird der Benutzername des aktuellen Eigentümers angezeigt. Als Eigentümer können Sie die Zugriffsberechtigungen ändern; andernfalls können Sie sich über die Zugriffsberechtigungen nur informieren (nachdem Sie für eine Datei einen anderen Eigentümer eingesetzt haben, können Sie dies nicht mehr rückgängig machen, d.h., Sie können sich nicht selbst wieder als Eigentümer einsetzen).

   `Owner's Permissions:`
   Für die Dateien und Kataloge in Ihrem Dateisystem sind im Normalfall Sie selbst der Eigentümer. Wahrscheinlich möchten Sie für Ihre Dateien und Kataloge sowohl die Leseberechtigung (die Berechtigung zum Abrufen des Dateiinhalts) als auch die Schreibberechtigung (die Berechtigung zum Ändern des Dateiinhalts) und die Ausführungs-/Suchberechtigung (die Berechtigung zum Aufrufen einer ausführbaren Datei bzw. zum Abrufen des Verzeichnisinhalts) haben; hierzu müssen Sie bei allen Zugriffsberechtigungen `yes` angeben.

   Warnung: Kataloge sowie einige durch ausführbare Dateien erstellte Dateien sind nicht dazu bestimmt, mit einem Editor gelesen oder geändert zu werden. Für diese Dateien werden Sie sich wahrscheinlich die Ausführungs-/Suchberechtigung erteilen, nicht aber die Lese- und Schreibberechtigung. Die Ausführungs-/Suchberechtigung des

Eigentümers sollte nur in Ausnahmefällen auf `no` eingestellt werden.

`Group's Permissions:`
Die hier angezeigten Zugriffsberechtigungen gelten für die Mitglieder Ihrer Gruppe. Damit werden die Funktionen angegeben, die die Mitglieder Ihrer Gruppe für eine Datei bzw. einen Katalog durchführen können.

`All Others' Permissions:`
Die hier angezeigten Zugriffsberechtigungen gelten für alle sonstigen Personen, die an Ihrem Rechner angemeldet sind. Sie geben die Funktionen an, die die anderen Benutzer für eine Datei bzw. ein Katalog durchführen können (im Normalfall handelt es sich hierbei um die Lese- oder Ausführungs-/Suchberechtigung, nicht aber um die Schreibberechtigung).

3. Wenn Sie den Mitgliedern Ihrer Gruppe z.B. das Editieren der Datei `workplans` erlauben möchten, müssen Sie den Eintrag bei `Group Write Permissions` von `no` in `yes` ändern.

   Bewegen Sie den Cursor zum Feld `Group Write Permissions`.

4. Betätigen Sie die Taste [CHOICES] (bzw. [CTRL-f] [2]) und beobachten Sie die Änderungen, die auf dem Bildschirm vor sich gehen; der Inhalt dieses Feld wechselt von `no` in `yes`.

5. Betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um den neuen Wert für dieses Feld abzuspeichern.

   Die Security-Schablone wird jetzt ausgeblendet; die Datei `workplans` in Ihrem Katalog /home/login kann jetzt auch von den anderen Mitgliedern Ihrer Gruppe editiert, versetzt oder gelöscht werden.

Zugriff auf die Dateisysteme der anderen Benutzer

In Ihrem Menü Office of login ist u.a. die Option `Other Users` enthalten, die Ihnen den Zugriff auf die Dateisysteme von anderen an Ihrem System arbeitenden Benutzern ermöglicht.

Wenn Sie die Option `Other Users` wählen, wird eine Liste mit Benutzernamen angezeigt; die Wahl eines der Benutzernamen ermöglicht Ihnen den Zugriff auf das Dateisystem dieses Benutzers (das UNIX-Home-Verzeichnis `$HOME`). Im Normalfall werden Sie im Dateisystem dieses Benutzers die Berechtigung zum Abrufen der Inhaltsübersicht einiger Kataloge und des Inhalts einiger Dateien haben, in manchen Fällen sogar zum Ändern eines Katalogs oder einer Datei. Eine Übersicht über die Zugriffsberechtigungen, die ein Benutzer Ihnen für seine Dateien bzw. Kataloge erteilt hat, erhalten Sie über das Kommando `security` (siehe Abschnitt "Zugriffsschutz für eine vorhandene Datei").

Die Option `Other Users` können Sie jetzt oder bei Gelegenheit einmal ausprobieren. Rufen Sie einige der FACE-Kommandos auf, die Sie bisher kennengelernt haben. Bei einem Teil wird es möglich sein, bei anderen nicht, je nachdem, welche Zugriffsberechtigungen der betreffende Benutzer Ihnen zugeteilt hat.

Einstellung Ihrer Office-Parameter

Als neuer FACE-Benutzer brauchen Sie sich nicht mit dem Einrichten Ihres FACE-Office zu befassen - FACE stellt Ihnen die Standard-Konfiguration zur Verfügung, mit der Sie bereits gearbeitet haben. Mit der Zeit wird sich bei Ihnen jedoch der Wunsch nach Änderungen an Ihrem FACE-Office ergeben.

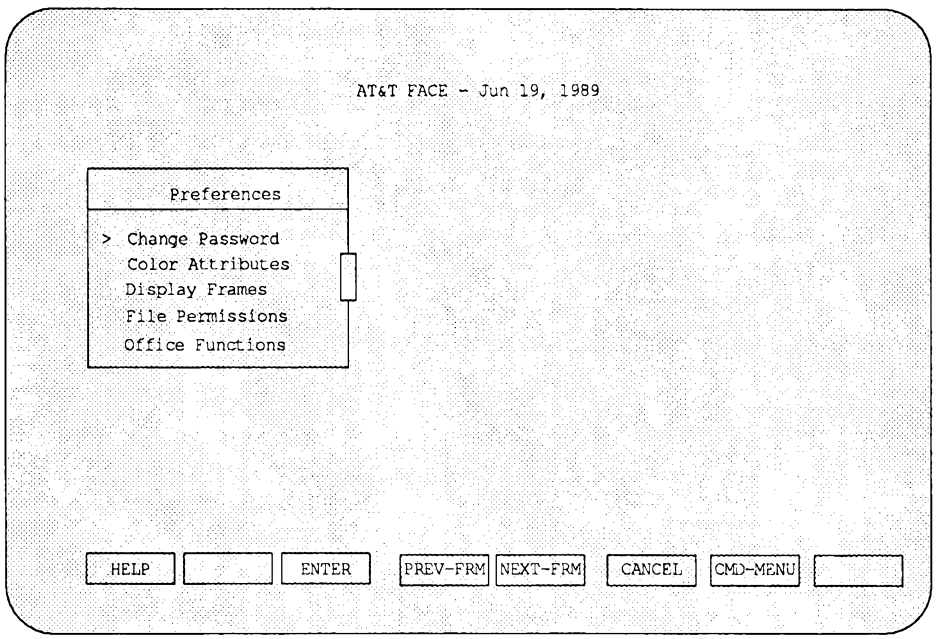
Wie Sie die Anordnung der einzelnen Datei- und Katalog-Einträge ändern können, wurde bereits in einem früheren Abschnitt beschrieben. Auch die Erweiterung oder Einschränkung der Zugriffsberechtigungen, die andere Benutzer für die Dateien und Kataloge in Ihrem Dateisystem haben sollen, haben Sie bereits kennengelernt.

In diesem Abschnitt geht es darum, wie Sie Ihr gesamtes Dateisystem umorganisieren, die Zugriffsberechtigungen für Ihr gesamtes Dateisystem ändern, den Abfall Ihren Anforderungen entsprechend konfigurieren und die Ausschnitte angeben können, die bei Ihrer Anmeldung automatisch eröffnet werden sollen; außerdem wird beschrieben, wie Sie Ihr Paßwort ändern und den Editor

angeben, der standardmäßig zum Editieren von Dateien aufgerufen werden soll.

Unter FACE spricht man hier von Standardparametern (preferences). Die Standard-Parameter, die Sie auf Ihre Bedürfnisse zuschneiden können, sind im Menü Preferences (siehe Bild 5-15) enthalten.

Bild 5-14: Das Menü Preferences



Die Optionen im Menü Preferences:

`Change Password` Hiermit können Sie Ihr Paßwort ändern.

`Color Attributes` Hiermit können Sie die Farbeinstellung für Ihr FACE-Office angeben. Diese Option ist nur dann im Menü Preferences verfügbar, wenn Ihr Terminal mit einem Farbbildschirm ausgestattet ist.

`Display Frames`	Hiermit können Sie angeben, daß bei Ihrer Anmeldung zwei oder mehr Menüs angezeigt werden sollen.
`File Permissions`	Hiermit können Sie die Zugriffsberechtigungen für die (bereits vorhandenen und zukünftig anzulegenden) Kataloge und Dateien festlegen. Dabei können Sie sich selbst (dem Eigentümer), Ihrer Gruppe sowie allen anderen Benutzern die Lese-, Schreib- und Aufführungs-/Suchberechtigung erteilen.
`Office Functions`	Hiermit können Sie angeben, wie lange eine Datei oder ein Katalog vor dem endgültigen Löschen im Abfall bleiben soll, und ob vor dem endgültigen Löschen eine Bestätigung angefordert werden soll. Außerdem können Sie festlegen, welcher Editor standardmäßig aufgerufen werden soll, in welchem Anzeigeformat die Kataloge (Umfang und Reihenfolge) auf dem Bildschirm dargestellt sollen, und ob FACE bei Ihrer Anmeldung automatisch aufgerufen werden soll.

Change Password

Das Paßwort, mit dem Sie sich beim Rechner anmelden, hat eine ähnliche Funktion wie Ihr Büroschlüssel. Sie sollten Ihr Paßwort aus Sicherheitsgründen in regelmäßigen Abständen durch ein neues ersetzen.

Wenn Sie noch kein Paßwort besitzen, Sie schon längere Zeit keines mehr gehabt haben oder andere Benutzer es in Erfahrung gebracht haben, so können Sie es über die Option `Change Password` im Menü Preferences ändern. Dabei gehen Sie folgendermaßen vor: Nach der Wahl der Option `Change Password` geben Sie die angeforderten Informationen ein. Weitere Informationen über UNIX-Paßwörter und ihre Einrichtung finden Sie in Kapitel 2, "Einführung in die Benutzung von UNIX", im Abschnitt "Das Paßwort".

1. Wählen Sie im Menü AT&T FACE die Option `Office of login`.
2. Wählen Sie im Menü Office of login die Option `Preferences`.
3. Wählen Sie im Menü Preferences die Option `Change Password`.

Color Attributes

Hinweis: Eine Änderung der Farbeinstellung ist nur dann möglich, wenn Ihr Terminal mit einem Farbbildschirm ausgestattet ist; andernfalls steht diese Menüoption nicht zur Wahl.

Über die Option Color Attributes können Sie die Standard-Farbeinstellungen Ihrer FACE-Ausschnitte ändern. Im einzelnen können Sie die Farbeinstellung der folgenden Bildschirmattribute ändern:

Bildschirmattribut	Standard-Farbe
Titel:	Zyanblau
Text im Ausschnitt :	Zyanblau
Rand aktiver Ausschnitt:	Rot
Titelleiste aktiver Ausschnitt :	Rot
Titeltext aktiver Ausschnitt:	Zyanblau
Rand inaktiver Ausschnitt:	Blau
Titelleiste inaktiver Ausschnitt:	Blau
Titeltext inaktiver Ausschnitt :	Zyanblau
Markierte Leiste:	Blau
Funktionstasten-Leiste:	Weiß

FACE überprüft jede von Ihnen gewählte Farbeinstellung, damit Sie für ein Attribut und den Hintergrund nicht dieselbe Farbeinstellung wählen.

1. Wählen Sie im Menü Office of login die Option `Preferences`.
2. Wählen Sie im Menü Preferences die Option `Color Attributes`.
3. Bewegen Sie den Cursor zu den verschiedenen Feldern und wählen Sie die gewünschten Farben über die Taste [CHOICES]
4. Abschließend betätigen Sie die Taste [SAVE]

   In der Meldungszeile erscheint jetzt die Bestätigung, daß die Farbeinstellungen aktualisiert worden sind (`The Color Attributes have been updated`).

   Die von Ihnen gewählten Farbeinstellungen werden sofort wirksam, sobald Sie die Taste [SAVE] betätigen. Da die Schablone weiterhin eröffnet bleibt, können Sie weitere Änderungen vornehmen, wenn Sie jetzt nicht mit dem Resultat zufrieden sind.

5. Betätigen Sie die Taste CANCEL, um die Schablone Color Attribute zu schließen.

Anzeige mehrerer Menüs nach der Anmeldung

Bei der Option Display Frames können Sie bis zu vier Menüs angeben, die nach Ihrer Anmeldung automatisch angezeigt werden sollen. Sie können hier den vollständigen Pfadnamen eines beliebigen (auch eines von Ihnen erstellten) Menüs in Ihrem FACE-Office angeben. Um Ihnen diese Arbeit zu erleichtern, können einige häufig benutzte FACE-Menüs über die Funktionstaste CHOICES in die Schablone eingetragen werden:

- Filecabinet
- Mail Services
- Office of login
- Preferences
- Printer Operations*
- Programs**

* Das Menü Printer Operations steht hier nur dann zur Wahl, wenn das Kommando `lp` im UNIX-System installiert ist.

** Das Menü Programs steht hier nur dann zur Wahl, wenn ein Programm in die FACE-Umgebung integriert worden ist.

1. Wählen Sie im Menü Office of login die Option `Preferences`.
2. Wählen Sie im Menü Preferences die Option `Display Frames`.
3. Bewegen Sie den Cursor zu den verschiedenen Feldern und betätigen Sie die Funktionstasten CHOICES (bzw. Hk CTRL-f 2), und ENTER, um die anzuzeigenden Menüs auszuwählen. Sie können aber auch den vollständigen Pfadnamen eines beliebigen Menüausschnitts in Ihrem FACE-Office eintragen.
4. Abschließend betätigen Sie die Taste SAVE. Wenn Sie einen ungültigen Pfadnamen eingegeben oder ein Menü zweimal angegeben haben, gibt FACE eine Meldung aus.

Andernfalls wird in der Meldungszeile eine Bestätigung angezeigt.

Bei Ihrer nächsten Anmeldung werden die von Ihnen angegebenen Menüs angezeigt.

File Permissions

Wenn Sie im Menü Preferences die Option `File Permissions` wählen, können Sie die Standard-Zugriffsberechtigungen für alle neuen Kataloge und Dateien festlegen (d.h. die Zugriffsberechtigungen, die ihnen beim Anlegen automatisch zugeordnet werden). Auf bereits angelegte Kataloge und Dateien wirkt sich diese Option nicht aus.

Im Normalfall haben Sie für Ihre eigenen Kataloge und Dateien sämtliche Zugriffsberechtigungen, während die Mitglieder Ihrer UNIX-Gruppe und die anderen Benutzer Ihre Daten zwar abrufen, aber nicht ändern oder löschen können. Besonders wichtig ist, daß die Ausführungs-/Suchberechtigung für Sie auf `yes` gesetzt sind; ist sie auf `no` gesetzt, so können Sie weder den Inhalt von Katalogen abrufen noch neue Dateien und Kataloge anlegen oder Programme aufrufen.

Vielleicht möchten Sie die Standard-Zugriffsberechtigungen, wie sie von FACE eingestellt worden sind, nicht neu setzen. Sie können jedoch folgendermaßen auf die Schablone File Permissions zugreifen:

1. Wählen Sie in Ihrem Menü Office of login die Option `Preferences`.
2. Wählen Sie im Menü Preferences die Option `File Permissions`.

Befolgen Sie jetzt die Anweisungen zur Änderung der Zugriffsberechtigungen im Abschnitt "Zugriffsschutz für eine vorhandene Datei" in diesem Kapitel. Beachten Sie jedoch, daß die hier angegebenen Zugriffsberechtigungen sich ausschließlich auf die Dateien und Kataloge beziehen, die Sie zukünftig anlegen werden.

Hinweis: Wie bereits an früherer Stelle beschrieben wurde, können Sie die Zugriffsberechtigungen für eine beliebige, bereits vorhandene Einzeldatei bzw. einen beliebigen, bereits vorhandenen Einzelkatalog über das Kommando `security` ändern.

Office Functions

Die Schablone `Office Functions` dürfte Ihnen nicht mehr ganz unbekannt sein. Sie enthält die beiden Felder, die Sie an einer früheren Stelle mit dem Kommando `organize` editiert haben, um die Beschreibung und Reihenfolge der Einträge im Katalog `alpha` zu ändern. Die Schablone Office Functions enthält einige weitere Felder, mit denen Sie Ihr FACE-Office auf Ihre Anforderungen zuschneiden können.

Da Sie sich jetzt mit der Cursor-Steuerung und der Änderung von Feldern in einer Schablone auskennen, möchten Sie bestimmt mehr über die übrigen Standard-Parameter in der Schablone Office Functions erfahren. Dafür gehen Sie folgendermaßen vor:

1. Wählen Sie im Menü Office of login die Option `Preferences`.
2. Wählen Sie die Option `Office Functions`.

Nachfolgend finden Sie eine Beschreibung jedes Felds in der Schablone Office Functions; ändern Sie die Felder nach Bedarf. Beachten Sie, daß Sie über die Taste [HELP] stets Informationen über das aktuelle Feld abrufen können.

```
Delete objects from my Wastebasket after (# of days):
```

In diesem Feld geben Sie die Verweildauer jeder Datei bzw. jedes Katalogs in Ihrem Abfall an, d.h. die Anzahl der Tage, bis sie/er permanent endgültig gelöscht wird. Hier können Sie eine beliebige Zahl im Bereich 1 bis 30 eintragen.

```
Prompt before deleting objects from my Wastebasket (yes or
no):
```

Wenn Sie dieses Feld auf `yes` setzen, so erhalten Sie die Gelegenheit, die Optionen nochmals zu überprüfen und darüber zu entscheiden, ob sie gelöscht oder abgespeichert werden sollen.

Ist dieses Feld auf `no` gesetzt, so werden Sie bei Ihrer Anmeldung nicht darüber informiert, wenn in Ihrem Abfall Dateien oder Kataloge enthalten sind, die an diesem Tag endgültig gelöscht werden sollen. FACE entfernt die Optionen dann ohne weitere Benachrichtigung aus dem Abfall.

Hinweis: Wenn Sie sich an dem Tag, für den das Löschen der Dateien und Kataloge angesetzt war, nicht anmelden, so werden sie nicht gelöscht. Wenn Sie das Löschen der Dateien bzw. Kataloge im Abfall zwar storniert haben, sie aber nicht aus dem Katalog /home/login/WASTEBASKET in einen anderen Katalog versetzt haben, so wird das Löschen der Dateien für Ihre nächste Anmeldung angesetzt.

`Default Editor:`

In diesem Feld können Sie den Editor angeben, den FACE bei der Eröffnung einer Standard-Datei aufruft. Der angegebene Editor muß in Ihrem System installiert sein. Das Feld `Default Editor` enthält standardmäßig den Eintrag `vi`, vorausgesetzt, dieser Editor ist auf Ihrem Rechner installiert; andernfalls enthält es standardmäßig den Eintrag `ed`. Statt des Standard-Editors können Sie ein beliebiges Editierprogramm eintragen, das auf Ihrem Rechner installiert ist.

`Folder Display Format:`

In diesem Feld geben Sie die Bezeichner-Felder an, die in Ihren Katalogen für jede Datei angezeigt werden sollen. Die hier gewählte Einstellung ist sowohl für die zukünftig angelegten Kataloge als auch die bereits vorhandenen gültig (jedoch nicht für diejenigen, die Sie explizit mit dem Kommando `organize` umorganisiert haben). Der Wert, den Sie jetzt in dieses Feld eintragen, ist für sämtliche Dateien und Kataloge in Ihrem FACE-Office gültig (mit Ausnahme derjenigen, die Sie einzeln mit dem Kommando `organize` umorganisiert haben). Weitere Informationen hierzu finden Sie im Abschnitt "Umorganisation des Inhalts eines einzelnen Katalogs".

`Folder Display Order:`

In diesem Feld geben Sie die Reihenfolge an, in der die Optionen innerhalb eines Katalogs aufgeführt sein sollen (in alphabetischer Reihenfolge usw.). In der hier angegebenen Anordnung werden künftig sowohl die neuen Kataloge als auch die bereits vorhandenen angezeigt

(dies gilt jedoch nicht für die Kataloge, die Sie explizit mit dem Kommando `organize` umorganisiert haben. Weitere Informationen hierzu finden Sie im Abschnitt "Umorganisation des Inhalts eines einzelnen Katalogs").

`Confirm at Exit:`

In diesem Feld geben Sie an, ob FACE Sie in einem Ausschnitt zur Eingabe einer Bestätigung auffordern soll, wenn Sie im Menü AT&T FACE die Option `Exit FACE` aufgerufen oder das Kommando `exit` im Command Menu gewählt bzw. in der Kommandozeile eingegeben haben. Ist dieses Feld auf `no` gesetzt, so wird FACE sofort beendet, ohne daß Sie die Möglichkeit haben, Ihre Entscheidung nochmals zu revidieren.

`Invoke FACE at login:`

Hier geben Sie an, ob FACE bei Ihrer Anmeldung automatisch aufgerufen werden soll. Dieses Feld ist in dieser Schablone nur dann vorhanden, wenn Ihr Benutzername unter UNIX eingetragen ist.

Hinweis: Wird FACE bei Ihrer Anmeldung nicht automatisch aufgerufen, so können Sie darauf zugreifen, indem Sie bei der UNIX-Eingabeaufforderung `face` eingeben.

Programs Administration

Über die Option `Programmverwaltung` im Menü `Office of login` können Sie die benutzer-eigenen Programme in Ihrem Menü `Programs` abrufen, ergänzen, ändern und löschen. Diese Einrichtung Ihres FACE-Office wird im nächsten Abschnitt beschrieben; zunächst geht es jedoch um das Aufrufen von Programmen innerhalb Ihres FACE-Office.

(Diese Menüoption bezieht sich nicht auf globale Programme. Informationen zur Verwaltung von globalen Programmen finden Sie unter "System Administration").

Wastebasket

Wenn Sie im Menü `Office of login` die Option `Wastebasket` auswählen, wird im Arbeitsbereich ein neuer Ausschnitt mit dem Titel `/home/login/WASTEBASKET` eröffnet. Die Optionen in diesem Ausschnitt sind die Dateien bzw. Kataloge, die später gelöscht werden sollen und daher von Ihnen in den Abfall gelöscht, kopiert oder versetzt worden sind. Die Dateien und Kataloge im Abfall können Sie wie jede beliebige Datei und jeden beliebigen Katalog in Ihrem FACE-Office manipulieren und verwalten. Beachten Sie jedoch stets, daß die in `/home/login/WASTEBASKET` enthaltenen Dateien und Kataloge von FACE zu einem bestimmten Termin aus dem Menüsystem entfernt werden, wenn Sie das Löschen nicht zuvor stornieren.

Informationen über das Versetzen von Dateien und Katalogen in den Abfall bzw. zum Löschen von Optionen aus dem Abfall finden Sie in den Abschnitten "Löschen einer Datei bzw. eines Katalogs" und "Löschen einer Datei bzw. eines Katalogs rückgängig machen". Im Abschnitt "Ändern von weiteren Standardparametern" finden Sie Informationen darüber, wie Sie die Verweildauer der Optionen in Ihrem Abfall festlegen können und Sie zur Anzeige einer entsprechenden Benachrichtigung anweisen können, wenn das Löschen der Optionen unmittelbar bevorsteht. Informationen darüber, wie Sie den Umfang der angezeigten Informationen steuern können, finden Sie im Abschnitt "Umorganisation des Inhalts eines einzelnen Katalogs".

Auch über das Hilfesystem können Sie Informationen über den Abfall abrufen.

1. Wählen Sie im Menü Office of login die Option `WASTEBASKET`.
2. Betätigen Sie die Taste [HELP] (bzw. [CTRL-f] [1]), um `Help on Wastebasket` abzurufen.

Hinweis: Die Dateien und Kataloge werden nicht gelöscht, wenn Sie sich nicht an Ihrem Rechner anmelden. Dies gilt selbst dann, wenn das Löschen für diesen Tag angesetzt worden ist. Wenn Sie das Löschen storniert haben, die Dateien dann aber nicht aus dem Katalog `WASTEBASKET` in einen anderen Katalog versetzen, so wird das Löschen dieser Dateien für jede Anmeldung angesetzt, die Sie nachfolgend durchführen.

Weitere FACE-Funktionen

Printer Operations

Wenn auch seitens der Software-Industrie immer wieder das "papierlose Büro" propagiert wird, weiß jeder Rechnerbenutzer, daß dies noch in weiter Ferne liegt.

Wenn Sie den Inhalt Ihrer Dateien auf Papier ausdrucken möchten, so rufen Sie die FACE-Funktion `Printer Operations` und das Kommando `print` auf. Mit `Printer Operations` können Sie das Druck-Kommando Ihres Systems an Ihre Anforderungen anpassen, und mit dem Kommando `print` können Sie die Ausdrucke erstellen. Unter FACE entspricht das Kommando `print` standardmäßig dem UNIX-Kommando `lp`.

Hinweis: Vor dem Durcharbeiten der nächsten Abschnitte sollten Sie sich bei Ihrem Systemverwalter erkundigen, ob auf Ihrem Rechner zum Ausdrucken das Kommando `lp` benutzt wird; andernfalls sollten Sie das installierte Druck-Kommando in Erfahrung bringen.

Printer Status

Bevor Sie eine Datei ausdrucken können, müssen Sie den Namen des betreffenden Druckers kennen; außerdem müssen Sie sicherstellen, daß er in Betrieb ist. Diese Informationen können Sie über die Option `Printer Status` im Menü Printer Operations abrufen. Dabei gehen Sie folgendermaßen vor:

1. Wählen Sie im Menü AT&T FACE die Option `Printer Operations`.
2. Wählen Sie im Menü Printer Operations die Option `Printer Status`.
3. Auf dem Bildschirm wird ein Ausschnitt mit den verfügbaren Druckern angezeigt.

   Wenn keine Drucker eingerichtet worden sind oder das Kommando `lp` auf Ihrem Rechner nicht installiert ist, so wird in einem Ausschnitt mit dem Titel `Warning` eine entsprechende Meldung angezeigt. In diesem Fall wenden Sie sich an Ihren Systemverwalter.

Die Ausgabe von `Printer Status` informiert Sie darüber, welche Drucker Ihnen aktuell zur Verfügung stehen.

Hinweis: Wenn es im Augenblick überhaupt keine Drucker gibt bzw. keine Drucker, die Druckanforderungen entgegennehmen, so können Sie keine Dateien ausdrucken, bis Ihr Systemverwalter diese Situation bereinigt hat. In diesem Fall wird beim Versuch, eine Datei auszudrucken, eine Fehlermeldung angezeigt.

Wenn es zwar einen betriebsbereiten Drucker gibt, dennoch aber keine Ausdrucke erstellt werden, so wenden Sie sich an Ihren Systemverwalter.

Print Options

In der nächsten Übung wird erklärt, wie Sie die Optionen und Argumente des Kommandos `lp` ändern oder ergänzen können (z.B. die Option, mit der `lp` zum Ausdrucken der Datei über einen bestimmten Drucker angewiesen wird). Nach derselben Methode können Sie auch `Print Command #2` und `Print Command #3` ändern.

Change Print Options for Login Die Optionen des Kommandos `lp` können Sie mit der Option `Print Options` im Menü Printer Operations ändern.

1. Wählen Sie im Menü AT&T FACE die Option `Printer Operations`.
2. Wählen Sie im Menü Printer Operations die Option `Print Options`.
3. Auf dem Bildschirm erscheint eine Schablone mit dem Titel `Change Print Options for login`. Die Schablone besteht aus drei Feldern, die den ersten drei Optionen im Menü des Kommandos `print` entsprechen.

   Hinweis: Mit der Eingabe eines Werts in dieses Feld wird der vorhandene Wert ausgeblendet. Sie sollten sich das ursprüngliche Kommando notieren, so daß Sie die Optionen später bei Bedarf wieder parat haben.

4. Ändern Sie die Angaben bei `Print Command #1:` ab bzw. ergänzen Sie sie.

   Da FACE die Argumente bzw. Optionen, mit denen Sie Ihre Dateien über das Kommando `lp` ausdrucken möchten, nicht automatisch ermitteln kann, müssen Sie die Optionen für `lp` gegebenenfalls ergänzen bzw. ändern. Ausführliche Informationen zum Kommando `lp` finden Sie in Kapitel 8 dieses Leitfadens, "Der Druck-Service LP", sowie im *Referenzhandbuch für Benutzer* unter dem Eintrag "lp(1)."

Hinweis: Den Namen der auszudruckenden Datei dürfen Sie nicht in das Feld `Print Command #1:` eintragen.

5. Betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um den neuen Wert im Feld `Print Command #1:` abzuspeichern. Sie können aber auch die Taste [CANCEL] betätigen, um diese Schablone ohne Änderungen zu verlassen.

   Sie kehren jetzt zum Menü Printer Operations zurück.

Im folgenden wird beschrieben, wie Sie eine Übersicht über die Druckanforderungen abrufen können, die in der Warteschlange für den/die Drucker enthalten sind. Über die Option `Printer Queue` können Sie sich auch darüber informieren, welcher der aktuell verfügbaren Drucker am wenigsten ausgelastet ist oder wann Ihr Druckauftrag an der Reihe ist.

1. Wählen Sie im Menü AT&T FACE die Option `Printer Operations`.
2. Wählen Sie im Menü Printer Operations die Option `Printer Queue`.

   Sobald das Menü Printer Queue angezeigt wird, befolgen Sie die Anweisungen auf dem Bildschirm; über die Taste [HELP] (bzw. [CTRL-f] [1]) können Sie zusätzliche Informationen abrufen.

   Hinweis: Befinden sich aktuell keine Druckanforderungen in der Warteschlange, so wird eine Warnung ausgegeben.

   Es können maximal 30 Druckanforderungen gleichzeitig dargestellt werden.

Ausdrucken einer Datei

1. Schalten Sie zu dem Katalog um, in dem die auszudruckende Datei enthalten ist.
2. Bewegen Sie den Cursor zu der betreffenden Datei.

3. Wählen Sie im Command Menu die Option `print`, oder geben Sie in der Kommandozeile `print` ein und betätigen Sie die Taste ENTER.
4. Wählen Sie eines der drei Druck-Kommandos, die im Print-Menü aufgelistet sind.

   Der Bildschirm wird gelöscht, dann werden Sie in einer Meldung über die Auftragsnummer informiert, die Ihrem Druckauftrag zugeordnet worden ist.
5. Betätigen Sie die Taste ENTER, um zu Ihrem FACE-Office zurückzukehren.

Hinweis: Die letzte Option im Print-Menü ist `Print Options`. Über diese Option können Sie auf die Schablone `Change Print Options for login` zugreifen, die im Abschnitt "Print Options" beschrieben worden ist. Diese Option wurde als Arbeitserleichterung zusätzlich in dieses Menü eingebracht.

Programs

Ein (Anwendungs-)Programm ist eine ausführbare Datei, bei der es sich einfach um eine einzeilige, von Ihnen eingegebene Shell-Prozedur handeln kann (Informationen über das Erstellen einer Shell-Prozedur finden Sie in Kapitel 9), um ein UNIX-Kommando (ausführliche Informationen über die UNIX-Kommandos finden Sie im *Referenzhandbuch für Benutzer*) oder ein komplexes, mit vielen Funktionen ausgestattetes Tabellenkalkulationsprogramm. Für die unterschiedlichen Rechnersysteme sind unterschiedliche Programme verfügbar.

Über das Menü Programs können Sie auf andere, auf Ihrem Rechner installierte Anwendungsprogramme zugreifen, ohne das FACE-Office zu verlassen. Hierzu gehören unmittelbar nach der Installation von FACE eine (englische) Rechtschreibprüfung und ein Mail-Programm (bei Ihnen sind möglicherweise weitere Programme aufgeführt, die von Ihrem Systemverwalter zur allgemeinen Nutzung installiert worden sind). Um ein Programm im Menü Programs aufzurufen, müssen Sie es einfach auswählen.

Mail Services

Zu den Programmen, die Sie im Menü Programs aufrufen können, gehört beispielsweise das Mail-Programm `Mail Services`, mit dem Sie Nachrichten abschicken und empfangen können. `Mail Services` ruft wiederum das UNIX-Kommando `mailx` auf. Im folgenden wird das Senden und Empfangen von Nachrichten mit `Mail Services` kurz beschrieben. Informationen über den gesamten Funktionsumfang von `mailx` finden Sie in diesem Leitfaden in Kapitel 11, "Die elektronische Post", oder im *Referenzhandbuch für Benutzer* unter dem Eintrag `mailx`(1). Nachfolgend finden Sie eine kurze Einführung in die Benutzung der beiden Optionen im Menü Mail Services:

`New Mail` Über diese Menüoption können Sie die Nachrichten abrufen, die andere Benutzer an Sie geschickt haben, und sie lesen. Bei der Wahl dieser Option wird der Bildschirm gelöscht, und die für Sie eingegangenen Nachrichten werden in chronologischer Reihenfolge aufgelistet. Wenn keine Post für Sie angekommen ist, so wird in der Meldungszeile eine entsprechende Meldung angezeigt. Über das Fragezeichen `(?)` können Sie Hilfe abrufen. Über `q` können Sie die Menüoption verlassen und zur FACE-Schnittstelle zurückkehren.

`Send Mail` Über diese Menüoption können Sie Nachrichten an andere Benutzer senden, die an diesem oder einem anderen System arbeiten. Auf dem Bildschirm wird eine Schablone angezeigt, in der Sie den Benutzernamen des Empfängers eintragen können. Nachdem Sie die Taste SAVE betätigt haben, wird der Bildschirm gelöscht; Sie können jetzt Ihre Nachricht eintragen. Über eine Tilde in Kombination mit einem Fragezeichen `(~?)` können Sie Hilfe abrufen. Zum Abschluß der Nachricht geben Sie eine Tilde in Kombination mit einem Punkt `(~.)` ein; damit wird die Nachricht abgeschickt, und Sie kehren zur FACE-Schnittstelle zurück. Eine andere Möglichkeit besteht darin, `mailx` durch die Eingabe einer Tilde in Kombination mit dem Kleinbuchstaben `x` `(~x)` zu verlassen, ohne die Nachricht abzuschicken.

Wenn die Nachricht an einen Benutzer geschickt werden soll, der an einem "fernen" Rechner arbeitet, so muß der Benutzername im Format `system_name!benutzer_name` eingegeben werden.

So würde die Nachricht bei der Eingabe `merc!dlt` an den Benutzer mit dem Benutzernamen `dlt` geschickt, der am System `merc` arbeitet. Der Systemname muß bei der Funktion Mail Setup im Menü Administration angegeben worden sein.

Nachdem Sie die Taste [SAVE] betätigt haben, wird der Bildschirm gelöscht, und `mailx` fordert Sie zur Eingabe des Titels der Nachricht ein; danach können Sie mit der Eingabe Ihrer Nachricht beginnen. Über `~?` können Sie Hilfe abrufen. Bei der Eingabe von `~.` wird Ihre Nachricht an den Empfänger geschickt, und Sie kehren zur FACE-Schnittstelle zurück. Wenn Sie `mailx` verlassen möchten, ohne eine Nachricht abzuschicken, geben Sie `~x` ein.

Spell Checker

Über die Menüoption `Spell Checker` im Menü Programs rufen Sie das UNIX-Programm `spell` auf, mit dem Sie Ihre Dateien auf (englische) Schreibfehler überprüfen können. Dabei gehen Sie folgendermaßen vor:

1. Wählen Sie im Menü Programs die Option `Spell Checker`.

   Auf dem Bildschirm wird eine Schablone mit dem Titel `Spell Checker` angezeigt, in der Sie den Namen der zu überprüfenden Datei angeben können.

2. Geben Sie den Pfadnamen der betreffenden Datei ein.

   Hierbei kann es sich um den absoluten Pfadnamen (Zugriffspfad ab `root`) oder den relativen Pfadnamen (Zugriffspfad ab dem aktuellen Katalog) der Datei handeln. Die Eingabe ausschließlich eines Dateinamens ist nur dann möglich, wenn diese Datei in dem Katalog enthalten ist, in dem Sie sich *vor* dem Umschalten zum Menü Programs befunden haben (über die Taste [CHOICES] können Sie eine Auflistung derartiger Dateien abrufen).

3. Betätigen Sie die Taste [SAVE].

   Im Arbeitsbereich wird ein neuer Ausschnitt eröffnet, in dem mögliche Fehler in der Datei aufgeführt sind.

4. Betätigen Sie die Taste CANCEL, um den Ausschnitt zu schließen.

Anhand der Fehlerliste können Sie die Datei mit einem Editor aufrufen und die fehlerhaften Wörter korrigieren.

Arbeiten mit anderen Programmen

Die nachfolgend beschriebene Vorgehensweise eignet sich für jedes Programm, das im Menü Programs aufgeführt ist.

Hinweis: Bei einigen Programmen werden Anweisungen zum Erstellen und Bearbeiten von vorhandenen Dateien angezeigt. Informieren Sie sich vor dem Aufrufen des Programmes auf jeden Fall darüber, wie Sie das gewählte Programm benutzen, Ihre Arbeit abspeichern und das Programm beenden können.

1. Wählen Sie im Menü Office of login die Option `Programs`.
2. Wählen Sie im Menü Programs das Programm, mit dem Sie arbeiten möchten.

   Das Programm wird jetzt aufgerufen.
3. Wenn Argumente in die Aufrufzeile des Programms eingegeben werden müssen, so wird die Eingabeaufforderung `Enter arguments for program:` angezeigt. Geben Sie die Argumente ein und betätigen Sie die Taste ENTER.

   Die Dateien können Sie über ihren absoluten Pfadnamen (Zugriffspfad ab `root`) oder ihren relativen Pfadnamen (Zugriffspfad ab dem aktuellen Katalog) angeben.
4. Das Aufrufen des Programms wird jetzt abgeschlossen.
5. Nachdem Sie ein Programm verlassen haben, werden Sie zur Betätigung der Taste ENTER aufgefordert.

   Danach wird wieder der FACE-Bildschirm angezeigt.

Programs Administration

Man unterscheidet zwischen globalen und benutzer-eigenen Programmen. Globale Programme werden unter FACE durch den FACE-Systemverwalter (bzw. durch die Person(en), die mit den Privilegien des FACE-Systemverwalters ausgestattet ist/sind), eingerichtet und verwaltet (Informationen über das Einrichten von globalen Programmen finden Sie unter "FACE Administration"). Benutzer-eigene Programme werden durch die einzelnen Benutzer unter FACE eingerichtet und verwaltet. Im Menü Programs sind sowohl die benutzer-eigenen als auch die globalen Programme aufgeführt; allerdings sind die globalen Programme im Menü Programs jedes Benutzers enthalten, während Ihre benutzer-eigenen Programme ausschließlich in *Ihrem* Menü Programs erscheinen.

Benutzer-eigene Programme

Ein benutzer-eigenes Programm können Sie über die Option `Programs Administration` im Menü Office of login in Ihr Menü Programs eintragen.

Da die Wahl eines Programms stets gleich abläuft, unabhängig davon, ob es sich um ein globales oder ein benutzer-eigenes Programm handelt, werden sie im Menü Programs nicht besonders gekennzeichnet.

Hinweis: Nachdem Sie im Menü Programs ein Programm eingerichtet, entfernt oder sonstige Änderungen vorgenommen haben, wird das Menü automatisch aktualisiert; beim nächsten Aufrufen präsentiert sich das Menü dann bereits mit seinem neuen Inhalt. Wird das Menü Programs aktuell auf Ihrem Bildschirm angezeigt, so können Sie es sofort aktualisieren, indem Sie es zum aktiven Ausschnitt machen und das Kommando `update` aufrufen.

Eintragen eines Programms

Die Programme werden im Normalfall von Ihrem Systemverwalter in Ihr Menü Programs eingetragen. Sie selbst werden hauptsächlich solche Programme in Ihr Menü Programs einfügen, die von Ihnen geschrieben worden sind oder für die anderen Benutzer nicht von Interesse sind.

In dieser Übung lernen Sie, wie Sie ein benutzer-eigenes Programm in Ihr Menü Programs einfügen können.

1. Wählen Sie im Menü AT&T FACE die Option `Office of login`.
2. Wählen Sie im Menü Office of login die Option `Programs Administration`.
3. Wählen Sie im Menü Programs Administration die Option `Add Programs`.

   Auf dem Bildschirm wird eine Schablone mit dem Titel `Add Programs` angezeigt. Diese Schablone enthält vier Felder, in denen Sie das einzufügende Programm angeben können.

   Beachten Sie, daß einige der Felder bereits Standard-Werte enthalten.
4. Das erste Feld ist `Program Menu Name:`.

   Geben Sie den Namen des Programms ein, das im Menü Programs angezeigt werden soll, und betätigen Sie die Taste ENTER.

   Der Name, den Sie eingeben, darf maximal 42 Zeichen lang sein und sollte Aufschluß über die Funktion des Programms geben.
5. Das zweite Feld ist `Name of Command:`.

   Geben Sie den absoluten Pfadnamen des Programms ein und betätigen Sie die Taste ENTER (wenn das Programm im Zugriffspfad der Umgebungsvariablen $PATH enthalten ist, so genügt die Eingabe des Programmnamens).

   Wenn Sie den absoluten Pfadnamen des Programms nicht kennen, so ermitteln Sie ihn anhand Ihrer schriftlichen Unterlagen, oder erkundigen Sie sich beim Systemverwalter.
6. Das dritte Feld ist `Working Directory:`. Dieses Feld enthält einen Standardwert, das sogenannte aktuelle Arbeitsverzeichnis. Bei den meisten Benutzern wird im Feld `Working Directory:` der Wert `/home/login` enthalten sein (also der absolute Pfadname Ihres bei `$HOME` angegebenen Katalogs).

   Das Feld `Working Directory:` enthält den Katalog, in dem das Programm aufgerufen werden soll, und in dem die vom Programm erzeugten Dateien abgespeichert werden sollen. Wenn von dem Programm keine Dateien generiert werden oder Sie im aktuellen Katalog bleiben möchten, so geben Sie lediglich einen Punkt (`.`) (das Kürzel für "den aktuellen Katalog") ein.

Beim Aufrufen dieses Programms im Menü Programs versetzt FACE Sie automatisch in den Katalog, den Sie in diesem Feld angegeben haben. Nach dem Verlassen des Programms kehren Sie automatisch wieder zum Menü Programs zurück.

Geben Sie einen Punkt (.) oder den absoluten Pfadnamen eines Katalogs ein und betätigen Sie die Taste [ENTER].

7. Das vierte Feld ist `Prompt for arguments:`.

   Soll der Benutzer dieses Programms beim Aufrufen Argumente (z.B. einen Dateinamen) oder eine Option angeben, so muß dieses Feld auf `yes` gesetzt werden; andernfalls setzen Sie es auf `no`.

8. Betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um die neuen Werte in der Schablone Add Programs abzuspeichern.

Beim nächsten Aufrufen der Option `Programs` im Menü Office of login wird das eben eingetragene Programm aufgeführt. Mit der Wahl des Programmnamens rufen Sie das Programm gleichzeitig auf.

Abrufen von benutzer-eigenen Programmen

Sie können auch eine Liste abrufen, in der ausschließlich die benutzer-eigenen Programme enthalten sind, die Sie über die Option `List Programs` im Menü Programs Administration eingetragen haben.

1. Wählen Sie im Menü Programs Administration die Option `List Programs`.

   Auf dem Bildschirm wird das Menü List Personal Programs angezeigt; beachten Sie, daß die globalen Programme nicht aufgeführt sind. In diesem Menü können Sie, ebenso wie im Menü Programs, Programme auswählen.

   Wenn Sie in Ihr Menü Programs keine benutzer-eigenen Programme eingetragen haben, wird in der Meldungszeile die Meldung `No Programs Installed` angezeigt.

2. Betätigen Sie die Taste [CANCEL] (bzw. [CTRL-f] [6]), um zum Menü Programs Administration zurückzukehren.

Ändern eines benutzer-eigenen Programms

Die benutzer-eigenen Programme dürfen Sie im Gegensatz zu den globalen Programmen (in Ihrem Menü Programs) beliebig ändern. Dabei gehen Sie folgendermaßen vor:

1. Wählen Sie im Menü Programs Administration die Option `Modify Programs`.

   Auf dem Bildschirm erscheint das Menü Modify Personal Programs mit sämtlichen benutzer-eigenen Programmen, die Sie eingetragen haben.

2. Wählen Sie im Menü Modify Personal Programs das Programm, das Sie ändern möchten.

3. Auf dem Bildschirm wird die Schablone Modify Programs angezeigt; die Felder dieser Schablone enthalten die Werte, die Sie beim Eintragen dieses Programms in Ihr Menü Programs eingegeben haben. Zum Editieren dieser Schablone können Sie sämtliche Cursor- und Editiertasten für Schablonen benutzen (siehe "Arbeiten mit Schablonen" in diesem Kapitel.

4. Betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um Ihre Änderungen abzuspeichern.

   Sie kehren jetzt zum Menü Programs Administration zurück.

Entfernen eines benutzer-eigenen Programms

Ihre benutzer-eigenen Programme können Sie im Gegensatz zu den globalen Programmen jederzeit aus dem Menü Programs entfernen. Dabei gehen Sie folgendermaßen vor:

1. Wählen Sie im Menü Programs Administration die Option `Remove Programs`.

   Auf dem Bildschirm erscheint das Menü Remove Personal Programs mit sämtlichen benutzer-eigenen Programmen, die Sie eingetragen haben.

2. Wählen Sie das Programm, das Sie entfernen möchten.

3. Auf dem Bildschirm erscheint jetzt der Ausschnitt `Confirm Delete of Program` mit der Meldung `You are about to delete program`.

- Wenn Sie das Programm doch nicht entfernen möchten, so betätigen Sie die Taste CANCEL (bzw. CTRL-f 6).
- Andernfalls betätigen Sie die Taste CONT (bzw. CTRL-f 3).

Sie kehren jetzt zum Menü Programs Administration zurück.

FACE Administration

Hinweis: Die Systemverwalter, die FACE installiert haben, sollten sich vor der Durchführung der folgenden Prozeduren über die Programme Operations, Administration und Maintenance selbst als FACE-Benutzer eintragen (siehe Abschnitt "Eintragen eines neuen FACE-Benutzers").

Ein FACE-Benutzer kann die folgenden Prozeduren nur dann durchführen, wenn er die Privilegien des Systemverwalters hat und das Paßwort des Systemverwalters kennt. Wenn Sie diese Privilegien benötigen, so wenden Sie sich an Ihren Systemverwalter.

Die Verwaltung von FACE ist aus zwei Gründen sehr einfach: Zum einen erfolgt sie über das UNIX-Betriebs-, Verwaltungs- und Wartungssystem OA&M (Operations, Administration, und Maintenance System), ein sehr bedienungsfreundliches, menügesteuertes Programm, in dem Sie die meisten Verwaltungsaufgaben im Dialogbetrieb erledigen können. Zum anderen müssen vom FACE-Systemverwalter nur Funktionen aus vier Gruppen erledigt werden: Benutzerverwaltung (User Administration), Verwaltung von globalen FACE-Programmen (Global Programs), Verwaltung der Post-Funktionen (Mail Services) sowie Verwaltung der Rechtschreibprüfung (Spell Checker).

Die Benutzer-Verwaltung besteht unter FACE aus vier Funktionen: das Eintragen von Benutzern als FACE-Benutzer (Einrichten der FACE-Umgebung für ihre Benutzernamen), die Änderung der FACE-Umgebung der Benutzer, das Entfernen von FACE-Benutzern (das Aufheben ihrer Zugriffsberechtigung für FACE) sowie das Abrufen einer FACE-Benutzerübersicht.

Zur Verwaltung von globalen Programmen gehören vier Funktionen: Das Einrichten, Entfernen, Ändern oder Abrufen der globalen Programme, die allen FACE-Benutzern zur Verfügung stehen.

Die Verwaltung der Funktionen Mail Services und Spell Checker besteht jeweils aus dem Eintragen und Entfernen der jeweiligen Funktion.

FACE User Administration

Eintragen eines neuen FACE-Benutzers

Für einen Benutzer können Sie nur dann eine FACE-Umgebung einrichten, wenn er bereits einen UNIX-Benutzernamen hat.

1. Wählen Sie im Menü AT&T FACE die Option `System Administration`.

   Der Bildschirm wird jetzt gelöscht, und Sie werden zur Eingabe des Systemverwalter-Paßworts aufgefordert. Bei der Eingabe des falschen Paßworts kehren Sie zu Ihrem FACE-Office zurück. Bei der Eingabe des korrekten Paßworts wird die Oberfläche des Menüs UNIX System V Administration dargestellt. Diese Oberfläche wird ähnlich wie FACE bedient; so können Sie zur Cursorsteuerung und Auswahl von Menüoptionen, zum Editieren von Schablonen und zur Ausführung von Kommandos häufig dieselben Tasten wie unter FACE benutzen.

2. Wählen Sie im Menü UNIX System V Administration die Option `applications`.

3. Wählen Sie im Menü Administration for Available Applications die Option `FACE`.

4. Wählen Sie im Menü FACE Administration die Option `users`.

5. Wählen Sie im Menü FACE User Administration die Option `add`.

   Auf dem Bildschirm erscheint ein Menü mit dem Titel `Add a FACE Environment for a UNIX System User`.

6. Füllen Sie die vier Felder dieser Schablone aus bzw. ändern Sie sie.

   Wenn Sie den Cursor zu einem Feld bewegen, wird in der Meldungszeile eine Erklärung angezeigt, wie Sie es ausfüllen können.

   Hinweis: Wenn Sie in das Feld `Show System Administration in FACE Menu: yes` eintragen, erscheint im Menü AT&T FACE des Benutzers die Option `System Administration`; der Benutzer kann die Menüoption `System Administration` nur dann aufrufen, wenn er das Paßwort des UNIX-Systemverwalters kennt.

Dieses Paßwort wird den jeweiligen Benutzern am besten mündlich mitgeteilt.

Wenn Sie in das Feld `Provide UNIX System Access: yes` eintragen, so erscheint im Menü AT&T FACE des Benutzers die Option `UNIX System`; der Benutzer kann dann über diese Menüoption eine Ganzseiten-UNIX-Subshell aufrufen.

7. Betätigen Sie abschließend die Taste [SAVE].

   Sie werden jetzt in einer Meldung darüber informiert, welche Rechte der Benutzer unter FACE hat.

8. Betätigen Sie die Taste [CANCEL], um zur Schablone Add a FACE Environment for a UNIX System User zurückzuschalten.

9. Wenn Sie keine Benutzer mehr eintragen möchten, so betätigen Sie die Taste [CMD-MENU] und wählen im Menü UNIX System V Administration die Option `exit`. Sie werden jetzt zur Betätigung der Taste [ENTER] aufgefordert, um zu Ihrem FACE-Office zurückzukehren.

Dem neuen Benutzer wird jetzt eine vollständige FACE-Umgebung zur Verfügung gestellt; dazu gehört auch die Datei `.profile` in seinem Katalog `$HOME`, die FACE bei seiner Anmeldung automatisch aufruft, vorausgesetzt, er hat in das Feld `Invoke FACE at Login: yes` eingetragen. Die nachfolgenden Kataloge werden als Sohnverzeichnisse des `$HOME`-Katalogs des neuen Benutzers angelegt, wenn sie noch nicht vorhanden sind:

`WASTEBASKET` Im Katalog `$HOME/WASTEBASKET` (dem sogenannten Abfall) werden gelöschte Dateien und Kataloge aufbewahrt, bis sie endgültig aus dem Dateisystem gelöscht werden.

`bin` Der Katalog `$HOME/bin` enthält die Shell-Prozeduren zum Aufrufen der Anwendungsprogramme, die der Benutzer als benutzer-eigene Programme installiert hat. Ist das Verzeichnis `$HOME/bin` aus irgendeinem Grund gelöscht worden, so wird es von FACE automatisch angelegt, wenn ein Benutzer ein benutzer-eigenes Programm installiert, und eine entsprechende Meldung ausgegeben.

`pref` Das Verzeichnis `$HOME/pref` enthält die Umgebungsvariablen, die von FACE benutzt werden; hier können Dateien mit zusätzlichen FMLI-Prozeduren für das Menü Office of Login und die Programs-Menüs des Benutzers enthalten sein. FMLI, eine Programmiersprache und ein Kommandointerpreter, kann von Programmierern zur Erstellung einer Schablone und Menüschnittstelle wie FACE oder zur Konfiguration von FACE benutzt werden. Weitere Informationen über die FMLI-Prozduren finden Sie im *Character User Interface Programmer's Guide.*

`tmp` Im Verzeichnis `$HOME/tmp` können Daten vorübergehend aufbewahrt werden. Ohne dieses Verzeichnis kann FACE nicht aufgerufen werden.

Sobald die Benutzer mit dem Anlegen von Dateien und Katalogen in AT&T FACE, `$HOME`, `$HOME/WASTEBASKET` und anderen Katalogen beginnen, legen sie auch eine Datei namens `.ott` mit einer Objekttyp-Tabelle an, die FACE über den Inhalt des Katalogs informiert. Der Benutzer hat im Normalfall mit den `.ott`-Dateien nichts zu tun; werden sie versehentlich einmal gelöscht, so werden sie von FACE automatisch wieder angelegt.

Die Benutzer sollten darauf aufmerksam gemacht werden, daß sie am Inhalt einer `.ott`-Datei nichts ändern dürfen.

Hinweis: Wechselt ein Benutzer in einen Katalog mit einer leeren `.ott`-Datei, so scheint dieser Katalog leer zu sein, selbst wenn darin Dateien und Kataloge enthalten sind. Wenn ein Benutzer versehentlich eine leere `.ott`-Datei anlegt, so sollte er die leere Datei löschen; FACE legt dann eine neue an, und die Dateien in diesem Katalog werden wie normal im Ausschnitt angezeigt (vorausgesetzt, der Pfadname des Katalogs enthält nicht mehr Zeichen als zulässig (siehe Abschnitt "Pfadnamen in FACE-Kommandos" an einer früheren Stelle dieses Kapitels).

Das Format der `.ott`-Dateien und einiger anderer Dateien, auf die durch FACE zugegriffen wird, werden im *Referenzhandbuch für Systemverwalter* unter dem Eintrag FACE im Abschnitt "Dateiformate(4)" beschrieben.

Ändern einer FACE-Benutzerumgebung

Die FACE-Umgebung eines Benutzers können Sie über die Option `modify` im Menü FACE User Administration ändern. Bei der Wahl von `modify` wird die Schablone angezeigt, die auch beim Eintragen eines neuen Benutzers erscheint. Geben Sie den Benutzernamen des betreffenden Benutzers ein; in den übrigen Feldern werden dann die aktuellen Werte für diesen Benutzer angezeigt. Zur Änderung der Werte können Sie die üblichen Cursor- und Editiertasten für Schablonen benutzen. Nach Abschluß der Änderungen wird in einer Meldung eine Bestätigung angezeigt.

Entfernen eines FACE-Benutzereintrags

Sie können den Eintrag eines Benutzers aus dem FACE-System entfernen. Der Benutzer hat dann zwar noch seinen Benutzernamen, jedoch keine Zugriffsberechtigung mehr für FACE.

1. Wählen Sie im Menü FACE User Administration die Option `remove`.

   In der Schablone Remove FACE Environment for a FACE User wird das Feld `User's login ID:` angezeigt.

2. Geben Sie den Benutzernamen des Benutzers ein, den Sie entfernen möchten.

   Sie können jetzt über die Taste [CHOICES] ein Menü aufrufen, in dem sämtliche am Rechner arbeitenden Benutzer aufgeführt sind.

3. Wenn Sie jetzt die Taste [SAVE] betätigen, wird der Benutzer aus dem FACE-System entfernt. Eine andere Möglichkeit ist, über die Taste [CANCEL] zum Menü FACE User Administration zurückzukehren, ohne den Benutzer aus dem FACE-System zu entfernen.

   Bei der Betätigung der Taste wird [SAVE] ein Ausschnitt mit dem Ergebnis der Prozedur angezeigt.

4. Schalten Sie zum Menü UNIX System V Administration um, wenn Sie einen weiteren Benutzer aus dem System entfernen möchten, oder betätigen Sie die Taste [CMD-MENU] und wählen Sie `exit`, um das Systemverwaltungs-Menü zu verlassen.

FACE-Benutzerübersicht

Sie können eine Übersicht über sämtliche am Rechner arbeitenden FACE-Benutzer abrufen, indem Sie im Menü FACE User Administration die Option `list` auswählen. Auf dem Bildschirm erscheint dann ein Ausschnitt mit dem Titel `FACE Users on This System`, in dem die Benutzernamen sämtlicher Benutzer aufgeführt sind, die als FACE-Benutzer eingetragen sind. Über diese Option können Sie sich einen Überblick über die FACE-Benutzernamen verschaffen, bevor Sie eine der Menüoptionen `add`, `modify` oder `remove` auswählen.

Verwaltung von globalen Programmen

Unter einem globalen Programm versteht man ein Programm, auf das jeder FACE-Benutzer auf dem Rechner Zugriff hat und dessen Name in den Programs-Menüs sämtlicher Benutzer aufgeführt ist. Zum Installieren, Entfernen oder Ändern eines globalen Programms ist nur der Systemverwalter bzw. ein Benutzer, dem Systemverwalter-Privilegien zugeteilt worden sind, berechtigt.

Ein globales Programm können Sie über die Menüschnittstelle UNIX System V Administration installieren, ändern oder entfernen. Die Verwaltung von globalen Programmen besteht aus denselben Aufgaben, die ein einzelner FACE-Benutzer im Menü Programs Administration für seine benutzer-eigenen Programme durchführt. Der einzige Unterschied besteht darin, daß sich die Durchführung dieser Aufgaben über das Menü UNIX System V Administration auf die Menü Programs sämtlicher FACE-Benutzer auf Ihrem Rechner auswirkt.

Sämtliche Aufgaben im Zusammenhang mit der Verwaltung von globalen Programmen werden im Menü FACE Administration des Menü UNIX System V Administration durchgeführt. Zuvor müssen Sie die folgenden Schritte durchführen:

1. Wählen Sie im Menü AT&T FACE die Option `System Administration`.

   Der Bildschirm wird jetzt gelöscht, und Sie werden zur Eingabe des Systemverwalter-Paßworts aufgefordert. Bei der Eingabe eines falschen Paßworts kehren Sie zu Ihrem FACE-Office zurück. Nach der Eingabe des korrekten Paßworts wird für Sie das Menü UNIX System V Administration aufgerufen. In diesem Menü können Sie zur Cursor-Steuerung, Wahl von Menüoptionen, zum Editieren von Schablonen und Aufrufen von Kommandos großteils dieselben Tasten wie unter FACE benutzen.

2. Wählen Sie im Menü UNIX System V Administration die Option `applications`.
3. Wählen Sie im Menü Administration for Available Applications die Option `FACE`.
4. Wählen Sie im Menü FACE Administration die Option `Programs`.

   Auf dem Bildschirm wird jetzt ein Menü mit vier Optionen angezeigt:

   ```
   add
   list
   modify
   remove
   ```

Eintragen eines globalen Programms

Nach der Wahl von `add` wird die Schablone Add Global Programs angezeigt; hier können Sie ein Programm angeben, das im Menü Programs sämtlicher FACE-Benutzer auf dem Rechner erscheinen soll.

In der folgenden Übung lernen Sie, wie Sie ein Programm in das Menü FACE Programs eintragen können:

1. Wählen Sie im Menü Global Programs Administration die Option `add`.
2. Auf dem Bildschirm erscheint die Schablone Add Global Programs mit vier Feldern.
3. Füllen Sie die vier Felder der Schablone aus.

   Sobald Sie den Cursor in ein Feld bewegen, wird in der Meldungszeile das Ausfüllen des Felds erklärt. Sie können auch die Taste **HELP** betätigen, um zusätzliche Informationen abzurufen.
4. Betätigen Sie die Taste **SAVE**, um Ihre Definition des neuen globalen Programms abzuspeichern.

   Sie kehren dann zum Menü Global Program Administration zurück.
5. Abschließend betätigen Sie die Taste **CANCEL**, um zum Menü FACE Administration zurückzukehren. Eine andere Möglichkeit besteht darin, das Menü UNIX System V Administration über die Taste **CMD-MENU** und die Wahl von `exit` zu verlassen.

6. Bei der Eingabeaufforderung betätigen Sie die Taste ENTER, um zu FACE zurückzukehren.
7. Rufen Sie im Menü Programs zur Kontrolle das neue globale Programm auf.

Nachdem Sie ein neues globales Programm eingetragen haben, sollten Sie die übrigen FACE-Benutzer auf Ihrem Rechner auf jeden Fall entsprechend informieren.

Sobald Sie wieder zu FACE zurückgekehrt sind und nicht mehr in der Funktion als Systemverwalter arbeiten, können weder Sie noch die übrigen FACE-Benutzer dieses globale Programm entfernen, ändern oder abrufen, indem Sie/sie im Menü Office of login die Option `Programs Administration` auswählen (wie bereits erwähnt, können ausschließlich benutzer-eigene Programme von `Programs Administration` aus verwaltet werden).

Abrufen, Ändern und Entfernen eines globalen

Programms" Beim Zugriff auf die Optionen `list`, `modify` und `remove` im Menü Global Program Administration gehen Sie exakt wie beim Zugriff auf `add` vor (siehe "Eintragen eines globalen Programms").

Nachdem Sie das Menü Global Program Administration im Menü UNIX System V Administration aufgerufen haben, wird erneut eine Übersicht über die vier zur Wahl stehenden Funktionen angezeigt:

```
add
list
modify
remove
```

Die Funktion `list` ist die einfachste Funktion zur Verwaltung von globalen Programmen. Hiermit rufen Sie einfach eine aktuelle Übersicht über die globalen, unter FACE eingetragenen Programme ab; nach dem Schließen des Ausschnitts kehren Sie zum Menü Global Programs Administration zurück.

Hinweis: Wenn Sie im Menü List Global Programs ein globales Programm auswählen, wird es ausgeführt.

Über die Option `modify` rufen Sie das Menü Modify Global Programs auf. In diesem Menü sind die globalen Programme aufgeführt, die unter FACE eingetragen worden sind. Allerdings fehlen die globalen Programme, die bereits von AT&T unter FACE eingetragen worden sind (z.B. `Mail Services` oder `Spell Checker`). Wenn Sie in diesem Menü das zu ändernde Programm auswählen, wird die Schablone `Modify Global Programs` angezeigt; diese Schablone enthält dieselben Felder wie die Schablone `Add Global Programs`. Auf diese Weise können Sie jedes Feld editieren. Nachdem Sie die betreffenden Felder geändert haben, betätigen Sie die Taste [SAVE], um die neuen Werte abzuspeichern und zum Menü Global Programs Administration zurückzukehren. Auch in diesem Fall können Sie zum Verlassen des Menüs UNIX System V Administration die Taste [CMD-MENU] und die Option `exit` benutzen. Befolgen Sie die Eingabeaufforderung `Press ENTER to continue` zur Betätigung der ENTER-Taste, um zu FACE zurückzukehren.

Nach der Wahl von `remove` erscheint ein Menü mit dem Titel `Remove Global Programs`, in dem sämtliche globalen Programme aufgeführt sind, die unter FACE eingetragen worden sind. Bei der Wahl eines Programms in diesem Menü wird zur Bestätigung ein Ausschnitt mit dem Namen des zu löschenden Programms angezeigt. Sie können das Programm jetzt über die Taste [CONT] entfernen oder die Prozedur über die Taste [CANCEL] abbrechen, ohne das Programm zu löschen. Sie kehren in beiden Fällen zum Menü Global Programs Administration zurück. Auch in diesem Fall können Sie das Menü UNIX System V Administration verlassen, indem Sie die Taste [CMD-MENU] betätigen und die Option `exit` wählen. Befolgen Sie die Aufforderung `Press ENTER to continue`, um zu Ihrem FACE-Office zurückzukehren.

Verwaltung von Mail Services und Spell Checker

FACE wird mit den beiden globalen Programmen `Mail Services` und `Spell Checker` ausgeliefert. Die Verwaltung dieser beiden Programme ist allerdings nicht über die Menüoption Global Program Administration möglich, da sie in FACE sozusagen "eingebaut" sind. Somit können sie nicht über die Menüoption `Programs` im Menü FACE Administration entfernt oder geändert werden. Zur Verwaltung dieser eingebauten Funktionen müssen Sie im Menü FACE Administration `mail_services` bzw. `spell_checker` wählen.

Verwaltung von Mail Services

1. Wählen Sie im Menü FACE Administration die Option `mail_services`.

   Auf dem Bildschirm erscheint ein Ausschnitt mit dem Titel `Mail Services Administration`.

2. Durch die Wahl von `add` in diesem Menü können Sie die Menüoption Mail Services in die Programs-Menüs der Benutzer eintragen; mit `remove` können Sie die Menüoption Mail Services aus den Programs-Menüs der Benutzer entfernen.

3. Nach beendeter Durchführung der gewünschten Funktion wird in der Meldungszeile eine entsprechende Bestätigung angezeigt.

Verwaltung von Spell Checker

1. Wählen Sie im Menü FACE Administration die Option `spell_checker`.

   Auf dem Bildschirm erscheint ein Ausschnitt mit dem Titel `Spell Checker Administration`.

2. Durch die Wahl der Option `add` in diesem Menü können Sie die Menüoption Spell Checker in die Programs-Menüs der einzelnen Benutzer eintragen; mit `remove` können Sie die Menüoption aus den Programs-Menüs der einzelnen Benutzer entfernen.

3. Nach beendeter Durchführung der gewünschten Funktion wird in der Meldungszeile eine entsprechende Bestätigung angezeigt.

Informieren Sie die FACE-Benutzer in jedem Fall, wenn ein globales Programm eingetragen, geändert oder gelöscht worden ist.

Das UNIX-System

Hinweis: Wenn Sie nicht über FACE auf das UNIX-System zugreifen können, ist die Menüoption `UNIX System` nicht in Ihrem Menü AT&T FACE enthalten; die in diesem Abschnitt beschriebenen Funktionen stehen Ihnen dann nicht zur Verfügung.

Die Wahl von `UNIX System` im Menü AT&T FACE stellt eine weitere Möglichkeit zum Aufrufen der UNIX-Shell dar. Dadurch wird eine Subshell generiert, die den gesamten Bildschirm belegt. Am oberen Bildschirmrand werden außer dem Namen des aktuellen Verzeichnisses die folgenden Anweisungen dargestellt:

```
To return, type 'exit' or control-d
```

Auf dem Bildschirm wird `UNIX:`, die Eingabeaufforderung der Subshell, angezeigt. Diese nur einmal dargestellte Eingabeaufforderung zeigt Ihnen, daß Sie sich in einer UNIX-Subshell befinden.

Um zu AT&T FACE zurückzukehren, geben Sie `exit` ein und betätigen die Taste ENTER bzw. die Taste CTRL-d.

Sie können jetzt in der Subshell ein Programm oder ein Kommando aufrufen; allerdings wird die Subshell nach ihrer Rückkehr zu AT&T FACE ebenso wie die unter der Subshell aufgerufenen Prozesse beendet. Weitere Informationen zur UNIX-Shell finden Sie in den Kapiteln 1, 2, 3 und 9 dieses Leitfadens.

Aufrufen von UNIX-Kommandos im FACE-Office

Zum Aufrufen von UNIX-Kommandos müssen Sie nicht im Menü AT&T FACE die Option `UNIX System` auswählen - Sie können sie ebensogut in der FACE-Kommandozeile nach einem Ausrufezeichen `(!` eingeben.

Die vollständige Beschreibung sämtlicher UNIX-Kommandos würde den Rahmen dieses Kapitels bei weitem sprengen; in den folgenden Abschnitten finden Sie lediglich eine Einführung in die beiden UNIX-Kommandos `news` und `pg`. Hierbei handelt es sich um zwei sehr nützliche Kommandos, die Ihre Möglichkeiten im FACE-Office beträchtlich erweitern. Eine ausführliche Beschreibung dieser und anderer UNIX-Kommandos finden Sie im *Referenzhandbuch für Benutzer* im Abschnitt

"Kommandos (1)".

Hinweis: Wenn Sie nicht als Benutzer unter UNIX eingetragen sind, können Sie nicht in der Kommandozeile ein Ausrufezeichen (! eingeben und UNIX-Kommandos aufrufen.

Das Kommando `news`

Bei Ihrer Anmeldung am Rechner werden Sie in manchen Fällen in einer einzeiligen Meldung darüber informiert, daß Nachrichten für Sie angekommen sind; möglicherweise werden die Nachrichten sogar aufgelistet. Die UNIX-Nachrichten können von allen Rechner-Benutzern gelesen werden; sie sind in ihrer Funktion mit einer elektronischen Zeitung vergleichbar. In dieser Übung lernen Sie, wie Sie diese elektronische Zeitung lesen und neue Nachrichten in sie einfügen können.

1. Betätigen Sie die Taste CTRL-j, um auf die Kommandozeile zuzugreifen. Geben Sie `!news` ein und betätigen Sie die Taste ENTER.

   Der Bildschirm wird zunächst gelöscht, dann werden die neuen Nachrichten (falls vorhanden) angezeigt. Wenn Sie gerade ein UNIX-Kommando ausführen, können Sie die Bildschirmausgabe über die Tasten CTRL-s anhalten; zur Fortsetzung der Bildschirmausgabe betätigen Sie dann CTRL-q.

2. Nachdem sämtliche Nachrichten ausgegeben worden sind (bzw. wenn es keine neuen auszugebenden Nachrichten mehr gibt), werden Sie in einer Meldung zur Betätigung der Taste ENTER aufgefordert, um zu FACE zurückzukehren. Sie kehren danach zu demselben Ausschnitt zurück, in dem Sie das Kommando aufgerufen haben.

Wenn Sie eine neue Option in diese elektronische Zeitung einbringen möchten, so legen Sie eine Datei mit Ihrer Mitteilung an und kopieren sie in das UNIX-Verzeichnis `/home/news`. Bei Fragen wenden Sie sich an Ihren Systemverwalter.

Das Kommando `pg` und Pipes

Das UNIX-Kommando `pg` gibt seine Eingabe bildschirm-seitenweise aus. Es ist immer dann hilfreich, wenn die Ausgabe eines Kommandos nicht in einem Durchgang angezeigt werden soll. Damit es mit der nächsten Bildschirmseite fortfährt, müssen Sie die Taste ENTER betätigen.

Zur Benutzung des Kommandos `pg` gibt es zwei Möglichkeiten. Bei der ersten geben Sie einen Dateinamen als Argument an, bei der zweiten übergeben Sie ihm die Ausgabe eines anderen Programms mit Hilfe einer Pipe (darauf kommen wir später in dieser Übung zurück).

1. Betätigen Sie die Taste CTRL-j, um auf die Kommandozeile zuzugreifen; dann geben Sie `!pg facetest2` ein und betätigen die Taste ENTER.

   Auf dem Bildschirm wird die erste Bildschirmseite der Datei `facetest2` angezeigt; dann legt das Kommando eine Pause ein und stellt die Eingabeaufforderung in Form eines Doppelpunkts `(:` dar.

   1. Wenn Sie mit der nächsten Bildschirmseite fortfahren möchten, betätigen Sie einfach die Taste ENTER.
   2. Um zur vorhergehenden Bildschirmseite zurückzukehren, geben Sie ein Minus-Zeichen (-) ein und betätigen die Taste ENTER.
   3. Um zum Ende der Datei zu springen, geben Sie ein Dollar-Zeichen `($` ein und betätigen die Taste ENTER.

2. Sobald Sie am Dateiende angelangt sind, ändert sich die Eingabeaufforderung in `(EOF) :` (für "end of file"). Wenn Sie jetzt die Taste ENTER betätigen, beenden Sie das Kommando `pg`.
3. Bei der entsprechenden Eingabeaufforderung betätigen Sie die Taste ENTER, um zu FACE zurückzukehren.

Die Möglichkeit zur seitenweise Ausgabe von Dateiinhalten ist beispielsweise bei einer umfangreichen Ausgabe des Kommandos `news` von Vorteil. Allerdings ist die Ausgabe von `news` keine Datei; Sie können unter UNIX jedoch ein Kommando aufrufen und seine Ausgabe statt zum Bildschirm mit Hilfe einer Pipe an das Kommando `pg` weiterleiten, das es dann wie eine Datei behandelt. Für eine derartige "Pipe" benutzen Sie das Pipe-Symbol (|).

1. Betätigen Sie die Taste CTRL-j, um auf die Kommandozeile zuzugreifen; geben Sie dann `!news | pg` ein und betätigen Sie die Taste ENTER.
2. Der Bildschirm wird gelöscht, und die elektronische Zeitung wird ausgegeben. Da die Ausgabe von `news` jedoch über eine Pipe an `pg` weitergeleitet wird, wenn sie nicht auf eine Bildschirmseite paßt, können Sie wie oben beschrieben vor- und zurück springen. Versuchen Sie, die nächste Bildschirmseite abzurufen, die letzte, oder zum Ende der Ausgabe

zu springen.

3. Beenden Sie das Kommando `pg` und kehren Sie zu FACE zurück.

Vergessen Sie nicht, daß UNIX-Kommandos mit einem Ausrufezeichen (`!`) beginnen müssen, wenn sie von der FACE-Kommandozeile aus aufgerufen werden.

Arbeiten mit ausführbaren Dateien

Auf Ihrem Rechner gibt es eine Vielzahl von ausführbaren Dateien. Eine ausführbare (bzw. binäre) Datei enthält Anweisungen an den Rechner, die von den Benutzern nicht gelesen werden können und es auch nicht sollen. Sie können eine ausführbare Datei in einen Ihrer Kataloge kopieren. Der Datei wird dann die Beschreibung `Executable` (ausführbar) zugeordnet.

Hinweis: Da die Beschreibung `Executable`, die von FACE automatisch zugewiesen wird, die einzige Möglichkeit zur Unterscheidung der ausführbaren Dateien von den nicht ausführbaren ist (es sei denn, Sie rufen die Dateien im ausführlichen Format in einem Menü ab), sollten Sie derartigen Dateien keine neue Beschreibung zuordnen.

Wenn Sie einer ausführbaren Datei eine neue Beschreibung zuordnen möchten, müssen Sie sie in einem Menü als ausführbar kennzeichnen, indem Sie den Wert des Felds `Display Type` in der Schablone Office Functions auf `long form` setzen, damit auch ihr Typ angezeigt wird.

Nachdem Sie eine ausführbare Datei in Ihr Dateisystem kopiert haben, müssen Sie sie lediglich auswählen, wenn Sie damit arbeiten möchten. Die folgende Übung zeigt Ihnen, wie Sie eine ausführbare Datei kopieren und damit arbeiten können.

1. Betätigen Sie CTRL-], um auf die Kommandozeile zuzugreifen, und geben Sie dann

   ```
   copy /usr/bin/news $HOME
   ```

   ein; abschließend betätigen Sie die Taste ENTER.

2. Eröffnen Sie jetzt den Katalog `/home/login`; die Datei `news` ist jetzt aufgeführt und mit `Executable` gekennzeichnet.

   Hinweis: Dateien, die aus Ihrem FACE-Office in das UNIX-System kopiert worden sind, haben möglicherweise nicht dieselben Zugriffsberechtigungen, die Sie in der Schablone Preferences definiert haben. Sie können jetzt die Zugriffsberechtigungen der Datei `news` (mit `security`) überprüfen, um sicherzustellen, daß die Lese-, Schreib- und Ausführungs-/Suchberechtigungen für Sie (den Eigentümer) auf `yes` gesetzt sind.

3. Um `news` aufzurufen, bewegen Sie den Cursor zum Dateinamen und betätigen die Taste [ENTER]. Das Kommando `news` nimmt den Bildschirm vollständig in Beschlag; wenn Nachrichten von anderen Benutzern auf Ihrem Rechner eingegangen sind, werden sie angezeigt.

Vorsicht: Von ausführbaren UNIX-Dateien sollten Sie innerhalb Ihres Dateisystems keine Kopien anfertigen, da Systemprogramme wie `news` im Normalfall an bestimmten Stellen innerhalb des Dateisystems (z.B. im Verzeichnis /usr/bin directory) gespeichert sind und problemlos als benutzer-eigene Programme in Ihrem Menü Programs eingetragen werden können.

Die Kopiermöglichkeit werden Sie vor allem dann nutzen, wenn ein anderer Benutzer Ihnen die Kopie eines Programms zur Verfügung gestellt hat, bei dem es sich nicht um ein Standard-UNIX-Programm handelt.

Aufrufen einer Shell-Prozedur

Unter UNIX gibt es noch einen weiteren Dateityp, der auf Ihrem Rechner ausgeführt werden kann. Eine derartige Standard-Datei wird als "Shell-Prozedur" bezeichnet. Vor der Shell-Prozedur muß jedoch zunächst die Funktion der Shell geklärt werden. Die UNIX-Shell ist ein sogenannter Kommandointerpreter, der ein von Ihnen eingegebenes Kommando einliest und das UNIX-System zur Durchführung einer bestimmten Funktion anweist.

Nachdem Sie das UNIX-System aufgerufen haben, zeigt die UNIX-Shell die Eingabeaufforderung `UNIX:` an. Damit signalisiert Ihnen die Shell, daß sie auf Ihre Kommandoeingabe wartet. Als Shell-Prozedur wird eine Datei mit ein oder mehreren Textzeilen bezeichnet; jede Textzeile ist ein Kommando, das von der

Shell interpretiert werden kann. Nach dem Anlegen der Datei müssen ihre Zugriffsberechtigungen geändert werden, um sie ausführbar zu machen. Selbst nach der Änderung der Zugriffsberechtigungen wird die Datei von FACE als Standard- und nicht als ausführbare Datei betrachtet.

Hinweis: Die Beschreibung `Executable`, die der Datei automatisch von FACE zugeordnet wird, ist für Sie die einzige Möglichkeit zur Unterscheidung der nicht ausführbaren Dateien von den ausführbaren (d.h. von den Dateien, die Sie ohne das Kommando `run` ausführen können). Den Shell-Prozeduren sollten Sie nicht die Beschreibung `Executable` zuordnen, obwohl es sich eigentlich um ausführbare Dateien handelt.

Wenn Sie einer Shell-Prozedur doch die Beschreibung `Executable` zuordnen, können Sie die Dateien mit der Beschreibung `Executable` nur dann erkennen, wenn Sie den Wert des Felds `Folder Display Type` in der Schablone auf `long form` gesetzt haben, so daß außer dem Dateityp auch ihre Beschreibung angezeigt wird.

In der folgenden Übung wird eine einfache Shell-Prozedur erstellt, ihr die Ausführungs-/Suchberechtigung zugeordnet und sie dann mit dem Kommando `run` ausgeführt.

1. Legen Sie in Ihrem Katalog `/home/login` eine Standard-Datei namens `test.sh` an und tragen Sie die folgenden Zeilen ein:

```
echo What is your name?
read name
echo Hello $name.
echo I\'m a shell script und now I am finished.
```

2. Speichern Sie die Datei ab und verlassen Sie den Editor.

   `test.sh` ist in Ihrem Menü `/home/login` jetzt als Standard-Datei aufgeführt.

3. Bewegen Sie den Cursor zur Menüoption `test.sh` im Menü `/home/login`.

4. Rufen Sie das Kommando `security` auf und erteilen Sie sich selbst die Ausführungs-/Suchberechtigung für diese Datei. Nach Ihrer Rückkehr zum Katalog `/home/login` werden Sie an der Beschreibung keine Änderung feststellen.

5. Wählen Sie dann `test.sh`.

   FACE interpretiert diese Eingabe jetzt nicht wie in der letzten Übung als Kommando zum Aufrufen der Datei, sondern als Kommando zum Editieren der Datei. Da Sie die Datei nicht editieren möchten, verlassen Sie den Editor, ohne Änderungen durchzuführen.

6. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]), und wählen Sie das Kommando `run` (der Cursor befindet sich noch bei `test.sh`).

   Dadurch wird der Bildschirm gelöscht, und die Shell- Prozedur `test.sh` wird aufgerufen.

7. Bei der Eingabeaufforderung `What is your name?` geben Sie Ihren Namen ein und betätigen die Taste [ENTER].

8. Nach der Abarbeitung der Shell-Prozedur betätigen Sie die Taste [ENTER], um zu FACE zurückzukehren.

Nachdem Sie eine Shell-Prozedur wie oben beschrieben in Ihrem FACE-Office bzw. Ihrem Dateisystem abgespeichert haben, müssen Sie zu ihrer Ausführung das Kommando `run` aufrufen.

Vorübergehendes Verlassen von und Rückkehr zu Dateien

Angenommen, Sie benötigen beim Editieren einer Datei Informationen aus einer anderen Datei. Eigentlich müßten Sie Ihre aktuelle Position jetzt markieren, die Datei abspeichern und verlassen, zu FACE zurückkehren, die andere Datei eröffnen und lesen, zur zuvor editierten Datei zurückkehren, die markierte Position suchen, die Markierung löschen und dann mit dem Editieren fortfahren. Wenn Sie beim Editieren der ersten Datei öfter Informationen aus anderen Dateien benötigen ....

Selbstverständlich können Sie Ihre wertvolle Arbeitszeit sinnvoller verbringen. Glücklicherweise gibt es das Kommando `facesuspend`, mit dem Sie bis zu fünf `vi`- Editor-Sitzungen vorübergehend verlassen und über `frm-mgmt` die Datei angeben können, zu der Sie zurückkehren möchten.

Hinweis: Das Kommando `facesuspend` kann jedoch nur in Programmen benutzt werden, in denen wie in `vi` die Möglichkeit zum vorübergehenden Aufrufen einer UNIX- Shell eingebaut ist. In einem Programm, in dem diese Möglichkeit nicht vorhanden ist, wird der Aufruf von `facesuspend` ignoriert.

Vorgehensweise

Die folgende Übung zeigt Ihnen, wie Sie eine gerade mit `vi` editierte Datei vorübergehend verlassen können.

1. Wechseln Sie zum Katalog /home/login und legen Sie die drei neuen Dateien `facetest3`, `facetest4` und `facetest5` an.

   Beim Anlegen jeder Datei wird `vi` aufgerufen. Tragen Sie in jede Datei (einen anderen) Text ein und speichern Sie die Datei mit `:wq` ab (dadurch beenden Sie gleichzeitig den Editor). Danach sind in Ihrem Katalog /home/login die drei neuen Dateien aufgeführt.

2. Wählen Sie die Datei `facetest3`.

   Angenommen, Sie benötigen Informationen aus der Datei `facetest4`. Dazu müssen Sie jetzt nicht `:wq` eingeben, um `facetest3` abzuspeichern und den Editor `vi` zu beenden, sondern Sie müssen nur `:!facesuspend` aufrufen und die Taste ENTER betätigen .

   Wie bereits erwähnt, können Sie im `vi`- Kommandomodus über die Eingabe von `!kommando` ein UNIX-Kommando ausführen, ohne `vi` zuvor zu verlassen. In diesem Fall führen Sie das Kommando `facesuspend` aus, das in FACE eingebaut ist. Das Kommando `facesuspend` beendet den aktuellen Prozeß (in diesem Fall `vi`) nicht permanent, sondern schaltet lediglich vorübergehend zum FACE-Bildschirm zurück.

3. Wiederholen Sie Schritt 2, um `facetest4` zu eröffnen und dann vorübergehend zu beenden.

4. Angenommen, Sie benötigen jetzt auch Informationen aus der Datei `facetest5`.

   Eröffnen Sie `facetest5` und verlassen Sie die Datei vorübergehend.

Obwohl Sie jetzt in Ihrem Arbeitsbereich drei Dateien eröffnet und vorübergehend verlassen haben, ist keine davon aktuell auf Ihrem Bildschirm zu sehen. Sie befinden sich jetzt wieder in Ihrem FACE-Office, und /home/login ist der aktive Ausschnitt.

Rückkehr zu vorübergehend verlassenen Dateien

Anhand der drei kurzen Dateien, die Sie eben angelegt haben, kann von der enormen Zeitersparnis, die `facesuspend` ermöglicht, selbstverständlich nur ein kleiner Eindruck vermittelt werden.

In der folgenden Übung lernen Sie, wie Sie über das Kommando `frm-mgmt` eine Übersicht über die vorübergehend verlassenen Dateien abrufen und die Datei angeben können, zu der Sie zurückkehren möchten.

1. Rufen Sie `frm-mgmt` vom Command Menu oder von der Kommandozeile aus auf.

   Dadurch wird das Menü Frame Management eröffnet.

2. Wählen Sie im Menü Frame Management die Option `list`.

   Auf dem Bildschirm wird jetzt ein Menü mit den eröffneten Ausschnitten angezeigt. Beachten Sie, daß die vorübergehend verlassenen Dateien aufgelistet werden, obwohl sie aktuell nicht auf dem Bildschirm sichtbar sind.

3. Wählen Sie `facetest3`.

   Sie kehren zu `facetest3` zurück; dabei werden Sie automatisch an die Stelle zurückgesetzt, an der Sie die Datei vorübergehend verlassen haben. Sie können jetzt mit dem Editieren der Datei fortfahren und die Datei dann entweder abspeichern und den Editor verlassen, oder sie erneut vorübergehend verlassen.

4. In dieser Übung verlassen Sie die Datei `facetest3` erneut nur vorübergehend.

5. Wiederholen Sie Schritt 1 bis 4 für die Dateien `facetest4` und `facetest5`.

   Nachdem Sie `facetest5` zum zweiten Mal vorübergehend verlassen haben, befinden Sie sich wieder im Ausschnitt /home/login.

6. Betätigen Sie die Taste CTRL-j, geben Sie `exit` ein und verlassen Sie FACE über die Taste ENTER.

   Im Normalfall könnten Sie FACE jetzt verlassen, obwohl der Ausschnitt /home/login und andere Ausschnitte eröffnet sind. Stattdessen kehren Sie zur ersten der vorübergehend verlassenen Optionen im Menü Open Frames zurück. Sie werden jetzt in einer weiteren Meldung darüber informiert, daß Sie diese Option vor der Beendigung von FACE verlassen müssen.

7. Schließen Sie die jeweils angezeigte `vi`- Sitzung mit `:q` oder `:wq`, und betätigen Sie dann die Taste ENTER. Sobald die letzte der vorübergehend verlassenen Dateien geschlossen worden ist, führt FACE abschließend das Kommando `exit` durch; gleichzeitig werden Sie von Ihrem FACE-Office abgemeldet.

Hinweis: Vorübergehend verlassene Dateien, die über die Option `UNIX System` im Menü Office of login aufgerufen worden sind, werden mit ihren Namen in der bei `frm-mgmt` angezeigten Liste aufgeführt. Vorübergehend verlassene Dateien, auf die Sie über das Menü Programs oder ein Shell-Kommando (!unix_kommando) zugegriffen haben, werden in der bei `frm-mgmt` ausgegebenen Liste mit ihrem absoluten Pfadnamen aufgeführt.

6 Der Zeileneditor ed

Einführung in den Zeileneditor 6-1

Benutzerhinweise 6-3

Einstieg 6-4
Aufrufen von ed 6-4
Erstellen von Text 6-5
Abrufen von Text 6-6
Löschen einer Textzeile 6-8
An eine andere Stelle der Datei springen 6-9
Pufferinhalt in einer Datei abspeichern 6-10
Verlassen des Editors 6-11

Übung 1 6-14

Allgemeines Format der ed-Kommandos 6-15

Zeilenadressierung 6-16
Numerische Adresse 6-17
Symbolische Adresse 6-18
- Die symbolische Adresse für die aktuelle Zeile 6-18
- Die symbolische Adresse für die letzte Zeile 6-19
- Die symbolische Adresse für alle Zeilen 6-19

- Die symbolische Adresse für den Zeilenbereich von der aktuellen bis zur letzten Zeile 6-20
- Zur aktuellen Zeile relative Adresse 6-20
- Alphanumerische Adressen 6-22
- Angabe eines Zeilenbereichs 6-25
- Globaler Suchvorgang 6-26

Übung 2 6-29

Abrufen des Dateiinhalts 6-30
Nur Text: Das Kommando `p` 6-30
Abrufen von Text über Zeilenadressen: Das Kommando `n` 6-31

Erstellen von Text 6-33
Anfügen von Text: Das Kommando `a` 6-33
Einfügen von Text: Das Kommando `i` 6-36
Überschreiben von Text: Das Kommando `c` 6-37

Übung 3 6-40

Löschen von Text und Aufheben von Änderungen 6-42
Zeilen löschen im Kommandomodus: Das Kommando `d` 6-42
Aufheben des letzten Kommandos im Kommandomodus: Das Kommando `u` 6-43

Ersetzen von Text 6-46
Beschränkung auf die aktuelle Zeile 6-47
Beschränkung auf eine bestimmte Zeile 6-48
Ersetzen innerhalb eines Zeilenbereichs 6-49
Globales Ersetzen 6-50

Übung 4 6-54

Mustervergleiche mit Sonderzeichen 6-56

Übung 5 6-66

Versetzen von Text 6-68
Textzeilen versetzen 6-68
Textzeilen kopieren 6-70
Verknüpfen von aufeinanderfolgenden Zeilen 6-72
Abspeichern von Textzeilen in einer Datei 6-73
Mögliche Probleme 6-74
Einlesen des Inhalts einer Datei 6-75

Übung 6 6-77

Weitere nützliche Kommandos und Informationen 6-78
Hilfe-Kommandos 6-78
Anzeige von nicht darstellbaren Zeichen 6-81
Der aktuelle Dateiname 6-82
Vorübergehendes Aufrufen einer Sub-Shell 6-84
Nach einem Absturz 6-85
Zusammenfassung 6-86

Übung 7 6-88

Auflösung der Übungen 6-89
Übung 1 6-89
Übung 2 6-91
Übung 3 6-94
Übung 4 6-98
Übung 5 6-100
Übung 6 6-103
Übung 7 6-105

Einführung in den Zeileneditor

Dieses Kapitel enthält eine Einführung in den Zeileneditor ed. Bei ed handelt es sich um einen flexiblen, relativ schnellen Editor, der auf so gut wie jedem Terminal lauffähig ist. Die Kommandozeilen und System- Ausgaben werden in diesem Kapitel exakt so aufgeführt, wie sie auch auf Ihrem Terminal (Datensichtgerät oder Drucker-Terminal) eingegeben bzw. angezeigt werden. Die ed-Kommandos können Sie direkt an Ihrem Terminal eingeben oder in ein Shell-Programm einbringen (siehe Kapitel 9, "Die Shell").

Bei ed handelt es sich um einen sogenannten Zeileneditor. Das heißt, beim Arbeiten mit ed editieren bzw. bearbeiten Sie stets eine einzige Zeile, die als aktuelle Zeile bezeichnet wird. Beim Aufrufen einer bereits vorhandenen Datei macht ed die letzte Zeile automatisch zur aktuellen Zeile, so daß Sie problemlos Text anfügen können. ed führt jedes Kommando automatisch für die aktuelle Zeile durch, es sei denn, Sie geben eine bestimmte Zeile bzw. einen bestimmten Zeilenbereich an. Mit ed können Sie nicht nur Text in ein oder mehreren Zeilen der Datei ändern, löschen oder einfügen, sondern auch Text aus einer anderen Datei in den Puffer laden.

Beim Arbeiten mit ed ändern Sie den Inhalt einer Datei, der in einem temporären Puffer enthalten ist. Bei der Änderung einer bereits vorhandenen Datei wird eine Kopie dieser Datei in den Puffer geladen; sämtliche Änderungen, die Sie an der Datei vornehmen, betreffen ausschließlich diese Kopie. In die ursprüngliche Datei werden die Änderungen erst dann eingebracht, wenn Sie ed über das Kommando write die Anweisung zum Abspeichern des Pufferinhalts in die Datei erteilen.

Nachdem Sie diese Einführung durchgearbeitet und die Beispiele und Übungen nachvollzogen haben, werden Sie gute Kenntnisse über ed besitzen, die Ihnen das Arbeiten mit den Editor-Funktionen ermöglichen. In diesem Kapitel werden die folgenden grundlegenden Funktionen beschrieben:

- Aufrufen des Zeileneditors ed, Eingeben von Text, Abspeichern des Texts in einer Datei und Verlassen von ed
- Adressierung von bestimmten Zeilen der Datei und Ausgabe der Textzeilen
- Löschen von Text
- Ersetzen von Text

- Benutzung von Sonderzeichen in Mustern zum Suchen und Ersetzen von Text
- Versetzen von Text innerhalb der Datei sowie weitere nützliche Kommandos und Informationen

Benutzerhinweise

Die Kommandos, die in jedem Abschnitt besprochen werden, werden am Ende des jeweiligen Abschnitts nochmals zusammengefaßt. In Anhang D finden Sie eine nach thematischen Gesichtspunkten geordnete Übersicht über sämtliche ed-Kommandos, die in diesem Kapitel besprochen werden.

Am Ende einiger Abschnitte sind Beispiele enthalten, damit Sie mit den Kommandos ein wenig experimentieren können. Die Auflösungen der Übungen finden Sie am Ende dieses Kapitels.

Die Beschreibung in diesem Kapitel hält sich an das Beschreibungsformat, das in diesem *Leitfaden* durchgängig verwendet wird. Informationen zum Beschreibungsformat finden Sie im Vorwort.

Einstieg

Um ed kennenzulernen, melden Sie sich am besten beim UNIX-System an und vollziehen die Beispiele beim Durcharbeiten dieser Einführung nach. Die Übungen sollten Sie auf jeden Fall selbst an Ihrem Terminal durchführen; experimentieren Sie auch mit selbst erdachten Beispielen. Durch das Ausprobieren der ed-Kommandos werden Sie eine schnelle und flexible Methode zum Editieren von Texten kennenlernen.

In diesem Abschnitt lernen Sie Kommandos für die folgenden Funktionen kennen:

- Aufrufen von ed
- Anfügen von Text
- Anspringen und Abzeigen einer bestimmten Textzeile innerhalb der Datei
- Löschen einer Textzeile
- Abspeichern des Pufferinhalts in einer Datei
- Verlassen von ed

Aufrufen von ed

Zum Aufrufen des Zeileneditors geben Sie ed sowie einen Dateinamen ein:

```
ed datei_name<CR>
```

Wählen Sie möglichst einen Dateinamen, aus dem der Dateiinhalt hervorgeht. Wenn die Datei noch nicht vorhanden ist, gibt das System ein Fragezeichen sowie den Dateinamen aus:

```
$ ed new-file<CR>
?new-file
```

Wenn Sie dagegen eine bereits vorhandene Datei abändern möchten, gibt ed die Größe der Datei in Byte aus:

```
$ ed old-file<CR>
235
```

Erstellen von Text

In einer ed-Sitzung sind an Ihrem Terminal zwei Arten von Eingaben möglich: Sie können Kommandos zum Editieren von Text sowie den Text selbst eingeben. Um Verwechslungen zwischen den beiden Eingabeearten zu vermeiden, wird unter ed zwischen zwei Modi unterschieden: Dem Kommando- und dem dem Eingabemodus. Im Kommandomodus werden sämtliche Zeichen, die Sie eingeben, als Kommandos interpretiert. Im Eingabemodus dagegen werden sämtliche Zeichen, die Sie eingeben, als Text interpretiert und in eine Datei einfügt.

Unmittelbar nach dem Aufrufen von ed befinden Sie sich auf jeden Fall im Kommandomodus. Wenn Sie in Ihre Datei Text einbringen möchten, so müssen Sie zunächst in den Eingabemodus wechseln; hierzu geben Sie in einer Zeile ausschließlich das Kommando a (für append) ein und betätigen die Taste RETURN:

```
a<CR>
```

Sie befinden sich jetzt im Eingabemodus; alle Zeichen, die Sie von nun an eingeben, werden als Text in Ihre Datei eingebracht. Wie bereits erwähnt, müssen Sie das Kommando a in einer separaten Zeile eingeben; andernfalls wird es vom Editor nicht ausgeführt.

Wenn Sie keinen Text mehr eingeben möchten, geben Sie in einer separaten Zeile einen Punkt ein. Dadurch verlassen Sie den Eingabemodus und kehren wieder in den Kommandomodus zurück. Sie können jetzt weitere ed-Kommandos eingeben.

Das folgende Beispiel zeigt, wie Sie ed aufrufen, in eine neue Datei namens try-me Text eingeben und den Eingabemodus wieder mit einem Punkt verlassen.

```
$ ed try-me<CR>
?try-me
a<CR>
This is the first line of text.<CR>
This is the second line,<CR>
and this is the third line.<CR>
.<CR>
```

Beachten Sie, daß ed nach der Eingabe des Punkts keine Meldung o.ä. ausgibt; er wartet lediglich auf ein neues Kommando. Wenn ed ein Kommando nicht wie erwartet durchführt, haben Sie möglicherweise vergessen, zum Abschluß der Kommandoeingabe einen Punkt einzugeben; Sie befinden sich dann noch im Eingabemodus. In einem solchen Fall geben Sie am Anfang einer neuen Zeile einen Punkt ein und betätigen die Taste RETURN. Sie kehren dadurch in den Kommandomodus zurück, in dem Sie Kommandos zum Editieren von Text eingeben können. Wenn Sie beispielsweise versehentlich Zeichen oder Zeilen in Ihren Text eingefügt haben, müssen Sie zunächst in den Kommandomodus zurückkehren, bevor Sie sie löschen können.

Abrufen von Text

Um eine Textzeile auf dem Bildschirm abzurufen, geben Sie in einer separaten Zeile das Kommando p (für print) ein. Das Kommando p gibt die aktuelle Zeile aus, d.h. diejenige Zeile, die Sie zuletzt bearbeitet haben. Im vorangegangenen Beispiel haben Sie abschließend einen Punkt eingegeben, um den Eingabemodus zu verlassen. Geben Sie jetzt das Kommando p ein, um die aktuelle Zeile abzurufen:

```
$ ed try-me<CR>
?try-me
a<CR>
This is the first line of text.<CR>
This is the second line,<CR>
and this is the third line.<CR>
.<CR>
p<CR>
and this is the third line.
```

Sie können eine beliebige Textzeile auf dem Bildschirm abrufen, indem Sie ihre Zeilennummer (bzw. Adresse) eingeben. Die erste Zeile hat die Adresse 1, die zweite die Adresse 2, usw.. Wenn Sie also die zweite Zeile der Datei try-me abrufen möchten, so geben Sie folgendes ein:

```
2p<CR>
This is the second line,
```

Über Zeilenadressen können Sie auch einen Zeilenbereich abrufen; hierzu geben Sie, durch Komma voneinander getrennt, die Adressen der ersten und letzten Zeile des betreffenden Abschnitts ein. Wenn Sie beispielsweise die ersten drei Zeilen einer Datei abrufen möchten, geben Sie das folgende Kommando ein:

```
1,3p<CR>
```

Auf diese Weise können Sie sogar den gesamten Dateiinhalt abrufen. So können Sie z.B. mit dem Kommando `1,20p` eine 20-zeilige Datei abrufen. Wenn Ihnen die Adresse der letzten Zeile in Ihrer Datei nicht bekannt ist, können Sie das Zeichen `$`, das `ed`-Symbol für die Adresse der letzten Zeile, eingeben (ausführliche Informationen hierzu finden Sie im Abschnitt "Zeilenadressierung"):

```
1,$p<CR>
This is the first line of text.
This is a second line,
and this is the third line.
```

Wenn Sie es versäumt haben, den Eingabemodus durch die Eingabe eines Punkts zu verlassen, geben Sie in Ihre Datei versehentlich unerwünschten Text ein. Fügen Sie in Ihre Datei `try-me` eine weitere Textzeile ein, und geben Sie dann das Kommando `p` ein, ohne zuvor den Eingabemodus zu verlassen. Verlassen Sie dann den Eingabemodus, und rufen Sie die gesamte Datei auf dem Bildschirm ab:

```
p<CR>
and this is the third line.
a<CR>
This is the fourth line.<CR>
p<CR>
.<CR>
1,$p<CR>
This is the first line of text.
This is the second line,
and this is the third line.
This is the fourth line.
p
```

Wie wirkt sich dieses Kommando aus? Im nächsten Abschnitt finden Sie eine Anleitung zum Löschen der unerwünschten Textzeile.

Löschen einer Textzeile

Beim Löschen von Text muß `ed` sich im Kommandomodus befinden. Durch die Eingabe von `d` löschen Sie die aktuelle Zeile, durch die Eingabe von `d` in Kombination mit einer Zeilennummer eine bestimmte Zeile. Probieren Sie dieses Kommando am letzten Beispiel aus, um die versehentlich eingefügte Zeile (die das Zeichen `p` enthält) zu löschen. Hierzu rufen Sie zunächst die aktuelle Zeile ab (Kommando `p`), löschen sie (Kommando `d`) und rufen die restlichen Zeilen der Datei ab (Kommando `p`). Ihr Bildschirm sollte jetzt folgendermaßen aussehen:

```
p<CR>
p
d<CR>
1,$p<CR>
This is the first line of text.
This is a second line,
and this is the third line.
This is the fourth line.
```

Das Löschen von Text wird von ed nicht durch eine Meldung bestätigt; wenn Sie also die erfolgreiche Durchführung des Kommandos d überprüfen möchten, müssen Sie den Dateiinhalt mit dem Kommando p auf dem Bildschirm abrufen. Um die Durchführung des Löschvorgangs zu überprüfen, können Sie die Kommandos d und p zusammen in eine einzige Kommandozeile setzen. Nachdem Sie das letzte Beispiel mit diesem Kommando wiederholt haben, sollte Ihr Bildschirm folgendermaßen aussehen:

```
p<CR>
p
dp<CR>
This is the fourth line.
```

An eine andere Stelle der Datei springen

Die Zeile, die in der Datei auf die aktuelle Zeile folgt, können Sie durch Betätigung der Taste RETURN abrufen (der Editor muß sich im Kommandomodus befinden). Ist die aktuelle Zeile die letzte Zeile der Datei, so gibt ed ein Fragezeichen (?) aus und behandelt die letzte Zeile der Datei weiterhin als aktuelle Zeile. Die Zeile, die in der Datei der aktuellen Zeile vorangeht, können Sie über die Minus-Taste (–) abrufen.

Das folgende Beispiel zeigt die Anwendung beider Kommandos:

```
p<CR>
This is the fourth line.
-<CR>
and this is the third line.
-<CR>
This is the second line,
-<CR>
This is the first line of text.
<CR>
This is the second line,
<CR>
and this is the third line.
```

Beachten Sie, daß Sie den Inhalt einer Textzeile auch nur durch die Eingabe von -<CR> oder <CR> - also ohne Umweg über das Kommando `p` - abrufen können. Diese Kommandos dienen auch als Zeilenadressen. Bei jeder Eingabe einer Zeilenadresse, auf die kein Kommando folgt, geht `ed` davon aus, daß es den Inhalt der angegebenen Zeile anzeigen soll. Geben Sie zur Übung Text ein, löschen Sie eine Zeile und rufen Sie den Inhalt Ihrer Datei ab.

Pufferinhalt in einer Datei abspeichern

Wie bereits an einer früheren Stelle erwähnt, befindet sich Ihr Text während einer Editor- Sitzung in einem temporären Speicherbereich, dem sogenannten Puffer. Wenn Sie an Ihrer Datei nichts mehr ändern möchten, können Sie die Änderungen absichern, indem Sie sie aus dem temporären Puffer in eine permanente Datei in einem Sekundärspeicher "schreiben". Dabei bringen Sie einfach eine Kopie des Pufferinhalts in die Datei ein. Der Text im Puffer bleibt erhalten, so daß Sie weitere Änderungen daran vornehmen können.

Hinweis: Das Abspeichern sollten Sie in regelmäßigen Abständen vornehmen, so daß Sie im Falle eines Systemabsturzes (z.B. aufgrund eines Stromausfalls) nur den Inhalt des Puffers verlieren, nicht aber die bereits in die Datei eingebrachte Kopie.

Um Ihren Text in einer Datei abzuspeichern, geben Sie das Kommando `w` ein. Sie brauchen noch keinen Dateinamen anzugeben; geben Sie zunächst einfach `w` ein und betätigen Sie die Taste RETURN. Wenn Sie den Text gerade neu erstellt haben, legt `ed` eine Datei an und ordnet ihr den Namen zu, den Sie beim Aufrufen des Editors angegeben haben. War die Datei beim Aufrufen des Editors bereits vorhanden, so bringt das Kommando `w` den Pufferinhalt standardmäßig in diese Datei ein.

Sie können Ihrer Datei auch einen neuen Namen zuordnen, den Sie als Argument in der `w`- Kommandozeile angeben. Achten Sie jedoch darauf, daß Sie nicht den Namen einer anderen, bereits vorhandenen Datei angeben - es sei denn, Sie möchten den Inhalt dieser Datei durch den aktuellen Pufferinhalt überschreiben. `ed` fordert Sie nicht zur Eingabe einer Bestätigung auf, sondern überschreibt den Inhalt der Datei ohne vorherige Warnung durch den aktuellen Pufferinhalt.

Soll die Datei `try-me` beispielsweise zukünftig `stuff` heißen, können Sie sie folgendermaßen umbenennen:

```
w stuff <CR>
110
```

Achten Sie auf die letzte Zeile auf dem Bildschirm. Hier wird die Anzahl der Zeichen angezeigt, die in Ihrem Text enthalten sind. Wenn der Editor so die Anzahl der Zeichen ausgibt, ist die Abspeicherung ordnungsgemäß durchgeführt worden.

Verlassen des Editors

Wenn Sie an der Datei `try-me` nichts mehr ändern möchten, können Sie sie mit dem Kommando `w` aus dem Puffer in eine Datei schreiben. Dann können Sie den Editor verlassen und über das Kommando `q` (für quit) zur Shell zurückkehren:

```
w<CR>
110
q<CR>
$
```

Das System gibt jetzt die Shell-Eingabeaufforderung aus. Der Inhalt des Editor-Puffers ist jetzt gelöscht. Wenn Sie ihn nicht zuvor mit dem Kommando `w` abgespeichert haben, ist auch Ihr Text im Puffer gelöscht worden. Dies hat keinerlei Folgen, wenn Sie während der Editor-Sitzung keine Änderung an Ihrem Text vorgenommen haben; andernfalls besteht jedoch die Gefahr, daß Ihre Arbeit verlorengeht. Daher warnt `ed` Sie immer dann mit einem Fragezeichen (?), wenn Sie das Kommando `q` ohne vorherige Abspeicherung des Texts aufgerufen haben. Dies gibt Ihnen die Möglichkeit, den Text vor dem Beenden des Editors abzuspeichern:

```
q<CR>
?
w<CR>
110
q<CR>
$
```

Wenn Sie jedoch den Text nicht abspeichern, sondern ein zweites Mal das Kommando `q` eingeben, so geht `ed` davon aus, daß Sie den Pufferinhalt nicht in einer Datei abspeichern möchten und schaltet sofort zur Shell um. Ihre Datei bleibt dadurch unverändert, und der Pufferinhalt geht unwiederbringlich verloren.

Sie kennen jetzt die grundlegenden Kommandos, mit denen Sie unter `ed` eine Datei anlegen und editieren können.

In Tabelle 6-1 finden Sie eine Kurzübersicht über diese Kommandos.

Tabelle 6-1: Kommandos des Editors `ed`

Kommando	Funktion
`ed` *datei*	`ed` zum Editieren von *datei* aufrufen
`a`	Text unter der aktuellen Zeile einfügen
`.`	Eingabemodus verlassen und zum `ed`- Kommandomodus umschalten
`p`	Text auf dem Bildschirm abrufen
`d`	Text löschen
`<CR>`	(carriage return) Nächste Zeile im Puffer abrufen
`-`	Inhalt der vorhergehenden Zeile im Puffer abrufen
`w`	Pufferinhalt in Datei abspeichern
`q`	`ed` beenden und Rückkehr zur Shell

Übung 1

Die Lösungen für sämtliche Übungen in diesem Kapitel finden Sie am Kapitelende. Beachten Sie jedoch, daß es in vielen Fällen mehr als eine Lösung gibt - entscheidend ist nicht die Vorgehensweise, sondern das Ergebnis!

1-1. Rufen Sie `ed` mit einer Datei namens `junk` auf. Geben Sie die Textzeile `Hello World` ein, speichern Sie sie in einer Datei ab und verlassen Sie `ed`.

Legen Sie nun mit `ed` eine Datei namens `stuff` an. Geben Sie eine Textzeile ein, die aus den beiden Wörtern `Goodbye world` besteht; speichern Sie diesen Text in der Datei ab und beenden Sie `ed`.

1-2. Rufen Sie `ed` erneut mit der Datei `junk` auf. Wie sah die Reaktion des Programms aus? Entsprach die ausgegebene Zeichenzahl derjenigen, die das Kommando `w` in Übung 1-1 ausgab?

Rufen Sie den Dateiinhalt ab. Wird der Inhalt der Datei `junk` korrekt ausgegeben?

Wie können Sie zur Shell umschalten? Versuchen Sie den Editor mit dem Kommando `q` ohne vorheriges Abspeichern der Datei zu beenden. Warum ist dies möglich?

1-3. Rufen Sie `ed` mit der Datei `junk` auf. Fügen Sie eine Zeile ein:

```
Wendy's horse came through the window.
```

Sie haben keine Zeilenadresse angegeben - an welcher Stelle im Puffer wurde die Zeile wohl eingefügt? Rufen Sie den Pufferinhalt ab. Versuchen Sie den Editor ohne vorheriges Abspeichern der Datei zu verlassen. Versuchen Sie den Pufferinhalt in einer anderen Datei namens `stuff` abzuspeichern. Beachten Sie, daß `ed` den Inhalt der Datei `stuff` ohne vorherige Warnung einfach überschreibt. Sie haben also den Inhalt der Datei `stuff` gelöscht und durch neuen Text ersetzt.

Geben Sie jetzt das Kommando `q` ein, um den Editor zu beenden. Beachten Sie, daß `ed` *keine* Warnung in Form eines Fragezeichens (`?`) ausgibt, obwohl Sie die Änderungen an der Datei `junk` nicht vor dem Verlassen des Editors mit `q` abgespeichert haben. Dies ist darauf zurückzuführen, daß `ed` den Pufferinhalt nicht mehr als geändert betrachtet, sobald Sie ihn in *irgendeiner* Datei abgespeichert haben.

Allgemeines Format der ed-Kommandos

Die ed-Kommandos werden nach einem einfachen, stets gleichen Schema gebildet:

[*adresse1*[,*adresse2*]]*kommando*[*argument*]<CR>

Die eckigen Klammern um *adresse1*, *adresse2* und *argument* bedeuten, daß es sich hier um sogenannte optionale Elemente handelt. Die eckigen Klammern werden bei der Kommandoeingabe weggelassen.

adresse1,*adresse2*	Die Adressen stehen für die Position der Zeilen innerhalb des Puffers. *adresse1* bis *adresse2* stehen für einen Zeilenbereich, der durch das *kommando* verarbeitet werden soll. Fehlt *adresse2*, so wird das Kommando nur für die Zeile durchgeführt, die durch *adresse1* angegeben wird.
kommando	Bei *kommando* handelt es sich um einen Einzelbuchstaben, der den Editor zur Durchführung einer bestimmten Funktion anweist.
argument	Die *argumente* eines *kommando*s sind die zu ändernden Textteile, ein Dateiname oder eine weitere Zeilenadresse.

Wenden Sie die ed-Kommandos auf einige selbsterdachte Beispiele an, um mit dem Kommandoformat vertraut zu werden.

Zeilenadressierung

Als Zeilenadresse werden ein oder mehrere Zeichen bezeichnet, durch die eine Textzeile angegeben wird. Damit `ed` Kommandos zum Einfügen, Löschen, Versetzen oder Ändern von Text ausführen kann, muß es die Zeilenadresse des betreffenden Texts kennen. Die Zeilenadresse geben Sie unmittelbar vor dem Kommando ein:

[adresse1],[adresse2]kommando<CR>

Sowohl *adresse1* als *adresse2* sind optionale Argumente. Mit *adresse1* weisen Sie den Editor an, nur eine einzige Textzeile zu bearbeiten, mit *adresse1* und *adresse2* einen Zeilenbereich. Wenn Sie überhaupt keine *adresse* angeben, geht `ed` davon aus, daß die Zeilenadresse sich auf die aktuelle Zeile bezieht.

Die Zeilenadressierung erfolgt unter `ed` in der Regel in folgendem Format:

- Sie geben Zeilennummern ein (dabei wird davon ausgegangen, daß die Zeilen innerhalb der Dateien, beginnend bei der ersten Zeile der Datei, in aufsteigender Folge von 1 bis *n* durchnumeriert sind).
- Sie geben Sonderzeichen ein, die für die aktuelle Zeile, die letzte Zeile oder einen Zeilenbereich stehen.
- Sie addieren Zeilennummern zur Nummer der aktuellen Zeile, oder subtrahieren Zeilennummern von der Nummer der aktuellen Zeile.
- Sie suchen nach einer Zeichenkette bzw. einem Wort, das in der betreffenden Zeile enthalten ist.

Sie können auf eine einzelne Zeile oder einen Zeilenbereich zugreifen, oder nach allen Zeilen suchen, die eine bestimmte Zeichenkette enthalten (als Zeichenkette wird eine Reihe von aufeinanderfolgenden Zeichen wie z.B. ein Wort bezeichnet).

Numerische Adresse

Unter ed ist jeder Zeile im Puffer eine numerische Adresse (Zeilennummer) zugeordnet. Die erste Zeile im Puffer hat die Adresse 1, die zweite die Adresse 2 usw. Unter ed ist der Zugriff auf jede Zeile über die numerische Zeilenadresse möglich. Versuchen Sie eine Zeile in der Datei try-me zu adressieren. Hierzu rufen Sie die Datei mit ed auf und geben eine Zeilennummer ein.

```
$ ed try-me<CR>
110
1<CR>
This is the first line of text.
3<CR>
and this is the third line.
```

Wie bereits erwähnt, wird bei der Eingabe einer Zeilenadresse ohne ein Kommando automatisch das Kommando p ausgeführt. Da Sie eine Zeilenadresse eingegeben haben, geht ed davon aus, daß Sie diese Zeile auf Ihrem Bildschirm abrufen möchten.

Die numerischen Zeilenadressen ändern sich während einer Editor-Sitzung ständig. An einer späteren Stelle dieses Kapitels finden Sie Informationen darüber, wie Sie Zeilen anlegen, löschen und an eine andere Stelle in der Datei versetzen und dadurch die numerischen Zeilenadressen einiger Zeilen ändern können. Die Nummer einer bestimmten Zeile entspricht in jedem Fall der aktuellen Position dieser Zeile im Editor-Puffer. Wenn Sie beispielsweise zwischen Zeile 5 und 6 fünf Textzeilen einfügen, so wird Zeile 6 zu Zeile 11. Wenn Sie dagegen Zeile 5 löschen, wird Zeile 6 zu Zeile 5.

Symbolische Adresse

Die symbolische Adresse für die aktuelle Zeile

Als aktuelle Zeile wird diejenige Zeile bezeichnet, die Sie zuletzt mit einem ed-Kommando bearbeitet haben. Unmittelbar nach dem Aufrufen von ed mit einer bereits vorhandenen Datei ist die letzte Zeile im Puffer die aktuelle Zeile. Das Symbol für die Adresse der aktuellen Zeile ist der Punkt (.). Somit können Sie den Inhalt der aktuellen Zeile abrufen, indem Sie einen Punkt eingeben und die Taste RETURN betätigen.

Probieren Sie dieses Kommando an der Datei try-me aus:

```
$ ed try-me<CR>
110
.<CR>
This is the fourth line.
```

Der Punkt (.) wurde hier als Adresse verwendet. Da auf den Punkt kein Kommando folgt, führt ed das Standard-Kommando p aus und gibt die Zeile aus, die sich an dieser Adresse des Puffers befindet.

Die Nummer der aktuellen Zeile können Sie mit dem folgenden Kommando abrufen:

```
.=<CR>
```

ed gibt jetzt die Zeilennummer aus. So hat in der Datei try-me die aktuelle Zeile die Nummer 4:

```
.<CR>
This is the fourth line.
.=<CR>
4
```

Die symbolische Adresse für die letzte Zeile

Als symbolische Adresse für die letzte Zeile einer Datei wird das Dollar-Zeichen (`$`) benutzt. Versuchen Sie mit dem Symbol `$` auf die letzte Zeile der Datei `try-me` zuzugreifen; dazu rufen Sie die Datei mit `ed` auf und geben diese Adresse in einer separaten Zeile ein (wie bereits erwähnt, befindet sich die aktuelle Zeile unmittelbar nach dem Aufrufen einer Datei automatisch am Dateiende).

```
$ ed try-me<CR>
110
.<CR>
This is the fourth line.
$<CR>
This is the fourth line.
```

Beachten Sie den Unterschied zwischen der Adresse `$`, die Sie beim Arbeiten nit `ed` benutzen, und der Shell- Eingabeaufforderung `$`.

Die symbolische Adresse für alle Zeilen

Als Adresse können Sie auch ein Komma (`,`) eingeben; hiermit adressieren Sie sämtliche Zeilen einer Datei (von der ersten bis zur letzten Zeile). Bei dieser Adresse handelt es sich um die kürzere Variante der bereits besprochenen Zeichenkette zur Adressierung sämtlicher Zeilen in einer Datei (Format: `1,$`). Rufen Sie mit dieser kürzeren Variante den Inhalt der Datei `try-me` auf dem Bildschirm ab:

```
,p<CR>
This is the first line of text.
This is the second line,
and this is the third line.
This is the fourth line.
```

Die symbolische Adresse für den Zeilenbereich von der aktuellen bis zur letzten Zeile

Das Semikolon (;) steht für den Zeilenbereich zwischen der aktuellen und der letzten Zeile der Datei. Hierbei handelt es sich um die funktionsgleiche Variante der symbolischen Adresse .,$. Benutzen Sie diese Adresse mit der Datei `try-me`:

```
2p<CR>
This is the second line,
;p<CR>
This is the second line,
and this is the third line.
This is the fourth line.
```

Zur aktuellen Zeile relative Adresse

Sie können auch eine Zeile angeben, die sich in einem bestimmten Abstand von der aktuellen Zeile befindet. Hierzu addieren Sie zur Nummer der aktuellen Zeile mit dem Plus-Zeichen (+) eine bestimmte Zeilenzahl bzw. subtrahieren mit dem Minus-Zeichen (–) von der Zeilennummer eine bestimmte Zeilenzahl. Eine so entstandene Adresse wird als relative Adresse bezeichnet. Probieren Sie die relativen Zeilenadressen an Ihrer Datei `try-me` aus, indem Sie einige Zeilen einfügen (siehe nächstes Beispiel). Sichern Sie die Änderungen abschließend in der Datei ab:

```
$ ed try-me<CR>
110
.<CR>
This is the fourth line.
a<CR>
five<CR>
six<CR>
seven<CR>
eight<CR>
nine<CR>
ten<CR>
.<CR>
w<CR>
140
```

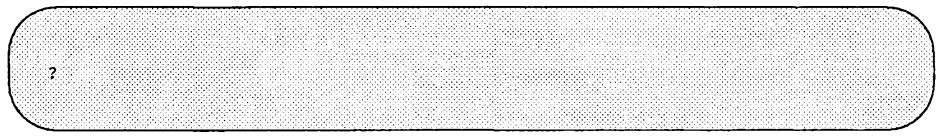
Subtrahieren Sie jetzt verschiedene Zeilennummern von der aktuellen Zeile bzw. addieren Sie Zeilennummern zu der aktuellen Zeile.

```
4<CR>
This is the fourth line.
+3<CR>
seven
-5<CR>
This is a second line,
```

Versuchen Sie eine Zeilennummer anzugeben, die höher ist als diejenige der letzten Zeile der Datei, bzw. eine Zeilennummer zu subtrahieren, die höher ist als diejenige der aktuellen Zeile:

```
5<CR>
five
-6<CR>
?
.=<CR>
5
+7<CR>
```

(Fortsetzung auf nächster Seite)



Beachten Sie, daß die aktuelle Zeile weiterhin bei Zeile 5 im Puffer bleibt. Die aktuelle Zeile ändert sich nur, wenn Sie unter `ed` eine korrekte Adresse angeben; andernfalls meldet es mit einem Fragezeichen (`?`) einen Fehler. Im Abschnitt "Weitere nützliche Kommandos und Informationen" am Ende dieses Kapitels finden Sie Informationen darüber, wie Sie eine etwas aussagekräftigere Meldung erhalten, aus der die Fehlerursache hervorgeht.

Alphanumerische Adressen

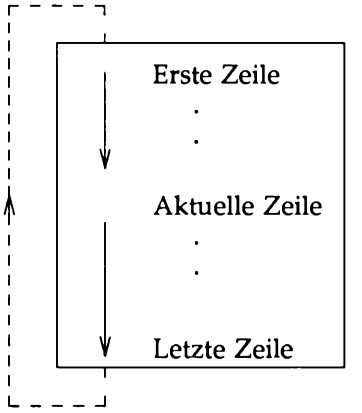
Eine Datei können Sie in Vor- und Rückwärtsrichtung nach einer bestimmten Zeichenkette durchsuchen. Hierzu geben Sie die Zeichenkette ein, der ein sogenannter Begrenzer vorangeht.

Durch die Begrenzer werden Zeichenketten voneinander abgegrenzt, damit `ed` deren Anfang und Ende erkennen kann. Der gebräuchlichste Begrenzer ist der Schrägstrich (`/`); er wird in folgendem Format benutzt:

/muster

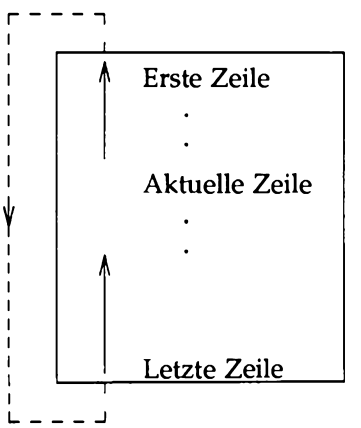
Mit einem Muster, dem ein Schrägstrich (`/`) vorangeht, starten Sie einen Suchvorgang in Vorwärtsrichtung; `ed` beginnt dann bei der aktuellen Zeile und durchsucht die Datei nach der nächsten Zeile, in der das Muster enthalten ist. Sobald es mit dem Suchvorgang bei der letzten Zeile im Puffer angelangt ist, setzt `ed` den Suchvorgang am Anfang des Puffers, d.h. bei Zeile 1, fort.

Das nachfolgende Rechteck stelle den Editor-Puffer dar. Die Pfeile zeigen den Ablauf des Suchvorgangs bei der Eingabe eines Schrägstrichs (`/`):



Mit einem Muster, dem ein Fragezeichen (?) vorangestellt ist, starten Sie einen Suchvorgang in Rückwärtsrichtung. ed beginnt den Suchvorgang dann bei der aktuellen Zeile und durchsucht die vorhergehenden Zeilen im Puffer nach der nächsten Zeile, die das Muster enthält. Ist es mit dem Suchvorgang bei der ersten Zeile der Datei angelangt, setzt ed den Suchvorgang bei der letzten Zeile der Datei fort, bis es wieder bei der aktuellen Zeile angelangt ist.

Das folgende Rechteck stelle den Editor-Puffer dar. Die Pfeile zeigen den Ablauf des Suchvorgangs bei der Eingabe eines Fragezeichens (?):



Rufen Sie die Datei `try-me` auf und suchen Sie nach beiden Methoden nach bestimmten Zeichenketten. Was passiert, wenn `ed` die angegebene Zeichenkette nicht finden kann?

```
$ ed try-me<CR>
140
.<CR>
ten
?first<CR>
This is the first line of text.
/fourth<CR>
This is the fourth line.
/junk<CR>
?
```

In diesem Beispiel fand `ed` die angegebenen Zeichenketten `first` und `fourth`. Da bei der Adresse kein Kommando angegeben worden ist, führte es das Standard-Kommando `p` aus und zeigt somit die gefundenen Zeilen an. Wenn `ed` eine Zeichenkette (z.B. `junk`) nicht ausfindig machen kann, gibt es ein Fragezeichen (`?`) aus.

Mit dem Schrägstrich (`/`) können Sie auch nach mehreren Stellen mit einem bestimmten Muster suchen, ohne das Muster mehrmals eingeben zu müssen. Geben Sie zuerst wie normal das Muster ein (Eingabe: */muster*). Nachdem `ed` die erste Stelle ausgegeben hat, wartet es auf ein weiteres Kommando. Wenn Sie jetzt einen Schrägstrich (`/`) eingeben und die Taste `RETURN` betätigen, so sucht `ed` nach einer weiteren Stelle in der Datei, an der das angegebene Muster vorkommt (der Suchvorgang findet hier in Vorwärtsrichtung statt). Suchen Sie mit diesem Kommando in der Datei `try-me` das Wort `line`:

```
.<CR>
This is the first line of text.
/line<CR>
This is the second line,
/<CR>
and this is the third line.
/<CR>
This is the fourth line.
/<CR>
This is the first line of text.
```

Nachdem ed alle Stellen zwischen der aktuellen Zeile und dem Dateiende ausfindig gemacht hat, an denen das Suchmuster vorkommt, setzt es den Suchvorgang am Dateianfang fort.

Angabe eines Zeilenbereichs

Wenn Sie mehrere Zeilen bearbeiten möchten, haben Sie zwei Möglichkeiten: Sie können entweder einen Zeilenbereich (z.B. *adresse1* bis *adresse2*) angeben, oder sämtliche Zeilen suchen, die ein bestimmtes Muster enthalten (sogenannter globaler Suchvorgang).

Die einfachste Methode zur Angabe eines Zeilenbereichs besteht darin, die Zeilennummern der ersten und der letzten Zeile innerhalb des Bereichs einzugeben; die beiden Zeilennummern müssen dann durch ein Komma voneinander getrennt sein. Diese Adresse setzen Sie in der Kommandozeile unmittelbar vor das Kommando. Wenn Sie beispielsweise die Zeilen 2 bis 7 des Editor-Puffers abrufen möchten, geben Sie *adresse1* (d.h. die Zahl 2) und *adresse2* (d.h. die Zahl 7) in folgendem Format ein:

```
2,7p<CR>
```

Benutzen Sie diese Adressierungsmöglichkeit in der Datei `try-me`:

```
2,7p<CR>
This is the second line,
and this is the third line.
This is the fourth line.
five
six
seven
```

Haben Sie vergessen, nach der Adresse `2,7` das Kommando `p` einzugeben? In diesem Fall gibt `ed` nur *adresse2* aus, also die letzte Zeile innerhalb des Zeilenbereichs.

Bei der Angabe eines Zeilenbereichs können Sie auch mit relativen Adressen arbeiten. Sie müssen aber auf jeden Fall sicherstellen, daß *adresse1* im Puffer vor *adresse2* steht. Relative Adressen werden, wie das folgende Beispiel zeigt, auf der Basis der aktuellen Zeile berechnet:

```
4<CR>
This is the fourth line
-2,+3p<CR>
This is the second line,
and this is the third line.
This is the fourth line.
five
six
seven
```

Globaler Suchvorgang

Die folgenden beiden Kommandos weichen vom üblichen `ed`-Kommandoformat ab: `g` und `v`. Hierbei handelt es sich um globale Suchkommandos, bei denen die Adressierung über Zeichenketten (Muster) erfolgt. Mit dem Kommando `g` werden sämtliche Zeilen gesucht, die das Muster enthalten; für diese Zeilen wird das angegebene Kommando dann ausgeführt. Das Kommando `v` sucht nach allen Zeilen, in denen das Muster nicht enthalten ist; für diese Zeilen wird das angegebene Kommando dann ausgeführt.

Diese Kommandos haben folgendes grundlegendes Format:

g/*muster*/*kommando*<CR>
v/*muster*/*kommando*<CR>

Suchen Sie mit diesen Kommandos in der Datei `try-me` nach dem Wort `line`:

```
g/line/p<CR>
This is the first line of text.
This is the second line,
and this is the third line.
This is the fourth line
```

```
v/line/p<CR>
five
six
seven
eight
nine
ten
```

Beachten Sie die Funktion des Kommandos `v`: Sie können damit nach allen Zeilen suchen, in denen das angegebene Wort (`line`) nicht enthalten ist.

Auch für eine mit `g` oder `v` adressierte Zeile wird das Kommando `p` standardmäßig ausgeführt, wenn nicht explizit ein Kommando angegeben worden ist; in diesem Fall kann das Kommando `p` als letzter Begrenzer innerhalb der Kommandozeile entfallen.

```
g/line<CR>
This is the first line of text.
This is the second line,
and this is the third line.
This is the fourth line
```

Wenn Sie dagegen die Adressen von Zeilen eingeben, die durch andere ed-Kommandos bearbeitet werden sollen, so sind die Begrenzer am Anfang und Ende obligatorisch. Die in diesem Abschnitt besprochenen Möglichkeiten zur Zeilenadressierung sind bei sämtlichen ed-Kommandos zulässig. In Tabelle 6-2 finden Sie eine Übersicht über die Symbole und Kommandos, mit denen Zeilen adressiert werden können.

Tabelle 6-2: Symbole und Kommandos zur Zeilenadressierung

Adresse	Funktion
n...	Die Nummer einer Zeile im Puffer
.	Die aktuelle Zeile (die zuletzt mit einem ed-Kommando bearbeitete Zeile)
.=	Kommando zur Abfrage der Zeilennummer der aktuellen Zeile
$	Die letzte Zeile der Datei
43,	Der Zeilenbereich, der bei Zeile 1 beginnt und mit der letzten Zeile endet
;	Der Zeilenbereich zwischen der aktuellen Zeile (einschließlich) und der letzten Zeile (einschließlich)
+ *n*	Die Zeile, die *n* Zeilen nach der aktuellen Zeile liegt.
− *n*	Die Zeile, die *n* Zeilen vor der aktuellen Zeile liegt.
/*abc*	Das Kommando, mit dem die erste Zeile im Puffer gesucht wird, in der das Muster *abc* vorkommt (Suche in Vorwärtsrichtung)
?*abc*	Das Kommando, mit dem die erste Zeile im Puffer gesucht wird, in der das Muster *abc* vorkommt (Suche in Rückwärtsrichtung)
g/*abc*	Alle Zeilen, in denen das Muster *abc* vorkommt.
v/*abc*	Alle Zeilen, in denen das Muster *abc* NICHT vorkommt.

Übung 2

2-1. Legen Sie die Datei `towns` mit dem folgenden Inhalt an:

```
My kind of town is
Chicago
Like being no where at all in
Toledo
I lost those little town blues in
New York
I lost my heart in
San Francisco
I lost $$ in
Las Vegas
```

2-2. Rufen Sie den Inhalt von Zeile 3 auf dem Bildschirm ab.

2-3. Welche Zeilen werden ausgegeben, wenn Sie einen Zeilenbereich mit der relativen Adresse –2,+3p angeben?

2-4. Welche Nummer hat die aktuelle Zeile? Rufen Sie den Inhalt der aktuellen Zeile auf dem Bildschirm ab.

2-5. Welchen Inhalt hat die letzte Zeile?

2-6. Welche Zeile wird beim folgenden Suchmuster angezeigt?

```
?town<CR>
```

Warten Sie bis `ed` das Ergebnis dargestellt hat, und geben Sie dann in einer separaten Zeile das folgende Kommando ein:

```
?<CR>
```

Was ist das Ergebnis?

2-7. Suchen Sie nach sämtlichen Zeilen, in denen das Muster `in` enthalten ist. Suchen Sie dann nach allen Zeilen, in denen das Muster `in` NICHT enthalten ist.

Abrufen des Dateiinhalts

Unter `ed` gibt es drei Kommandos, mit denen der Inhalt des Editor-Puffers auf dem Bildschirm abgerufen werden kann: `p`, `n` und `l` (das Kommando `l` wird an einer späteren Stelle dieses Kapitels besprochen).

Nur Text: Das Kommando `p`

Mit dem Kommando `p` haben Sie bereits in einigen Beispielen gearbeitet, so daß Ihnen sein allgemeines Format bekannt sein dürfte:

[*adresse1*,*adresse2*]p<CR>

Beim Kommando `p` gibt es keine Argumente. Allerdings kann es mit einem Ersetzen-Kommando kombiniert werden. Nähere Informationen zu diesem Thema finden Sie an einer späteren Stelle dieses Kapitels.

Geben Sie die in Tabelle 6-3 aufgeführten Zeilenadressen für eine Datei in Ihrem Home- Verzeichnis an. Kombinieren Sie jede Adresse mit dem Kommando `p` und überprüfen Sie, ob `ed` erwartungsgemäß reagiert.

Tabelle 6-3: Abrufen von Text über Zeilenadressen - Beispiele

Einzugebende Adresse	Erwartete Reaktion
`1,$p<CR>`	`ed` sollte den Inhalt der gesamten Datei auf Ihrem Bildschirm ausgeben.
`-5p<CR>`	`ed` sollte diejenige Zeile ausgeben, die fünf Zeilen über der aktuellen Zeile liegt.
`+2p<CR>`	`ed` sollte diejenige Zeile ausgeben, die zwei Zeilen unter der aktuellen Zeile liegt.
`1,/`*x*`/p<CR>`	`ed` gibt sämtliche Zeilen zwischen Zeile 1 und der ersten Zeile nach der aktuellen Zeile aus, in der das Zeichen *x* enthalten ist. Vergessen Sie nicht, den Buchstaben *x* in Schrägstriche einzuschließen, damit `ed` das Suchmuster zur Zeilenadressierung (*x*) vom `ed`- Kommando (`p`) unterscheiden kann.

Abrufen von Text über Zeilenadressen: Das Kommando `n`

Das Kommando `n` rufen Sie auf, wenn nicht nur der Inhalt einer Zeile angezeigt werden soll, sondern auch ihre Zeilennummer. Diese Möglichkeit ist besonders beim Löschen, Anlegen oder Ändern von Zeilen hilfreich. Das Kommando `n` hat dasselbe allgemeine Format wie das Kommando `p`:

[*adresse1*,*adresse2*]n<CR>

Bei `n` werden, ebenso wie bei `p`, keine Argumente eingegeben; allerdings kann es mit dem Ersetzen-Kommando kombiniert werden.

Geben Sie n in die Datei try-me ein:

```
$ ed try-me<CR>
140
1,$n<CR>
1       This is the first line of text.
2       This is the second line,
3       and this is the third line.
4       This is the fourth line.
5       five
6       six
7       seven
8       eight
9       nine
10      ten
```

Tabelle 6-4 enthält einen Überblick über die ed-Kommandos zum Abrufen von Text.

Tabelle 6-4: Kommandos zum Abrufen von Text

Kommando	Funktion
p	Die angegebenen Zeilen werden auf dem Bildschirm angezeigt.
n	Der Inhalt der angegebenen Zeilen wird, zusammen mit ihren Zeilennummern, auf dem Bildschirm angezeigt.

Erstellen von Text

Zum Erstellen von neuen Textzeilen in einer Datei gibt es unter `ed` drei grundlegende Kommandos:

`a`	(für append) Text anfügen
`i`	(für insert) Text einfügen
`c`	(für change) Text überschreiben

Anfügen von Text: Das Kommando `a`

Mit dem Kommando `a` (für append) können Sie Text NACH der aktuellen Zeile bzw. einer bestimmten Adresse einfügen. Das Kommando haben Sie bereits im Abschnitt "Einstieg" dieses Kapitels kennengelernt. Das Kommando `a` hat folgendes allgemeines Format:

[*adresse1*]a<CR>

Die Adresse kann weggelassen werden; wenn Sie keine Adresse angeben, wird für *adresse1* standardmäßig die Adresse der aktuellen Zeile eingesetzt.

In einigen der vorhergehenden Übungen haben Sie dieses Kommando mit der Standard-Adresse benutzt. Setzen Sie nun probehalber für *adresse1* andere Zeilennummern ein. Im nächsten Beispiel wird eine neue Datei namens `newfile` angelegt. In der ersten `a`- Kommandozeile wird die aktuelle Zeile (also die Standardadresse) benutzt. In der zweiten `a`-Kommandozeile wird für *adresse1* dann Zeile 1 eingesetzt. Da zum Abrufen der Zeilen das Kommando `n` benutzt wird, können Sie sich gleichzeitig über ihre numerischen Zeilenadressen informieren. Zum Verlassen des Anfügen-Modus geben Sie in einer separaten Zeile den bereits bekannten Punkt (`.`) ein.

```
$ ed new-file<CR>
?new-file
a<CR>
Create some lines<CR>
of text in<CR>
this file.<CR>
.<CR>
1,$n<CR>
1       Create some lines
2       of text in
3       this file.
1a<CR>
This will be line 2<CR>
This will be line 3<CR>
.<CR>
1,$n<CR>
1       Create some lines
2       This will be line 2
3       This will be line 3
4       of text in
5       this file.
```

Beachten Sie, daß die ursprüngliche Zeile 2 (`of text in`) nach dem Anfügen der beiden neuen Zeilen jetzt zu Zeile 4 geworden ist.

Zur Angabe der Stellen in der Datei, an denen Sie Text anfügen möchten, können Sie in der a-Kommandozeile auch mit symbolischen Adressen arbeiten. Mit den folgenden drei Kommandozeilen können Sie schnell an die verschiedenen Stellen im Text springen, an denen Sie Text anfügen möchten.

`.a<CR>`	Der Text wird nach der aktuellen Zeile eingefügt
`$a<CR>`	Der Text wird nach der letzten Zeile der Datei eingefügt.
`0a<CR>`	Der Text wird vor der ersten Zeile der Datei eingefügt (für Zeile 0 wird eine symbolische Adresse eingegeben).

Um diese Adressen kennenzulernen, legen Sie die Datei `lines` (Umfang: eine einzige Zeile) an und geben die folgenden drei Beispiele ein (die Beispiele sind der Übersichtlichkeit halber in separaten Abbildungen enthalten. Zur Benutzung der drei Symbole müssen Sie die Datei `lines` nicht drei mal aufrufen - rufen

Sie die Datei `lines` einmal auf und probieren Sie die drei Symbole hintereinander aus).

```
$ ed lines<CR>
?lines
a<CR>
This is the current line.<CR>
.<CR>
p<CR>
This is the current line.
.a<CR>
This line is after the current line.<CR>
.<CR>
-1,.p<CR>
This is the current line.
This line is after the current line.
```

```
$a<CR>
This is the last line now.<CR>
.<CR>
$<CR>
This is the last line now.
```

```
0a<CR>
This is the first line now.<CR>
This is the second line now.<CR>
The line numbers change<CR>
as lines are added.<CR>
.<CR>
1,4n<CR>
1       This is the first line now.
2       This is the second line now.
3       The line numbers change
4       as lines are added.
```

Da das Kommando a den Text hinter einer bestimmten Adresse erstellt, wird die Zeile vor Zeile 1 im letzten Beispiel als diejenige Zeile angegeben, die auf Zeile 0 folgt. Um derartige zweideutige Adressen zu vermeiden, sollten Sie für diese Zwecke ein weiteres ed- Kommando benutzen: i (für insert).

Einfügen von Text: Das Kommando i

Mit dem Kommando i (für insert) können Sie VOR einer bestimmten Zeile im Puffer Text einfügen. Das Kommando i hat dasselbe allgemeine Format wie a:

[*adresse1*]i<CR>

Wie mit dem Kommando a können Sie auch mit dem Kommando i eine oder mehrere Zeilen einfügen. Um den Einfügemodus zu verlassen, geben Sie in einer separaten Zeile einen Punkt (.) ein.

Legen Sie eine Datei namens insert an, an der Sie das Einfügen-Kommando (i) ausprobieren können:

```
$ ed insert<CR>
?insert
a<CR>
Line 1<CR>
Line 2<CR>
Line 3<CR>
Line 4<CR>
.<CR>
w<CR>
28
```

Fügen Sie jetzt vor Zeile 2 eine Textzeile ein und eine weitere über Zeile 1. Rufen Sie mit dem Kommando n sämtliche Zeilen auf dem Bildschirm ab, die im Puffer enthalten sind:

```
2i<CR>
This is the new line 2.<CR>
.<CR>
1,$n<CR>
1       Line 1
2       This is the new line 2.
3       Line 2
4       Line 3
5       Line 4
1i<CR>
This is the beginning.<CR>
.<CR>
1,$n<CR>
1       This is the beginning.
2       Line 1
3       This is the new line 2.
4       Line 2
5       Line 3
6       Line 4
```

Kombinieren Sie das Kommando `i` probehalber mit symbolischen Zeilenadressen:

- `.i<CR>`
- `$i<CR>`

Überschreiben von Text: Das Kommando `c`

Mit dem Kommando `c` (für change) werden sämtliche angegebenen Zeilen gelöscht und durch ein oder mehrere Textzeilen ersetzt. Da Sie mit `c` einen Zeilenbereich löschen können, enthält das allgemeine Format für die Kommandozeile zwei Adressen:

[*adresse1*,*adresse2*]c<CR>

Das Kommando `c` schaltet automatisch in den Eingabemodus um. Zum Verlassen des Eingabemodus müssen Sie in einer separaten Zeile einen Punkt (.) eingeben.

adresse1 ist die erste Zeile, *adresse2* die letzte Zeile innerhalb des Textbereichs, der durch neuen Text ersetzt werden soll. Wenn Sie lediglich eine einzige Zeile löschen möchten, so genügt die Eingabe von *adresse1*. Fehlt die Adresse völlig, so geht `ed` davon aus, daß die aktuelle Zeile überschrieben werden soll.

Legen Sie jetzt eine Datei namens `change` an, an der Sie dieses Kommando ausprobieren können. Nach der Eingabe des Texts, der im folgenden Beispiel gezeigt wird, ändern Sie die Zeilen 1 bis 4 mit dem Kommando `1,4c`:

```
1,5n<CR>
1       line 1
2       line 2
3       line 3
4       line 4
5       line 5
1,4c<CR>
Change line 1<CR>
and lines 2 through 4<CR>
.<CR>
1,$n<CR>
1       change line 1
2       and lines 2 through 4
3       line 5
```

Ändern Sie nun probehalber die aktuelle Zeile:

```
.<CR>
line 5
c<CR>
This is the new line 5.<CR>
.<CR>
.<CR>
This is the new line 5.
```

Wenn Sie sich nicht sicher sind, ob Sie den Eingabemodus verlassen haben, so können Sie sicherheitshalber einen weiteren Punkt eingeben. Wenn die aktuelle Zeile angezeigt wird, befinden Sie sich auf jeden Fall im `ed`- Kommandomodus.

Tabelle 6-5 zeigt eine Übersicht über die ed-Kommandos zum Erstellen von Text.

Tabelle 6-5: Kommandos zum Erstellen von Text

Kommando	Funktion
a	Nach der angegebenen Zeile im Puffer wird Text eingefügt.
i	Vor der angegebenen Zeile im Puffer wird Text eingefügt.
c	Der Text in der bzw. den angegebenen Zeile(n) wird ersetzt.
.	Verlassen des Eingabemodus und Umschalten in den ed-Kommandomodus.

Übung 3

3-1. Legen Sie eine Datei namens `ex3` an. Fpgen Sie in den leeren Puffer statt mit dem Kommando `a` mit dem Kommando `i` Text ein. Was ist das Ergebnis?

3-2. Rufen Sie `ed` mit der Datei `towns` auf. Welche Adresse hat die aktuelle Zeile?

Fügen Sie folgendes vor der dritten Zeile ein:

```
Illinois<CR>
```

Fügen Sie folgendes über der aktuellen Zeile ein:

```
or<CR>
Naperville<CR>
```

Fügen Sie folgendes vor der letzten Zeile ein:

```
hotels in<CR>
```

Rufen Sie den Inhalt der Textzeilen mit Zeilennummern ab.

3-3. Rufen Sie die Zeilen 1 bis 5 der Datei `towns` auf dem Bildschirm ab und ersetzen Sie die Zeilen 2 bis 5 durch:

```
London<CR>
```

Rufen Sie den Inhalt der Zeile 1 bis 3 auf dem Bildschirm ab.

3-4. Welche Zeilennummer hat die aktuelle Zeile nach der Beendigung von Übung 3-3?

Suchen Sie nach der Textzeile mit folgendem Inhalt:

```
Toledo
```

Wechseln Sie

```
Toledo
```

gegen

```
Peoria
```

aus. Rufen Sie den Inhalt der aktuellen Zeile auf dem Bildschirm ab.

3-5 Suchen Sie mit einer einzigen Kommandozeile nach

```
New York
```

und ersetzen Sie es durch:

```
Iron City
```

Löschen von Text und Aufheben von Änderungen

In diesem Abschnitt geht es um diejenigen Kommandos, mit denen Sie unter ed Text löschen und Änderungen wieder rückgängig machen können. Diese Kommandos können nur im Kommandomodus aufgerufen werden. Mit dem Kommando d können Sie Zeilen löschen; mit dem Kommando u können Sie die Änderungen, die Sie mit dem letzten Kommando durchgeführt haben, wieder rückgängig machen.

Zeilen löschen im Kommandomodus: Das Kommando d

Die Vorgehensweise beim Löschen von Textzeilen mit dem Kommando (d) haben Sie bereits im Abschnitt "Einstieg" zu Beginn dieses Kapitels kennengelernt.

Das Kommando d hat folgendes allgemeines Format:

[*adresse1*,*adresse2*]d<CR>

Hiermit können Sie einen Zeilenbereich (*adresse1* bis *adresse2*) löschen, aber auch eine einzelne Zeile (*adresse1*). Wenn keine Adresse angegeben wird, löscht ed die aktuelle Zeile.

Im nächsten Beispiel werden zunächst die Zeilen 1 bis 5 angezeigt und dann die Zeilen 2 bis 4 gelöscht:

```
1,5n<CR>
1       1 horse
2       2 chickens
3       3 ham tacos
4       4 cans of mustard
5       5 bails of hay
2,4d<CR>
1,$n<CR>
1       1 horse
2       5 bails of hay
```

Wenn Sie lediglich die letzte Zeile einer Datei löschen möchten, so ist dies problemlos mit einer symbolischen Zeilenadresse möglich:

```
$d<CR>
```

Wie wird die aktuelle Zeile gelöscht? Zu den häufigsten Bedienungsfehlern beim Arbeiten mit `ed` gehört, daß die Eingabe des Punkts zum Verlassen des Eingabemodus vergessen wird; dadurch wird häufig unerwünschter Text in den Puffer eingefügt. Im nächsten Beispiel wird eine Zeile mit dem Ausgabe-Kommando (`1,$p`) versehentlich in den Text eingefügt, bevor der Benutzer den Eingabe-
Modus verläßt. Da diese Zeile als letzte in den Text eingefügt worden ist, ist sie jetzt die aktuelle Zeile. Zum Löschen dieser Zeile geben Sie die symbolische Adresse Punkt (.) ein.

```
a<CR>
Last line of text<CR>
1,$p<CR>
.<CR>
p<CR>
1,$p
.d<CR>
p<CR>
Last line of text.
```

Vor Ihren ersten Versuchen mit dem Kommando `d` sollten Sie zuerst das Kommando `u` (für undo) kennenlernen, mit dem Sie Änderungen rückgängig machen können.

Aufheben des letzten Kommandos im Kommandomodus: Das Kommando `u`

Das Kommando `u` (für undo) macht das letzte Kommando rückgängig, d.h., es stellt den Text, der durch das Kommando geändert oder gelöscht worden ist, wieder vollständig her. Bei diesem Kommando werden weder Adressen noch Argumente eingegeben. Die Kommandozeile hat folgendes Format:

```
u<CR>
```

Das Kommando u ist hauptsächlich zur Wiederherstellung von Text hilfreich, der versehentlich gelöscht worden ist. Wenn Sie alle Zeilen in einer Datei löschen und dann das Kommando p eingeben, so gibt ped ein Fragezeichen (?) aus, da in der Datei keine Zeilen mehr enthalten sind. Mit dem Kommando u können Sie dann den Dateiinhalt wiederherstellen.

```
1,$p<CR>
This is the first line.
This is the middle line.
This is the last line.
1,$d<CR>
p<CR>
?
u<CR>
p<CR>
This is the last line.
```

Versuchen Sie jetzt, mit Hilfe des Kommandos u das Kommando a rückgängig zu machen.

```
.<CR>
This is the only line of text
a<CR>
Add this line<CR>
.<CR>
1,$p<CR>
This is the only line of text
Add this line
u<CR>
1,$p<CR>
This is the only line of text
```

Hinweis: Das Abspeichern einer Datei (Kommando w) bzw. das Verlassen des Editors (Kommando q) kann mit dem Kommando u nicht rückgängig gemacht werden. Das Kommando u dagegen kann mit einem weiteren Kommando u selbst aufgehoben werden.

Tabelle 6-6 enthält eine Übersicht über die Kommandos und Tasten, mit denen Sie unter ed Text löschen können.

Tabelle 6-6: Kommandos zum Löschen von Text

Kommando	Funktion
d	Löschen von ein oder mehreren Textzeilen
u	Aufheben des letzten Kommandos

Ersetzen von Text

Mit dem Ersetzen-Kommando können Sie den Inhalt Ihrer Datei abändern. Mit diesem Kommando wird eine Zeichenkette bei ihrem ersten Auftreten durch neuen Text ausgetauscht. Die Kommandozeile hat folgendes allgemeines Format:

[*adresse1*,*adresse2*]s/*text_alt*/*text_neu*/[*kommando*]<CR>

Die einzelnen Komponenten der Kommandozeile haben folgende Funktion:

adresse1,*adresse2*	Der Zeilenbereich, der durch s adressiert wird. Die Adresse kann sich auf eine einzelne Zeile (*adresse1*) oder einen Zeilenbereich (*adresse1* bis *adresse2*) beziehen oder eine globale Adresse sein. Wenn keine Adresse angegeben wird, führt ed das Ersetzen nur in der aktuellen Zeile durch.
s	Das Ersetzen-Kommando selbst (s steht für substitute)
/*text_alt*	Das Argument, das den zu ersetzenden Text angibt, ist in der Regel in Schrägstriche eingeschlossen; als Begrenzer sind aber auch andere Zeichen wie z.B. das Fragezeichen (?) oder der Punkt (.) zulässig. Das Argument besteht aus den Wörtern bzw. Zeichen, die ausgetauscht werden sollen. Kommt *text_alt* in einer adressierten Zeile mehr als ein Mal vor, so wird es lediglich beim ersten Auftreten in der Zeile ausgetauscht. In diesem Kapitel wird *text_alt* als Suchmuster bezeichnet.
/*text_neu*	Dieses Argument steht für den Text, durch den *text_alt* ersetzt werden soll. Er ist in Schrägstriche (bzw. in dieselben Begrenzer wie *text_alt*) eingeschlossen und besteht aus den Wörtern bzw. Zeichen, die für *text_alt* eingesetzt werden sollen. In diesem Kapitel wird *text_neu* als Ersetzungs-Muster bezeichnet.
/*kommando*	Eines der folgenden vier Kommandos:

g	*text_alt* soll bei jedem Auftreten in jeder angegebenen Zeile ausgetauscht werden.
l	Die letzte Zeile des ersetzten Texts soll, einschließlich der nicht-darstellbaren Zeichen, ausgegeben werden (siehe den letzten Abschnitt dieses Kapitels, "Weitere nützliche Kommandos und Informationen").
n	Die letzte Zeile des neuen Texts wird, zusammen mit ihrer Zeilennummer, ausgegeben.
p	Die letzte Zeile des geänderten Texts wird ausgegeben.

Beschränkung auf die aktuelle Zeile

In seiner einfachsten Form ändert das Ersetzen- Kommando nur die aktuelle Zeile. Für die aktuelle Zeile brauchen Sie keine Zeilenadresse anzugeben:

s/*text_alt*/*text_neu*/<CR>

Das nächste Beispiel enthält einen Eingabefehler. Da er in der aktuellen Zeile enthalten ist, können Sie ihn mit dem Ersetzen- Kommando ohne Angabe einer Zeilenadresse korrigieren. Der zu ersetzende Text ist `ai` in `airor`, der neue Text `er`.

```
a<CR>
In the beginning, I made an airor.<CR>
.<CR>
.p<CR>
In the beginning, I made an airor.
s/ai/er/<CR>
```

Beachten Sie, daß `ed` nach der Ausführung des Ersetzen-Kommandos nichts (kein Ergebnis und keine Meldung) ausgibt. Wenn Sie die ordnungsgemäße Durchführung des Kommandos überprüfen möchten, so müssen Sie daher die Zeile entweder mit `p` oder mit `n` abrufen oder das Kommando `p` bzw. `n` in die

s-Kommandozeile einbringen. Im folgenden Beispiel wird mit Hilfe des Kommandos n überprüft, ob das Wort `toad` durch `file` ersetzt worden ist.

```
.p<CR>
This is a test toad
s/toad/file/n<CR>
1       This is a test file
```

Allerdings gibt es dafür unter ed noch eine kürzere Möglichkeit. Es gibt das Ergebnis des Kommandos immer dann automatisch aus, wenn Sie den letzten Begrenzer hinter dem Argument *text_neu* weglassen:

```
.p<CR>
This is a test file
s/file/frog<CR>
This is a test frog
```

Beschränkung auf eine bestimmte Zeile

Um Text zu ersetzen, der nicht in der aktuellen Zeile enthalten ist, müssen Sie in der Kommandozeile eine Adresse eingeben:

[*adresse1*]s/*text_alt*/*text_neu*/<CR>

So enthält das folgende Beispiel eine Kommandozeile mit der Adresse der zu ändernden Zeile (1), da die aktuelle Zeile sich in Zeile 3 befindet:

```
1,3p<CR>
This is a pest toad
testing testing
come in toad
.<CR>
come in toad
1s/pest/test<CR>
This is a test toad
```

In diesem Beispiel hat ed die neue Zeile nach der Änderung automatisch ausgegeben, da der letzte Begrenzer in der Kommandozeile weggelassen worden ist.

Ersetzen innerhalb eines Zeilenbereichs

Wenn Sie eine bestimmte Zeichenkette bei jedem Auftreten innerhalb eines Zeilenbereichs gegen eine andere austauschen möchten, so müssen Sie die Adresse der ersten Zeile (*adresse1*) und der letzten Zeile (*adresse2*) angeben:

[*adresse1*,*adresse2*]s/*text_alt*/*text_neu*/<CR>

Kann ed das zu ersetzende Muster innerhalb einer Zeile nicht finden, so läßt es diese Zeile unverändert.

Im folgenden Beispiel sind im Zeilenbereich beim Ersetzen-Kommando sämtliche Zeilen der Datei enthalten. Dennoch werden lediglich diejenigen Zeilen abgeändert, in denen die Zeichenkette es (das Argument *text_alt*) enthalten ist.

```
1,$p<CR>
This is a test toad
testing testing
come in toad
testing 1, 2, 3
1,$s/es/ES/n<CR>
4       tESting 1, 2, 3
```

Wenn am Ende einer Ersetzen-Kommandozeile, in der ein Zeilenbereich adressiert wird, das Kommando `p` oder `n` enthalten ist, so wird nur die letzte der abgeänderten Zeilen ausgegeben.

Um alle Zeilen auszugeben, in denen Text geändert worden ist, müssen Sie beim Kommando `n` bzw. `p` die Adresse `1,$` benutzen.

```
1,$n<CR>
1       This is a tESt toad
2       tESting testing
3       come in toad
4       tESting 1, 2, 3
```

Beachten Sie, daß die Zeichenkette `es` nur bei ihrem ersten Auftreten (in Zeile 2) geändert worden ist. Um ein Muster bei jedem Auftreten zu ändern, benutzen Sie das Kommando `g`, das im nächsten Abschnitt beschrieben wird.

Globales Ersetzen

Eine der vielseitigsten Funktionen von `ed` ist das globale Ersetzen. Wenn Sie hinter dem letzten Begrenzer in einer Ersetzen-Kommandozeile das Kommando `g` einfügen, können Sie ein Muster an jeder Stelle, an der es in den angegebenen Zeilen vorkommt, gegen eine andere Zeichenkette austauschen. Versuchen Sie die Zeichenkette `es` im letzten Beispiel bei jedem Auftreten zu ersetzen. Beim Nachvollziehen der folgenden Beispiele sollten Sie stets daran denken, daß Sie das letzte Ersetzen-Kommando mit dem Kommando `u` wieder rückgängig machen können.

```
u<CR>
1,$p<CR>
This is a test toad
testing, testing
come in toad
testing 1, 2, 3
1,$s/es/ES/g<CR>
1,$p<CR>
This is a tESt toad
tESting tESting
come in toad
tESting 1, 2, 3
```

Anstelle eines Zeilenbereichs (`1,$`) können Sie als Adresse auch ein globales Suchmuster verwenden:

```
1,$p<CR>
This is a test toad
testing testing
come in toad
testing 1, 2, 3
g/test/s/es/ES/g<CR>
1,$p<CR>
This is a tESt toad
tESting tESting
come in toad
tESting 1, 2, 3
```

Wenn das globale Suchmuster eindeutig ist und zum Argument *text_alt* (zu ersetzender Text) paßt, so gibt es unter `ed` eine kürzere Variante: Geben Sie das Muster einmal als globale Suchadresse an, und wiederholen Sie es nicht als Argument *text_alt*. `ed` speichert das Muster aus der Suchadresse ab und benutzt es erneut als zu ersetzendes Muster.

`g/`*text_alt*`/s//`*text_neu*`/g<CR>`

Hinweis: Bei dieser kürzeren Variante müssen Sie hinter dem Kommando `s` auf jeden Fall zwei Schrägstriche (`//`) in die Kommandozeile einfügen.

```
1,$p<CR>
This is a test toad
testing testing
come in toad
testing 1, 2, 3
g/es/s//ES/g<CR>
1,$p<CR>
This is a tESt toad
tESting tESting
come in toad
tESting 1, 2, 3
```

Probieren Sie auch andere Adressen in Form von Suchmustern aus:

/muster<CR>
?*muster*<CR>
v/*muster*<CR>

Verknüpfen Sie diese Kommandos mit dem Ersetzen- Kommandos und beobachten Sie das Ergebnis. Im folgenden Beispiel werden mit Hilfe des Suchmusters `v/`*muster* sämtliche Zeilen ausfindig gemacht, die nicht das Muster `testing` enthalten. Mit dem Ersetzen- Kommando (`s`) kann das vorhandene Muster (`in`) in diesen Zeilen durch ein neues Muster (`out`) ersetzt werden.

```
v/testing/s/in/out<CR>
This is a test toad
come out toad
```

Beachten Sie, daß auch die Zeile `This is a test toad` ausgegeben wurde, obwohl in ihr kein Text ersetzt worden ist. Wenn der letzte Begrenzer weggelassen wird, werden alle Zeilen, die durch das Suchmuster ausfindig gemacht worden sind, ausgegeben; dabei spielt es keine Rolle, ob Text in ihnen ausgetauscht worden ist, oder nicht.

Suchen Sie jetzt mit dem Kommando `g` nach allen Zeilen, in denen das Muster `testing` enthalten ist:

```
g/testing/s//jumping<CR>
jumping testing
jumping 1, 2, 3
```

Beachten Sie, daß dieses Kommando das Suchmuster (`testing`) nur beim jeweils ersten Auftreten in jeder Zeile austauscht. Auch in diesem Fall werden die Zeilen auf dem Bildschirm ausgegeben, da der letzte Begrenzer weggelassen worden ist.

Übung 4

4-1. Wechseln Sie in der Datei `towns` die Zeichenkette `town` in allen Zeilen gegen `city` aus, mit Ausnahme der Zeile, die die Wörter `little town` enthält.

Der Inhalt der Datei sollte folgendermaßen aussehen:

```
My kind of city is
London
Like being no where at all in
Peoria
I lost those little town blues in
Iron City
I lost my heart in
San Francisco
I lost $$ in
hotels in
Las Vegas
```

4-2. Verwenden Sie probehalber das Fragezeichen (`?`) als Begrenzer. Wechseln Sie in der aktuellen Zeile

```
Las Vegas
```

gegen

```
Toledo
```

aus. Da Sie in diesem Fall die ganze Zeile auswechseln, können Sie stattdessen auch das Kommando `c` (für change) benutzen.

4-3. Durchsuchen Sie die Datei in Rückwärtsrichtung nach dem Wort

```
lost
```

und ersetzen Sie es durch

```
found
```

Verwenden Sie dabei das Fragezeichen (`?`) als Begrenzer. Hat dies funktioniert?

4-4. Durchsuchen Sie die Datei in Vorwärtsrichtung nach

```
no
```

und ersetzen Sie es durch

```
NO
```

Wie sieht das Ergebnis aus, wenn Sie als Begrenzer das Fragezeichen (?) benutzen?

Probieren Sie die unterschiedlichen Kommandovarianten zur Adressierung eines Zeilenbereichs und zum globalen Suchen einer Zeichenkette aus.

Versuchen Sie `$$` zu ersetzen. Was hat das zur Folge? Versuchen Sie `$` in Zeile 9 Ihrer Datei gegen `Big $` auszutauschen. Geben Sie folgendes ein:

```
9s/$/Big $<CR>
```

Wie sieht die Datei jetzt aus?

Mustervergleiche mit Sonderzeichen

Wenn Sie versuchen, das Symbol $ in der Zeile

```
I lost my $ in Las Vegas
```

zu ersetzen, so werden Sie feststellen, daß nicht $ ausgetauscht, sondern der neuer Text am Ende der Zeile eingesetzt worden ist. Bei $ handelt es sich um ein ed-Sonderzeichen, das für das Zeilenende steht.

Unter ed gibt es eine Reihe von Sonderzeichen, die Sie als Kürzel in Such- und Ersetzungs-Mustern verwenden können. Wenn diese Zeichen in einem Such- oder Ersetzungs-Muster enthalten wären, sähe das Ergebnis anders aus, als Sie es erwarteten.

Einige der Sonderzeichen sind im folgenden aufgeführt:

`.`	Paßt zu einem beliebigen Einzelzeichen.
`*`	Paßt zu Null, einem oder mehr Auftreten des vorhergehenden Zeichens bzw. des in eckigen Klammern eingeschlossenen Ausdrucks.
`.*`	Paßt zu Null, einem oder mehr Auftreten eines beliebigen Zeichens.
`^`	Paßt zum Zeilenanfang.
`$`	Paßt zum Zeilenende.
`\`	Die Bedeutung des nachfolgenden Sonderzeichens wird aufgehoben.
`%`	Das letzte Ersetzungs-Muster wird erneut benutzt.
`&`	Dieses Zeichen wird innerhalb des Ersetzungs-Musters durch die erste Zeichenkette ersetzt, die zum Suchmuster paßt.
`[...]`	Paßt zum ersten Auftreten eines der in eckigen Klammern eingeschlossenen Zeichens.

`[^...]` Paßt zum jeweils ersten Zeichen, das NICHT in den eckigen Klammern enthalten ist.

Im folgenden Beispiel sucht `ed` nach jeder dreistelligen Zeichenkette, die mit dem Muster `at` endet:

```
1,$p<CR>
rat
cat
turtle
cow
goat
g/.at<CR>
rat
cat
goat
```

Beachten Sie, daß auch das Wort `goat` als passend betrachtet wird, da die Zeichenkette `oat` zum Muster `.at` paßt.

Der Stern (*) steht in einem Such- oder Ersetzungsmuster für Null, ein- oder mehrfaches Auftreten eines bestimmten Zeichens. Dieses Zeichen kann benutzt werden, um ein versehentlich wiederholt eingefügtes Zeichen zu löschen. Angenommen, Sie halten die Taste R bei der Eingabe des Worts `broke` zu lange gedrückt. In diesem Fall können Sie mit * jeden überzähligen Buchstaben `r` mit Hilfe eines einzigen Ersetzen-Kommandos austauschen.

```
p<CR>
brrroke
s/br*/br<CR>
broke
```

Beachten Sie, daß im Suchmuster vor dem ersten `r` der Kleinbuchstabe `b` enthalten ist. Wäre der Kleinbuchstabe `b` nicht im Suchmuster enthalten, so würde das Zeichen * ihn während des Suchvorgangs als Null Auftreten von `r` interpretieren, austauschen und den Suchvorgang beenden (wie bereits erwähnt, wird ein Muster nur bei seinem ersten Auftreten ausgetauscht, es sei denn, Sie haben mit dem Kommando `g` einen globalen Suchvorgang gestartet). Das

folgende Beispiel zeigt den Ablauf des Ersetzungs-Vorgangs, wenn Sie vor dem * nicht sowohl `b` als auch `r` angegeben hätten.

```
p<CR>
brrroke
s/r*/r<CR>
rbrrroke
```

Die Kombination aus Punkt und * paßt zu allen Zeichen. Mit dieser Kombination können Sie alle Zeichen im letzten Teil einer Zeile auswechseln:

```
p<CR>
Toads are slimy, cold creatures
s/are.*/are wonderful and warm<CR>
Toads are wonderful and warm
```

Außerdem können Sie mit `.*` alle Zeichen zwischen zwei Mustern auswechseln:

```
p<CR>
Toads are slimy, cold creatures
s/are.*cre/are wonderful and warm cre<CR>
Toads are wonderful and warm creatures
```

Wenn Sie am Anfang einer Zeile ein Wort einfügen möchten, so geben Sie das Zeichen ^ (Zierkumflex) als auszutauschenden Text an. Dadurch können Sie beispielsweise am Anfang mehrerer Zeilen ein bestimmtes Muster einfügen. Im nächsten Beispiel wird am Anfang jeder Zeile das Wort `all` eingefügt:

```
1,$p<CR>
creatures great and small
things wise and wonderful
things bright and beautiful
1,$s/^/all /<CR>
1,$p<CR>
all creatures great and small
all things wise and wonderful
all things bright and beautiful
```

Mit dem Symbol $ können Sie beispielsweise am Ende von einer Zeile bzw. einem Zeilenbereich Zeichen einfügen:

```
1,$p<CR>
I love
I need
I use
The IRS wants my
1,$s/$/ money.<CR>
1,$p<CR>
I love money.
I need money.
I use money.
The IRS wants my money.
```

In diesen Beispielen muß hinter dem Wort `all` bzw. vor dem Wort `money` ein Leerzeichen eingefügt werden, da `ed` die angegebenen Zeichen ganz am Anfang bzw. ganz am Ende des Satzes einfügt. Ohne Leerzeichen vor dem Wort `money` sähe Ihre Datei folgendermaßen aus:

```
1,$s/$/money/<CR>
1,$p<CR>
I lovemoney
I needmoney
I usemoney
The IRS wants mymoney
```

Das Symbol $ ist eine praktische Möglichkeit zum Einfügen eines Satzzeichens am Ende einer Zeile:

```
1,$p<CR>
I love money
I need money
I use money
The IRS wants my money
1,$s/$/./<CR>
1,$p/<CR>
I love money.
I need money.
I use money.
The IRS wants my money.
```

Da der Punkt (.) nicht im Suchmuster (text_alt) enthalten ist, sondern im Ersetzungs-Muster (text_neu), hat er keine spezielle Bedeutung. Zum Auswechseln eines Punkts, der in der Mitte einer Zeile enthalten ist, müssen Sie die Sonderbedeutung des Punkts im Argument text_alt aufheben. Hierzu stellen Sie dem Punkt einfach einen Gegenschrägstrich (\) voran. Nach dieser Methode können Sie generell die spezielle Bedeutung bestimmter Sonderzeichen aufheben, die Sie als normale Textzeichen in Such- oder Ersetzungs-Mustern verwenden möchten. Das folgende Beispiel zeigt, wie die spezielle Bedeutung des Punkts aufgehoben werden kann:

```
p<CR>
Way to go.  Wow!
s/\./!<CR>
Way to go!  Wow!
```

Auf dieselbe Weise kann auch die spezielle Bedeutung des Gegenschrägstrichs (\) selbst aufgehoben werden. Damit \ wie ein normales Textzeichen behandelt wird, müssen Sie ihm ebenfalls \ voranstellen. Wenn Sie beispielsweise das Zeichen \ gegen das Wort "Gegenschrägstrich" austauschen möchten, benutzen Sie das folgende Ersetzen-Kommando:

```
1,2p<CR>
This chapter explains
how to use the \.
s/\\/backslash<CR>
how to use the backslash.
```

Wenn Sie Text ersetzen möchten, ohne den Text im Ersetzungs-Muster zu wiederholen, können Sie als Kürzel das Et-Zeichen (&) benutzen. Für & innerhalb des Ersetzungs-Musters wird die Zeichenkette eingesetzt, die in der aktuellen Zeile zum Suchmuster paßt; dadurch müssen Sie das Suchmuster nicht zweimal eingeben. Beispiel:

```
p<CR>fP
The neanderthal skeletal remains
s/thal/& man's/<CR>
p<CR>
The neanderthal man's skeletal remains
```

Die Zeichenkette im letzten Suchmuster (text_alt) wird von `ed` automatisch zwischengespeichert. Zur Wiederholung des einzusetzenden Zeichens (d.h. des Ersetzungs-Musters) dagegen müssen Sie `ed` explizit anweisen; hierzu benutzen Sie das Symbol %. Mit dem Symbol % können Sie einen Ersetzungs-Vorgang in

mehreren Zeilen durchführen, ohne ein globales Ersetzen anzufordern. Wenn Sie z.B. das Wort `money` gegen das Wort `gold` auswechseln möchten, so wiederholen Sie den letzten Ersetzungs-Vorgang aus Zeile 1 in Zeile 3, nicht aber in Zeile 4.

```
1,$n<CR>
 1        I love money
 2        I need food
 3        I use money
 4        The IRS wants my money
1s/money/gold<CR>
I love gold
3s//%<CR>
I use gold
1,$n<CR>
 1        I love gold
 2        I need food
 3        I use gold
 4        The IRS wants my money
```

Das Wort `money` (d.h. der zu ersetzende Text) wird von `ed` automatisch zwischengespeichert, so daß die Zeichenkette nicht innerhalb der ersten beiden Begrenzer wiederholt werden muß. Das Symbol `%` weist `ed` an, das letzte Ersetzungs-Muster zu verwenden, also `gold`.

Sind im Suchmuster in eckige Klammern enthalten, so sucht `ed` nach der ersten Stelle, an der eines der in eckigen Klammern eingeschlossenen Zeichen vorkommt und setzt dafür das Ersetzungs-Muster ein. Die eckigen Klammern können im Suchmuster an einer beliebigen Stelle stehen.

So wechselt `ed` im folgenden Beispiel die Zahlen `6`, `7`, `8` oder `9` bei ihrem ersten Auftreten in jeder Zeile gegebenenfalls gegen `4` aus:

```
1,$p<CR>
Monday          33,000
Tuesday         75,000
Wednesday       88,000
Thursday        62,000
1,$s/[6789]/4<CR>
Monday          33,000
Tuesday         45,000
Wednesday       48,000
Thursday        42,000
```

Im nächsten Beispiel werden die Zeichenketten Mr und Ms aus der Namensliste gestrichen:

```
1,$p<CR>
Mr Arthur Middleton
Mr Matt Lewis
Ms Anna Kelley
Ms M. L. Hodel
1,$s/M[rs] //<CR>
1,$p<CR>
Arthur Middleton
Matt Lewis
Anna Kelley
M. L. Hodel
```

Handelt es sich beim ersten Zeichen innerhalb der eckigen Klammern um ein ^ (Zirkumflex), so wird es von `ed` als Anweisung zum Suchen von Zeichen interpretiert, die NICHT innerhalb der eckigen Klammern stehen. An jeder anderen Position innerhalb der eckigen Klammern dagegen wird der Zirkumflex von `ed` als darstellbares Zeichen interpretiert.

```
1,$p<CR>
grade  A   Computer Science
grade  B   Robot Design
grade  A   Boolean Algebra
grade  D   Jogging
grade  C   Tennis
1,$s/grade [^AB]/grade A<CR>
1,$p<CR>
grade  A   Computer Science
grade  B   Robot Design
grade  A   Boolean Algebra
grade  A   Jogging
grade  A   Tennis
```

Achten Sie bei der Benutzung von Sonderzeichen in Suchmustern darauf, daß Sie ihren Aktionsradius möglichst eng eingrenzen. Wenn Sie z.B. im letzten Beispiel nur

```
1,$s/[^AB]/A<CR>
```

verwendet hätten, so hätten Sie den Kleinbuchstaben g in jeder Zeile gegen den Großbuchstaben A ausgetauscht.

Probieren Sie diese Sonderzeichen an eigenen Beispielen und in verschiedenen Kombinationen und beobachten Sie die Ergebnisse.

In Tabelle 6-7 sind die Sonderzeichen für Such- bzw. Ersetzungs-Muster nochmals zusammengefaßt.

Tabelle 6-7: Sonderzeichen

Zeichen	Funktion
.	Paßt in einem Muster zu jedem beliebigen Zeichen.
*	Paßt zu Null, einem oder mehr Auftreten des vorhergehenden Zeichens bzw. des in eckigen Klammern eingeschlossenen Ausdrucks.
.*	Paßt zu Null, einem oder mehr Auftreten eines beliebigen Zeichens.
^	Paßt zum Zeilenanfang.
$	Paßt zum Zeilenende.
\	Die Bedeutung des unmittelbar nachfolgenden Sonderzeichens wird aufgehoben.
&	Wird innerhalb des Ersetzungs-Musters durch die erste Zeichenkette ausgewechselt, die zum Suchmuster paßt.
%	Das letzte Ersetzungs-Muster wird erneut benutzt.
[...]	Paßt zum ersten Auftreten eines der in eckigen Klammern eingeschlossenen Zeichens.
[^...]	Paßt zu jedem Zeichen, das NICHT in den eckigen Klammern enthalten ist.

Übung 5

5-1. Legen Sie eine Datei mit dem folgenden Inhalt an.

```
A     Computer Science
D     Jogging
C     Tennis
```

Welches Ergebnis hat dann die folgende Kommandozeile:

```
1,$s/[^AB]/A/<CR>
```

Heben Sie das letzte Kommando wieder auf. Welche Sonderzeichen müssen Sie verwenden, wenn Sie nur C und D austauschen möchten? (Tip: Diese Buchstaben befinden sich am Zeilenanfang, also an der Position, die zu ^ paßt). Nur keine Scheu vor Experimenten!

5-2. Fügen Sie vor Zeile 2 die folgende Zeile ein:

```
These are not really my grades.
```

Geben Sie mit eckigen Klammern und dem Zeichen ^ ein Suchmuster ein, mit dem Sie die eingefügte Zeile ausfindig machen können. Zur Adressierung einer Zeile gibt es mehrere Möglichkeiten. Wählen Sie die für Sie einfachste und schnellste Möglichkeit.

5-3. Legen Sie eine Datei folgenden Inhalts an:

```
I love money
I need money
The IRS wants my money
```

Ändern Sie diese Zeilen mit einem einzigen Kommando in:

```
It's my money
It's my money
The IRS wants my money
```

Führen Sie mit zwei Kommandozeilen die folgenden Funktionen durch: Wechseln Sie das Wort `money` zunächst in der ersten Zeile gegen `gold` aus, dann in den letzten beiden Zeilen, ohne die Wörter `money` und `gold` selbst einzugeben.

5-4. Wie können Sie die Zeile

```
1020231020
```

in

```
10202031020
```

ändern, ohne die ursprünglichen Ziffern im Ersetzungs-Muster anzugeben?

5-5. Geben Sie eine neue Textzeile mit den folgenden Zeichen ein:

```
* . \ & % ^ *
```

Setzen Sie für jedes Zeichen einen anderen Buchstaben ein. Müssen Sie für jeden Ersetzungs-Vorgang einen Gegenschrägstrich benutzen?

Versetzen von Text

In den bisherigen Abschnitten dieses Kapitels haben Sie das Adressieren von Zeilen, das Erstellen und Löschen von Text sowie das Ersetzen von Zeichenketten kennengelernt. `ed` verfügt über eine weitere Gruppe von vielseitigen und wichtigen Kommandos, mit denen Sie Textzeilen versetzen, kopieren oder miteinander im Editor-Puffer verknüpfen können. Außerdem können Sie Text aus einer Datei, die nicht im Editor-Puffer vorhanden ist, einlesen oder Zeilen aus der Datei im Puffer in eine andere Datei innerhalb des aktuellen Dateiverzeichnisses kopieren. Zum Versetzen von Text gibt es folgende Kommandos:

`m`	Textzeilen versetzen
`t`	Textzeilen kopieren
`j`	Aufeinanderfolgende Textzeilen verknüpfen
`w`	Textzeilen in eine Datei schreiben
`r`	Inhalt einer Datei einlesen

Textzeilen versetzen

Mit dem Kommando `m` (für move) können Sie Textblöcke an eine andere Stelle der Datei versetzen. Die Kommandozeile hat folgendes allgemeines Format:

[*adresse1*,*adresse2*]m[*adresse3*]<CR>

Die Kommandozeile besteht im einzelnen aus den folgenden Komponenten:

adresse1,*adresse2*	Der Zeilenbereich, der versetzt werden soll. Bei nur einer zu vesetzenden Zeile wird nur *adresse1* angegeben. Ohne die Angabe einer Adresse wird die aktuelle Zeile vesetzt.
`m`	Das Versetzen-Kommando selbst.
adresse3	Unter dieser Zeile wird Text eingefügt.

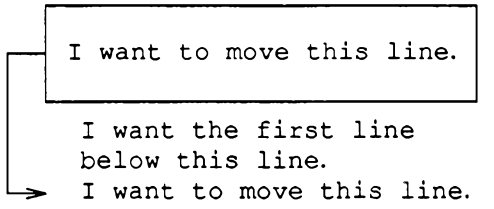
Legen Sie eine Datei mit den folgenden drei Textzeilen an und beobachten Sie das Ergebnis des Kommandos:

```
I want to move this line.
I want the first line
below this line.
```

Geben Sie das folgende Kommando ein:

```
1m3<CR>
```

ed versetzt Zeile 1 und fügt sie unter Zeile 3 ein.



Das folgende Beispiel zeigt, wie sich das Ergebnis auf Ihrem Bildschirm darstellt:

```
1,$p<CR>
I want to move this line.
I want the first line
below this line.
1m3<CR>
1,$p<CR>
I want the first line
below this line.
I want to move this line.
```

Wenn Sie einen ganzen Absatz versetzen möchten, so können Sie über *adresse1* und *adresse2* die erste und letzte Zeile des Absatzes angeben.

Im folgenden Beispiel wird ein Textblock (Zeile 8 bis 12) versetzt und unter Zeile 65 eingefügt. Beachten Sie das Kommando n, mit dem die Zeilennummern ausgegeben werden:

```
8,12n<CR>
8       This is line 8.
9       It is the beginning of a
10      very short paragraph.
11      This paragraph ends
12      on this line.
64,65n<CR>
64      Move the block of text
65      below this line.
8,12m65<CR>
59,65n<CR>
59      Move the block of text
60      below this line.
61      This is line 8.
62      It is the beginning of a
63      very short paragraph.
64      This paragraph ends
65      on this line.
```

Wie können Sie Zeilen oberhalb der ersten Zeile der Datei einfügen? Geben Sie das folgende Kommando ein:

```
3,4m0<CR>
```

Mit 0 als *adresse3* werden die Zeilen zum Dateianfang versetzt.

Textzeilen kopieren

Das Kommando `t` (für transfer) funktioniert wie das Kommando `m`, nur wird der Textblock an der ursprünglichen Stelle nicht gelöscht; stattdessen wird eine Kopie dieses Textblocks unter der angegebenen Textzeile eingefügt.

Das Kommando `t` hat dasselbe allgemeine Format wie das Kommando `m`.

[*adresse1*,*adresse2*]t[*adresse3*]<CR>

adresse1,*adresse2* Der zu kopierende Zeilenbereich. Wenn nur eine Zeile kopiert wird, muß nur *adresse1* angegeben werden. Ohne die Angabe einer Adresse wird die aktuelle Zeile kopiert.

`t`	Das Kopier-Kommando.
adresse3	Die Kopie der Textzeile wird unter der angegebenen Textzeile eingefügt.

Das nächste Beispiel zeigt, wie drei Textzeilen kopiert und unter der letzten Zeile eingefügt werden.

```
Safety procedures:

If there is a fire in the building:
Close the door of the room to seal off the fire

Break glass of nearest alarm.
Pull lever.
Locate and use fire extinguisher.

.
.
.

A chemical fire in the lab requires that you:

Break glass of nearest alarm
Pull lever
Locate and use fire extinguisher
```

Die nächste Abbildung zeigt die Kommandos sowie die von `ed` ermittelten und ausgegebenen Ergebnisse. Auch hier wird das Kommando `n` zur Ausgabe der Zeilennummern benutzt:

```
5,8n<CR>
5           Close the door of the room, to seal off the fire.
6           Break glass of nearest alarm.
7           Pull lever.
8           Locate and use fire extinguisher.
30n<CR>
30          A chemical fire in the lab requires that you:
6,8t30<CR>
30,$n<CR>
30          A chemical fire in the lab requires that you:
31          Break glass of nearest alarm
32          Pull lever
33          Locate and use fire extinguisher
6,8n<CR>
6           Break glass of nearest alarm
7           Pull lever
8           Locate and use fire extinguisher
```

Der Text in den Zeilen 6 bis 8 bleibt erhalten. Eine Kopie dieser drei Zeilen wird unter Zeile 50 eingefügt.

Üben Sie das Versetzen und Kopieren von Zeilen mit `m` und `t` an einer Ihrer Dateien .

Verknüpfen von aufeinanderfolgenden Zeilen

Mit dem Kommando `j` (für join) können Sie aufeinanderfolgende Zeilen verknüpfen. Das Kommando hat folgendes allgemeines Format:

[*adresse1*,*adresse2*]`j`<CR>

Diese Kommandozeile besteht aus den folgenden Einzelelementen:

adresse1,adresse2
: Der Bereich, in dem die zu verknüpfenden Zeilen enthalten sind. Ohne die Angabe einer Adresse wird die aktuelle Zeile mit der nachfolgenden verknüpft. Wird nur eine Adresse angegeben, hat das Kommando keinerlei Auswirkung.

`j`
: Das Verknüpfen-Kommando.

Das folgende Beispiel zeigt, wie zwei Zeilen miteinander verknüpft werden.

```
1,2p<CR>
Now is the time to join
the team.
1,2j<CR>
1p<CR>
Now is the time to join the team.
```

Beachten Sie, daß das letzte Wort der ersten Zeile (`join`) und das erste Wort der darauffolgenden Zeile (`the`) nicht durch ein Leerzeichen voneinander getrennt sind. Daher müssen Sie die beiden Wörter mit dem Kommando `s` durch ein Leerzeichen trennen.

Abspeichern von Textzeilen in einer Datei

Mit dem Kommando `w` (für write) können Sie Text aus dem Puffer in eine Datei "schreiben". Das Kommando hat folgendes allgemeines Format:

[*adresse1,adresse2*]`w`[*datei_name*]<CR>

adresse1,adresse2
: Der Zeilenbereich, der in eine andere Datei eingebracht werden soll. Ohne die Angabe von *adresse1* oder *adresse2* wird die gesamte Datei in eine neue Datei eingebracht.

`w`
: Das Schreiben-Kommando.

datei_name Der Name der neuen Datei, in der die Kopie des Textblocks enthalten ist.

Im nächsten Beispiel werden einige Zeilen eines Briefs in einer Datei namens memo abgespeichert.

```
1,$n<CR>
1               March 20, 1990
2       Dear Kelly,
3       There will be a meeting in the
4       green room at 4:30 P.M. today.
5       Refreshments will be served.
6       Please plan to attend.
7       Other divisions and locations
8       will also be represented.
9       We will discuss plans
10      for marketing several
11      new products during the
12      coming fiscal year,
13      as well as long range
14      research activities that
15      should yield profitable products
16      during the next decade.
3,5w  memo<CR>
91
```

Das Kommando w bringt eine Kopie der Zeilen 3 bis 5 in eine neue Datei namens memo ein. Als Reaktion darauf gibt ed die Anzahl von Zeichen in der neuen Datei aus.

Mögliche Probleme

Das Kommando w überschreibt eine bereits vorhandene Datei; der Inhalt der aktuellen Datei wird gelöscht und ohne entsprechende Warnung durch den neuen Text ersetzt. Angenommen, in unserem Beispiel wäre die Datei memo bereits vorhanden gewesen, als das Kommando zum Abspeichern der neuen Datei unter diesem Namen aufgerufen wurde; in diesem Fall wäre die ursprüngliche Datei gelöscht worden.

Im Abschnitt "Weitere nützliche Kommandos und Informationen" an einer späteren Stelle dieses Kapitels werden Sie die Durchführung von Shell-Kommandos unter `ed` kennenlernen. Dies gibt Ihnen die Möglichkeit, die Dateinamen im Dateiverzeichnis vor dem Abspeichern abzurufen und so das versehentliche Überschreiben einer Datei zu verhindern.

Ein weiteres Problem könnte darin bestehen, daß Sie die Datei `memo` nicht erweitern können. Wenn Sie also die Zeilen 13 bis 16 einfügen, werden die vorhandenen Zeilen (3 bis 5) gelöscht; die Datei enthält dann also nur die neuen Zeilen (13 bis 16). Dieses Problem läßt sich allerdings mit dem Kommando `W` umgehen, mit dem der Inhalt des aktuellen `ed`-Puffers am Ende der Datei eingefügt wird.

Einlesen des Inhalts einer Datei

Mit dem Kommando `r` (für read) können Sie Text aus einer Datei an den Pufferinhalt amfügen. Das Kommando hat folgendes allgemeines Format:

[*adresse1*]`r` *datei_name*<CR>

adresse1	Der Text wird unter Zeile *adresse1* eingefügt. Ohne die Angabe einer *adresse1* wird die Datei am Puffer-Ende eingefügt.
`r`	Das Kommando zum Einlesen der Datei.
datei_name	Der Name der Datei, die in den Editor-Puffer kopiert wird.

Im nächsten Beispiel wird die oben mit dem Kommando `w` abgespeicherte Datei abgeändert und durch neuen Text erweitert, der mit dem Kommando `r` eingelesen worden ist.

```
1,$n<CR>
1                March 20, 1990
2        Dear Michael,
3        Are you free later today?
4        Hope to see you there.
3r memo<CR>
91
3,$n<CR>
3        Are you free later today?
4        There will be a meeting in the
5        green room at 4:30 P.M. today.
6        Refreshments will be served.
7        Hope to see you there.
```

Als Ergebnis gibt ed die Anzahl der Zeichen in der Datei aus, die in den Puffer eingefügt worden ist (in diesem Beispiel hatte sie den Namen memo).

Neue oder geänderte Textzeilen sollten Sie zur Überprüfung in jedem Fall auf dem Bildschirm abrufen.

Tabelle 6-8 enthält einen Überblick über die ed-Kommandos zum Versetzen von Text.

Tabelle 6-8: Die ed-Kommandos zum Versetzen von Text

Kommando	Funktion
m	Textzeilen versetzen
t	Textzeilen kopieren
j	Aufeinanderfolgende Zeilen verknüpfen
w	Text in eine neue Datei einbringen
W	Text an eine vorhandene Datei anfügen
r	Text aus einer anderen Datei einlesen

Übung 6

6-1. Zum Kopieren von Textzeilen im Puffer gibt es zwei Möglichkeiten: Sie können entweder das Kopieren-Kommando aufrufen oder den Text zunächst mit dem Schreiben-Kommando in eine Datei einbringen, die Sie dann mit dem Lesen-Kommando in den Puffer einlesen.

Die zweite Möglichkeit (zuerst den Text in eine Datei einbringen und die Datei in den Puffer einlesen) ist zeitaufwendiger als die erste. Können Sie sich ein Beispiel vorstellen, bei dem die zweite Möglichkeit trotzdem vorzuziehen wäre?

Mit welchen Kommandos können Sie die Zeilen 10 bis 17 der Datei `exer` bei Zeile 7 der Datei `exer6` einfügen?

6-2. Die Zeilen 33 bis 46 sollen beispielsweise versetzt und unter Zeile 3 statt unter Zeile 32 eingefügt werden. Welche Kommandos sind dazu notwendig?

6-3. Angenommen, Sie befinden sich in Zeile 10 einer Datei und möchten die Zeilen 13 und 14 miteinander verknüpfen. Welche Kommandos sind hierzu notwendig?

Weitere nützliche Kommandos und Informationen

Im folgenden geht es um vier weitere Kommandos und eine spezielle die Ihnen während einer Editor-Sitzung zur Verfügung stehen.

`h,H`	Zugriff auf die Hilfe-Kommandos, durch die Fehlermeldungen ausgegeben werden.
`l`	Anzeige von Zeichen, die normalerweise nicht angezeigt werden.
`f`	Anzeige des aktuellen Dateinamens
`!`	Vorübergehendes Beenden von `ed`, um ein Shell-Kommando auszuführen.
`ed.hup`	Abspeichern des Pufferinhalts in einer speziellen Datei namens `ed.hup` für den Fall, daß `ed` oder der Rechner einmal abstürzen sollte.

Hilfe-Kommandos

Sie haben bestimmt bereits bemerkt, daß `ed` während einer Editor-Sitzung auf einige Ihrer Kommandos mit einem Fragezeichen (`?`) reagiert. Bei dem Fragezeichen handelt es sich um eine Meldung, die `ed` bei einem Fehler ausgibt. Mit den Hilfe-Kommandos erreichen Sie, daß Ihnen in einer kurzen Meldung die Ursache für den letzten Fehler mitgeteilt wird.

Es gibt zwei Hilfe-Kommandos:

`h` Es wird eine kurze Fehlermeldung angezeigt, in der die Ursache für das letzte Fragezeichen (`?`) erläutert wird.

`H` Der Hilfe-Modus von `ed` wird aufgerufen, so daß gleichzeitig mit jedem Fragezeichen (`?`) eine kurze Fehlermeldung angezeigt wird (dies können Sie durch das erneute Aufrufen des Kommandos `H` ausschalten).

Wie bereits erwähnt, zeigt ed jedesmal ein Fragezeichen (?) an, wenn Sie versuchen, den Editor zu verlassen, ohne zuvor die Änderungen am Puffer in eine Datei einzubringen. Versuchen Sie jetzt, den Editor auf diese Weise zu verlassen. Sobald das Fragezeichen (?) erscheint, geben Sie das Kommando h ein:

```
q<CR>
?
h<CR>
warning: expecting 'w'
```

Das Fragezeichen (?) wird auch dann angezeigt, wenn Sie in der ed-Kommandozeile einen neuen Dateinamen angeben. Rufen Sie ed mit einem neuen Dateinamen auf. Sobald auf dem Bildschirm das Fragezeichen (?) erscheint, geben Sie h ein, um Informationen über die Bedeutung der Fehlermeldung zu erhalten.

```
ed newfile<CR>
?newfile
h<CR>
cannot open input file
```

Diese Meldung kann zwei verschiedene Bedeutungen haben: Entweder, daß es keine Datei mit dem Namen newfile gibt, oder daß es eine Datei gibt, die von ed jedoch nicht gelesen werden darf.

Wie bereits erwähnt, können Sie mit dem Kommando H den Hilfe-Modus einschalten, in dem nach nach jedem Fragezeichen (?) eine kurze Erläuterung der Fehlerursache ausgegeben wird. Den Hilfe-Modus können Sie durch erneute Eingabe von H wieder ausschalten. Im nächsten Beispiel wird mit dem Kommando H der Hilfe-Modus eingeschaltet. Außerdem werden einige Fehlermeldungen gezeigt, die bei gängigen Fehlern erscheinen:

```
$ ed newfile<CR>
H<CR>
cannot open input file
/hello<CR>
?
search string not found
1,22p<CR>
?
line out of range
a<CR>
I am appending this line to the buffer.<CR>
.<CR>
s/$ tea party<CR>
?
illegal or missing delimiter
,$s/$/ tea party<CR>
?
unknown kommando
H<CR>
q<CR>
?
h<CR>
warning: expecting 'w'
```

Im folgenden sind die Fehlermeldungen aufgeführt, die während einer Editor-Sitzung bei den gängigsten Fehlern ausgegeben werden:

`search string not found`
: ed kann das Suchmuster `hello` nicht finden.

`line out of range`
: ed kann keine Zeilen ausgeben, da entweder der Puffer leer ist oder die angegebene Zeile nicht im Puffer vorhanden ist.

An den Pufferinhalt wird eine Textzeile angefügt, um Ihnen einige Fehlermeldungen vorzuführen, die beim Kommando `s` auftreten können.

`illegal or missing delimiter`
: Der Begrenzer zwischen dem ursprünglichen (zu ersetzenden) und dem neuen Text fehlt.

`unknown command`
: Vor dem Komma fehlt *adresse1*; , `$` kann von `ed` nicht interpretiert werden.

Nach dem Ausschalten des Hilfe-Modus muß zur Ermittlung der Ursache des letzten `?` das Kommando `h` aufgerufen werden. In der Einarbeitungsphase sollten Sie `ed` bei eingeschaltetem Hilfe-Modus benutzen; hierzu rufen Sie das Kommando `H` auf. Mit wachsendem Kenntnisstand werden Sie die Fehlermeldungen jedoch immer seltener benötigen und sie bei Bedarf mit dem Kommando `h` abrufen können.

Anzeige von nicht darstellbaren Zeichen

Steuerzeichen werden im Normalfall nicht auf dem Bildschirm angezeigt. Ein Beispiel für ein Steuerzeichen ist <^g> (control-g), bei der am Terminal ein akustisches Signal ertönt.

Bei der Eingabe eines Tabulatorzeichens werden auf dem Bildschirm normalerweise bis zu acht Leerzeichen angezeigt, um den Platz bis zum nächsten Tabulatorstopp aufzufüllen (die Tabulatorzeichen können auch in mehr oder weniger als acht Leerzeichen umgesetzt werden; siehe auch Kapitel 9, "Die Shell").

Wenn Sie sich über die Tabulator- oder sonstigen Steuerzeichen informieren möchten, die in Ihrem Text enthalten sind, so rufen Sie das Kommando `l` (für list) auf. Dieses Kommando hat dasselbe allgemeine Format wie die Kommandos `n` und `p`.

[*adresse1*,*adresse2*]`l`<CR>

Die Kommandozeile dieses Kommandos besteht aus den folgenden Einzelelementen:

adresse1,adresse2	Der darzustellende Zeilenbereich. Ohne die Angabe einer Adresse wird die aktuelle Zeile angezeigt. Wenn nur *adresse1* angegeben wird, so wird nur diese eine Zeile angezeigt.
`l`	Das Kommando, mit dem außer dem Text auch die nicht darstellbaren Zeichen angezeigt werden.

Das Kommando `l` zeigt die Tabulatorzeichen zusammen mit einem Größer-als-Zeichen (>) an. Zur Eingabe von Steuerzeichen halten Sie die CONTROL-(CTRL)-Taste gedrückt und betätigen die entsprechende Buchstabentaste. Ein akustisches Signal können Sie mit ^g (Control-g) erzeugen. Dieses Steuerzeichen wird im Format \007 (also als Oktalwert) angezeigt.

Geben Sie zwei Textzeilen ein, die das Steuerzeichen <^g> (Control-g) sowie ein Tabulatorzeichen enthalten. Rufen Sie dann die Textzeilen mit Hilfe des Kommandos `l` auf Ihrem Terminal ab.

```
a<CR>
Add a <^g> (Control-g) to this line.<CR>
Add a <tab> (tab) to this line.<CR>
.<CR>
1,2l<CR>
Add a \007 (Control-g) to this line.
Add a >  (tab) to this line.
```

Ertönte bei der Eingabe von <^g> ein akustisches Signal?

Der aktuelle Dateiname

Bei einer längeren Editor-Sitzung kann es vorkommen, daß Sie den Dateinamen vergessen. In diesem Fall rufen Sie das Kommando `f` auf, mit dem der Name der Datei ausgegeben wird, die aktuell im Puffer enthalten ist. Vielleicht möchten Sie jedoch den Inhalt der Datei, wie er ursprünglich in den Editor-Puffer geladen worden ist, beibehalten und den Puffer-Inhalt in eine neue Datei einbringen. Hier besteht bei einer langen Editor-Sitzung das Risiko, daß Sie vergessen, dies zu tun, und die ursprüngliche Datei mit der bekannten Kommandofolge `w` und `q` überschreiben. Um das zu verhindern, teilen Sie dem Editor während einer Editor-Sitzung mit, daß dem Puffer-Inhalt ein neuer Dateiname

zugeordnet werden soll. Hierzu rufen Sie das Kommando `f` mit einem neuen Dateinamen auf.

Der aktuelle Dateiname wird ausgegeben, wenn Sie in der Kommandozeile ausschließlich das Kommando `f` angeben:

```
f<CR>
```

Um die Funktion dieses Kommandos kennenzulernen, rufen Sie `ed` mit einer Datei auf. Wenn Ihre Datei beispielsweise den Namen `oldfile` hat, gibt `ed` folgendes aus:

```
ed oldfile<CR>
323
f<CR>
oldfile
```

Wenn Sie dem Inhalt des Editor-Puffers einen neuen Dateinamen zuordnen möchten, so rufen Sie ein Kommando in folgendem Format auf:

```
f datei_neu<CR>
```

Wenn Sie beim Kommando `w` keinen Dateinamen angeben, setzt `ed` automatisch den Dateinamen ein, den Sie zu Beginn der Editor-Sitzung angegeben haben; der Inhalt des Editor- Puffers wird also in diese Datei eingebracht. Wenn Sie den Inhalt der ursprünglichen Datei nicht überschreiben möchten, so geben Sie entweder beim Kommando `w` einen neuen Dateinamen ein, oder Sie ändern den aktuellen Dateinamen mit dem Kommando `f` und mit dem neuen Dateinamen. Da Sie `f` jederzeit während einer Editor-Sitzung aufrufen können, können Sie den Dateinamen sofort ändern. Dann können Sie die Editor- Sitzung fortsetzen, ohne das versehentliche Überschreiben der ursprünglichen Datei zu riskieren.

Das nächste Beispiel zeigt die Kommandos, mit denen der Editor mit der Datei `oldfile` aufgerufen und dann der Dateiname in `newfile` geändert werden kann. In den Puffer wird eine Textzeile eingefügt; dann werden die Kommandos zum Abspeichern und zum Beenden des Editors aufgerufen:

```
ed oldfile<CR>
323
f<CR>
oldfile
f newfile<CR>
newfile
a<CR>
Add a line of text.<CR>
.<CR>
w<CR>
343
q<CR>
```

Nach Ihrer Rückkehr zur Shell können Sie überprüfen, ob die neue Datei `newfile` tatsächlich im Dateiverzeichnis vorhanden ist. `newfile` sollte eine Kopie von `oldfile`, erweitert um die neue Textzeile, enthalten.

Vorübergehendes Aufrufen einer Sub-Shell

Wie können Sie das Risiko ausschalten, daß Sie eine vorhandene Datei überschreiben, wenn Sie den Inhalt des Puffers in eine neue Datei einbringen? Sie rufen eine Sub-Shell auf und rufen eine Dateiliste auf dem Bildschirm ab. Mit dem Ausrufezeichen (`!`) können Sie vorübergehend eine Sub-Shell aufrufen, ein Shell-Kommando ausführen und dann die Editor-Sitzung wieder bei der aktuellen Zeile fortsetzen.

Hierzu geben Sie eine Kommandozeile in folgendem Format ein:

```
!shell_kommando_zeile<CR>
Antwort der Shell
!
```

Wenn Sie `!` ganz am Anfang einer Zeile eingeben, muß unmittelbar darauf (in derselben Zeile) das Shell-Kommando folgen. Das jeweils aufgerufene Programm gibt während der Ausführung das Ergebnis aus. Nach beendeter Kommandoausführung wird eine Zeile angezeigt, in der ausschließlich `!` enthalten ist. Dies zeigt Ihnen, daß Sie sich wieder in der aktuellen Zeile Ihrer gerade editierten Datei befinden.

Angenommen, Sie möchten vorübergehend eine Sub-Shell aufrufen, um das aktuelle Datum zu ermitteln; hierzu geben Sie ! sowie das Shell-Kommando `date` ein:

```
p<CR>
This is the current line
! date<CR>
Sun  Apr 1  14:24:22  EST  1990
!
p<CR>
This is the current line.
```

Auf dem Bildschirm wird zunächst die aktuelle Zeile angezeigt, gefolgt von dem Kommando zum vorübergehenden Verlassen des Editors und zur Anzeige des Datums. Anschließend befinden Sie sich wieder in der aktuellen Zeile der editierten Datei.

Informationen darüber, wie Sie in einer Shell- Kommandozeile zwei oder mehr Kommandos durchführen können, finden Sie in Kapitel 9, "Die Shell", im Abschnitt "Sonderzeichen", im Zusammenhang mit der Diskussion des Zeichens `;`.

Nach einem Absturz

Was geschieht, wenn Sie unter `ed` einen Text erstellen und beispielsweise das Verbindungskabel zwischen Ihrem Terminal und dem Rechner oder der Netzstecker Ihres Terminals abgezogen wird? Kommt es aus irgendeinem Grund zu einem Systemabsturz, versucht das UNIX-System, den Inhalt des Editor-Puffers in eine spezielle Datei namens `ed.hup` zu retten. Zur Wiederherstellung Ihres Texts anhand dieser Datei haben Sie zwei Möglichkeiten. Zum einen können Sie den Inhalt von `ed.hup` mit einem Shell-Kommando unter einem anderen Dateinamen abspeichern (z.B. dem Namen, den die Datei vor dem Absturz hatte). Die zweite Möglichkeit besteht darin, `ed` aufzurufen und den Inhalt des Puffers mit dem Kommando `f` umzubenennen. Im folgenden Beispiel wurde die zweite Möglichkeit benutzt:

```
ed ed.hup<CR>
928
f myfile<CR>
myfile
```

Nachdem Sie den Inhalt des Puffers nach dieser Methode wiederhergestellt haben, sollten Sie die Datei `ed.hup` wieder löschen.

Zusammenfassung

Sie haben in diesem Kapitel eine Vielzahl von Kommandos kennengelernt, die Sie beim Arbeiten mit `ed` benötigen. Informationen über diejenigen Kommandos, die hier nicht besprochen wurden (z.B. `G`, `P` und `Q`) sowie über die speziellen Zeichen für Suchmuster (z.B. `(`, `)`, `{` und `}`) finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `ed`(1). Arbeiten Sie ein wenig mit diesem Kommandos, um ihre Funktionsweise kennenzulernen.

Tabelle 6-9 enthält einen Überblick über die Kommandos, die in diesem Abschnitt vorgestellt wurden.

Tabelle 6-9: Weitere nützliche Kommandos

Kommando	Funktion
`h`	Für die vorausgegangene Fehleranzeige in Form eines Fragezeichens (?) wird eine Erläuterung in Form einer kurzen Fehlermeldung angezeigt.
`H`	Der Hilfe-Modus wird eingeschaltet. Bei jeder Fehleranzeige in Form eines Fragezeichens (?) wird eine Fehlermeldung ausgegeben. Durch die erneute Eingabe von `H` wird der Hilfe-Modus wieder ausgeschaltet.
`l`	Die nicht darstellbaren Zeichen, die im Text enthalten sind, werden angezeigt.
`f`	Der aktuelle Dateiname wird angezeigt.
`f` *datei_neu*	Der aktuelle Dateiname, der dem Editor-Puffer zugeordnet ist, wird in *datei_neu* geändert.
!kommando	Vorübergehendes Aufrufen einer Sub-Shell, um das Shell-Kommando *kommando* durchzuführen.
`ed.hup`	Der Puffer wird bei einem Systemabsturz o.ä. in einer speziellen Datei namens `ed.hup` abgespeichert.

Übung 7

7-1. Legen Sie eine neue Datei namens `newfile1` an. Rufen Sie `ed` auf und ändern Sie den Namen der Datei in `current1`. Geben Sie dann einen kurzen Text ein, speichern Sie ihn in einer Datei ab, und verlassen Sie `ed`. Rufen Sie das Kommando `ls` auf, um sicherzustellen, daß in Ihrem Dateiverzeichnis noch keine Datei namens `newfile1` vorhanden ist.

7-2. Legen Sie eine Datei namens `file1` an. Fügen Sie am Ende der Datei einige Textzeilen ein (Kommando `a`). Verlassen Sie den Anfüge-Modus; speichern Sie die Datei jedoch nicht ab. Schalten Sie Ihr Terminal aus. Schalten Sie Ihr Terminal dann wieder ein und melden Sie sich erneut an. Rufen Sie von der Shell aus das Kommando `ls` auf. Gibt es die neue Datei namens `ed.hup`? Bearbeiten Sie die Datei `ed.hup` unter `ed`. Wie können Sie den aktuellen Dateinamen in `file1` ändern? Rufen Sie den Inhalt der Datei auf dem Bildschirm ab. Sind in der Datei dieselben Zeilen wie vor dem Ausschalten des Terminals enthalten?

7-3. Rufen Sie von `ed` aus vorübergehend eine Sub-Shell auf, und schicken Sie über die elektronische Post eine Nachricht an sich selbst.

Auflösung der Übungen

Übung 1

1-1.

```
$ ed junk<CR>
?junk
a<CR>
Hello world.<CR>
.<CR>
w<CR>
13
q<CR>
$
```

```
$ ed stuff<CR>
?stuff
a<CR>
Goodby world.<CR>
.<CR>
w<CR>
15
q<CR>
$
```

1-2.

```
$ ed junk<CR>
13
1,$p<CR>
Hello world.
q<CR>
$
```

Das System gab keine Warnung in Form eines Fragezeichen aus, da Sie am Inhalt des Puffers nichts geändert haben.

1-3.

```
$ ed junk<CR>
13
a<CR>
Wendy's horse came through the window.<CR>
.<CR>
1,$p<CR>
Hello world.
Wendy's horse came through the window.
q<CR>
?
w stuff<CR>
52
q<CR>
$
```

Übung 2

2-1.

```
$ ed towns<CR>
?towns
a<CR>
My kind of town is<CR>
Chicago<CR>
Like being no where at all in<CR>
Toledo<CR>
I lost those little town blues in<CR>
New York<CR>
I lost my heart in<CR>
San Francisco<CR>
I lost $$ in<CR>
Las Vegas<CR>
.<CR>
w<CR>
163
```

2-2.

```
3<CR>
Like being no where at all in
```

2-3.

```
-2,+3p<CR>
My kind of town is
Chicago
Like being no where at all in
Toledo
I lost those little town blues in
New York
```

2-4.

```
.=<CR>
6
6<CR>
New York
```

2-5.

```
$<CR>
Las Vegas
```

2-6.

```
?town<CR>
I lost those little town blues in
?<CR>
My kind of town is
```

2-7.

```
g/in<CR>
My kind of town is
Like being no where at all in
I lost those little town blues in
I lost my heart in
I lost $$ in

v/in<CR>
Chicago
Toledo
New York
San Francisco
Las Vegas
```

Übung 3

3-1.

```
$ ed ex3<CR>
?ex3
i<CR>
?
q<CR>
```

Das Fragezeichen (`?`) hinter dem `i` bedeutet, daß die Kommandozeile einen Fehler enthält. Es gibt keine aktuelle Zeile, vor der Sie Text einfügen könnten.

3-2.

```
$ ed towns<CR>
163
.n<CR>
10      Las Vegas
3i<CR>
Illinois<CR>
.<CR>
.i<CR>
or<CR>
Naperville<CR>
.<CR>
$i<CR>
hotels in<CR>
.<CR>
1,$n<CR>
1       my kind of town is
2       Chicago
3       or
4       Naperville
5       Illinois
6       Like being no where at all in
7       Toledo
8       I lost those little town blues in
9       New York
10      I lost my heart in
11      San Francisco
12      I lost $$ in
13      hotels in
14      Las Vegas
```

3-3.

```
1,5n<CR>
1       My kind of town is
2       Chicago
3       or
4       Naperville
5       Illinois
2,5c<CR>
London<CR>
.<CR>
1,3n<CR>
1       My kind of town is
2       London
3       Like being nowhere at all in
```

3-4.

```
.<CR>
Like being nowhere at all in
/Tol<CR>
Toledo
c<CR>
Peoria<CR>
.<CR>
.<CR>
Peoria
```

3-5.

```
.<CR>
/New Y/c<CR>
Iron City<CR>
.<CR>
.<CR>
Iron City
```

Die Suchfolge muß nicht unbedingt das vollständige Wort oder die vollständige Zeile sein - sie muß nur eindeutig sein.

Übung 4

4-1.

```
v/little town/s/town/city<CR>
My kind of city is
London
Like being no where at all in
Peoria
Iron City
I lost my heart in
San Francisco
I lost $$ in
hotels in
Las Vegas
```

Die Zeile

```
I lost those little town blues in
```

ist nicht ausgegeben worden, da sie NICHT durch das Kommando v adressiert worden ist.

4-2.

```
.<CR>
Las Vegas
s?Las Vegas?Toledo<CR>
Toledo
```

4-3.

```
?lost?s??found<CR>
I found $$ in
```

4-4.

```
/no?s??NO<CR>
?
/no/s//NO<CR>
Like being NO where at all in
```

In ein und derselben Kommandozeile können Sie nicht mit verschiedenen Begrenzern wie dem Schrägstrich (/) oder dem Fragezeichen (?) arbeiten.

Das Ersetzen-Kommando in Zeile 9 führte zu folgender Ausgabe:

```
I found $$ inBig $
```

Das Problem lag darin, daß das Symbol $ unter ed ein Sonderzeichen ist.

Übung 5

5-1.

```
$ ed file1<CR>
?file1
a<CR>
A  Computer Science<CR>
D  Jogging<CR>
C  Tennis<CR>
.<CR>
1,$s/[^AB]/A/<CR>
1,$p<CR>
AA  Computer Science
A  Jogging
A  Tennis
u<CR>
```

```
1,$s/^[^AB]/A/<CR>
1,$p<CR>
A  Computer Science
A  Jogging
A  Tennis
```

5-2.

```
2i<CR>
These are not really my grades.<CR>
.<CR>
1,$p<CR>
A  Computer Science
These are not really my grades.
A  Tennis
A  Jogging
/^[^A]<CR>
These are not really my grades
?^[T]<CR>
These are not really my grades
```

5-3.

```
1,$p<CR>
I love money
I need money
The IRS wants my money
g/^I/s/I.*m /It's m<CR>
It's my money
It's my money
```

```
1s/money/gold<CR>
It's my gold
2,$s//%<CR>
The IRS wants my gold
```

5-4.

```
s/10202/&0<CR>
10202031020
```

5-5.

```
a<CR>
* . \ & % ^ *<CR>
.<CR>
s/*/a<CR>
a . \ & % ^ *
s/*/b<CR>
a . \ & % ^ b
```

Da dem * keine Zeichen vorangestellt waren, wurde es selbst als Suchmuster benutzt und ausgewechselt.

```
s/ \./c<CR>
a c \ & % ^ b
s/ \\/d<CR>
a c d & % ^ b
s/&/e<CR>
a c d e % ^ b
s/%/f<CR>
a c d e f ^ b
```

Die Zeichen & und % sind nur im Ersetzungs-Muster Sonderzeichen.

```
s/ \^/g<CR>
a c d e f g b
```

Übung 6

6-1. Wenn Sie Textzeilen mehrmals eingeben möchten, so ist es in vielen Fällen einfacher, diese Zeilen in eine Datei einzubringen und diese Datei dann an den entsprechenden Stellen in den Text einzulesen.

Wenn Sie die Zeilen in eine andere Datei kopieren möchten, so müssen Sie sie zunächst in eine Datei einbringen und diese Datei dann in den Puffer einlesen, in dem bereits die andere Datei enthalten ist.

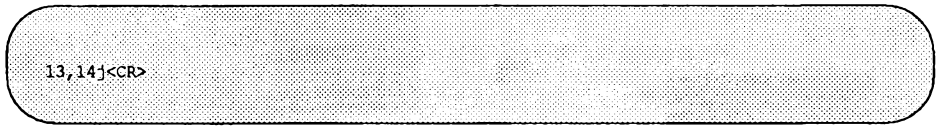
```
ed exer<CR>
725
10,17 w temp<CR>
210
q<CR>
ed exer6<CR>
305
7r temp<CR>
210
```

In diesem Beispiel heißt die temporäre Datei `temp`.

6-2.

```
33,46m3<CR>
```

6-3



Übung 7

7-1.

```
$ ed newfile1<CR>
?newfile1
f current1<CR>
current1
a<CR>
This is a line of text<CR>
Will it go into newfile1<CR>
or into current1<CR>
.<CR>
w<CR>
66
q<CR>
$ ls<CR>
bin
current1
```

7-2.

```
ed file1<CR>
?file1
a<CR>
I am adding text to this file.<CR>
Will it show up in ed.hup?<CR>
.<CR>
```

Schalten Sie Ihr Terminal aus.

Melden Sie sich erneut an.

```
ed ed.hup<CR>
58
f file1<CR>
file1
1,$p<CR>
I am adding text to this file.
Will it show up in ed.hup?
```

7-3.

```
$ ed file1<CR>
58
! mail mylogin<CR>
You will get mail when<CR>
you are done editing!<CR>
.<CR>
!
```

7 Der Bildschirmeditor vi

Einführung 7-1
Benutzerhinweise 7-3

Einstieg 7-4
Konfiguration des Terminals 7-4
Einrichten der Benutzerumgebung 7-5
Einstellung des automatischen Wagenrücklaufs 7-5

Anlegen einer Datei 7-7
Betriebsmodi 7-8
Erstellen von Text im Anfügen-Modus 7-8
Beenden des Anfügen-Modus 7-9

Editieren von Text im Kommandomodus 7-10
Cursorsteuerung 7-10
Cursor nach links oder rechts bewegen 7-12
Löschen von Text 7-15
Eintragen von neuem Text 7-16

Beenden von `vi` 7-18

Übung 1 7-21

Positionierung des Cursors auf dem Bildschirm 7-22
Cursor zu einem Zeichen bewegen 7-23
- Cursor zum Zeilenanfang oder -ende bewegen 7-23
- Zeichen in einer Zeile suchen 7-25

Zeilenweises Positionieren 7-26
- Das Positionierungs-Kommando - Minus-Zeichen (-) 7-26
- Das Positionierungs-Kommando: Plus-Zeichen (+) 7-27

Wortweises Positionieren 7-27
Satzweises Positionieren 7-31
Absatzweises Positionieren 7-32
Bildschirmbezogenes Positionieren 7-34

Cursor an eine aktuell nicht angezeigte Stelle bewegen 7-40
Text durchblättern 7-40
- Das Kommando Control-f 7-40
- Das Kommando Control-d 7-41
- Das Kommando Control-b 7-41
- Das Kommando Control-u 7-43

In eine bestimmte Zeile springen 7-43
Zeilennummern 7-43
Suchen von Zeichenmustern: Die Kommandos / und ? 7-45

Übung 2 7-51

Erstellen von Text 7-53
Anfügen von Text 7-53
Einfügen von Text 7-53
Einfügen einer Leerzeile 7-55

Übung 3 7-58

Löschen von Text 7-59
Löschen von Text, der im Eingabemodus eingegeben worden ist 7-59
Aufheben des letzten Kommandos 7-61
Löschen von Text im Kommandomodus 7-61
- Löschen von Wörtern 7-62
- Löschen von Absätzen 7-63
- Löschen von Zeilen 7-63
- Löschen von Text hinter der aktuellen Cursor-Position 7-64

Übung 4 7-66

Ändern von Text 7-67
Ersetzen von Text (Kommando r) 7-67
Ersetzen von Text (Kommando s) 7-68
Überschreiben von Text (Kommando c) 7-69

Text ausschneiden und einfügen 7-74
Versetzen von Text 7-74
Korrigieren von vertauschten Zeichen 7-75
Kopieren von Text 7-75
Text über Register kopieren bzw. versetzen 7-77

Übung 5 7-79

Spezielle Kommandos 7-80
Wiederholen des letzten Kommandos 7-80
Verknüpfen von zwei Zeilen 7-81
Bildschirm löschen und wieder aufbauen 7-81
Groß- gegen Kleinbuchstaben austauschen und umgekehrt 7-82

Arbeiten mit Zeileneditor-Kommandos unter `vi` 7-83
Vorübergehendes Aufrufen einer Shell: Die Kommandos `:sh` und `:!` 7-83
Text in einer neuen Datei abspeichern: Das Kommando `:w` 7-84
In eine bestimmte Zeile springen 7-85
Zeilen bis zum Puffer-Ende löschen 7-85
Datei in den Puffer einlesen 7-86
Globale Änderungen 7-86

Beenden von `vi` 7-89

Spezielle `vi`-Optionen 7-92
Wiederherstellen einer Datei nach einem Systemabsturz 7-92
Editieren von mehreren Dateien 7-92
Abrufen des Dateiinhalts 7-93

Übung 6 7-95

Auflösung der Übungen 7-96
Übung 1 7-96
Übung 2 7-97
Übung 3 7-99
Übung 4 7-100
Übung 5 7-101
Übung 6 7-102

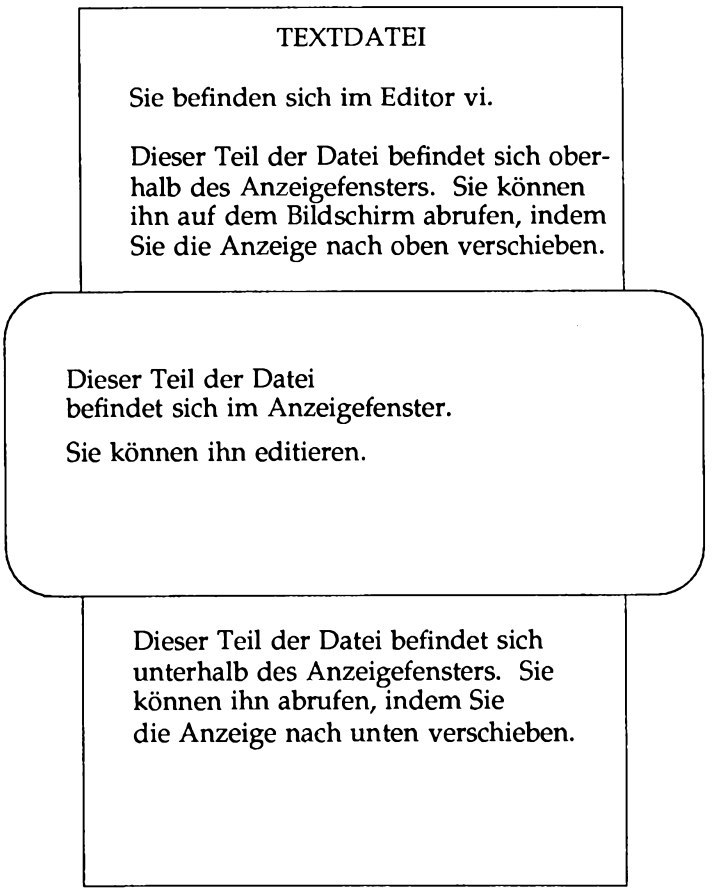
Einführung

Dieses Kapitel enthält eine Einführung in den Bildschirmeditor `vi` (für visual). Der Editor `vi` ist ein leistungsfähiges, ausgereiftes Werkzeug zum Erstellen und Bearbeiten von Dateien. Er wurde für Terminals konzipiert, auf denen der Inhalt einer Datei auf dem Bildschirm angezeigt werden kann (also für Datensichtgeräte). Unter `vi` können Sie mit wenigen einfachen Kommandos Änderungen am Text vornehmen, die sofort am Bildschirm angezeigt werden.

Der Editor `vi` kann eine bis mehrere Zeilen anzeigen. Sie können in einer `vi`-Sitzung den Cursor an eine beliebige Stelle auf dem Bildschirm bzw. der Datei bewegen (hierzu geben Sie z.B. den Beginn oder das Ende eines Worts, einer Zeile, eines Absatzes oder der Datei an) und von dieser Stelle aus^ Text erstellen, ändern oder löschen. Mit weiteren Editier-Kommandos wie z.B. den leistungsfähigen "globalen" Kommandos können Sie eine bestimmte Zeichenkette mit einem einzigen Kommando an mehreren Stellen in der Datei auswechseln. Um an eine andere Stelle der Datei zu gelangen, verschieben Sie den Text in Vor- oder Rückwärtsrichtung; dadurch werden die Zeilen angezeigt, die sich oberhalb oder unterhalb des aktuellen Fensters befinden (siehe Bild 7-1).

Hinweis: Die Möglichkeit zum Verschieben der Bildschirmanzeige ist nicht bei allen Terminals vorhanden; ob Sie diese Funktion nutzen können, hängt vom Typ Ihres Terminals ab.

Bild 7-1: Anzeige einer Datei in einem `vi`-Fenster



Es gibt über 100 `vi`-Kommandos. In diesem Kapitel können nur die grundlegenden Kommandos besprochen werden, die Ihnen die einfache und effektive Nutzung von `vi` ermöglichen. Im einzelnen werden die folgenden Funktionen besprochen:

- Änderung Ihrer Shell-Umgebung zur Konfiguration Ihres Terminals.
- Einschalten des automatischen Zeilenumbruchs.
- Aufrufen von `vi`, Erstellung von Text, Korrigieren von Fehlern, Abspeichern des Texts in einer Datei und Beenden des Editors.
- Versetzen von Text innerhalb einer Datei.
- Text elektronisch ausschneiden und einfügen.
- Benutzung von speziellen Kommandos und Abkürzungstasten.
- Arbeiten mit den Zeileneditor-Kommandos von `vi`.
- Vorübergehendes Aufrufen einer Shell, um Shell-Kommandos durchzuführen.
- Wiederherstellung einer Datei, die bei einer Unterbrechung der Editor-Sitzung. verlorengegangen ist.
- Editieren von mehreren Dateien in ein und derselben Sitzung.

Benutzerhinweise

Die Beschreibung der Kommandos in diesem Kapitel hält sich an das Beschreibungsformat, das im Vorwort vorgestellt worden ist. In den abgebildeten Bildschirmen in diesem Kapitel wird die aktuelle Cursor-Position durch Pfeile angezeigt.

Die Kommandos, die in jedem Abschnitt besprochen werden, sind am Ende des jeweiligen Abschnitts nochmals kurz zusammengefaßt. Eine vollständige, nach thematischen Gesichtspunkten geordnete Übersicht über die `vi`-Kommandos finden Sie in Anhang E. Am Ende einiger Abschnitte wurden Übungen eingefügt, damit Sie sich etwas Praxis aneignen können. Die Auflösung sämtlicher Übungen finden Sie am Ende dieses Kapitels. `vi` lernen Sie am besten kennen, indem Sie beim Durcharbeiten der Einführung die Beispiele und Übungen nachvollziehen. Melden Sie sich vor dem Durcharbeiten dieses Kapitels bei UNIX an.

Einstieg

In diesem Abschnitt geht es um die Konfiguration des Terminals als wesentliche Voraussetzung für die Arbeit mit `vi` sowie die optionale Einstellung des Zeilenumbruchs bzw. automatischen Wagenrücklaufs.

Konfiguration des Terminals

Bevor Sie `vi` das erste Mal aufrufen, müssen Sie Ihr Terminal geeignet konfigurieren. Bei der Konfiguration geben Sie lediglich den Typ des benutzten Terminals an; dies ist notwendig, da das Programm `vi` sich auf unterschiedlichen Terminals unterschiedlich verhält.

Jeder Terminal-Typ hat mehrere Codenamen, die durch das UNIX-System erkannt werden. In Anhang G, "Einstellung des Terminals," finden Sie Informationen über die zulässigen Namen für Ihr Terminal. Beachten Sie, daß bei der Installation vieler Rechner die Liste der von Ihrem UNIX-System unterstützten Terminal-Typen erweitert wird. Daher sollten Sie sich auf jeden Fall bei Ihrem Systemverwalter nach der aktuellen Liste der unterstützten Terminal-Typen erkundigen.

Die Konfiguration Ihres Terminals stellen Sie folgendermaßen ein:

```
TERM=terminal_name<CR>
export TERM<CR>
tput init<CR>
```

In der ersten Zeile wird einer Variablen namens TERM ein Wert (für den Typ des Terminals) zugeordnet. Dieser Wert wird in der zweiten Zeile "exportiert", so daß er für sämtliche UNIX-Programme gültig ist, deren Ausführung vom Typ des benutzten Terminals abhängig ist.

Mit dem Kommando `tput` in der dritten Zeile wird die Software auf Ihrem Terminal so initialisiert (konfiguriert), daß sie unter dem UNIX-System ordnungsgemäß funktioniert. Das Aufrufen des Kommandos `tput init` ist bei der Konfiguration Ihres Terminals besonders wichtig, da andernfalls bestimmte Funktionen wie z.B. die Umsetzung der Tabulatorzeichen nicht ordnungsgemäß eingestellt werden.

Arbeiten Sie beispielsweise an einem Terminal vom Typ Teletype® 5425, so sehen die Kommandos auf Ihrem Bildschirm folgendermaßen aus:

```
$ TERM=5425<CR>
$ export TERM<CR>
$ tput init<CR>
```

Versuchen Sie auf keinen Fall, einen Namen anzugeben, der nicht zu Ihrem Terminal gehört - Sie könnten damit Störungen der Bildschirmanzeige unter `vi` verursachen!

Einrichten der Benutzerumgebung

Wenn Sie regelmäßig mit `vi` arbeiten möchten, sollten Sie Ihre Benutzerumgebung permanent ändern, so daß Sie Ihr Terminal nicht immer zu Arbeitsbeginn (beim Anmelden) neu konfigurieren müssen. Das Aussehen Ihrer Benutzerumgebung hängt von der Datei `.profile` in Ihrem Home-Verzeichnis ab. Weitere Informationen hierzu finden Sie in Kapitel 9, "Die Shell."

Das Eintragen der Konfiguration Ihres Terminals in die Datei `.profile` hat für Sie den Vorteil, daß Ihr Terminal bei jeder Anmeldung automatisch korrekt eingerichtet wird. Sie sollten daher die drei Zeilen, die auf dem letzten Bildschirm gezeigt wurden (die Belegung der Variablen `TERM`, das Kommando `export` sowie das Kommando `tput`), in Ihre Datei `.profile` eintragen. Ausführliche Informationen hierzu schlagen Sie bitte in Kapitel 9, "Die Shell" nach.

Einstellung des automatischen Wagenrücklaufs

Soll das Steuerzeichen, das der Taste `RETURN` zugeordnet ist, unter `vi` automatisch in den Text eingefügt werden, so legen Sie in Ihrem Home-Verzeichnis eine Datei namens `.exrc` an. In die Datei `.exrc` können Sie Optionen eintragen, durch die Ihre `vi`-Arbeitsumgebung festgelegt wird.

Um die Datei `.exrc` anzulegen, rufen Sie einen Editor mit diesem Dateinamen auf. Geben Sie dann eine Textzeile ein, in der der Zeilenumbruch (automatischer Wagenrücklauf) ein- oder ausgeschaltet wird. Diese Option stellen Sie in folgendem Format ein:

```
:set wm=n<CR>
```

n gibt an, in welchem Abstand vom rechten Bildschirmrand (Anzahl der Zeichen) der automatische Wagenrücklauf durchgeführt werden soll. Hierbei gilt allerdings die Wortgrenze; ein Wort kann also nicht in seine Silben aufgeteilt werden. Soll der Wagenrücklauf also in einem Abstand von 20 Zeichen vom rechten Bildschirmrand aus durchgeführt werden, so geben Sie folgendes ein:

```
:set wm=20<C
```

Speichern Sie den Inhalt des Puffers abschließend ab und beenden Sie den Editor (siehe Kapitel 4, Abschnitt "Editor-Puffer"). Beim nächsten Aufrufen von `vi` sorgt diese Datei dafür, daß der automatische Wagenrücklauf durchgeführt wird.

Die Einstellung der Optionen für den Terminal-Typ und den automatischen Zeilenumbruch können Sie unter `vi` mit dem folgenden Kommando abrufen:

```
:set<CR>
```

`vi` gibt die Einstellungen für den Terminal-Typ und den automatischen Zeilenumbruch sowie weitere Optionen (falls vorhanden) aus. Das Kommando `:set` können Sie auch zur Eingabe oder Änderung der Zeilenumbruch-Option benutzen; probieren Sie verschiedene Einstellungen dieser Optionen aus.

Anlegen einer Datei

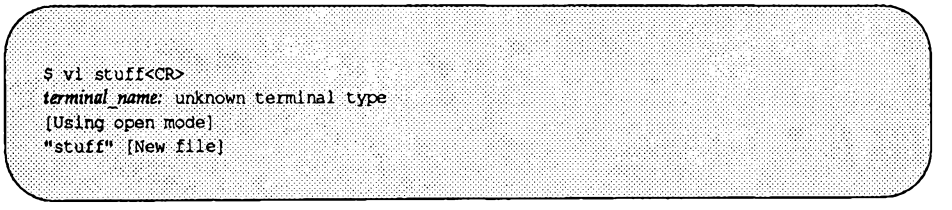
Rufen Sie zunächst den Editor auf, indem Sie das Kommando `vi` mit dem Namen der anzulegenden bzw. zu editierenden Datei eingeben:

`vi` *dateiname*<CR>

Angenommen, Sie möchten eine Datei namens `stuff` anlegen. Sobald Sie das Kommando `vi` mit dem Dateinamen `stuff` eingeben, löscht `vi` den Bildschirm und zeigt ein Fenster an, in dem Sie Text eingeben und editieren können.

Der Unterstrich (_) in der obersten Zeile zeigt die Position des Cursors an (auf Datensichtgeräten ist der Cursor entweder ein blinkender Unterstrich oder ein invers angezeigtes, farbiges Rechteck). Am Anfang aller anderen Zeilen ist lediglich eine Tilde (~), das Symbol für eine leere Zeile, enthalten.

Wenn Sie es vor dem Aufrufen von `vi` versäumt haben, Ihr Terminal zu konfigurieren oder Sie nicht den korrekten Terminal-Typ angegeben haben, wird stattdessen eine Fehlermeldung angezeigt:



Während einer `vi`-Sitzung können Sie das Terminal nicht konfigurieren; hierfür müssen Sie sich auf Shell-Ebene befinden. Beenden Sie daher den Editor mit

dem Kommando

```
:q<CR>
```

Jetzt können Sie Ihr Terminal ordnungsgemäß konfigurieren.

Betriebsmodi

Beim Editor `vi` wird zwischen zwei verschiedenen Betriebsmodi unterschieden: Zwischen dem Eingabemodus, in dem Sie Text eingeben und ändern können, und dem Kommandomodus, in dem Sie folgende Funktionen durchführen können:

- Editieren und Änderung von bereits vorhandenem Text.
- Löschen, Verschieben und Kopieren von Text.
- An eine andere Stelle in der Datei springen.
- Weitere Aufgaben.

Erstellen von Text im Anfügen-Modus

Unmittelbar nach dem Aufrufen von `vi` befinden Sie sich im Kommandomodus, in dem `vi` auf Ihre Kommandos wartet. Wie können Sie jetzt Text eingeben?

- Geben Sie das Kommando `a` (für append) ein, um den Eingabemodus von `vi` aufzurufen (betätigen Sie abschließend NICHT die Taste `RETURN`). Sie können jetzt Text in die Datei einfügen (das Kommando `a` wird nicht auf dem Bildschirm angezeigt).
- Geben Sie Text ein.
- Wenn Sie in einer neuen Zeile fortfahren möchten, betätigen Sie die Taste `RETURN`.

  Wenn Sie in der Datei `.exrc` die Option für den automatischen Zeilenumbruch eingeschaltet haben, springt der Cursor automatisch in eine neue Zeile, sobald die aktuelle Zeile gefüllt ist und daher ein automatischer Wagenrücklauf eingefügt werden muß (siehe Abschnitt "Einstellung des automatischen Wagenrücklaufs").

Beenden des Anfügen-Modus

Wenn Sie keinen Text mehr eingeben möchten, betätigen Sie die Taste ESCAPE, um den Eingabemodus zu beenden und in den Kommandomodus zurückzuschalten. Sie können jetzt den eben erstellten Text editieren oder den im Puffer enthaltenen Text in einer Datei abspeichern .

```
<a>
Create some text<CR>
in the screen editor<CR>
and return to<CR>
command mode.<ESC>
~
~
```

Wenn bei der Betätigung der Taste ESCAPE ein akustisches Signal ertönt, befinden Sie sich bereits im Kommandomodus. Die Betätigung der Taste ESCAPE im Kommandomodus hat keinerlei Auswirkung auf den Text in der Datei, unabhängig davon, wie oft Sie die Taste betätigen.

Hinweis: Auf manchen Terminals gibt `vi` anstelle eines akustischen ein optisches Signal (die Bildschirmanzeige leuchtet auf) ab.

Editieren von Text im Kommandomodus

Das Editieren einer vorhandenen Datei besteht aus dem Einfügen, Ändern und Löschen von Text. Allerdings können Sie diese Funktionen erst durchführen, wenn die zu editierende Stelle auf dem Bildschirm angezeigt wird. Unter `vi` gibt es eine Reihe von Möglichkeiten, um eine Datei seitenweise "durchzublättern" oder den Cursor in eine andere Zeile bzw. an eine andere Stelle der aktuellen Zeile zu bewegen. Um diese Kommandos geht es in diesem Abschnitt. Ein weiteres Thema sind die Kommandos zum Löschen und Einfügen von Text.

Cursorsteuerung

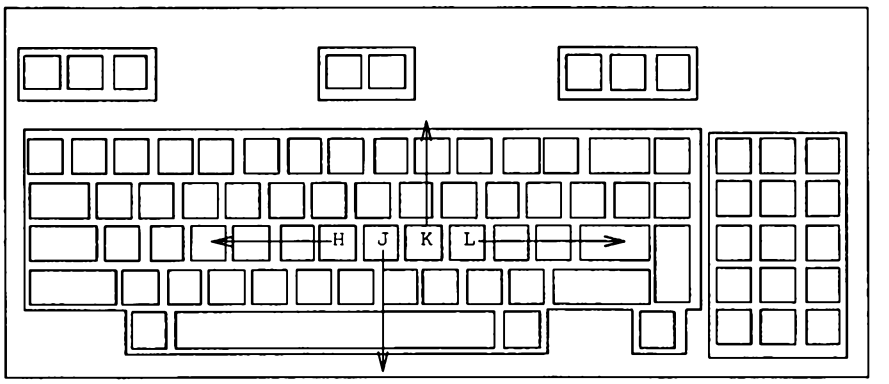
Damit Sie eine bestimmte Textstelle editieren können, müssen Sie den Cursor an diese Stelle auf dem Bildschirm bewegen. Dies ist problemlos über die vier Tasten möglich, die auf der Tastatur nebeneinander angeordnet sind: h, j, k und l (siehe Bild 7-2).

`h`	Cursor um ein Zeichen nach links bewegen.
`j`	Cursor um eine Zeile nach unten bewegen.
`k`	Cursor um eine Zeile nach oben bewegen.
`l`	Cursor um ein Zeichen nach rechts bewegen.

Bei den Kommandos `j` und `k` wird der Cursor innerhalb der aktuellen Spalte nach oben oder unten bewegt. Befindet sich der Cursor beispielsweise in Spalte 7, so wird er bei der Eingabe von `j` oder `k` in der neuen Zeile ebenfalls zum siebten Zeichen bewegt. Enthält die neue Zeile beispielsweise nur fünf Zeichen, so wird der Cursor zum letzten Zeichen in der Zeile bewegt.

Viele `vi`-Benutzer versehen diese vier Tasten mit Pfeilen, die die Richtung der Cursor- Bewegungen anzeigen.

Bild 7-2: Tastatur mit Pfeilen zur Anzeige der Richtung der Cursor-Bewegungen



Hinweis: Auf einigen Terminals gibt es spezielle Cursor-Steuerungstasten, die bereits mit Pfeilen beschriftet sind. Über diese Tasten können Sie dieselben Funktionen durchführen wie über die Kommandos `h`, `j`, `k` und `l`.

Betätigen Sie die Tasten `h`, `j`, `k`, und `l`, und beobachten Sie die Auswirkung auf die Cursor-Position. Wenn Sie den Cursor um mehrere Zeichen nach links oder rechts bzw. um mehrere Zeilen nach oben oder unten bewegen möchten, so gibt es außer den Cursor-Steuerungstasten noch eine weitere Möglichkeit: Sie können dem Kommando die gewünschte Zahl von Zeichen bzw. Zeilen voranstellen. Wenn Sie den Cursor also um zwei Stellen nach rechts bewegen möchten, können Sie entweder zweimal die Taste `l` betätigen oder das Kommando `2l` eingeben. Um den Cursor um vier Zeilen nach oben zu bewegen, betätigen Sie entweder viermal die Taste `k` oder geben das Kommando `4k` ein. Wenn eine Cursor-Bewegung in der angegebenen Richtung nicht mehr möglich ist, erzeugt `vi` ein akustisches Signal.

Versuchen Sie den Cursor mit den Kommandos `j` und `k` an eine andere Stelle zu bewegen. Bewegen Sie ihn zunächst um sieben Zeilen nach oben:

```
7k
```

Der Cursor wird, ausgehend von der aktuellen Zeile, um sieben Zeilen nach oben bewegt. Gibt es oberhalb der aktuellen Zeile beispielsweise nur noch fünf Zeilen, so ertönt ein akustisches Signal, und der Cursor bleibt in der aktuellen Zeile.

Bewegen Sie den Cursor jetzt um 35 Zeilen nach unten: Geben Sie folgendes ein:

```
35j
```

Der Bildschirm wird von `vi` gelöscht und neu aufgebaut. Auf der neuen Bildschirmanzeige befindet sich der Cursor in der 35. Zeile unterhalb der aktuellen Zeile, die sich in der Mitte der neuen Bildschirmanzeige befindet. Gibt es unterhalb der aktuellen Zeile beispielsweise nur noch 33 Zeilen, so ertönt ein akustisches Signal, und der Cursor bleibt in der aktuellen Zeile. Geben Sie das folgende Kommando ein, und beachten Sie das Ergebnis:

```
35k
```

Die Positionierungs-Kommandos `h`, `j`, `k` und `l` laufen, wie die meisten `vi`-Kommandos, "stillschweigend" ab: Sie werden bei der Eingabe nicht auf dem Bildschirm angezeigt. Zeichen sollten nur dann auf dem Bildschirm angezeigt werden, wenn Sie sich im Eingabemodus befinden und Text in Ihre Datei einfügen. Wenn Sie also ein Positionierungs-Kommando eingeben und dieses Kommandos auf dem Bildschirm erscheint, befinden Sie sich noch im Eingabemodus. In einem solchen Fall schalten Sie über die Taste `ESCAPE` in den Kommandomodus zurück und geben die Kommandos erneut ein.

Cursor nach links oder rechts bewegen

Außer den Positionierungs-Kommandos `h` and `l` können Sie den Cursor auch über die Leertaste sowie die Taste `BACKSPACE` zu einem Zeichen bewegen, das sich in der aktuellen Zeile links oder rechts von der aktuellen Cursor-Position befindet.

<Leertaste>	Cursor um ein Zeichen nach rechts bewegen.
<*n*Leertaste>	Cursor um *n* Zeichen nach rechts bewegen.

<BACKSPACE>	Cursor um ein Zeichen nach links bewegen.
<*n*BACKSPACE>	Cursor um *n* Zeichen nach links bewegen.

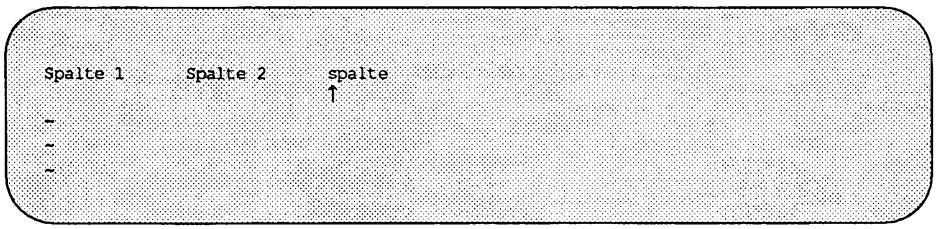
Geben Sie eine Zahl ein, bevor Sie die Kommandotaste betätigen. Der Cursor sollte jetzt um die angegebene Zeichenzahl nach links oder rechts bewegt werden. Im folgenden Beispiel wird die Cursor-Bewegung anhand von Pfeilen gezeigt.

Um den Cursor um mehrere Zeichen gleichzeitig nach links oder rechts zu bewegen, geben Sie vor dem Kommando eine Zahl ein. Angenommen, Sie möchten auf Ihrem Bildschirm vier Spalten erstellen; nachdem Sie die Titel für die ersten drei Spalten eingegeben haben, stellen Sie einen Eingabefehler fest:

<ESC>

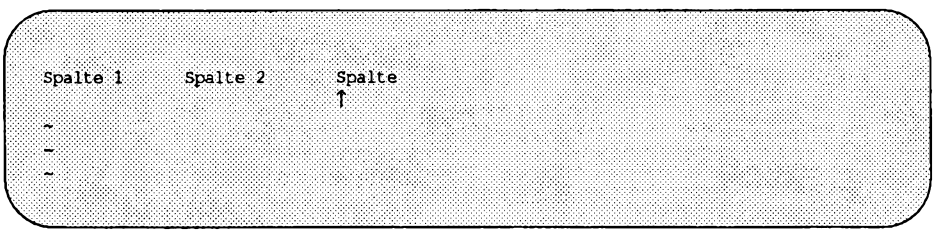
Wenn Sie den Eingabefehler sofort, noch vor der Eingabe weiterer Zeichen, korrigieren möchten, so beenden Sie den Einfügen-/Eingabemodus und schalten über die Taste ESCAPE in den Kommandomodus zurück; der Cursor wird zum Buchstaben e bewegt. Jetzt können Sie den Cursor über die Taste h um fünf Stellen zurückbewegen. In diesem Beispiel wird, wie auch in vielen anderen Beispielen in diesem Kapitel, sowohl die ursprüngliche Bildschirmanzeige als auch auf einem zweiten Bildschirm das Ergebnis gezeigt.

```
5h
```

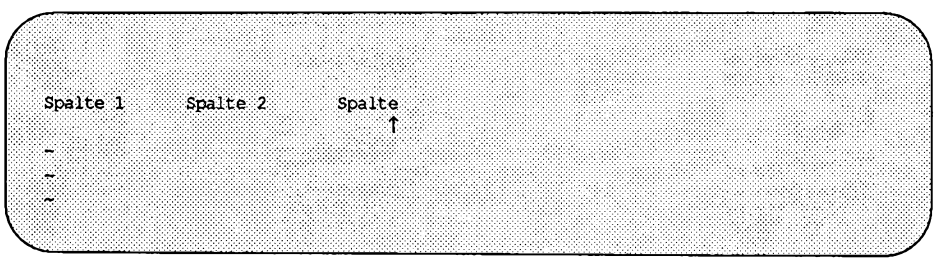



```
xiC<ESC>
```

Um den Buchstaben `s` zu löschen, geben Sie `x` ein. Dann schalten Sie in den Eingabemodus um (`i`), geben ein `S` ein und betätigen die Taste `ESCAPE`. Bewegen Sie den Cursor über das Positionierungs-Kommando `l` zu der ursprünglichen Stelle zurück. Die Kommandos `x` und `i` werden an einer späteren Stelle dieses Kapitels besprochen.



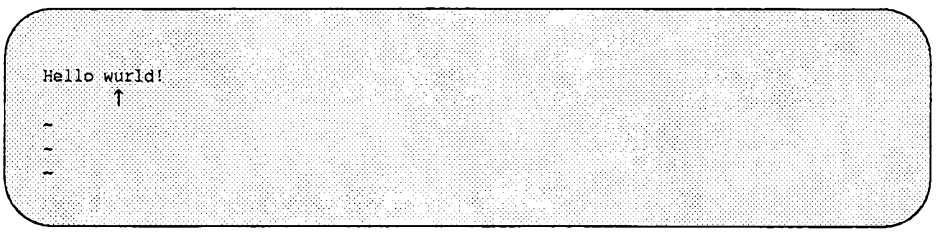

```
5l
```



Löschen von Text

Um ein Zeichen zu löschen, bewegen Sie den Cursor zu diesem Zeichen und betätigen die Taste x. Beachten Sie dabei die Bildschirmanzeige; das Zeichen verschwindet vom Bildschirm, und die nachfolgenden Zeichen rücken um ein Zeichen nach vorn. Drei aufeinanderfolgende Zeichen können Sie löschen, indem Sie x drei Mal betätigen. Im folgenden Beispiel ist die Position des Cursors durch einen Pfeil unter dem jeweiligen Buchstaben gekennzeichnet.

x	Ein Zeichen wird gelöscht.
*n*x	*n* Zeichen werden gelöscht; *n* steht dabei für die Anzahl der zu löschenden Zeichen.



x

Geben Sie probehalber x mit der Anzahl der zu löschenden Zeichen ein. Wenn Sie beispielsweise das Wort deep bei seinem zweiten Auftreten aus dem folgenden Beispieltext löschen möchten, bewegen Sie den Cursor zum ersten Wort der zu löschenden Zeichenkette und löschen fünf Zeichen (die vier Buchstaben des Worts deep plus eine Leerstelle).

```
Tomorrow the Loch Ness monster
shall slither forth from
the deep dark deep depths of the lake.
              ↑
~
~
~
```

5x

```
Tomorrow the Loch Ness monster
shall slither forth from
the deep dark depths of the lake.
              ↑
~
~
~
```

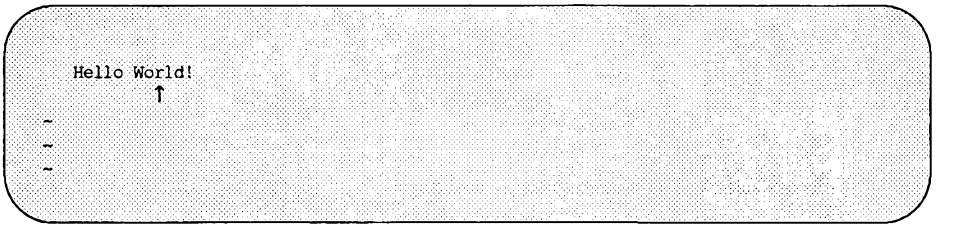
Beachten Sie, daß die Zeichen hinter dem gelöschten Wort von `vi` automatisch nach vorn rücken, so daß im Text keine Lücke entsteht. Handelt es sich, wie in diesem Fall, bei der zu löschenden Zeichenkette um ein ganzes Wort, so gibt es unter `vi` ein spezielles Kommando (siehe Abschnitt "Wortweises Positionieren").

Eintragen von neuem Text

Zum Eintragen von neuem Text gibt es zwei grundlegende Kommandos: Einfügen (`i` für insert) und Anfügen (`a` für append). Um mit dem Einfügen-Kommando an einer aktuell sichtbaren Textstelle Text einzufügen, bewegen Sie den Cursor über die Kommandos `h`, `j`, `k` und `l` zu dieser Stelle. Dann betätigen Sie die Taste `i` und beginnen mit der Eingabe des Texts. Die neuen Zeichen erscheinen während der Eingabe links vor der aktuellen Cursor-Position. Das Zeichen an der aktuellen Cursor-Position sowie sämtliche darauffolgenden Zeichen rücken nach rechts, um Platz für den neuen Text zu schaffen. Die

eingegebenen Zeichen werden von `vi` auf diese Weise verarbeitet, bis Sie wieder die Taste `ESCAPE` betätigen. Die ursprünglichen Zeichen werden gegebenenfalls sogar in die nächste Zeile gesetzt.


```
io
```




```
<ESC>
```

Das Anfügen-Kommando hat im Prinzip dieselbe Funktion, nur werden die Zeichen hinter der aktuellen Cursor-Position in den Text eingefügt.

Informationen darüber, wie Sie zu verschiedenen Stellen auf dem Bildschirm springen können oder eine Datei durchblättern können, um Zeichen, Wörter bzw. Zeilen einzufügen oder zu löschen, finden Sie an einer späteren Stelle dieses Kapitels.

Beenden von `vi`

Nach Beendigung der Texteingabe können Sie den Inhalt des Puffers in einer Datei abspeichern und zur Shell zurückkehren. Hierzu halten Sie die Taste SHIFT gedrückt und betätigen zweimal die Taste `Z` (`ZZ`). Der Editor speichert den Dateinamen, den Sie zu Beginn der Editor-Sitzung beim Kommando `vi` angegeben haben, und bringt den Text im Puffer in die Datei dieses Namens ein. In einer Meldung am unteren Bildschirmrand werden Sie über den Namen der Datei und ihre Größe (Anzahl der enthaltenen Zeilen und Zeichen) informiert. Dann gibt die Shell eine Eingabeaufforderung aus.

```
<a>
This is a test file.<CR>
I am adding text to<CR>
a temporary buffer and<CR>
now it is perfect.<CR>
I want to write this file,<CR>
and return to the shell.<ESC><ZZ>
~
~
~
~
"stuff" [New file] 6 lines, 135 characters
$
```

Das Abspeichern der Datei und das Beenden des Editors können Sie auch mit den Zeileneditor-Kommandos `:w` und `:q` durchführen (die Zeileneditor-Kommandos beginnen mit einem Doppelpunkt und werden in der untersten Bildschirmzeile angezeigt). Das Kommando `:w` bringt den Puffer-Inhalt in eine Datei ein. Mit dem Kommando `:q` beenden Sie den Editor und kehren zur Shell zurück. Die beiden Kommandos können entweder in separate Zeilen gesetzt oder in einer einzigen Zeile unmittelbar hintereinander geschrieben werden (`:wq`).

```
<a>This is a test file.<CR>
I am adding text to<CR>
a temporary buffer and<CR>
now it is perfect.<CR>
I want to write this file,<CR>
and return to the shell.<ESC>
~
~
~
~
:wq<CR>
```

Tabelle 7-1 enthält einen Überblick über die grundlegenden Kommandos zum Aufrufen und Benutzen von `vi`.

Tabelle 7-1: Die Kommandos des Editors `vi`

Kommando	Funktion
`TERM=`*terminal_name* `export TERM`	Konfiguration des Terminals angeben.
`tput init`	Initialisierung des Terminals entsprechend des Eintrags *terminal_name*.
`vi` *datei_name*	Aufrufen des Editors `vi` zum Editieren der Datei *datei_name*.
`a`	Einfügen von Text hinter dem Cursor (Anfügen).
`h`	Cursor um ein Zeichen nach links bewegen.
`j`	Cursor um eine Zeile nach unten bewegen.
`k`	Cursor um eine Zeile nach oben bewegen.
`l`	Cursor um ein Zeichen nach rechts bewegen.
`x`	Löschen eines Zeichens.
`<CR>`	Wagenrücklauf
`<ESC>`	Verlassen des Eingabemodus und Umschalten in den `vi`-Kommandomodus.
`:w`	Abspeichern des Puffer-Inhalts in einer Datei.
`:q`	Beenden von `vi`.
`:wq`	Abspeichern des Puffer-Inhalts in einer Datei und Beenden von `vi`.
`ZZ`	Abspeichern der Änderungen in einer Datei und Beenden von `vi`.

Übung 1

Die Auflösung der Übungen finden Sie am Ende dieses Kapitels. Beachten Sie jedoch, daß es in vielen Fällen zwei oder mehr Möglichkeiten zur Durchführung einer bestimmten Funktion unter `vi` gibt - nicht die Methode ist entscheidend, sondern das Ergebnis!

Bei der Eingabe der Kommandos in den folgenden Übungen sollten Sie stets auf die Änderungen achten, die auf dem Bildschirm vor sich gehen, ebenso auf die Cursor-Bewegungen.

1-1. Melden Sie sich bei UNIX an, falls Sie dies noch nicht erledigt haben. Konfigurieren Sie dann Ihr Terminal.

1-2. Rufen Sie `vi` auf und fügen Sie in eine neue Datei namens `exer1` die folgenden fünf Zeilen ein:

```
This is an exercise!
Up, down,
left, right,
build your terminal's
muscles bit by bit
```

1-3. Bewegen Sie den Cursor in die erste Zeile der Datei, in die siebte Spalte von rechts. Beachten Sie, daß der Cursor immer nur dann zum letzten Zeichen der vorhergehenden Zeile springt, wenn diese Zeile kürzer ist als die aktuelle.

1-4. Löschen Sie das siebte und achte Zeichen von rechts.

1-5. Bewegen Sie den Cursor zum letzten Zeichen in der letzten Textzeile.

1-6. Fügen Sie die folgende neue Textzeile ein:

```
and byte by byte
```

1-7. Speichern Sie den Puffer-Inhalt in einer Datei ab und beenden Sie `vi`.

1-8. Rufen Sie `vi` erneut auf und fügen Sie zwei weitere Textzeilen in die Datei `exer1` ein.
Was wird in der untersten Bildschirmzeile angezeigt, nachdem Sie `vi` zum Editieren der Datei `exer1` aufgerufen haben?

Positionierung des Cursors auf dem Bildschirm

Bisher haben Sie lediglich mit den Positionierungs-Kommandos `h`, `j`, `k`, `l`, der Taste BACKSPACE und der Leertaste gearbeitet. Unter `vi` stehen Ihnen eine Reihe weiterer Kommandos zur Verfügung, mit denen Sie den Cursor schnell an eine andere Stelle auf dem Bildschirm bewegen können. In diesem Abschnitt werden die folgenden Methoden zur Cursor-Positionierung beschrieben:

- Zeichenweises Positionieren innerhalb einer Zeile.
- Zeilenweises Positionieren.
- Textobjekt-bezogenes Positionieren:
  - Wortweises Positionieren.
  - Satzweises Positionieren.
  - Absatzweises Positionieren.
- Bildschirmbezogenes Positionieren.

Mit weiteren Kommandos können Sie den Cursor zu Stellen innerhalb des `vi`-Editor-Puffers bewegen, die nicht auf dem Bildschirm sichtbar sind. Diese Kommandos werden im Abschnitt "Cursor an eine aktuell nicht angezeigte Stelle bewegen" im Anschluß an diesen Abschnitt beschrieben.

Für diesen Abschnitt sollten Sie `vi` mit einer Datei aufrufen, die mindestens 40 Zeilen enthält. Legen Sie die Datei gegebenenfalls neu an, wenn Ihnen noch keine Datei dieser Länge verfügbar ist. Beachten Sie stets, daß die hier beschriebenen Kommandos im Kommandomodus von `vi` aufgerufen werden müssen. Betätigen Sie sicherheitshalber die Taste ESCAPE, damit Sie sich ganz bestimmt nicht im Eingabemodus befinden.

Cursor zu einem Zeichen bewegen

Um den Cursor innerhalb der aktuellen Zeile zu einem Zeichen zu bewegen, gibt es drei Möglichkeiten:

- Sie bewegen den Cursor um ein Zeichen nach links oder rechts.
- Sie geben das Zeichen an, das sich entweder am Anfang oder am Ende der Zeile befindet.
- Sie durchsuchen die Zeile nach dem Zeichen.

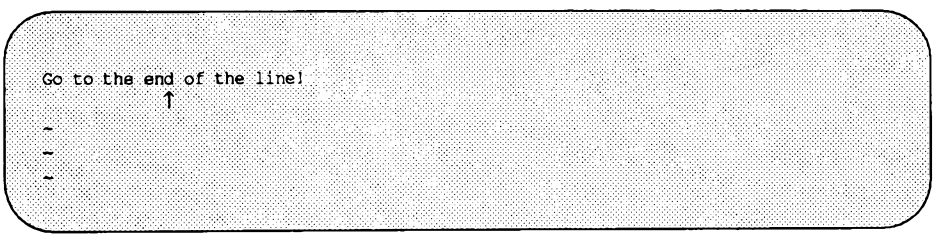
Die erste Möglichkeit wurde im Abschnitt "Cusor nach links oder rechts bewegen" in diesem Kapitel beschrieben. In den folgenden Abschnitten geht es um die anderen zwei Möglichkeiten.

Cursor zum Zeilenanfang oder -ende bewegen

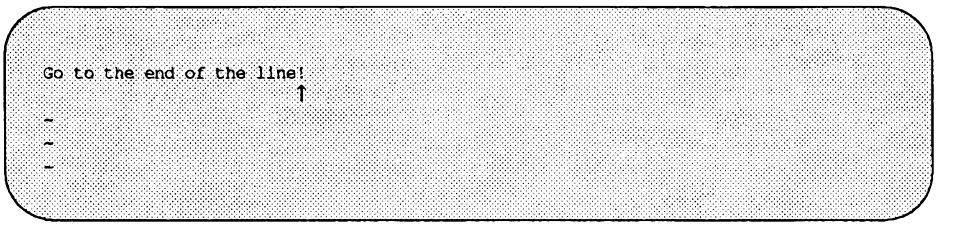
Zur zweiten Möglichkeit der Cursor-Positionierung innerhalb einer Zeile gibt es drei Kommandos, mit denen der Cursor zum letzten oder ersten Zeichen einer Zeile bewegt wird.

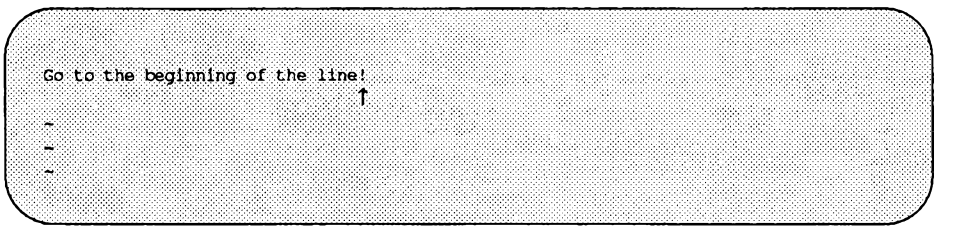
$	Cursor zum letzten Zeichen einer Zeile bewegen.
0 (Null)	Cursor zum ersten Zeichen einer Zeile bewegen.
^ (Zirkumflex)	Cursor zum ersten Zeichen einer Zeile bewegen, bei dem es sich nicht um ein Leerzeichen handelt.

Die folgenden Beispiele zeigen die Auswirkung dieser drei Kommandos auf die Position des Cursors.

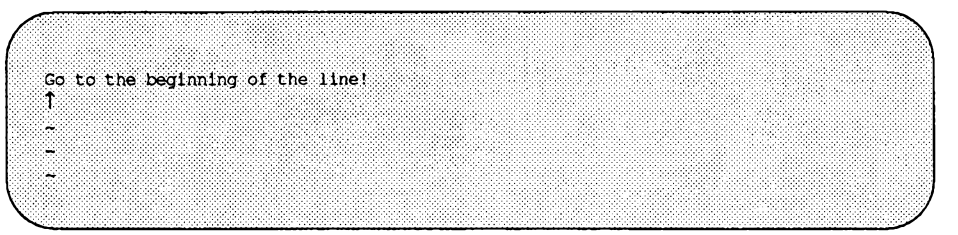


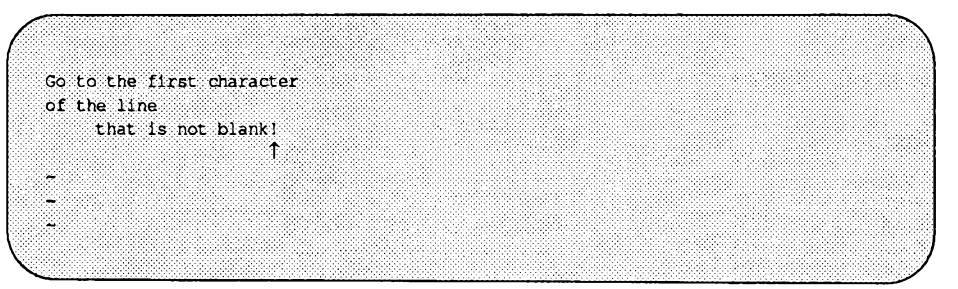
$


Go to the end of the line!
~
~
~


Go to the beginning of the line!
~
~
~

0


Go to the beginning of the line!
~
~
~
~


Go to the first character
of the line
that is not blank!
~
~
~

```
^
```

```
Go to the first character
of the line
     that is not blank!
     ↑
~
~
~
```

Zeichen in einer Zeile suchen

Bei der dritten Möglichkeit zur Positionierung des Cursors innerhalb einer Zeile wird die aktuelle Zeile nach einem bestimmten Zeichen durchsucht. Ist das Zeichen nicht in der aktuellen Zeile enthalten, so wird ein akustisches Signal abgegeben, und die Position des Cursors ändert sich nicht (ein weiteres Kommando, mit dem in einer Datei nach Mustern gesucht wird, wird im nächsten Abschnitt beschrieben). Zum Durchsuchen einer Zeile gibt es die sechs folgenden
Kommandos: `f`, `F`, `t`, `T`, `;` und `,`. Jedem dieser Kommandos mit Ausnahme von `;` und `,` muß ein Zeichen folgen.

`f`*x*	Der Cursor wird nach rechts zum Zeichen *x* bewegt.
`F`*x*	Der Cursor wird nach links zum Zeichen *x* bewegt.
`t`*x*	Der Cursor wird nach rechts und zu dem Zeichen bewegt, das sich unmittelbar vor dem Zeichen *x* befindet.
`T`*x*	Der Cursor wird nach links und zu dem Zeichen bewegt, das sich unmittelbar hinter dem Zeichen *x* befindet.
`;`	Der Suchvorgang, der im letzten Kommando gestartet wurde, wird in derselben Richtung fortgesetzt. Das Kommando `;`, das das gesuchte Zeichen abspeichert, sucht dann nach dem nächsten Auftreten dieses Zeichens in der aktuellen Zeile.
`,`	Der Suchvorgang, der im letzten Kommando gestartet wurde, wird in der entgegengesetzten Richtung fortgesetzt. Das Kommando `,`, das das gesuchte Zeichen abspeichert, sucht dann nach dem vorhergehenden Auftreten dieses Zeichens in der

aktuellen Zeile.

Im nächsten Beispiel durchsucht `vi` die nachfolgenden Zeichen in der aktuellen Zeile nach dem ersten Auftreten des Buchstabens A.

```
Go forward to the letter A on this line.
          ↑
~
~
~
```

```
fA
```

```
Go forward to the letter A on this line.
                        ↑
~
~
~
```

Probieren Sie die Suchen-Kommandos an einer Ihrer Dateien aus.

Zeilenweises Positionieren

Außer den bereits bekannten Kommandos `j` und `k` können zur Positionierung des Cursors in eine andere Zeile auch die folgenden Kommandos benutzt werden: `+`, `-` und `<CR>`.

Das Positionierungs-Kommando - Minus-Zeichen (-)

Mit dem Kommando - (Minus) wird der Cursor zum ersten Zeichen in der darüberliegenden Zeile bewegt, bei dem es sich nicht um ein Leerzeichen handelt (falls vorhanden). Wenn Sie gleichzeitig mehrere Zeilen überspringen möchten, so stellen Sie dem Kommando - die Anzahl der Zeilen voran. Mit dem folgenden Kommando wird der Cursor z.B. um 13 Zeilen nach oben

bewegt:

```
13-
```

Befinden sich einige dieser Zeilen oberhalb des aktuellen Fensters, wird die Bildschirmanzeige nach oben verschoben, um die Zeilen sichtbar zu machen. Dies stellt eine schnelle Möglichkeit zum Überspringen mehrerer Zeilen dar.

Versuchen Sie jetzt 100 Zeilen zu überspringen:

```
100-
```

Was geschieht mit dem Fenster? Wenn es oberhalb der aktuellen Zeile z.B. nur 80 Zeilen gibt, so werden Sie über ein akustisches Signal auf den Fehler aufmerksam gemacht; die Position des Cursors ändert sich dann nicht.

Das Positionierungs-Kommando: Plus-Zeichen (+)

Mit dem Kommando (+) (Plus) sowie dem Kommando <CR> können Sie den Cursor zum ersten Zeichen in der nächsten Zeile bewegen, bei dem es sich nicht um ein Leerzeichen handelt. Auch mit dem Kommando + können Sie mehrere Zeilen überspringen. Wenn Sie den Cursor beispielsweise um neun Zeilen nach unten bewegen möchten, geben Sie folgendes ein:

```
9+
```

Befinden sich einige dieser Zeilen unterhalb des aktuellen Fensters, so wird der Bildschirminhalt nach unten verschoben, um die Zeilen sichtbar zu machen.

Wiederholen Sie das Beispiel, nur betätigen Sie jetzt die Taste RETURN. Ist das Ergebnis mit dem ersten (nach der Betätigung der Taste +) identisch?

Wortweises Positionieren

Unter `vi` wird ein Wort als eine Zeichenkette behandelt, die Buchstaben, Zahlen und Unterstriche enthalten kann. Zur wortweisen Positionierung gibt es sechs Kommandos: `w`, `b`, `e`, `W`, `B` und `E`. Bei den kleingeschriebenen Kommandos (`w`, `b` und `e`) wird jedes Zeichen, bei dem es sich nicht um einen Buchstaben, eine Zahl oder ein Unterstrich handelt, als Begrenzer behandelt; Begrenzer markieren den Anfang bzw. das Ende eines Worts. Satzzeichen, die vor oder hinter einem Leerzeichen stehen, werden ebenfalls als Wort interpretiert. Auch Zeilenanfang und -ende werden als Begrenzer interpretiert.

Bei den großgeschriebenen Kommandos (`W`, `B` und `E`) werden Satzzeichen als Teil des Worts interpretiert; als Begrenzer zwischen den einzelnen Wörtern wird ein Leerraum (das Leerzeichen, das Tabulatorzeichen sowie das Zeilenendezeichen) benutzt.

Die einzelnen Kommandos zum wortweisen Positionieren haben folgende Funktion:

`w` Der Cursor wird zum ersten Zeichen des nächsten Worts bewegt. Wenn Sie mehrere Wörter überspringen möchten, so können Sie entweder mehrmals die Taste `w` betätigen, oder dem Kommando `w` die entsprechende Zahl voranstellen.

n`w` Der Cursor wird zum ersten Zeichen des Worts bewegt, das *n* Wörter hinter der aktuellen Cursor-Position steht. Ist dieses Wort nicht in der aktuellen Zeile enthalten, so springt der Cursor automatisch in die entsprechende Zeile; die Wörter werden über Zeilen-Grenzen hinweg fortlaufend gezählt.

```
The <w> command
leaps word by word through the
file.  Move from THIS word forward

six words to THIS word.
~
~
```

```
6w
```

```
The w command
leaps word by word through the
file.  Move from THIS word forward
six words to THIS word.
          ↑
~
~
```

`W` Der Cursor wird zu dem Wort bewegt, das auf das nächste Leerzeichen folgt; alle Satzzeichen werden dabei ignoriert.

`e` Der Cursor wird zum letzten Zeichen des nächsten Worts in derselben Zeile bewegt.

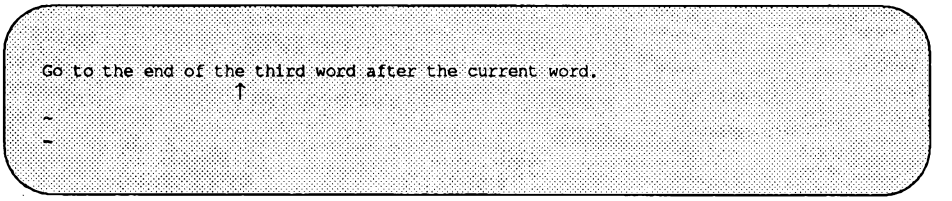
```
Go forward one word to the end of
the next word in this line
  ↑
~
~
```

`e`

```
Go forward one word to the end of
the next word in this line
      ↑
~
~
```

```
Go to the end of the third word after the current word.
       ↑
~
~
```

`3e`

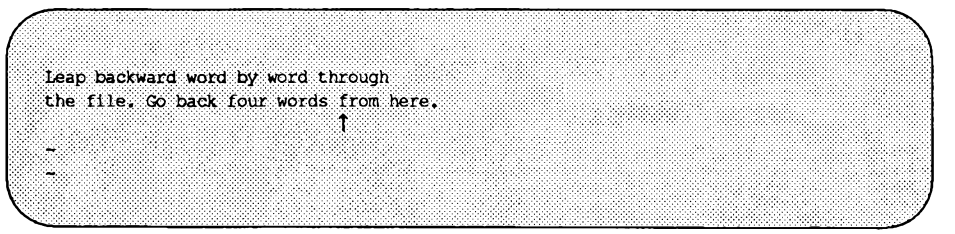


`E` Alle Satzzeichen mit Ausnahme des Leerzeichens werden ignoriert; als Begrenzer wird nur der Leerraum interpretiert.

`b` Der Cursor wird innerhalb der aktuellen Zeile zum ersten Zeichen des vorhergehenden Worts bewegt.

n`b` Der Cursor wird um *n* Wörter zurückbewegt, zum ersten Zeichen des *n*-ten Worts. Der Zeilenbeginn stellt für das Kommando `b` keine Begrenzung dar; der Cursor wird gegebenenfalls zum Ende der vorhergehenden Zeile bewegt und in dieser Zeile zurückbewegt.

`B` Dieses Kommando funktioniert wie das Kommando `b`, jedoch werden als Begrenzer nur die Leer- und Zeilenendezeichen interpretiert. Alle anderen Satzzeichen werden wie normale, zu einem Wort gehörige Buchstaben behandelt.




```
4b
```

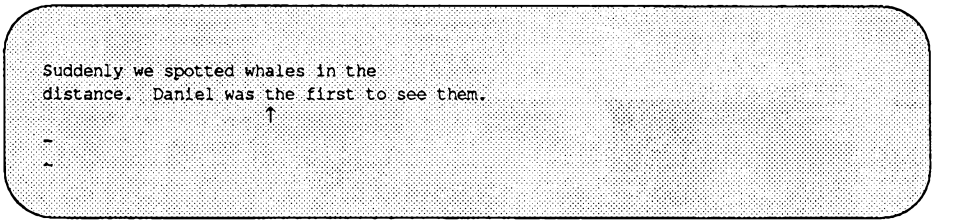
Satzweises Positionieren

Der Editor `vi` erkennt auch Sätze. Unter `vi` endet ein Satz mit einem Ausrufezeichen, einem Punkt oder einem Fragezeichen (`!` oder `.` oder `?`). Sind diese Begrenzer in der Zeilenmitte enthalten, so müssen darauf zwei Leerzeichen folgen; andernfalls werden sie von `vi` nicht korrekt interpretiert. Sie sollten es sich zur Gewohnheit machen, nach einem Punkt (.) am Ende eines Satzes zwei Leerzeichen einzugeben - dies ermöglicht Ihnen die Manipulation von Sätzen als Ganzes.

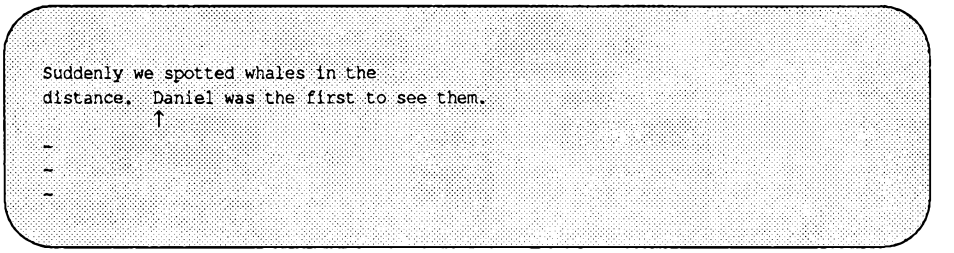
Für das satzweise Positionieren des Cursors innerhalb der Datei gibt es die Kommandos `(` (öffende Klammer) und `)` (schließende Klammer).

`(`	Der Cursor wird zum Anfang des aktuellen Satzes bewegt.
n `(`	Der Cursor wird zum Anfang des *n*-ten Satzes über dem aktuellen Satz bewegt.
`)`	Der Cursor wird zum Anfang des nächsten Satzes bewegt.
n `)`	Der Cursor wird zum Anfang des *n*-ten Satzes unter dem aktuellen Satz bewegt.

Das folgende Beispiel zeigt, wie der Cursor über die öffnende Klammer an verschiedene Stellen auf dem Bildschirm bewegt werden kann:



(



Wiederholen Sie jetzt das Kommando; stellen Sie ihm jetzt aber wie im folgenden Beispiel eine Zahl voran:

3 ((oder)
5)

Ist der Cursor um die korrekte Anzahl von Sätzen bewegt worden?

Absatzweises Positionieren

Als Absatz wird von `vi` Text interpretiert, dem eine Leerzeile vorangeht. Wenn Sie den Cursor später eventuell zum Anfang eines Absatzes bewegen (oder einen ganzen Absatz löschen oder überschreiben) möchten, so müssen Sie hinter jedem Absatz eine Leerzeile einfügen.

{	Der Cursor wird zum Anfang des aktuellen Absatzes bewegt (Begrenzer: vorangestellte Leerzeile).
n{	Der Cursor wird zum Anfang des *n*-ten Absatzes über dem aktuellen Absatz bewegt.
}	Der Cursor wird zum Anfang des nächsten Absatzes bewegt.
n}	Der Cursor wird zum *n*-ten Absatz unter der aktuellen Zeile bewegt.

Die folgenden Beispiele zeigen, wie der Cursor zum Anfang eines anderen Absatzes bewegt werden kann.

```
Suddenly, we spotted whales in the
distance.  Daniel was the first to see them.
          ↑


"Hey look!  Here come the whales!" he cried excitedly.
~
~
```

}

```
Suddenly, we spotted whales in the
distance.  Daniel was the first to see them.

<----------
"Hey look!  Here come the whales!" he cried excitedly.
~
~
```

Bildschirmbezogenes Positionieren

Der Editor `vi` weist auch drei Kommandos auf, mit denen Sie die Position des Cursors innerhalb des Fensters festlegen können. Diese Kommandos müssen auf jeden Fall in Großschreibweise eingegeben werden.

`H`	(Home) Der Cursor wird zur ersten Zeile auf dem Bildschirm bewegt.
`M`	(Middle) Der Cursor wird in die mittlere Zeile des Bildschirms bewegt.
`L`	(Last Line on Screen) Der Cursor wird in die letzte Zeile auf dem Bildschirm bewegt.

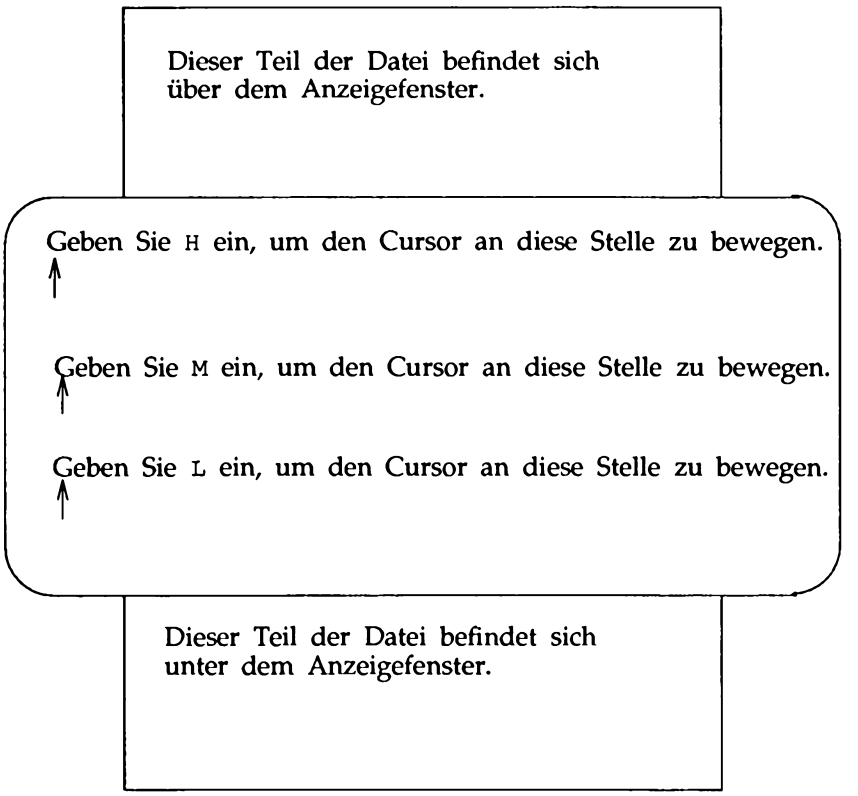


Tabelle 7-2 (Blatt 1 bis 4) zeigt die `vi`-Kommandos, mit denen Sie den Cursor zeilen-, zeichen-, wort-, satz-, absatzweise oder bildschirmbezogen positionieren können (weitere `vi`-Kommandos zur Cursor-Positionierung finden Sie in Tabelle 7-3 an einer späteren Stelle dieses Kapitels).

Tabelle 7-2: Die `vi`-Positionierungs-Kommandos

Zeichenweises Positionieren	
`h`	Cursor um ein Zeichen nach links bewegen.
`l`	Cursor um ein Zeichen nach rechts bewegen.
`<BACKSPACE>`	Cursor um ein Zeichen nach links bewegen.
`<Leertaste>`	Cursor um ein Zeichen nach rechts bewegen.
`f`*x*	Cursor nach rechts zum Zeichen *x* bewegen.
`F`*x*	Cursor nach links zum Zeichen *x* bewegen.
`t`*x*	Cursor nach rechts bewegen, zu dem Zeichen, das unmittelbar vor dem Zeichen *x* steht.
`T`*x*	Cursor nach links bewegen, zu dem Zeichen, das unmittelbar hinter dem Zeichen *x* steht.
`;`	Das Zeichen, das zuletzt bei `f`, `F`, `t` oder `T` angegeben wurde, wird erneut in derselben Richtung gesucht; das Kommando `;` speichert das Zeichen ab und sucht nach dem nächsten Auftreten in der aktuellen Zeile.
`,`	Das Zeichen, das zuletzt bei `f`, `F`,`t` oder `T` angegeben wurde, wird erneut in der entgegengesetzten Richtung gesucht; das Kommando `,` speichert das Zeichen ab und sucht nach dem nächsten Auftreten in der aktuellen Zeile.

Tabelle 7-2: Die `vi`-Positionierungs-Kommandos (Fortsetzung)

	Zeichenweises Positionieren
`k`	Der Cursor wird innerhalb der aktuellen Spalte in die vorhergehende Zeile bewegt (vorausgesetzt, in dieser Spalte ist ein Zeichen vorhanden).
`j`	Der Cursor wird innerhalb der aktuellen Spalte in die nächste Zeile bewegt (vorausgesetzt, in dieser Spalte ist ein Zeichen vorhanden).
`-`	Cursor zum Anfang der vorhergehenden Zeile bewegen.
`+`	Cursor zum Anfang der nächsten Zeile bewegen.
`<CR>`	Cursor zum Anfang der nächsten Zeile bewegen.
	Cursor-Positionierung innerhalb der aktuellen Zeile
`$`	Cursor zum letzten Zeichen der Zeile bewegen.
`0 (zero)`	Cursor zum ersten Zeichen der Zeile bewegen.
`^ (Zirkumflex)`	Cursor zum ersten Zeichen der Zeile bewegen, bei dem es sich nicht um ein Leerzeichen handelt.

Tabelle 7-2: Die `vi`-Positionierungs-Kommandos (Fortsetzung)

	Wortweises Positionieren
`w`	Cursor zum ersten Zeichen des nächsten Worts bewegen.
`W`	Cursor zum Anfang des nächsten Worts bewegen. Als Begrenzer wird nur der Leerraum interpretiert; alle Satzzeichen werden ignoriert.
`b`	Cursor zum Anfang des vorhergehenden Worts bewegen.
`B`	Cursor um ein Wort nach links bewegen; als Begrenzer wird nur der Leerraum interpretiert.
`e`	Cursor zum Ende des aktuellen Worts bewegen.
`E`	Als Begrenzer zwischen den Wörtern soll nur der Leerraum interpretiert werden. Der Cursor wird zum letzten Zeichen vor dem nächsten Leerraum bewegt oder zum Zeilenende.

Tabelle 7-2: Die `vi`-Positionierungs-Kommandos (Fortsetzung)

	Satzweises Positionieren
(	Cursor zum Anfang des aktuellen Satzes bewegen.
)	Cursor zum Anfang des nächsten Satzes bewegen.
	Absatzweises Positionieren
{	Cursor zum Anfang des aktuellen Absatzes bewegen.
}	Cursor zum Anfang des nächsten Absatzes bewegen.
	Bildschirmbezogenes Positionieren
H	Cursor in die erste Zeile auf dem Bildschirm bewegen.
M	Cursor in die mittlere Zeile des Bildschrims bewegen.
L	Cursor in die letzte Zeile auf dem Bildschirm bewegen.

Cursor an eine aktuell nicht angezeigte Stelle bewegen

Unter `vi` gibt es auch die Möglichkeit, den Cursor an eine Stelle in der Datei zu bewegen, die nicht im aktuellen Editor-Fenster angezeigt wird. Hierzu können Sie beispielsweise das Kommando `20j` oder `20k` benutzen. Allerdings ist dies beim Editieren einer großen Datei eine ziemlich umständliche und ungenaue Methode, um an eine andere Stelle in der Datei zu gelangen. In diesem Abschnitt werden dafür die folgenden komfortableren Methoden beschrieben:

- Datei in Vor- und Rückwärtsrichtung durchblättern
- Cursor in eine bestimmte Zeile der Datei bewegen
- Datei nach einem Muster durchsuchen

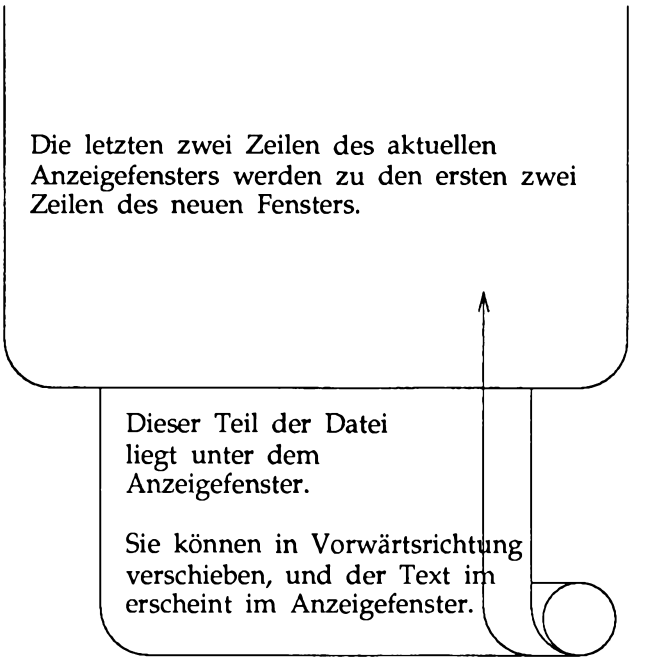
Text durchblättern

Zum Durchblättern eines Texts, der in einer Datei enthalten ist, stehen vier Kommandos zur Verfügung. Mit `^f` (Control-f) und `^d` (Control-d) wird die Bildschirmanzeige in Vorwärtsrichtung verschoben. Mit `^b` (Control-b) und `^u` (control-u) wird die Bildschirmanzeige in Rückwärtsrichtung verschoben.

Das Kommando Control-f

Mit dem Kommando `^f` (Control-f) können Sie die Bildschirmanzeige um ein vollständiges Fenster in Vorwärtsrichtung verschieben. Dabei löscht `vi` den Bildschirm und baut ihn vollständig neu auf. Die beiden untersten Zeilen des ursprünglichen Fensters befinden sich im neuen Fenster ganz oben. Füllen die nachfolgenden Zeilen das neue Fenster nicht aus, so werden in den Leerzeilen Tilden (~) angezeigt.

Der Bildschirm wird von `vi` folgendermaßen gelöscht und wieder aufgebaut:



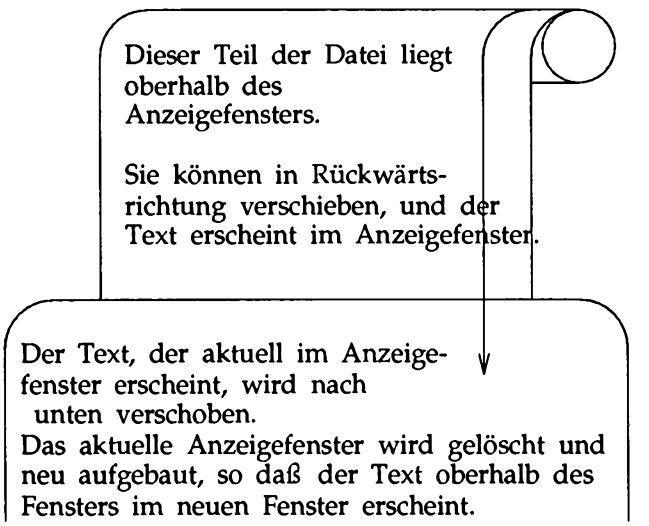
Das Kommando Control-d

Mit dem Kommando `^d` (Control-d) wird die Bildschirmanzeige um ein halbes Fenster in Vorwärtsrichtung verschoben. Bei der Eingabe von `^d` scheint die Bildschirmanzeige nur am oberen Bildschirmrand verschoben zu werden, nicht aber am unteren Rand. Dadurch können die Zeilen, die sich unter der aktuellen Bildschirmanzeige befinden, auf dem Bildschirm abgerufen werden, während die Zeilen am oberen Bildschirmrand verschwinden. Befindet sich der Cursor bei der Eingabe von `^d` in der letzten Zeile der Datei, ertönt ein akustisches Signal.

Das Kommando Control-b

Mit dem Kommando `^b` (Control-b) wird die Bildschirmanzeige um ein volles

Fenster in Rückwärtsrichtung verschoben; es wird also die vorhergehende Bildschirmseite angezeigt. Hierfür löscht `vi` den Bildschirm und baut ihn wieder neu auf; auf dem Bildschirm wird dann also der Text angezeigt, der sich ursprünglich oberhalb der aktuellen Bildschirmseite befunden hat. Anders als beim Kommando `^f` werden beim Kommando `^b` keine Zeilen aus der ursprünglichen Bildschirmanzeige übernommen. Füllen die Zeilen oberhalb der aktuellen Bildschirmanzeige kein vollständiges Fenster mehr aus, ertönt ein akustisches Signal; die aktuelle Bildschirmanzeige bleibt erhalten.



Versuchen Sie jetzt, die Bildschirmanzeige in Rückwärtsrichtung zu verschieben:

```
<^b>
```

`vi` löscht den Bildschirm und baut eine neue Bildschirmanzeige auf. Der Text in der ursprünglichen Bildschirmanzeige befindet sich jetzt unterhalb der neuen Bildschirmanzeige.

Das Kommando Control-u

Mit dem Kommando ^u (Control-u) wird die Bildschirmanzeige um eine halbe Bildschirmseite in Rückwärtsrichtung verschoben; dadurch werden jetzt die Zeilen angezeigt, die sich ursprünglich oberhalb der Bildschirmanzeige befunden haben. Die Zeilen am unteren Bildschirmrand werden gelöscht. Verschieben Sie die Datei jetzt in Rückwärtsrichtung, damit der Text oberhalb der aktuellen Bildschirmanzeige angezeigt wird:

```
<^u>
```

Sobald der Cursor am Dateianfang angelangt ist, ertönt ein akustisches Signal, da ein weiteres Verschieben in Rückwärtsrichtung nicht möglich ist.

In eine bestimmte Zeile springen

Mit dem Kommando G wird der Cursor in eine bestimmte Zeile auf dem Bildschirm bewegt; wird diese Zeile aktuell nicht dargestellt, löscht G den Bildschirm und baut die Bildschirmanzeige neu auf; die adressierte Zeile befindet sich dann in der Bildschirmmitte. Ohne die Angabe einer bestimmten Zeile bewegt das Kommando G den Cursor in die letzte Zeile der Datei.

G	Cursor in die letzte Zeile der Datei bewegen.
*n*G	Cursor in die *n*-te Zeile der Datei bewegen.

Zeilennummern

Die Zeilen in einer Datei sind, ihrer Position im Puffer entsprechend, durchnumeriert. Die Nummer einer bestimmten Zeile können Sie abrufen, indem Sie den Cursor in diese Zeile bewegen und das Kommando ^g eingeben. Dieses Kommando gibt am unteren Bildschirmrand eine Statusmeldung aus, die folgende Informationen enthält:

- Den Namen der Datei.
- Ob am Puffer seit der letzten Abspeicherung der Datei Änderungen vorgenommen worden sind.

- Die Nummer der Zeile, in der der Cursor sich aktuell befindet.
- Die gesamte Anzahl der Zeilen im Puffer.
- Die Position der aktuellen Zeile relativ zum Gesamtumfang des Puffer-Inhalts in Prozent.

```
This line is the 35th line of the buffer.
The cursor is on this line.
                  ↑


There are several more lines in the
buffer.
The last line of the buffer is line 116.
~
~
```

<^g>

```
This line is the 35th line of the buffer.
The cursor is on this line.
                  ↑


There are several more lines in the
buffer.
The last line of the buffer is line 116.
~
~
"file.name" [modified] line 36 of 116 --34%--
```

Suchen von Zeichenmustern: Die Kommandos / und ?

Am schnellsten gelangen Sie über eines der folgenden Such-Kommandos an eine bestimmte Stelle im Text: `/`, `?`, `n` oder `N`. Mit diesen Kommandos können Sie den Puffer in Vor- und Rückwärtsrichtung nach der nächsten Stelle durchsuchen, an der ein bestimmtes Zeichenmuster vorkommt. Die Kommandos / und ? erscheinen bei ihrer Eingabe im Gegensatz zu anderen `vi`-Kommandos auf dem Bildschirm. Mit den Kommandos `n` und `N` können Sie den letzten Suchvorgang wiederholen.

Mit / und einem Muster (*/muster*) können Sie im Puffer nach der nächsten Stelle suchen, an der die im Muster enthaltenen Zeichen enthalten sind; der Cursor wird zum ersten dieser Zeichen bewegt. So wird mit der Kommandozeile

```
/Hello world<CR>
```

nach der nächsten Stelle im Puffer gesucht, an der die Wörter `Hello world` vorkommen; der Cursor wird zum Buchstaben `H` bewegt.

Mit ? und einem Muster (*?muster*) wird der Puffer in Rückwärtsrichtung nach der ersten Stelle durchsucht, an der die Zeichen im Muster vorkommen; der Cursor wird zum ersten dieser Zeichen bewegt. So wird mit der Kommandozeile

```
?data set design<CR>
```

die erste Stelle im Puffer oberhalb der aktuellen Cursor-Position gesucht, an der die Wörter `data set design` vorkommen; der Cursor wird zum Anfangsbuchstaben `d` des Wortes `data` bewegt.

Handelt es sich bei den Suchmustern um zwei oder mehr Wörter, so müssen diese in derselben Zeile enthalten sein. Wenn Sie beispielsweise die Wörter `Hello world` suchen und das Wort `Hello` sich am Ende einer Zeile und das Wort `world` am Anfang der darauffolgenden Zeile befindet, so kann das Suchen-Kommando die Suchfolge `Hello world` nicht ausfindig machen.

Ende und Anfang des Puffers dagegen stellen keine Begrenzung dar - der Suchvorgang wird gegebenenfalls am Ende bzw. Anfang des Puffers fortgesetzt. Wenn Sie sich beispielsweise am Ende des Puffers befinden und das (mit dem Kommando */muster*) gesuchte Muster am Anfang des Puffers enthalten ist, so wird das Muster von dem Kommando gefunden.

Die Kommandos `n` und `N` ermöglichen Ihnen die Fortsetzung eines mit */muster* oder `?`*muster* angeforderten Suchvorgangs, ohne die Suchen-Kommandos erneut einzugeben.

`n`	Wiederholung des letzten Suchen-Kommandos.
`N`	Wiederholung des letzten Suchen-Kommandos, jedoch in der entgegengesetzten Richtung.

Wenn Sie die Datei beispielsweise in Rückwärtsrichtung nach dem dreistelligen Muster `the` durchsuchen möchten, starten Sie den Suchvorgang mit `?the` und setzen ihn dann mit `n` fort. Das folgende Beispiel zeigt die einzelnen Schritte eines mit `n` in Rückwärtsrichtung gestarteten Suchvorgangs, bei dem die Zeichenkette `the` vier mal gefunden wird.

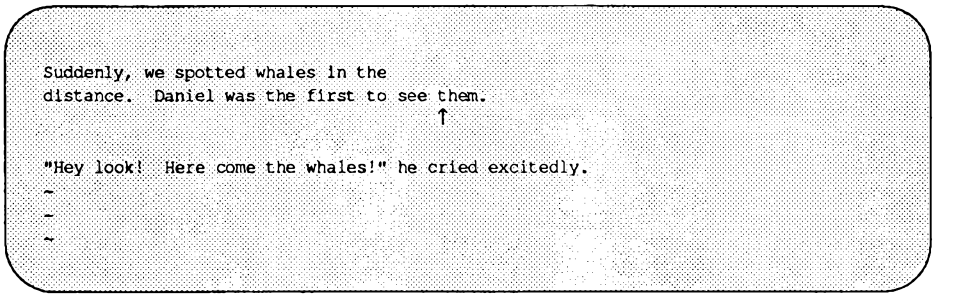
```
↓
Suddenly, we spotted whales in the
distance.  Daniel was the first to see them.

"Hey look!  Here come the whales!" he cried excitedly.
~
~
~
?the
```

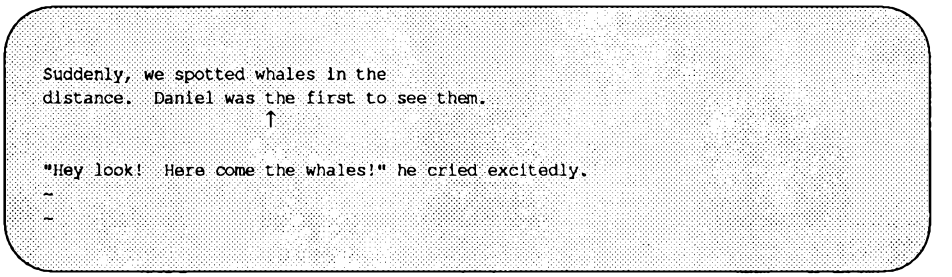
```
Suddenly, we spotted whales in the
distance.  Daniel was the first to see them.

"Hey look!  Here come the whales!" he cried excitedly.
                   ↑
~
~
~
```

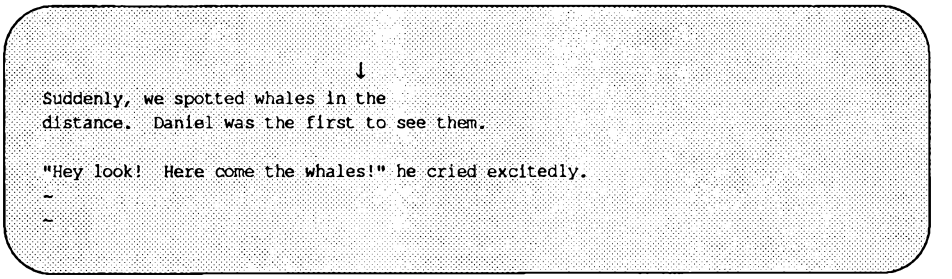
```
n
```



n



n



Bei den Suchen-Kommandos / und ? können Sie nicht angeben, die wievielte Stelle mit dem Suchmuster gesucht werden soll. D. h., Sie können

beispielsweise nicht die dritte Stelle (ab der aktuellen Position) suchen, an der ein Muster vorkommt.

Tabelle 7-3 enthält einen Überblick über die Möglichkeiten zur Cursor-Positionierung, die Ihnen unter `vi` zur Verfügung stehen (Durchblättern des Texts, Angabe einer Zeilennummer, Suchen eines Musters).

Tabelle 7-3: Weitere `vi`-Kommandos zur Cursor-Positionierung

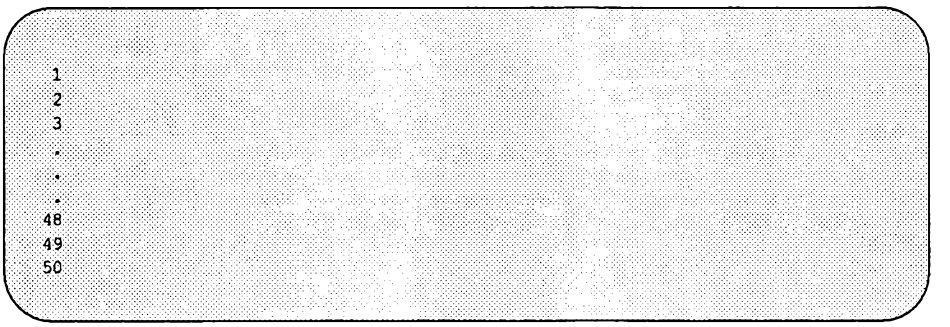
Durchblättern	
`^f`	Die Bildschirmanzeige wird um eine volle Bildschirmseite in Vorwärtsrichtung verschoben; dadurch wird der Text unterhalb der aktuellen Bildschirmanzeige dargestellt.
`^d`	Die Bildschirmanzeige wird um eine halbe Bildschirmseite in Vorwärtsrichtung verschoben; dadurch wird der Text unterhalb der aktuellen Bildschirmanzeige dargestellt.
`^b`	Die Bildschirmanzeige wird um eine volle Bildschirmseite in Rückwärtsrichtung verschoben; dadurch wird die Bildschirmseite vor der aktuellen Bildschirmseite angezeigt.
`^u`	Die Bildschirmanzeige wird um eine halbe Bildschirmseite in Rückwärtsrichtung verschoben; dadurch wird die Bildschirmseite vor der aktuellen Bildschirmseite angezeigt.
Zeilenadressierung über eine Zeilennummer	
`1G`	Der Cursor wird in die erste Zeile der Datei bewegt.
`G`	Der Cursor wird in die letzte Zeile der Datei bewegt.
`^g`	Die Zeilennummer sowie weitere Informationen über die Datei werden ausgegeben.

Tabelle 7-3: Weitere `vi`-Kommandos zur Cursor-Positionierung (Fortsetzung)

Suchen von Zeichenmustern	
/muster	Der Puffer wird in Vorwärtsrichtung nach der nächsten Stelle durchsucht, an der das Muster vorkommt. Der Cursor wird zum ersten Zeichen des Musters bewegt.
?*muster*	Der Puffer wird in Rückwärtsrichtung nach der ersten Stelle durchsucht, an der das Muster vorkommt. Der Cursor wird zum ersten Zeichen des Musters bewegt.
`n`	Das letzte Suchen-Kommando wird wiederholt.
`N`	Das letzte Suchen-Kommando wird in der entgegengesetzten Richtung wiederholt.

Übung 2

2-1. Legen Sie eine Datei namens `exer2` an. Numerieren Sie die Zeilen von 1 bis 50. Ihre Datei sollte in etwa folgendermaßen aussehen:



2-2. Führen Sie die folgenden Kommandos zum Durchblättern der Datei durch und beachten Sie, um wieviele Zeilen die Bildschirmanzeige jeweils verschoben wird:

```
^f
^b
^u
^d
```

2-3. Bewegen Sie den Cursor zum Dateiende. Fügen Sie die folgende Textzeile ein:

```
123456789 123456789
```

Zu welcher Nummer wird der Cursor beim Kommando `7h` bewegt? Zu welcher wird der Cursor beim Kommando `31` bewegt?

2-4. Geben Sie probehalber die Kommandos `$` und `0` (Null) ein.

2-5. Bewegen Sie den Cursor zum ersten Zeichen in der Zeile, bei dem es sich nicht um ein Leerzeichen handelt. Bewegen Sie den Cursor zum ersten Zeichen im nächsten Wort. Bewegen Sie den Cursor zum ersten Zeichen des vorhergehenden Worts. Bewegen Sie den Cursor zum Ende des Worts.

2-6. Bewegen Sie den Cursor in die erste Zeile der Datei. Rufen Sie die Kommandos auf, mit denen der Cursor in die Bildschirmmitte sowie die erste und letzte Zeile des Bildschirms bewegt wird.

2-7. Suchen Sie nach der Zahl 8. Suchen Sie dann nach der nächsten Stelle, an der die Zahl 8 vorkommt (Ergebnis: 48).

Erstellen von Text

Zum Erstellen von Text gibt es drei grundlegende Kommandos:

a	(append) Text anfügen
i	(insert) Text einfügen
o	(open) Leerzeile erzeugen, in der Text eingetragen werden kann

Nach der Beendigung der Texteingsabe mit einem dieser Kommandos können Sie über die Taste ESCAPE wieder in den vi-Kommandomodus zurückschalten.

Anfügen von Text

a	Hinter der aktuellen Cursor-Position wird Text angefügt.
A	Am Ende der aktuellen Zeile wird Text angefügt.

Das Kommando a haben Sie bereits im Abschnitt "Anlegen einer Datei" kennengelernt. Legen Sie eine neue Datei namens junk2 an. Tragen Sie mit dem Kommando a Text ein. Schalten Sie dann über die Taste ESCAPE in den vi-Kommandomodus zurück. Rufen Sie dann das Kommando A auf, und vergleichen Sie das Ergebnis beider Kommandos.

Einfügen von Text

i	Vor der aktuellen Cursor-Position wird Text eingefügt.
I	Am Anfang der aktuellen Zeile wird, vor dem ersten Zeichen, bei dem es sich nicht um ein Leerzeichen handelt, Text eingefügt.

Schalten Sie dann über die Taste ESCAPE in den vi-Kommandomodus zurück.

Die folgenden Beispiele veranschaulichen die unterschiedliche Funktionsweise der Kommandos a und i. Die aktuelle Cursor-Position wird durch Pfeile angezeigt; an dieser Stelle wird der neue Text eingefügt.

```
Append three spaces AFTER the H of Here.
                                  ↑
~
~
~
```

a <ESC>

```
Append three spaces AFTER the H of H   ere.
                                    ↑
~
~
~
```

```
Insert three spaces BEFORE the H of Here.
                                  ↑
```

i <ESC>

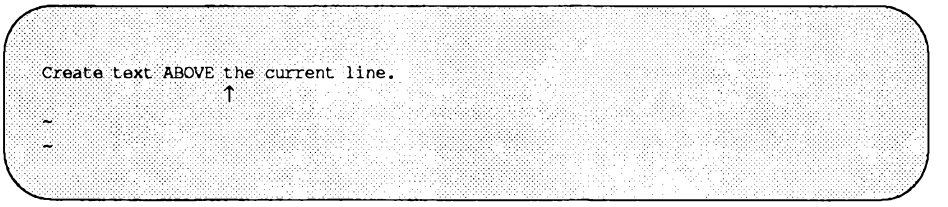
```
Insert three spaces BEFORE the H of   Here.
                                  ↑
~
~
~
```

Beachten Sie, daß in beiden Fällen der `vi`- Eingabemodus über die Taste `ESCAPE` verlassen worden ist.

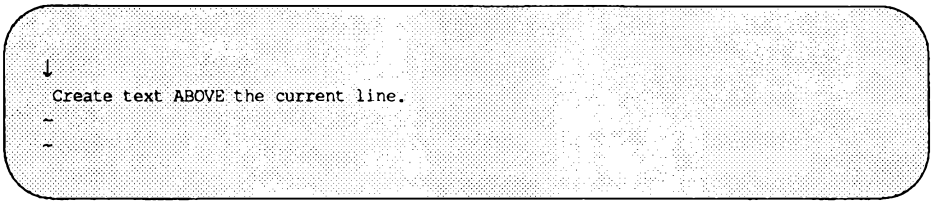
Einfügen einer Leerzeile

- o — Unterhalb der aktuellen Zeile wird eine neue Zeile eingefügt, in die Sie Text eingeben können. Dieses Kommando können Sie an einer beliebigen Stelle der aktuellen Zeile aufrufen.
- O — Oberhalb der aktuellen Zeile wird eine neue Zeile eingefügt, in die Sie Text eingeben können. Dieses Kommando können Sie ebenfalls an einer beliebigen Stelle der aktuellen Zeile aufrufen.

Mit den Kommandos o und O wird ober- oder unterhalb der aktuellen Zeile eine Leerzeile eingefügt; gleichzeitig wird der vi-Eingabemodus eingeschaltet. In den folgenden Beispielen wird mit dem Kommando O über der aktuellen Zeile eine Leerzeile eingefügt, mit dem Kommando o unter der aktuellen Zeile. In beiden Fällen wird der Cursor zum Anfang der neuen Zeile bewegt, so daß Sie sofort mit der Eingabe des Texts beginnen können.



O



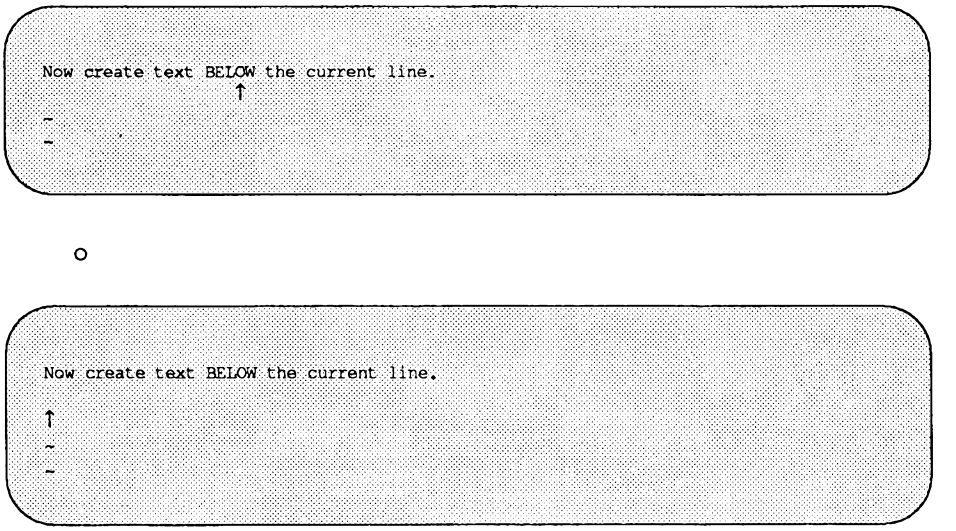


Tabelle 7-4 enthält einen Überblick über die `vi`-Kommandos, mit denen eine Datei erstellt und ergänzt werden kann.

Tabelle 7-4: Die `vi`-Kommandos zum Erstellen von Text

Kommando	Funktion
`a`	Hinter der aktuellen Cursor-Position wird Text eingegeben.
`A`	Am Ende der aktuellen Zeile wird Text eingegeben.
`i`	Vor der aktuellen Cursor-Position wird Text eingegeben.
`I`	Vor dem ersten Zeichen der aktuellen Zeile, bei dem es sich nicht um ein Leerzeichen handelt, wird Text eingegeben.
`o`	Am Anfang einer neuen Zeile, die unterhalb der aktuellen Zeile eingefügt wird, wird Text eingegeben.
`O`	Am Anfang einer neuen Zeile, die oberhalb der aktuellen Zeile eingefügt wird, wird Text eingegeben.
`<ESC>`	Einer der obengenannten Eingabemodi wird aus- und der `vi`-Kommandomodus wieder eingeschaltet.

Übung 3

3-1. Legen Sie eine Datei namens `exer3` an.

3-2. Tragen Sie die folgenden vier Textzeilen ein:.

```
Append text
Insert text
a computer's
job is boring.
```

3-3. Tragen Sie über der letzten Zeile den folgenden Text ein:

```
financial statement and
```

3-4. Fügen Sie über der dritten Zeile die folgende Textzeile ein (Kommando `i`):

```
Delete text
```

3-5. Tragen Sie unter der aktuellen Zeile die folgende Textzeile ein:

```
byte of the budget
```

3-6. Tragen Sie unter der letzten Zeile die folgende Textzeile ein (Kommando `a`):

```
But, it is an exciting machine.
```

3-7. Bewegen Sie den Cursor in die erste Zeile und fügen Sie vor dem Wort text das Wort `some` ein.

Üben Sie jetzt den Umgang mit den sechs Kommandos anhand eigener Beispiele.

3-8. Beenden Sie `vi`. Im nächsten Abschnitt erfahren Sie, wie Sie Fehler bei der Texteingabe korrigieren können.

Löschen von Text

Zum Löschen von Text stehen Ihnen eine Reihe von Kommandos zur Verfügung, die im Kommandomodus aufgerufen werden; im Eingabemodus können Sie den Eintrag von kleineren Textmengen wieder rückgängig machen. Außerdem können Sie die Auswirkungen Ihres zuletzt aufgerufenen Kommandos wieder vollständig aufheben.

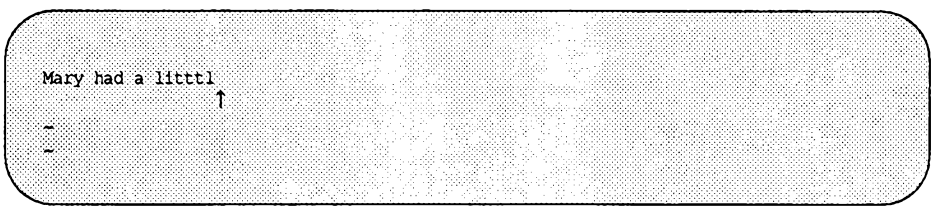
Löschen von Text, der im Eingabemodus eingegeben worden ist

Im Eingabemodus können Sie ein Zeichen jederzeit über die Taste BACKSPACE löschen.

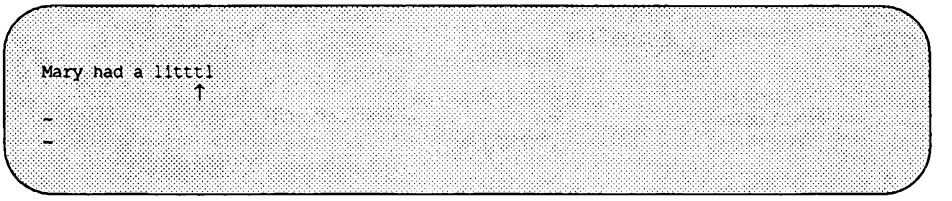
<BACKSPACE>	Das zuletzt eingegebene Zeichen (links neben der aktuellen Cursor-Position) wird gelöscht.

Wenn Sie die Taste BACKSPACE im Eingabemodus betätigen, wird der Cursor zeichenweise zurückbewegt; jedes Zeichen, zu dem der Cursor bewegt wird, wird gelöscht. Allerdings sind die gelöschten Zeichen noch auf dem Bildschirm sichtbar, bis Sie sie überschreiben oder die Taste ESCAPE betätigen, um in den Kommandomodus zurückzukehren.

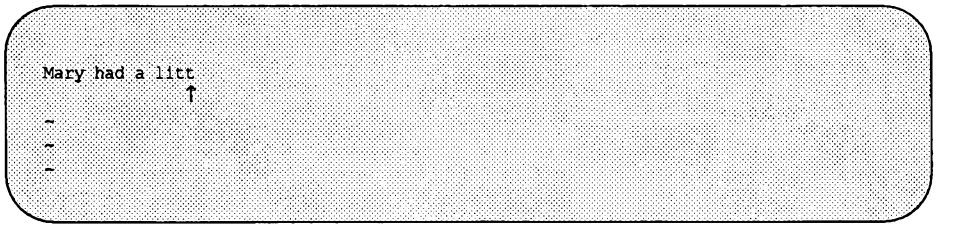
In den folgenden Beispielen wird die aktuelle Cursor-Position jeweils durch einen Pfeil angezeigt.



<BACKSPACE> <BACKSPACE>



<ESC>



Beachten Sie, daß die Zeichen erst dann vom Bildschirm gelöscht werden, wenn Sie die Taste ESCAPE betätigen.

Das Löschen von Text im Eingabemodus kann noch über zwei weitere Tasten vorgenommen werden. Diese Möglichkeit wird zwar nicht sehr häufig verwendet, wird hier aber der Vollständigkeit halber angegeben. Wenn die spezielle Belegung dieser Tasten aufgehoben werden soll, so daß die entsprechenden Buchstaben in den Text eingegeben werden können, müssen Sie vor der Betätigung dieser Tasten ^v eingeben.

^w	Die Eingabe des aktuellen Worts wird rückgängig gemacht
@	Der gesamte Text, der seit dem Aufrufen des Eingabemodus in der aktuellen Zeile eingegeben worden ist, wird gelöscht
^v	Die spezielle Bedeutung des nachfolgend eingegebenen Zeichens wird gegebenenfalls aufgehoben

Bei der Eingabe von ^w wird der Cursor zum ersten Zeichen des zuletzt eingegebenen Worts zurückbewegt. Das Wort wird erst dann vom Bildschirm gelöscht, wenn Sie die Taste ESCAPE betätigen oder neue Zeichen eingeben, durch die die ursprünglichen Zeichen überschrieben werden. Das Zeichen @ hat eine ähnliche Funktion, nur wird damit sämtlicher Text gelöscht, den Sie seit dem Aufrufen des Eingabemodus in der aktuellen Zeile eingegeben haben.

Aufheben des letzten Kommandos

Vor dem Löschen-Kommando sollten Sie zunächst das Kommando `u` ausprobieren, mit dem Sie das zuletzt aufgerufene Kommando rückgängig machen können.

`u`	Das letzte Kommando wird aufgehoben.
`U`	Die aktuelle Zeile wird in den Zustand zurückgesetzt, in dem sie sich vor der letzten Änderung befunden hat.

Wenn Sie versehentlich ein oder mehrere Zeilen gelöscht haben und Sie das Kommando `u` eingeben, werden Ihre Zeilen wieder auf dem Bildschrim angezeigt. Ebenso können Sie die Auswirkung anderer versehentlich aufgerufener Kommandos über das Kommando `u` rückgängig machen. Mit dem Kommando `U` werden alle Änderungen, die Sie an der aktuellen Zeile vorgenommen haben, wieder aufgehoben - vorausgesetzt, der Cursor ist zwischendurch nicht in eine andere Zeile bewegt worden.

Wenn Sie das Kommando `u` zweimal hintereinander eingeben, wird das erste Kommando durch das zweite aufgehoben - Sie machen also das Kommando selbst rückgängig! Wenn Sie beispielsweise versehentlich eine Zeile gelöscht und sie über das Kommando `u` wiederhergestellt haben, so wird die Zeile bei der zweiten Eingabe von `u` erneut gelöscht. Mit diesem Kommando können Sie sich also sehr viel Zeit und Arbeit ersparen.

Löschen von Text im Kommandomodus

Wie bereits erwähnt, können Sie einem Kommando eine Zahl voranstellen. Bei vielen `vi`- Kommandos (z.B. zum Löschen oder Überschreiben von Text) können Sie zusätzlich ein nachgestelltes Positionierungs-Kommando eingeben. Das Positionierungs-Kommando kann ein Textobjekt wie z.B. ein Wort, eine Zeile, ein Satz oder ein Absatz sein. Ein derartiges `vi`-Kommando hat folgendes allgemeines Format:

[anzahl][kommando]text_objekt

Die optionalen Komponenten sind in diesem Kommandoformat in eckigen Klammern eingeschlossen.

Mit allen Löschen-Kommandos, die im Kommandomodus aufgerufen werden, wird unerwünschter Text vom Bildschirm entfernt; auf den meisten Terminals wird außerdem der betreffende Teil des Bildschirms neu aufgebaut.

Das Löschen-Kommando hat das allgemeine Format eines `vi`-Kommandos:

[anzahl]`d`*text_objekt*

Löschen von Wörtern

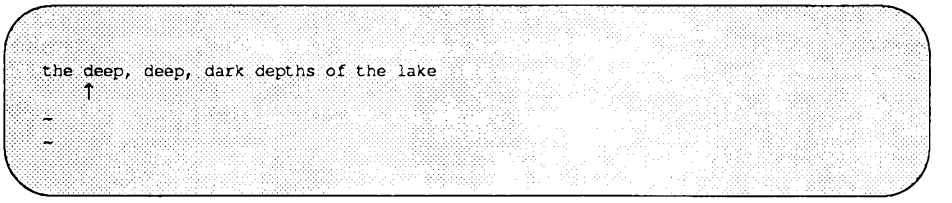
Zum vollständigen oder teilweisen Löschen eines Worts geben Sie das Kommando `d` ein. Bewegen Sie den Cursor zum ersten zu löschenden Zeichens und geben Sie `dw` ein. Das Zeichen an der aktuellen Cursor-Position sowie alle nachfolgenden Zeichen werden gelöscht.

```
the deep dark depths of the lake.
    ↑
~
~
```

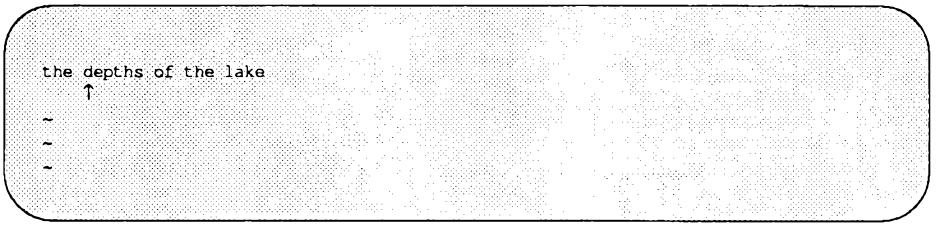
```
2dw
```

```
the depths of the lake.
    ↑
~
~
```

Mit dem Kommando `dw` können Sie ein Wort bzw. Satzzeichen samt den nachfolgenden Leerzeichen löschen. Wenn Sie mehrere Wörter oder Satzzeichen auf einmal löschen möchten, so stellen Sie dem Kommando eine Zahl voran. Wenn Sie beispielsweise Sie drei Wörter und zwei Kommata löschen möchten, geben Sie das Kommando `5dw` ein:




```
5dw
```



Löschen von Absätzen

Absätze können Sie mit den folgenden Kommandos löschen:

```
d{ oder d}
```

Beachten Sie stets, daß Sie gelöschten Text mit dem Kommando `u` wiederherstellen können.

Löschen von Zeilen

Zum Löschen einer Zeile geben Sie das Kommando `dd` ein. Mehrere Zeilen können Sie löschen, indem Sie dem Kommando eine Zahl voranstellen. So können Sie beispielsweise mit dem Kommando

```
10dd
```

zehn Zeilen löschen. Wenn Sie eine größere Anzahl von Zeilen löschen, stellt `vi` am unteren Bildschirmrand die folgende Meldung dar:

```
10 lines deleted
```

Wenn bis zum Dateiende weniger als 10 Zeilen vorhanden sind, ertönt ein akustisches Signal; in diesem Fall werden keine Zeilen gelöscht.

Löschen von Text hinter der aktuellen Cursor-Position

Um den gesamten Text zwischen der aktuellen Cursor-Position und dem Zeilenende zu löschen, bewegen Sie den Cursor zum ersten zu löschenden Zeichen und geben folgendes ein:

`D` oder `d$`.

Bei keinem dieser Kommandos können Sie mehrere Zeilen angeben; diese Kommandos sind nur für die aktuelle Zeile gültig.

Tabelle 7-5 enthält einen Überblick über die `vi`-Kommandos zum Löschen von Text.

Tabelle 7-5: Kommandos zum Löschen von Text

Kommando	Funktion
Im Eingabemodus:	
<BACKSPACE>	Das Zeichen an der aktuellen Cursor-Position wird gelöscht.
<^w>	Das aktuelle Wort wird gelöscht.
@	Die aktuelle Textzeile bzw. sämtlicher Text, der neu in die aktuelle Zeile eingegeben worden ist, wird gelöscht.
Im Kommandomodus:	
u	Das letzte Kommando wird rückgängig gemacht.
U	Der ursprüngliche Inhalt der aktuellen Zeile wird wiederhergestellt.
x	Das aktuelle Zeichen wird gelöscht.
*n*d*x*	*n* Textobjekte vom Typ *x* werden gelöscht.
dw	Das Wort zwischen der aktuellen Cursor-Position und dem nächsten Leerzeichen bzw. Satzzeichen wird gelöscht.
dW	Das Wort bzw. Satzzeichen zwischen der aktuellen Cursor-Position und dem nächsten Leerzeichen wird gelöscht.
dd	Die aktuelle Zeile wird gelöscht.
D	Der Text zwischen der aktuellen Cursor-Position und dem Zeilenende wird gelöscht.
d)	Der Text zwischen der aktuellen Cursor-Position und dem Ende des aktuellen Satzes wird gelöscht.
d}	Der Text zwischen der aktuellen Cursor-Position und dem Ende des aktuellen Absatzes wird gelöscht.

Übung 4

4-1. Legen Sie eine Datei namens `exer4` an und geben Sie die folgenden vier Textzeilen ein:

```
When in the course of human events
there are many repetitive, boring
chores, then one ought to get a
robot to perform those chores.
```

4-2. Bewegen Sie den Cursor in Zeilen 2 und fügen Sie hier folgenden Text an:

```
, tedious, and unsavory
```

Löschen Sie im Anfügen-Modus das Wort unsavory.

Löschen Sie das Wort boring im Kommandomodus.

Gibt es noch eine andere Möglichkeit zum Löschen des Worts boring?

4-3. Geben Sie am Anfang von Zeile 4 folgendes ein:

```
congenial and computerized
```

Löschen Sie die Zeile.

Gibt es eine Möglichkeit, den Inhalt der Zeile zu löschen, ohne die Zeile selbst zu entfernen?

Löschen Sie sämtliche Zeilen mit einem einzigen Kommando.

4-4. Beenden Sie den Bildschirmeditor und löschen Sie die leere Datei aus Ihrem Verzeichnis.

Ändern von Text

Eine Möglichkeit zum Ändern von Text haben Sie mit den Löschen-Kommandos sowie mit den Kommandos, die Sie im Eingabemodus durchführen können. Bei einer weiteren Möglichkeit können Sie ein Kommando aufrufen, mit dem Sie Zeichen überschreiben, d.h. Text löschen und gleichzeitig neuen Text eingeben können. Zum Überschreiben von Text gibt es drei grundlegende Kommandos: `r` (für replace), `s` (für substitute) und `c` (für change).

Ersetzen von Text (Kommando r)

`r`*x*	Das aktuelle Zeichen (an der aktuellen Cursor-Position) wird durch *x* ersetzt. Da durch dieses Kommando nicht der Eingabemodus aufgerufen wird, muß danach nicht die Taste `ESCAPE` betätigt werden.
n`r`*x*	*n* Zeichen werden durch *x* ersetzt. Dieses Kommando endet automatisch mit dem Ersetzen des *n-ten* Zeichens. Auch nach diesem Kommando müssen Sie nicht die Taste `ESCAPE` betätigen.
`R`	Nur diejenigen Zeichen, die vor der Betätigung der Taste `ESCAPE` eingegeben worden sind, werden überschrieben. Passen die neuen Zeichen nicht mehr in die aktuelle Zeile, so werden sie durch das Kommando als neuer Text angefügt.

Das Kommando `r` ersetzt das aktuelle Zeichen durch das nächste eingegebene Zeichen. Angenommen, im folgenden Satz soll das Wort acts gegen ants ausgetauscht werden:

```
The circus has many acts.
```

Bewegen Sie den Cursor zum Zeichen `c` im Wort `acts` und geben Sie folgendes ein:

```
rn
```

Der Satz sieht schließlich folgendermaßen aus:

```
The circus has many ants.
```

Um das Wort `many` durch `7777` zu ersetzen, bewegen Sie den Cursor zum

Zeichen m im Wort many und geben folgendes ein:

```
4r7
```

Das Kommando r ersetzt die vier Buchstaben im Wort many jeweils durch die Zahl sieben:

```
The circus has 7777 ants.
```

Ersetzen von Text (Kommando s)

Mit einem weiteren Kommando können Sie ebenfalls Text ersetzen; nur können Sie dann mit der Texteingabe fortfahren, bis Sie die Taste ESCAPE betätigen.

s	Das Zeichen an der Cursor-Position wird gelöscht, und Text wird amgefügt. Der Eingabemodus kann über die Taste ESCAPE beendet werden.
*n*s	*n* Zeichen werden gelöscht, und Text wird angefügt. Der Eingabemodus kann über die Taste ESCAPE beendet werden.
S	Alle Zeichen in der Zeile werden ausgetauscht.

Bei der Eingabe des Kommandos s wird das letzte Zeichen in der zu ersetzenden Zeichenkette durch das Zeichen $ ersetzt. Die Zeichen werden erst dann vom Bildschirm gelöscht, wenn sie von Ihnen überschrieben werden oder Sie den Eingabemodus über die Taste ESCAPE verlassen.

Beachten Sie, daß Sie weder bei r noch bei s ein Textobjekt als Argument eingeben können.

Wenn Sie in dem Satz My salary is one hundred dollars das Wort hundred durch million austauschen möchten, bewegen Sie den Cursor zum Zeichen h im Wort hundred und geben 7s ein. Beachten Sie die Stelle, an der das Symbol $ eingesetzt wird:

```
My salary is one hundred dollars.
                ↑
~
~
~
```

```
7s million<ESC>
```

```
My salary is one million dollars.
                ↑
~
~
~
```

Überschreiben von Text (Kommando c)

Im Gegensatz zum Kommando `s`, mit dem nur Zeichen überschrieben ersetzt, können Sie mit dem Kommando `c` (für change) Textobjekte überschreiben und dann mit der Texteingabe fortfahren, bis Sie die Taste `ESCAPE` betätigen. Zum Abschluß des Überschreiben-Kommandos betätigen Sie die Taste `ESCAPE`.

Als Option kann beim Überschreiben-Kommando ein Textobjekt angegeben werden. Dies gibt Ihnen die Möglichkeit, ein Zeichen, ein Wort, eine ganze Zeile oder sogar den ganzen Text zu überschreiben.

ncx	*n* Textobjekte vom Typ *x*, wie z.B. Sätze (angegeben durch `)` und Absätze (angegeben durch `}`) werden überschrieben.
`cw`	Ein Wort bzw. die restlichen Zeichen eines Worts werden durch neuen Text überschrieben. Der Editor `vi` gibt das Zeichen `$` aus, um das letzte zu überschreibende Zeichen anzuzeigen.

*n*cw	*n* Wörter werden überschrieben.
cc	Alle Zeichen in der Zeile werden überschrieben.
*n*cc	Alle Zeichen in der aktuellen Zeile und bis zu *n* Textzeilen werden überschrieben.
C	Die Zeichen zwischen der aktuellen Cursor-Position und dem Zeilenende werden überschrieben.
*n*C	Die Zeichen zwischen der aktuellen Cursor-Position und dem Zeilenende sowie sämtliche Zeilen unter der aktuellen Zeile (maximal *n* Zeilen) werden überschrieben.

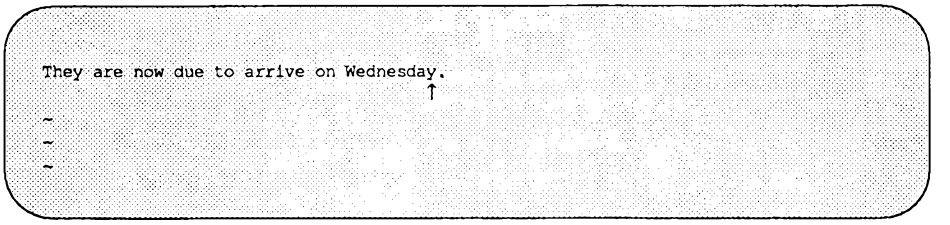
Beim Kommando c wird das letzte zu überschreibende Zeichen durch das Dollar-Zeichen ($) markiert. Dies funktioniert folgendermaßen:

```
They are now due to arrive on Tuesday.
                              ↑
~
~
```

```
cw
```

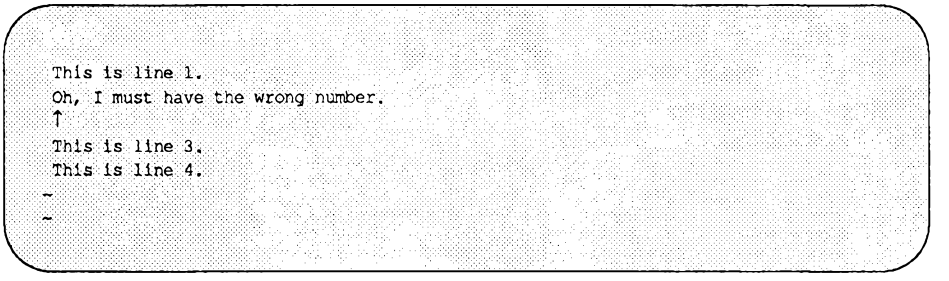
```
They are now due to arrive on Tuesda$.
                              ↑
~
~
```

```
Wednesday<ESC>
```

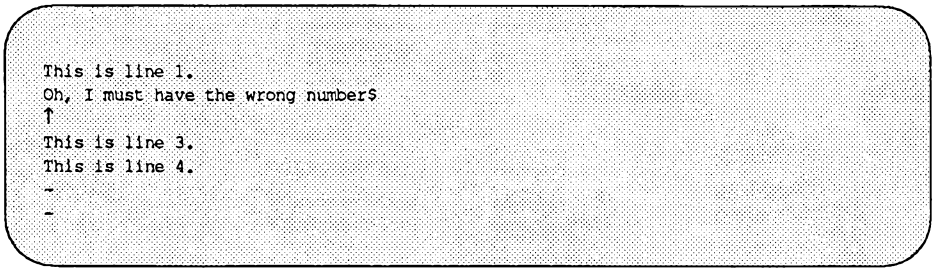


Beachten Sie, daß das neue Wort (`Wednesday`) aus mehr Buchstaben besteht als das zu überschreibende Wort (`Tuesday`). Nach beendeter Durchführung des Überschreiben- Kommandos werden Sie automatisch wieder in den Eingabemodus zurückgeschaltet; Sie können dann beliebig Text eingeben. Sobald Sie keinen Text mehr eingeben möchten, betätigen Sie die Taste `ESCAPE`.

Das Kommando `C` funktioniert ähnlich, nur wird damit der Rest der aktuellen Zeile überschrieben. Bei der Eingabe des Kommandos markiert das Kommando das Ende des zu überschreibenden Texts mit dem Dollar-Symbol (`$`). Dann werden Sie in den Eingabemodus zurückgeschaltet, in dem Sie den alten Text durch den neuen überschreiben können. Nachfolgend finden Sie einige Anwendungsbeispiele für das Kommando `C`.



C




```
This is line 2.<ESC>
```

Die Argumente können Sie auch miteinander kombinieren. Geben Sie beispielsweise folgendes ein:

```
c{
```

Probieren Sie ruhig verschiedene Argument-Kombinationen oder Kombinationen von Zahl und Kommando aus - schließlich können Sie alles problemlos mit dem Kommando `u` wieder rückgängig machen. Vor dem Aufrufen des Kommandos `u` müssen Sie die Taste `ESCAPE` betätigen, da das Kommando `c` den Eingabemodus einschaltet.

Vergleichen Sie die Funktion der Kommandos `S` und `cc`. Beide führen zu demselben Ergebnis.

Tabelle 7-6 enthält einen Überblick über die `vi`-Kommandos zum Überschreiben von Text.

Tabelle 7-6: `vi`-Kommandos zum Überschreiben von Text

Kommando	Funktion
`r`	Das aktuelle Zeichen wird ersetzt.
`R`	Nur diejenigen Zeichen werden überschrieben, die vor der Betätigung der Taste `ESCAPE` eingegeben worden sind.
`s`	Das Zeichen an der aktuellen Cursor-Position wird gelöscht, und Text wird angefügt. Zur Beendigung des Eingabemodus betätigen Sie die Taste `ESCAPE`.
`S`	Sämtliche Zeichen in der Zeile werden ersetzt.
`cc`	Sämtliche Zeichen in der Zeile werden überschrieben.
n`cx`	*n* Textobjekte vom Typ *x* werden überschrieben, z.B. Sätze (angegeben durch `)`) und Absätze (angegeben durch `}`).
`cw`	Ein Wort bzw. die restlichen Zeichen eines Worts werden überschrieben.
`C`	Die Zeichen zwischen der aktuellen Cursor-Position und dem Zeilenende werden überschrieben.

Text ausschneiden und einfügen

Unter `vi` gibt es eine Reihe von weiteren Kommandos, mit denen Text aus einer Datei ausgeschnitten und wieder in eine Datei eingefügt werden kann. Mit weiteren Kommandos kann von einem Textabschnitt eine Kopie erstellt und an einer anderen Stelle der Datei eingefügt werden.

Versetzen von Text

Sie können Text innerhalb des `vi`-Puffers an eine andere Stelle versetzen, indem Sie die Zeilen löschen und dann an der neuen Stelle einfügen. Der zuletzt gelöschte Text wird in einen temporären Puffer eingebracht. Sobald Sie den Cursor an die Stelle bewegen, an der die gelöschten Zeilen eingefügt werden sollen, und die Taste `p` betätigen, werden die gelöschten Zeilen unterhalb der aktuellen Zeile eingefügt.

`p`	Der Inhalt des temporären Puffers wird hinter der aktuellen Cursor-Position bzw. unterhalb der aktuellen Zeile eingefügt.

Wurde mit dem Kommando `D` ein Teil einer Zeile gelöscht, so kann er in einer beliebigen anderen Zeile eingefügt werden. Hierzu bewegen Sie den Cursor zum Leerzeichen zwischen zwei Wörtern und betätigen die Taste `p`. Der Zeilenausschnitt wird hinter der aktuellen Cursor-Position eingefügt.

Zeichen, die mit dem Kommando *n*`x` gelöscht worden sind, werden ebenfalls in einen temporären Puffer gesetzt. Sie können jedes beliebige Textobjekt, das Sie zuletzt gelöscht haben, mit dem Kommando `p` an einer beliebigen Stelle der Datei einfügen.

Das Kommando `p` muß unmittelbar nach dem Löschen der betreffenden Textstelle aufgerufen werden, da der temporäre Puffer nur das Ergebnis eines einzigen Kommandos speichern kann. Mit dem Kommando `p` können Sie auch die Kopie eines Textblocks, dem Sie mit dem Kommando `y` (für yank) in den temporären Puffer gesetzt haben, an einer anderen Stelle der Datei einfügen (das Kommando `y` wird im Abschnitt "Kopieren von Text" beschrieben).

Korrigieren von vertauschten Zeichen

Eine schnelle Möglichkeit, vertauschte Buchstaben in die korrekte Reihenfolge zu bringen, ist die Kombination der Kommandos `x` und `p` zum Kommando `xp`. Das Kommando `x` löscht den Buchstaben, und `p` fügt ihn hinter dem nächsten Zeichen ein.

So kann der Fehler in der Zeile

```
A line of tetx
```

schnell korrigiert werden, indem der Cursor zum Buchstaben `t` in `tx` bewegt wird und dann hintereinander die Tasten `x` und `p` betätigt werden (achten Sie auf die korrekte Reihenfolge!). Die Textzeile sieht danach folgendermaßen aus:

```
A line of text
```

Fügen Sie in Ihre Textzeile absichtlich einen Fehler ein und korrigieren Sie ihn mit dem Kommando `xp`. Warum funktioniert dieses Kommando?

Kopieren von Text

Unter `vi` haben Sie die Möglichkeit, eine oder mehrere Textzeilen mit dem Kommando `y` (für yank) in einen temporären Puffer zu sichern (kopieren) und die Kopie dieses Textblocks dann an einer beliebigen Stelle der Datei einzufügen. Um die Kopie an der neuen Stelle einzufügen, müssen Sie lediglich das Kommando `p` aufrufen; der Text erscheint dann in der darauffolgenden Zeile.

Das Kommandoformat von `y` entspricht dem allgemeinen `vi`-Kommandoformat.

[*anzahl*]y[*text_objekt*]

Beim Kommando `y` werden die Zeilen an ihrer ursprünglichen Position in der Datei nicht gelöscht. Vielmehr bietet es Ihnen eine bequeme Möglichkeit, einen bestimmten Textblock an mehreren Stellen einer Datei einzufügen, ohne ihn mehrmals einzugeben. Soll der Text jedoch nur an einer einzigen Stelle vorkommen, so vergessen Sie nicht, ihn an der ursprünglichen Stelle zu löschen, nachdem Sie ihn an der neuen Stelle eingefügt haben.

Tabelle 7-7 enthält einen Überblick über die Anwendungsmöglichkeiten des Kommandos y

Tabelle 7-7: Das Kommando y

Kommando	Funktion
*n*y*x*	*n* Textobjekte vom Typ *x* werden in einen temporären Puffer gesichert (z.B. Sätze) und Absätze }).
yw	Die Kopie eines Worts wird im temporären Puffer gesichert.
yy	Die Kopie der aktuellen Zeile wird im temporären Puffer gesichert.
*n*yy	*n* Zeilen werden im temporären Puffer gesichert.
y)	Alle Zeichen bis zum Ende des Satzes werden im temporären Puffer gesichert.
y}	Alle Zeichen bis zum Ende des Absatzes werden im temporären Puffer gesichert.

Bei diesem Kommando können Sie die Anzahl der zu sichernden Textobjekte angeben.

Geben Sie probehalber die folgenden Kommandozeilen ein und verfolgen Sie die Entwicklung der Bildschirmausgabe. Denken Sie stets daran, daß Sie das letzte Kommando wieder rückgängig machen können. Geben Sie folgendes ein

```
5yw
```

Bewegen Sie den Cursor an eine andere Stelle und geben Sie folgendes ein:

```
p
```

Versuchen Sie, einen Absatz (y}) in einem temporären Puffer zu sichern und seine Kopie hinter dem aktuellen Absatz einzufügen. Bewegen Sie den Cursor

dann an das Ende der Datei (G) und fügen Sie die Kopie des Absatzes am Ende der Datei ein.

Text über Register kopieren bzw. versetzen

Das Versetzen bzw. Kopieren von mehreren Textabschnitten ist eine sehr mühsame Angelegenheit. Unter `vi` können Sie dies einfacher und schneller über sogenannte benannte Register erledigen. Ein benanntes Register können Sie zur Aufbewahrung von Text benutzen, bis Sie ihn in die Datei einfügen möchten. Den aufzubewahrenden Text können Sie entweder im Puffer sichern (Kommando `y`) oder an der ursprünglichen Stelle löschen.

Das Arbeiten mit Registern ist immer dann empfehlenswert, wenn ein Textblock an mehreren Stellen der Datei vorkommen soll. Der extrahierte Text bleibt im angegebenen Register erhalten, bis Sie entweder die Editor-Sitzung beenden, oder einen weiteren Textblock (über das Kommando `y` oder `d`) in dieses Register einbringen.

Das Kommando hat folgendes allgemeines Format:

[anzahl]["x]kommando[text_objekt]

x ist der Name des Registers; hierbei kann es sich um einen beliebigen einzelnen Kleinbuchstaben handeln. Dem Namen des Registers muß ein Anführungszeichen (") vorangehen. Bewegen Sie den Cursor beispielsweise zum Anfang einer Zeile und geben Sie folgendes ein:

```
3"ayy
```

Geben Sie weiteren Text ein und bewegen Sie den Cursor dann zum Ende der Datei. Geben Sie folgendes ein:

```
"ap
```

Sind die Zeilen, die Sie im Register `a` gesichert haben, am Ende der Datei eingefügt worden?

Tabelle 7-8 enthält einen Überblick über die Kommandos zum Ausschneiden und Einfügen von Text.

Tabelle 7-8: `vi`-Kommandos zum Ausschneiden und Einfügen von Text

Kommando	Funktion
`p`	Der Inhalt des temporären Puffers (der zuletzt gelöschte bzw. gesicherte Text) wird hinter der aktuellen Cursor-Position eingefügt.
`yy`	Eine Textzeile wird in einem temporären Puffer gesichert.
n`y`*x*	Eine Kopie von *n* Textobjekten vom Typ *x* wird in einen temporären Puffer gesetzt.
`"xy`*n*	Die Kopie eines Textobjekts vom Typ *n* wird in das Register mit dem Namen x gesetzt.
`"xp`	Der Inhalt des Registers x wird hinter der aktuellen Cursor-Position eingefügt.

Übung 5

5-1. Rufen Sie `vi` mit der Datei `exer2` auf (diese Datei haben Sie in Übung 2 erstellt).

Bewegen Sie den Cursor zu Zeile 8 und ändern Sie ihren Inhalt in `END OF FILE` ab.

5-2. Sichern Sie die ersten acht Zeilen der Datei im Register `z`. Fügen Sie den Inhalt des Registers `z` hinter der letzten Textzeile ein.

5-3. Bewegen Sie den Cursor zu Zeile 8 und ändern Sie ihren Inhalt in `eight is great` ab.

5-4. Bewegen Sie den Cursor zur letzten Zeile der Datei. Überschreiben Sie `FILE` mit `EXERCISE`. Ersetzen Sie `OF` durch `TO`.

Spezielle Kommandos

Die folgenden speziellen Kommandos werden Ihnen das Arbeiten mit `vi` erleichtern:

`.`	Das letzte Kommando wird wiederholt.
`J`	Zwei Zeilen werden miteinander verknüpft.
`^l`	Der Bildschirm wird gelöscht und neu aufgebaut.
`~`	Kleinbuchstaben werden gegen Großbuchstaben ausgetauschtund umgekehrt.

Wiederholen des letzten Kommandos

Wenn Sie einen Punkt (`.`) eingeben, wird das letzte Kommando wiederholt, mit dem Sie Text in der Datei eingefügt, gelöscht oder geändert haben. Dieses Kommando wird häufig in Kombination mit dem Suchen-Kommando benutzt.

Angenommen, Sie haben den Fluß "Oder" versehentlich klein geschrieben, möchten aber den Kleinbuchstaben in der Konjunktion "oder" nicht ebenfalls in Großbuchstaben umwandeln. Eine Lösungsmöglichkeit besteht darin, nach dem Wort "oder" zu suchen und es bei der ersten passenden Stelle zu korrigieren. Wenn Sie dann in einem weiteren Suchvorgang eine weitere Stelle finden, können Sie einfach einen Punkt (.) eingeben; da `vi` Ihr letztes Kommando zwischengespeichert hat, ersetzt es den Kleinbuchstaben o erneut durch den Großbuchstaben O.

Probieren Sie dieses Kommando an weiteren Beispielen aus. Wenn Sie sich beispielsweise im Kommandomodus befinden und am Ende eines Satzes einen Punkt (.) einfügen möchten, so wird auf dem Bildschirm plötzlich die letzte Textänderung angezeigt. Beachten Sie die Änderungen, die auf dem Bildschirm vor sich gehen.

Verknüpfen von zwei Zeilen

Mit dem Kommando `J` können Sie Zeilen miteinander verknüpfen. Dazu betätigen Sie in der ersten der zu verknüpfenden Zeilen gleichzeitig die Tasten SHIFT und `j`; die aktuelle Zeile wird dann mit der nachfolgenden verknüpft.

Angenommen, in Ihrer Datei sind die folgenden zwei Textzeilen enthalten:

```
Dear Mr.
Smith:
```

Um diese beiden Zeilen miteinander zu verknüpfen, bewegen Sie den Cursor an eine beliebige Stelle der ersten Zeile und geben folgendes ein:

```
J
```

Auf dem Bildschirm erscheint sofort die folgende Ausgabe:

```
Dear Mr. Smith:
```

Zwischen dem letzten Wort der ersten Zeile und dem ersten Wort der zweiten Zeile wird von `vi` automatisch ein Leerzeichen eingefügt.

Bildschirm löschen und wieder aufbauen

Wenn Ihnen ein anderer UNIX-Benutzer während einer `vi`-Sitzung mit dem Kommando write eine Nachricht äschickt, erscheint die Nachricht in Ihrem aktuellen Fenster; dadurch werden Teile des gerade editierten Texts möglicherweise überlagert. Um die aktuelle Bildschirmanzeige wiederherzustellen, müssen Sie `vi` gegebenenfalls in den Kommandomodus umschalten (im Eingabemodus betätigen Sie dazu die Taste `ESCAPE`. Geben Sie dann Control-l (`^l`) ein. Die Nachricht wird von `vi` daraufhin gelöscht, und der Bildschirm wird wieder in den Zustand zurückgesetzt, in dem er sich beim Empfang der Nachricht befunden hat.

Groß- gegen Kleinbuchstaben austauschen und umgekehrt

Am schnellsten können Sie einen beliebigen Groß- gegen den entsprechenden Kleinbuchstaben austauschen, indem Sie den Cursor zu diesem Buchstaben bewegen und eine Tilde (~) eingeben. Um z.B. den Kleinbuchstaben a gegen den Großbuchstaben A auszutauschen, geben Sie eine Tilde (~) ein. Um mehrere Buchstaben auf diese Weise auszutauschen, geben Sie entweder mehrmals eine Tilde (~) ein oder stellen dem Kommando eine Zahl voran.

Tabelle 7-9 enthält einen Überblick über die speziellen `vi`-Kommandos.

Tabelle 7-9: Die speziellen `vi`-Kommandos

Kommando	Funktion
.	Das letzte Kommando wird wiederholt.
J	Die Zeile unter der aktuellen Zeile wird mit der aktuellen Zeile verknüpft.
^l	Die aktuelle Bildschirmanzeige wird gelöscht und neu aufgebaut.
~	Kleinbuchstaben werden gegen Großbuchstaben ausgetauscht und umgekehrt.

Arbeiten mit Zeileneditor-Kommandos unter `vi`

Unter dem Editor `vi` können Sie auf eine Reihe von Kommandos des Zeileneditors `ex` zugreifen (eine vollständige Übersicht über die `ex`-Kommandos finden Sie im Referenzhandbuch für Benutzer) unter dem Eintrag `ex`(1). In diesem Abschnitt werden einige der gängigsten Zeileneditor-Kommandos beschrieben.

Die `ex`-Kommandos entsprechen im Großen und Ganzen den `ed`-Kommandos, die Sie in Kapitel 6 kennengelernt haben. Wenn Sie bereits Erfahrung im Umgang mit `ed` haben, so können Sie an einer Textdatei ausprobieren, wie viele der `ed`-Kommandos auch unter `vi` funktionieren.

Die Zeileneditor-Kommandos beginnen mit einem Doppelpunkt (`:`). Nach der Eingabe des Doppelpunkts (`:`) springt der Cursor in die letzte Bildschirmzeile und gibt den Doppelpunkt (`:`) aus. Das von Ihnen eingegebene Kommando wird ebenfalls angezeigt.

Vorübergehendes Aufrufen einer Shell: Die Kommandos `:sh` und `:!`

Unmittelbar nach dem Aufrufen von `vi` wird der gesamte Bildschirm durch den Puffer-Inhalt belegt, so daß Sie keine Shell-Kommandos durchführen können. In manchen Situationen kann dies jedoch notwendig sein, z.B. um Informationen aus einer anderen Datei in Ihren aktuellen Text einzubringen. Hierzu könnten Sie eines der Shell-Kommandos zum Anzeigen des Inhalts einer Textdatei (z.B. `cat` oder `pg`) aufrufen. Dies ist allerdings ein zeitaufwendiges, umständliches Verfahren, da Sie dazu den Editor extra beenden und wieder aufrufen müssen. Unter `vi` gibt es zwei Möglichkeiten, um den Editor vorübergehend zu beenden und Shell-Kommandos (auch zum Editieren von anderen Dateien) aufzurufen, ohne den Puffer-Inhalt abzuspeichern und den Editor zu beenden. Das vorübergehende Beenden erreichen Sie über die Kommandos `:!` und `:sh`.

Mit dem Kommando `:!` können Sie den Editor beenden und ein einzeiliges Shell-Kommando aufrufen. Hierzu geben Sie im `vi`-Kommandomodus die Zeichenkombination `:!` ein. Die Zeichen werden am unteren Bildschirmrand dargestellt. Geben Sie unmittelbar hinter dem `!` ein Shell-Kommando ein. Die Shell ruft das angegebene Kommando auf, erzeugt eine Ausgabe und stellt die Meldung `[Hit return to continue]` dar. Sobald Sie jetzt die Taste `RETURN` betätigen, baut `vi` den Bildschirm neu auf, und der Cursor wird wieder an der Position dargestellt, an der er sich beim Beenden des Editors befunden hat.

Das ex-Kommando :sh hat dieselbe Funktion, erzeugt aber eine andere Bildschirmausgabe. Geben Sie im vi-Kommandomodus :sh ein und betätigen Sie die Taste RETURN. In der nächsten Zeile erscheint die Shell-Eingabeaufforderung, hinter der Sie Ihr(e) Kommando(s) so eingeben können, als befänden Sie sich bei der Shell. Sobald Sie wieder zu Ihrer vi-Sitzung zurückkehren möchten, geben Sie ^d oder exit ein. Dadurch wird der Bildschirm wieder neu aufgebaut (mit dem Inhalt des Puffers); der Cursor befindet sich wieder an der ursprünglichen Position.

Selbst wenn Sie beim vorübergehenden Aufrufen einer Shell ein Kommando zum Wechseln des Verzeichnisses eingegeben haben, wird nach der Eingabe von exit oder ^d wieder der Inhalt des vi-Puffers angezeigt, den Sie vor dem Beenden des Editors bearbeitet haben.

Text in einer neuen Datei abspeichern: Das Kommando :w

Mit dem Kommando :w (für write) können Sie eine Datei anlegen, indem Sie Textzeilen aus der aktuell editierten Datei in eine von Ihnen angegebene Datei kopieren. Zum Anlegen der neuen Datei müssen Sie in der Kommandozeile eine Zeile bzw. einen Zeilenbereich (mit Zeilennummern) sowie den Namen der neuen Datei angeben. Sie können beliebig viele Zeilen angeben. Das Kommando hat folgendes allgemeines Format:

:*zeilen_nr[,zeilen_nr]*w *datei_name*

Um beispielsweise die dritte Zeile im Puffer in eine Datei namens three zu schreiben, geben Sie folgendes ein:

```
:3w three<CR>
```

vi meldet das erfolgreiche Anlegen einer neuen Datei mit den folgenden Informationen:

```
"three" [New file] 1 line, 20 characters
```

Um die aktuelle Zeile in eine Datei einzubringen, können Sie als Zeilenadresse

einen Punkt (.) eingeben:

```
:.w junk<CR>
```

Eine neue Datei namens junk wird angelegt, in der lediglich die aktuelle Zeile des vi-Puffers enthalten ist.

Wenn Sie einen ganzen Abschnitt in einer neuen Datei abspeichern möchten, müssen Sie einen Zeilenbereich angeben. Um beispielsweise die Zeilen 23 bis 37 in einer Datei abzuspeichern, geben Sie folgendes ein:

```
:23,37w datei_neu<CR>
```

In eine bestimmte Zeile springen

Sie können den Cursor in eine beliebige Zeile bewegen, indem Sie das Kommando : in Kombination mit der Nummer der betreffenden Zeile eingeben. Mit der Kommandozeile

```
:n<CR>
```

geben Sie an, daß der Cursor in die *n*-te Zeile im Puffer springen soll.

Zeilen bis zum Puffer-Ende löschen

Eine der einfachsten Möglichkeiten zum Löschen sämtlicher Zeilen zwischen der aktuellen Zeile und dem Puffer-Ende besteht darin, das Zeileneditor-Kommando d mit den Sonderzeichen für die aktuelle und die letzte Zeile aufzurufen:

```
:.,$d<CR>
```

Der Punkt (.) steht für die aktuelle Zeile, das Dollar-Zeichen ($) für die letzte Zeile.

Datei in den Puffer einlesen

Um Text aus einer anderen Datei unter einer bestimmten Zeile des Editor-Puffers einzulesen, rufen Sie das Kommando `:r` (für read) auf. Wenn Sie beispielsweise den Inhalt der Datei `data` in Ihre aktuelle Datei einbringen möchten, bewegen Sie den Cursor in die Zeile oberhalb der betreffenden Stelle und geben folgendes ein:

```
:r data<CR>
```

Eine andere Möglichkeit besteht darin, die Nummer der betreffenden Zeile anzugeben. Um beispielsweise die Datei `data` unter Zeile 56 des Puffers einzufügen, geben Sie folgendes ein:

```
:56r data<CR>
```

Denken Sie stets daran, daß Sie ex-Kommandos mit dem Kommando `u` rückgängig machen können - also keine Angst vor Experimenten!

Globale Änderungen

Zu den leistungsfähigsten ex-Kommandos gehört das Kommando `g` (für global). Die Verwendung dieses Kommandos ist allerdings hauptsächlich für Benutzer empfehlenswert, die Erfahrung im Umgang mit dem Zeileneditor haben; unerfahrene Benutzer sollten das Kommando zunächst an einer Testdatei ausprobieren.

Angenommen, in einem mehrseitigen Text über das DNA-Molekül geht es um dessen helixförmige Struktur. Wenn Sie dann das Wort Helix an jeder Stelle, an der es vorkommt, gegen Doppel-Helix austauschen möchten, so können Sie dies mit dem Global-Kommando des Editors ex mit einer einzigen Kommandozeile ausführen. Zunächst müssen Sie jedoch die Funktion einer Reihe von Kommandos kennenlernen.

`:g/`*muster*`/`*kommando*`<CR>`

> Für jede Zeile, in der *muster* enthalten ist, wird das ex-Kommando *kommando* durchgeführt. Geben Sie beispielsweise das folgende Kommando ein: `:g/Helix/p<CR>`. Der Zeileneditor gibt jetzt alle Zeilen aus in denen das Muster Helix enthalten ist.

:s/*muster*/*text_neu*/<CR>

Das Ersetzen-Kommando. Der Zeileneditor durchsucht die Datei nach der ersten Stelle in der aktuellen Zeile, an der die Zeichen *muster* vorkommen, und ersetzt sie durch *text_neu*.

:s/*muster*/*text_neu*/g<CR>

Wenn Sie hinter dem letzten Begrenzer dieser Kommandozeile den Kleinbuchstaben g eingeben, wechselt ex das Muster *muster* an jeder Stelle aus, an der es in der aktuellen Zeile vorkommt. Ohne das Kommando g tauscht ex das Muster lediglich bei seinem ersten Auftreten aus.

:g/Helix/s//Doppel-Helix/g<CR>

Mit dieser Kommandozeile sucht ex nach dem Wort Helix und tauscht es bei jedem Auftreten in dieser Zeile gegen die beiden Wörter Doppel-Helix aus. Zwischen den beiden Begrenzern hinter dem Kommando s muß das Wort helix nicht erneut eingegeben werden, da das Kommando automatisch das Wort zwischen den beiden Begrenzern verwendet, die auf das Kommando g folgen. Weitere Informationen über die leistungsfähigen Global- und Ersetzungs-Kommandos finden Sie in Kapitel 6, "Der Zeileneditor ed."

Tabelle 7-10 enthält einen Überblick über die Zeileneditor-Kommandos, die Sie unter vi aufrufen können.

Tabelle 7-10: Die Zeileneditor-Kommandos

Kommando	Funktion
:	Die nachfolgend eingegebenen Kommandos sind Zeileneditor-Kommandos.
:sh<CR>	Der Editor wird vorübergehend verlassen, um Shell-Kommandos durchzuführen.
^d	Rückkehr von der vorübergehend aufgerufenen Shell zum aktuellen `vi`-Fenster, um die Datei weiter zu editieren.
:*n*<CR>	Sprung zur *n*-ten Zeile des Puffers.
:*x*,*y*w *daten*<CR>	Die Zeilen im Bereich *x* bis *y* werden in eine neue Datei (*daten*) eingebracht.
:$<CR>	Sprung zur letzten Zeile des Puffers.
:.,$d<CR>	Sämtliche Zeilen zwischen der aktuellen Zeile und dem Ende des Puffers werden gelöscht.
:r *shell.datei*<CR>	Der Inhalt von *shell.datei* wird unter der aktuellen Zeile des Puffers eingefügt.
:s/*text_alt*/*text_neu*/<CR>	Die Zeichen *text_alt* werden bei ihrem ersten Auftreten in der aktuellen Zeile gegen *text_neu* ausgetauscht.
:s/*text_alt*/*text_neu*/g<CR>	Die Zeichen *text_alt* werden bei jedem Auftreten in der aktuellen Zeile gegen *text_neu* ausgetauscht.
:g/*text_alt*/s//*text_neu*/g<CR>	Die Zeichen *text_alt* werden bei jedem Auftreten in der Datei gegen *text_neu* ausgetauscht.

Beenden von vi

Zum Beenden des Editors vi gibt es grundsätzlich fünf Möglichkeiten, u.a. Zeileneditor-Kommandos, die mit einem Doppelpunkt (:) eingeleitet werden müssen.

:wq<CR>	Der Inhalt des vi-Puffers wird in der aktuell editierten UNIX-Datei abgespeichert; danach wird vi beendet.
ZZ	Der Puffer-Inhalt wird nur dann abgespeichert, wenn er seit dem letzten Kommando w geändert worden ist; danach wird vi beendet.
:w *datei_name*<CR> :q<CR>	Der Inhalt des temporären Puffers wird in die neue Datei *datei_name* eingebracht; danach wird vi beendet.
:w! *datei_name*<CR> :q<CR>	Die vorhandene Datei *datei_name* wird durch den Inhalt des Puffers überschrieben; danach wird vi beendet.
:q!<CR>	vi wird beendet, ohne den Puffer-Inhalt in einer Datei abzuspeichern; sämtliche am Puffer vorgenommenen Änderungen gehen verloren.
:q<CR>	vi wird beendet, ohne den Puffer-Inhalt in einer UNIX-Datei abzuspeichern. Dies funktioniert jedoch nur dann, wenn Sie am Puffer keine Änderungen vorgenommen haben. Andernfalls gibt vi den Hinweis aus, daß Sie den Puffer-Inhalt entweder abspeichern oder den Editor über das Kommando :q!<CR> ohne vorheriges Abspeichern beenden müssen.

Die Kommandos :wq und ZZ speichern den Puffer-Inhalt unter den oben beschriebenen Bedingungen in einer Datei ab, beenden den Editor vi und schalten zur Shell zurück. Angenommen, Sie haben zuerst versucht, das Kommando ZZ aufzurufen und möchten den Editor nun mit :wq beenden. In diesem Fall wird der Name der aktuell editierten Datei von vi zwischengespeichert, so daß Sie ihn nicht mehr eingeben müssen, wenn Sie den Inhalt des Puffers wieder in

einer Datei abspeichern möchten. Geben Sie folgendes ein:

```
:wq<CR>
```

Der Editor `vi` gibt den Namen der Datei sowie die Anzahl der Zeilen und Zeichen in der Datei aus.

Wie können Sie der Datei auch einen anderen Namen zuordnen? Möchten Sie den Puffer-Inhalt beispielsweise in einer neuen Datei namens `junk` abspeichern, geben Sie folgendes ein:

```
:w junk<CR>
```

Nach dem Abspeichern des Puffer-Inhalts in der neuen Datei beenden Sie `vi` über das folgende Kommando:

```
:q<CR>
```

Ist die angegebene Datei bereits vorhanden, gibt der Editor eine Warnung aus. Wenn Sie den Puffer-Inhalt beispielsweise in einer Datei namens `johnson` abspeichern möchten, gibt das System folgendes aus:

```
"johnson" File exists - use "w! johnson" to overwrite
```

Soll der Inhalt der bestehenden Datei (`johnson`) durch den Inhalt des Puffers überschrieben werden, so rufen Sie das Kommando `:w!` auf:

```
:w! johnson<CR>
```

Die vorhandene Datei wird durch die neue überschrieben.

Wenn Sie eine Datei namens `memo` editieren und einige Änderungen daran vornehmen, die Sie dann nicht erhalten möchten, so beenden Sie `vi` ohne vorherige Abspeicherung:

```
:q!<CR>
```

Tabelle 7-11 enthält einen Überblick über die Kommandos zum Beenden des Editors `vi`.

Tabelle 7-11: Die Kommandos zum Beenden von vi

Kommando	Funktion
ZZ	Die Datei wird abgespeichert, wenn sie seit dem letzten Abspeichern geändert worden ist; dann wird vi beendet.
:wq<CR>	Die Datei wird abgespeichert und vi wird beendet.
:w *datei_name*<CR> :q<CR>	Der Inhalt des Editor-Puffers wird in eine neue Datei eingebracht (*datei_name*); dann wird vi beendet.
:w! *datei_name*<CR> :q<CR>	Die vorhandene Datei (*datei_name*) wird durch den Inhalt des Editor-Puffers überschrieben; dann wird vi beendet.
:q!<CR>	vi wird beendet, ohne daß der Puffer- Inhalt zuvor in einer Datei abgespeichert worden ist - unabhängig davon, ob am Puffer-Inhalt Änderungen vorgenommen worden sind, oder nicht.
:q<CR>	vi wird nur dann ohne vorheriges Abspeichern des Puffer-Inhalts beendet, wenn am Puffer-Inhalt keine Änderungen vorgenommen worden sind.

Spezielle `vi`-Optionen

Für das Kommando `vi` gibt es eine Reihe von speziellen Optionen, die Ihnen folgende Möglichkeiten bieten:

- Wiederherstellung einer Datei nach einem Systemabsturz.
- Mehrere Dateien in den Editor-Puffer holen und nacheinander editieren.
- Abrufen des Datei-Inhalts ohne das Risiko einer versehentlichen Änderung der Datei.

Wiederherstellen einer Datei nach einem Systemabsturz

Bei einem Absturz des Systems, einem Stromausfall o.ä. wird `vi` beendet, ohne zuvor den Puffer-Inhalt in die Datei zurückzuschreiben. Da `vi` vom Puffer-Inhalt jedoch eine Kopie anlegt, können Sie die Datei nach der erneuten Anmeldung bei UNIX wiederherstellen; dies wird durch die `vi`- Option `-r` ermöglicht, die Sie in folgendem Format eingeben:

`vi -r` *datei_name*<CR>

Dadurch werden alle bzw. ein Großteil der Änderungen, die Sie vor dem Absturz an der Datei *datei_name* vorgenommen haben, in den `vi`-Puffer geladen. Sie können die Datei nun entweder editieren, oder sie abspeichern und den Editor beenden. Beim Abspeichern benutzt der Editor `vi` automatisch den ursprünglichen Dateinamen.

Editieren von mehreren Dateien

Sie können während ein und derselben Editor-Sitzung auch zwei oder mehr Dateien editieren; hierzu rufen Sie das Kommando `vi` mit den jeweiligen Dateinamen auf:

`vi` *datei1 datei2*<CR>

`vi` gibt die Anzahl der Dateien aus, die Sie nach dem Kommando editieren werden:

```
2 files to edit
```

Nach dem Editieren der ersten Datei speichern Sie die Änderungen (die Sie am Puffer-Inhalt vorgenommen haben) in der Datei (*datei1*) ab:

```
:w<CR>
```

Das System reagiert auf das Kommando `:w<CR>` mit einer Meldung am unteren Bildschirmrand, die Sie über den Namen der Datei sowie ihre Größe (Anzahl der Zeilen/Zeichen) informiert. Jetzt können Sie mit dem Kommando `:n` mit dem Editieren der nächsten Datei fortfahren:

```
:n<CR>
```

Das System gibt daraufhin am unteren Bildschirmrand eine Meldung aus, die Sie über den Namen der nächsten zu editierenden Datei sowie ihre Größe (Anzahl der Zeichen/Zeilen) informiert.

Rufen Sie den Editor `vi` auf und holen Sie zwei Dateien gleichzeitig aus Ihrem aktuellen Verzeichnis in den Editor-Puffer. Achten Sie auf die Meldungen, die das System am unteren Bildschirmrand ausgibt.

Abrufen des Dateiinhalts

Zur Überprüfung des Inhalts einer Datei stehen Ihnen unter `vi` leistungsfähige Funktionen zum Durchsuchen und Durchblättern einer Datei zur Verfügung. In derartigen Fällen ist es häufig empfehlenswert, sich gegen das versehentliche Änderung einer Datei abzusichern. Hierzu rufen Sie die Datei mit einem Schreibschutz auf, indem Sie den Editor mit `view` statt mit `vi` starten.

Tabelle 7-12 enthält einen Überblick über die speziellen `vi`-Optionen.

Tabelle 7-12: Die speziellen vi-Optionen

Option	Funktion
vi *datei1 datei2 datei3*<CR>	In den vi-Puffer werden drei Dateien (*datei1, datei2* und *datei3*) zum Editieren geladen.
:w<CR> :n<CR>	Die aktuelle Datei wird abgespeichert, und die Editor-Sitzung wird mit der nächsten Datei fortgesetzt.
vi -r *datei1*<CR>	Die an *datei1* vorgenommenen Änderungen werden wiederhergestellt.
view *datei*<CR>	Der Inhalt der Datei wird mit einem Schreibschutz abgerufen, um versehentliche Änderungen an *datei* zu verhindern.

Übung 6

6-1. Versuchen Sie, eine bei einem Systemabsturz verlorengegangene Datei wiederherzustellen.

Rufen Sie `vi` auf und geben Sie in eine Datei namens `exer6` etwas Text ein. Schalten Sie Ihr Terminal aus, ohne die Datei zuvor abzuspeichern oder `vi` zu beenden. Schalten Sie dann Ihr Terminal wieder ein und melden Sie sich erneut bei UNIX an. Versuchen Sie jetzt, `vi` wieder mit der Datei `exer6` aufzurufen.

6-2. Holen Sie die Dateien `exer1` und `exer2` zum Editieren in den `vi`-Puffer. Speichern Sie `exer1` ab, und rufen Sie die nächste Datei im Puffer (`exer2`) auf.

Speichern Sie `exer2` in der Datei `junk` ab.

Beenden Sie `vi`.

6-3. Rufen Sie probehalber das folgende Kommando auf:

```
vi exer*<CR>
```

Welche Auswirkung hat dieses Kommando? Versuchen Sie alle Dateien möglichst schnell zu verlassen.

6-4. Rufen Sie den Inhalt von `exer4` mit einem Schreibschutz ab.

Verschieben Sie die Bildschirmanzeige um ein volles Fenster nach unten.

Verschieben Sie die Bildschirmanzeige um ein halbes Fenster nach unten.

Verschieben Sie die Bildschirmanzeige um ein volles Fenster nach oben.

Verschieben Sie die Bildschirmanzeige um ein halbes Fenster nach oben.

Beenden Sie den Editor, um zur Shell zurückzukehren.

Auflösung der Übungen

Unter `vi` gibt es zur Durchführung einer bestimmten Funktion häufig mehrere Möglichkeiten. Die folgenden Lösungen stellen lediglich Vorschläge dar. Wenn Ihnen eine andere Lösung besser liegt, so können Sie diese ohne weiteres benutzen - entscheidend ist nicht der Weg, sondern das Ergebnis.

Übung 1

1-1. Erkundigen Sie sich bei Ihrem Systemverwalter nach dem Systemnamen Ihres Terminals. Geben Sie folgendes ein:

```
TERM=terminal_name<CR>
export TERM<CR>
tput init<CR>
```

1-2. Rufen Sie `vi` mit der Datei `exer1` auf:

```
vi exer1<CR>
```

Rufen Sie dann das Kommando `a` auf und geben Sie den folgenden Text in Ihre Datei ein:

```
This is an exercise!<CR>
Up, down,<CR>
left, right,<CR>
build your terminal's<CR>
muscles bit by bit<ESC>
~
~
```

1-3. Rufen Sie die Kommandos `k` und `h` auf.

1-4. Rufen Sie das Kommando `x` auf.

1-5. Rufen Sie die Kommandos `j` und `l` auf.

1-6. Rufen Sie das Kommando a auf und geben Sie folgenden Text ein:

```
<CR>
and byte by byte<ESC>
```

1-7. Geben Sie das folgende Kommando ein:

```
ZZ
```

1-8. Geben Sie das folgende Kommando ein:

```
vi exer1<CR>
```

Beachten Sie die Meldung, die vom System ausgegeben wird:

```
"exer1" 6 lines, 100 characters
```

Übung 2

2-1. Geben Sie die folgenden Kommandos ein:

```
vi exer2<CR>
a
1<CR>
2<CR>
3<CR>
  .
  .
  .
48<CR>
49<CR>
50<ESC>
```

2-2. Geben Sie die folgenden Kommandos ein:

```
<^f>
<^b>
<^u>
<^d>
```

Beachten Sie die die geänderten Zeilennummern.

2-3. Geben Sie die folgenden Kommandos ein:

```
G
$
a
<CR>
123456789 123456789<ESC>
7h
31
```

Beim Kommando `7h` springt der Cursor zur Zahl `2` in der zweiten Zahlengruppe. Beim Kommando `31` springt der Cursor zur Zahl `5` in der zweiten Zahlengruppe.

2-4. `$` = Zeilen-Ende
`0` = Erstes Zeichen in der Zeile

2-5. Geben Sie folgendes ein:

```
^
w
b
e
```

2-6. Geben Sie folgendes ein:

```
1G
M
L
H
```

2-7. Geben Sie folgendes ein:

```
/8<CR>
n
/48<CR>
```

Übung 3

3-1. Geben Sie folgendes ein:

```
vi exer3<CR>
```

3-2. Geben Sie folgendes ein:

```
a
Append text<CR>
Insert text<CR>
a computer's<CR>
job is boring.<ESC>
```

3-3. Geben Sie folgendes ein:

```
O
financial statement and<ESC>
```

3-4. Geben Sie folgendes ein:

```
3G
iDelete text<CR><ESC>
```

Der Inhalt Ihrer Datei sieht jetzt folgendermaßen aus:

```
Append text
Insert text
Delete text
a computer's
financial statement and
job is boring.
```

3-5. `a computer's` ist die aktuelle Zeile. Wenn Sie unter dieser Zeile eine neue Textzeile einfügen möchten, so rufen Sie das Kommando `o` auf.

3-6. Die aktuelle Zeile ist `byte of the budget.`
Bei der Eingabe von `G` springt der Cursor in die unterste Zeile auf dem Bildschirm.
Über das Kommando `A` können Sie am Ende der Zeile Text anfügen.
Bei Betätigung von `<CR>` wird die neue Zeile erstellt.
Fügen Sie den folgenden Satz an: `But, it is an exciting machine.`
Über die Taste `<ESC>` beenden Sie den Eingabemodus.

3-7. Geben Sie folgendes ein:

```
1G
/text
i
some<space bar><ESC>
```

3-8. Über das Kommando `ZZ` wird der Puffer-Inhalt in die Datei `exer3` eingebracht, und Sie kehren zur Shell zurück.

Übung 4

4-1. Geben Sie folgendes ein:

```
vi exer4<CR>
a
When in the course of human events<CR>
there are many repetitive, boring<CR>
chores, then one ought to get a<CR>
robot to perform those chores.<ESC>
```

4-2. Geben Sie folgendes ein:

```
2G
A
, tedious, and unsavory<^w><ESC>
```

Geben Sie `Fb` ein, um den Cursor zum Buchstaben `b` in `boring` zu bewegen. Geben Sie dann folgendes ein:
`dw` (hierzu können Sie auch das Kommando `6x` benutzen).

4-3. Sie befinden sich in der zweiten Zeile. Geben Sie folgendes ein:

```
2j
I congenial and computerized <ESC>
dd
```

Wenn Sie die Zeile löschen und leer lassen möchten, so geben Sie folgendes ein:

`u` (Aufheben des Kommandos `dd`)
`0` (Bei der Eingabe der Zahl Null wird der Cursor zum Zeilenanfang bewegt).
`D`

```
d1G
```

4-4. Abspeichern und `vi` beenden.

```
ZZ
```

Löschen Sie die Datei:

```
rm exer4<CR>
```

Übung 5

5-1. Geben Sie folgendes ein:

```
vi exer2<CR>
8G
cc
END OF FILE <ESC>
```

5-2. Geben Sie folgendes ein:

```
1G
8"zyy
G
"zp
```

5-3. Geben Sie folgendes ein:

```
8G
cc
8 is great<ESC>
```

5-4. Geben Sie folgendes ein:

```
G
2w
cw
EXERCISE<ESC>
2b
cw
TO<ESC>
```

Übung 6

6-1. Geben Sie folgendes ein:

```
vi exer6<CR>
a (Anfügen von mehreren Textzeilen)
<ESC>
```

Schalten Sie das Terminal aus.

Schalten Sie das Terminal ein.
Melden Sie sich bei UNIX an. Geben Sie folgendes ein:

```
vi -r exer6<CR>
:wq<CR>
```

6-2. Geben Sie folgendes ein:

```
vi exer1 exer2<CR>
:w<CR>
:n<CR>

:w junk<CR>
ZZ
```

6-3. Geben Sie folgendes ein:

```
vi exer*<CR>
```

(Meldung:)
`8 files to edit` (`vi` ruft alle Dateien auf, deren Namen mit `exer` beginnen).

```
:q!
```

6-4. Geben Sie folgendes ein:

```
view exer4<CR>
<^f>
<^d>
<^b>
<^u>
:q<CR>
```

8 Der Druck-Service LP

Einführung 8-1
Einstellung von benutzer-definierten Druckparametern 8-2
Komponenten des LP-Druckprozesses 8-3
Zum Inhalt dieses Kapitels 8-4

Steuerung des Druckprozesses 8-5
Angabe eines Ziels 8-5
- Benutzung eines fernen Druckers 8-6

Prioritäten in der Drucker-Warteschlange steuern 8-6
Anforderung einer Bestätigung durch den Print-Service 8-7
Abfrage von Statusinformationen über die Drucker 8-7
Änderung einer Druckanforderung 8-11
Stornieren einer Anforderung: das Kommando `cancel` 8-12
Aktivieren und Deaktivieren eines Druckers 8-13

Formatierung der Druckausgabe mit dem Kommando lp 8-15
Angabe des Dateiformats 8-15
Angabe von Papierformat, Zeilenabstand und Zeichendichte 8-17
Seitenumbruch zwischen Dateien ausschalten 8-18
Ausdrucken ohne Titelseite 8-19
Benutzung von vorgedruckten Formularen 8-19
Wahl des Zeichensatzes bzw. Typenrads 8-20
Spezielle Druck-Modi 8-20
Anfordern von Mehrfach-Kopien 8-21
Arbeiten mit PostScript-Druckern 8-21
- Unterstützung von Nicht-PostScript-Druckanforderungen 8-22
- Zusätzliche PostScript-Möglichkeiten durch Filter 8-23

■ Arbeiten mit PostScript-Schriften 8-25

Die Kommandos des Druck-Service LP 8-28

Einführung

Bei dem Druck-Service LP handelt es sich um eine Gruppe von UNIX- Programmen, mit denen Sie Dateien auf Papier ausgeben können. Der Name "LP" steht für "line printer" (Zeilendrucker). Auf diesen Druckertyp ist der Druck-Service ursprünglich zugeschnitten gewesen; da er in seiner heutigen Form jedoch eine Vielzahl von Druckergeräten ansteuern kann, ist "LP" weniger eine Funktionsbeschreibung als vielmehr ein historisch begründeter Name.

Die einfachste Möglichkeit zur Nutzung dieses Druck-Service besteht darin, das Kommando `lp` mit dem Namen der auszudruckenden Datei aufzurufen. Das Kommando `lp` leitet einen Druckauftrag an ein Ziel (z.B. einen Zeilen- oder Laserdrucker) weiter; hier wird er in eine Warteschlange gesetzt, wo er auf das Ausdrucken wartet. Das Ziel kann ein Drucker oder eine Drucker-Klasse sein. Ohne die Angabe eines Ziels wird der Druckauftrag zum Standard-Ziel weitergeleitet.

Nach der Eingabe einer derartigen Kommandozeile gibt das System eine Meldung aus, die (a) den Namen des Druckers, über den die Druckausgabe vorgenommen wird, bestätigt, Ihrem Druckauftrag b) eine Auftragsnummer (request id) zuordnet und (c) die Anzahl der Dateien bestätigt, für die Sie eine Druckanforderung abgeschickt haben.

```
$ lp datei_name
request id is laser-9885 (1 file)
```

Aus dieser Antwort des Systems geht hervor, daß Ihre Datei auf einem Drucker namens "laser" (dem Standard-Drucker Ihres Systems) ausgedruckt wird und der Druckauftrag aus nur einer Datei besteht. Die Zeichenkette `laser-9885` wird als "Auftragsnummer" (request-ID) bezeichnet; über diese Nummer können Sie den Status eines Druckauftrags abfragen.

Wenn Sie eine Datei mit diesem einfachen Kommando ausdrucken, müssen Sie keine Druckparameter wie das Papierformat festlegen; im Normalfall können Sie davon ausgehen, daß Ihr Systemverwalter beim Einrichten des Druck-Service für Parameter wie das Papierformat Standardwerte eingegeben hat (sind Sie auf Ihrem Rechner für die Verwaltung des Druck-Service zuständig, so schlagen Sie im Kapitel "LP Print Service Administration" des *System Administrator's Guide* nach). Allerdings können Sie diese Standardwerte jederzeit ändern. Im folgenden Abschnitt finden Sie Informationen darüber, wie Sie die Parameter Ihres Druckauftrags auf Ihre Anforderungen zuschneiden können.

Einstellung von benutzer-definierten Druckparametern

Beim Druck-Service LP können Sie eine Reihe von Parametern Ihres Druckauftrags festlegen.

- Möchten Sie Ihre Datei auf einem bestimmten Drucker oder Drucker-Typ ausgeben?
  (Standard: Der vom Druck-Service-Verwalter festgelegte Drucker).
- Möchten Sie Ihre Datei auf normalem Papier (wie vom Verwalter angegeben) oder auf Formularen (z.B. Rechnungsformularen oder Briefbögen) ausdrucken?
  (Standard: Normales Papier).
- Möchten Sie in einer bestimmten Schrift ausdrucken?
  (Standard: Die vom Druck-Service-Verwalter festgelegte Schrift).
- Möchten Sie Papier mit einem bestimmten Papierformat bedrucken?
  (Standard: Das vom Druck-Service-Verwalter festgelegte Format).
- Möchten Sie mit einem größeren oder kleineren Zeilenabstand (mehr oder weniger Zeilen pro Zoll) ausdrucken?
  (Standard: Der vom Druck-Service-Verwalter festgelegte Zeilenabstand).
- Soll vor jeder Datei eine "Titelseite" (banner page) ausgegeben werden?
  (Standard: Eine Titelseite wird ausgegeben).
- Beim Ausdrucken von zwei oder mehr Dateien: Sollen die Dateien als ein Druckauftrag behandelt (d.h. ohne Seitenumbrüche zwischen den verschiedenen Dateien ausgegeben) werden, oder soll mit jeder Datei auch eine neue Seite beginnen?
  (Standard: Die Dateien werden getrennt ausgegeben).
- Möchten Sie Ihr Schriftstück in zwei- oder mehrfacher Ausfertigung ausgeben?
  (Standard: Nur eine Kopie).

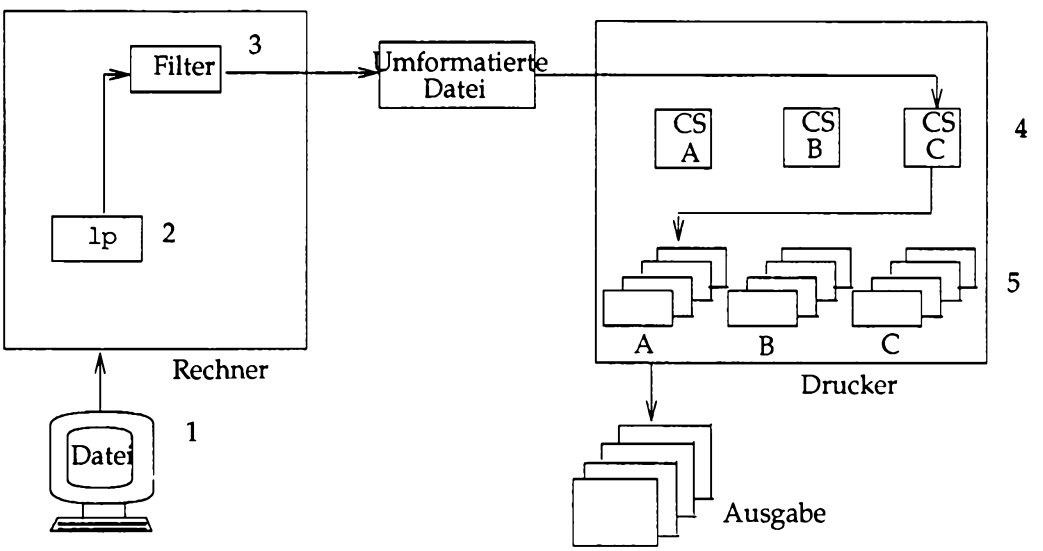
Wenn Sie beim Aufrufen des Kommandos `lp` nur einen Dateinamen angeben, müssen Sie keine dieser Fragen beantworten, da Ihr Verwalter die entsprechenden Parameter bereits beim Einrichten des Druck-Service eingestellt hat. Möchten Sie jedoch mit "Nicht- Standard-"Parametern arbeiten, so müssen Sie die Kommandozeile entsprechend abändern.

Zur Angabe von benutzer-definierten Druckparametern in der Kommandozeile gibt es eine Reihe von Optionen. Um Ihnen das Verständnis für diese Optionen zu erleichtern, werden Ihnen im folgenden zunächst die wichtigsten Komponenten eines Druckauftrags erläutert.

Komponenten des LP-Druckprozesses

Das Ausdrucken eines Textes mit dem Kommando `lp` erfordert das Zusammenspiel von fünf wichtigen Komponenten: (1) Ihre Datei, (2) das Kommando `lp`, (3) gegebenenfalls durch Ihren Verwalter installierte Filter (falls von Ihnen angefordert), (4) gegebenenfalls durch Ihren Verwalter installierte Zeichensätze (ZS) oder Typenräder (falls von Ihnen angefordert) sowie (5) das Papier, auf dem Ihre Datei ausgedruckt werden soll. Bild 8-1 zeigt die Funktion jeder dieser Komponenten innerhalb des Druckprozesses.

Bild 8-1: Die zentralen Komponenten eines Druckauftrags



Im Beispiel in Bild 8-1 wurden zum Ausdrucken der Zeichensatz C sowie Formulare vom Typ A gewählt. Beide Parameter werden beim Anfordern eines Druckauftrags eingestellt.

Zum Inhalt dieses Kapitels

In diesem Kapitel werden die drei Aufgaben beschrieben, die Sie während eines Druckvorgangs möglicherweise ausführen müssen:

- Aktivieren und Deaktivieren des Druckers.
- Erscheinungsbild des Ausdrucks über Druckparameter (z.B. Schriften und Papierformat) festlegen.
- Steuerung des Druckprozesses durch die Wahl eines bestimmten Druckers, Kontrolle des Druckprozesses über Meldungen und Statusinformationen sowie Änderung und Stornierung von Druckanforderungen.

Steuerung des Druckprozesses

Beim Druck-Service LP können Sie den Druckprozeß folgendermaßen kontrollieren und steuern:

- Durch die Angabe des Druckers.
- Durch die Heraufsetzung der Priorität für Druckaufträge, die in der Warteschlange an die Spitze gesetzt werden sollen.
- Durch die Abfrage von Statusinformationen über die Drucker, die Möglichkeiten des Druck-Service (z.B. Formulare und Zeichensätze) sowie die gerade laufenden Druckaufträge.
- Durch die Änderung der Parameter eines bereits abgesetzten Druckauftrags.
- Durch den Abbruch eines bereits laufenden Druckauftrags.

In diesem Abschnitt wird erklärt, wie Sie diese Funktionen durchführen können.

Angabe eines Ziels

Unter dem Begriff "Druck-Ziel" werden alle Geräte zusammengefaßt, die Ihr Systemverwalter als Drucker oder Drucker-Klasse definiert hat. Eine Drucker-Klasse besteht aus einer Reihe von Druckern, die ein Verwalter unter praktischen Gesichtspunkten zu einer Gruppe zusammengefaßt hat. So kann ein Verwalter alle Drucker eines bestimmten oder ähnlichen Typs (z.B. Laser- oder Zeilendrucker) zu einer Drucker-Klasse zusammenfassen. Eine andere Möglichkeit ist, daß alle Drucker im zweiten Stock eines Gebäudes derselben Klasse zugeordnet werden. Die Zusammensetzung einer Klasse richtet sich ausschließlich nach der Praktikabilität für die Benutzer und/oder den Verwalter des Druck-Service; beim Druck-Service LP muß ein Drucker keinerlei Bedingungen erfüllen, damit er einer Klasse zugeordnet werden kann. Eine Klasse kann also beliebig zusammengesetzt sein.

Die Option `-d` *ziel* (für destination) in der Kommandozeile bewirkt, daß Ihre Datei auf dem angegebenen `-d` *ziel* ausgedruckt wird, vorausgesetzt, der Drucker ist verfügbar und erfüllt Ihre Anforderungen für den Druckauftrag. Im folgenden Beispiel wird eine Anforderung zum Ausdrucken der Datei `memo` auf dem drucker3 abgeschickt:

```
$ lp -d drucker3 memo RETURN
```

Benutzung eines fernen Druckers

Beim Druck-Service LP können Rechner, die an dasselbe Netzwerk angeschlossen sind, einen Drucker gemeinsam nutzen. Ist der von Ihnen gewählte Drucker an einen fernen Rechner angeschlossen, so kann es beim Starten des Druckvorgangs sowie beim Abrufen von Statusinformationen über Druckauftrag und Drucker von einem fernen Rechner aus zu Verzögerungen kommen. Ansonsten sollte der Drucker, an den Sie Ihre Druckaufträge und die zugehörigen Kommandos schicken, überhaupt keine Rolle spielen; die Benutzung des Druck-Service läuft grundsätzlich immer gleich ab, unabhängig davon, ob Sie einen lokalen oder einen fernen Drucker benutzen.

Prioritäten in der Drucker-Warteschlange steuern

Die Druckanforderungen, die Sie und andere Benutzer an die Drucker Ihres Systems schicken, werden in eine Drucker-Warteschlange gesetzt, die über die Reihenfolge bestimmt, in der die Druckaufträge abgearbeitet werden. Die höchste Priorität erhalten diejenigen Anforderungen, denen die Prioritätsstufe 0 zugeordnet ist; die niedrigste Priorität erhalten Anforderungen mit der Prioritätsstufe 39. Die Priorität Ihres Druckauftrags hängt von einer Reihe von Faktoren ab.

Den Druckaufträgen wird standardmäßig die Priorität 20 zugeordnet, es sei denn, dieser Wert ist von Ihrem Systemverwalter geändert worden. Diese mittlere Priorität wird jedem Druckauftrag zugeordnet, den Sie an den Drucker schicken. Hat der Systemverwalter die Standard-Priorität auf beispielsweise 10 festgelegt, so erhalten alle von Ihnen abgesetzten Druckaufträge diese höhere Priorität.

Sie können die Prioritätsstufe eines Druckauftrags jedoch auch selbst festlegen, indem Sie eine vom Standardwert abweichende Priorität eingeben. Hierzu geben Sie das Kommando `lp` mit der Option `-q` ein. So können Sie einem besonders eiligen Druckauftrag die höchste Priorität (0) zuordnen:

```
$ lp -d drucker3 -q 0 urgent.memo RETURN
```

Beachten Sie, daß der Systemverwalter Ihren Spielraum bei der Vergabe von Prioritäten einschränken kann. Wenn der Systemverwalter für Sie eine Priorität festgelegt hat und Sie eine höhere Priorität eingeben, so wird standardmäßig die Priorität benutzt, die der Systemverwalter eingestellt hat. Erkundigen Sie sich bei Ihrem Systemverwalter über die Standard-Priorität und eine eventuelle

Obergrenze für Ihre Druckaufträge.

Anforderung einer Bestätigung durch den Print-Service

Der Druck-Service LP gibt nach dem Ende eines Druckvorgangs nicht automatisch eine Bestätigung aus. Um eine Bestätigung zu erhalten, geben Sie in der `lp`-Kommandozeile die Option `-w` ein:

```
$ lp -d drucker3 -w dateiname (RETURN)
```

Der Druck-Service gibt auf dem Bildschirm Ihres Terminals eine Meldung aus, die Ihnen das vollständige Ausdrucken Ihrer Dateien bestätigt; wenn Sie zu dem Zeitpunkt, zu dem die Meldung abgeschickt werden sollte, nicht angemeldet sind, wird die Meldung Ihnen über die elektronische Post zugestellt.

Wenn Sie über die elektronische Post über das vollständige Ausdrucken Ihrer Dateien benachrichtigt werden möchten, so geben Sie hinter dem Kommando `lp` die Option `-m` ein:

```
$ lp -d drucker3 -m dateiname (RETURN)
```

Abfrage von Statusinformationen über die Drucker

Wenn Sie wissen möchten, ob ein Druckauftrag störungsfrei abläuft, so können Sie den Status sämtlicher Druckanforderungen mit dem Kommando `lpstat` ermitteln: Wird dieses Kommando ohne Optionen eingegeben, so liefert es Ihnen den Status sämtlicher Druckanforderungen, die Sie mit dem Druck-Service LP abgeschickt haben:

```
$ lpstat
dqp10_1-25    pr2cms    1942    July 19 13:09
dqp10_1-26    pr2cms    3893    July 19 13:15
dqp10_1-27    pr2cms    942     July 19 14:09
```

Sie können aber auch selektiv nur Informationen über bestimmte Druckanforderungen abfragen, indem Sie in der Kommandozeile die Auftragsnummern dieser Druckanforderungen eingeben (für jede abgeschickte Druckanforderung wird auf dem Bildschirm eine Auftragsnummer angezeigt).

```
$ lpstat laser-6885 drucker- 227 RETURN
```

In diesem Beispiel wurde der Status der beiden Druckanforderungen mit den Auftragsnummern `laser-6885` und `drucker-227` abgefragt.

Mit `lpstat` können Sie außerdem über eine Reihe von Optionen die Art der angezeigten Informationen steuern. Es stehen folgende Informationen zur Wahl:

- Der Status der lokalen Drucker.
- Eine Auflistung der verfügbaren Formulare.
- Eine Auflistung der verfügbaren Zeichensätze und Typenräder.
- Eine Auflistung der verfügbaren Drucker.

Die nachfolgenden Absätze dieses Abschnitts enthalten Informationen darüber, wie Sie diese Informationen selektiv mit den `lpstat`-Optionen abrufen können.

Abfrage des Drucker-Status

Wenn Ihnen die Namen Ihrer Drucker nicht bekannt sind, können Sie eine Übersicht darüber abrufen. Die verfügbaren Drucker hängen von Ihrem installierten UNIX- System ab. Erkundigen Sie sich bei Ihrem Systemverwalter nach den Namen der verfügbaren Drucker, oder geben Sie die folgende Kommandozeile ein:

```
$ lpstat -p all RETURN
```

Hiermit wird eine Übersicht über sämtliche Drucker und ihren Status (enabled/disabled für aktiviert/deaktiviert) ausgegeben:

```
printer druckerl enabled since Aug 22 16:00. available.
printer druckerl disabled since Aug 26 22:00. available.
```

Wenn die Namen der Drucker in Ihrem System Ihnen bereits bekannt sind, können Sie Statusinformationen über einen oder mehrere Drucker abrufen, indem Sie nicht wie im letzten Beispiel das Argument `all` eingeben, sondern die Namen der betreffenden Drucker.

```
$ lpstat -p drucker1,drucker3 RETURN
```

Ausführlichere Statusinformationen können Sie abrufen, indem Sie die Option `-l` in die `lpstat`- Kommandozeile eintragen:

```
$ lpstat -p drucker1,drucker3 -l RETURN
```

Für jeden von Ihnen angegebenen Drucker werden Statusinformationen ausgegeben, die sich aus den folgenden Einzelinformationen zusammensetzen: Der Drucker-Typ, die unterstützten und eingelegten Formulare, die zulässigen Dateiformate, die Namen der nutzungsberechtigten Benutzer, die Standardwerte für Papierformat und Zeichendichte usw.

Der Systemverwalter kann den Zugriff auf bestimmte Drucker einschränken. Wenn Sie aus diesem Grund für einen Drucker nicht nutzungsberechtigt sind, wird die Meldung `not available` ausgegeben.

Die verfügbaren Formulare

Um einen Überblick über die Formulare zu erhalten, die auf Ihrem System verfügbar sind, rufen Sie das Kommando `lpstat` mit der Option `-f` sowie dem Argument `all` auf:

```
$ lpstat -f all RETURN
```

Das Kommando gibt eine Übersicht über sämtliche Formulare aus, die von Ihrem System verarbeitet werden können. Die Formulare, mit denen die Drucker Ihres Systems bestückt sind, werden folgendermaßen angegeben:

```
form payroll_check is available to you, mounted on printer4
```

Die Formulare, die von Ihrem System zwar verarbeitet werden können, aber in keinem Drucker eingelegt sind, werden folgendermaßen aufgelistet:

```
form payroll_check is available to you
```

Der Systemverwalter kann für bestimmte Formulare eine Nutzungsbeschränkung angeben. Wenn Sie für ein Formular aus diesem Grund nicht nutzungsberechtigt sind, wird die Meldung `is not available to you` angezeigt.

Informationen darüber, ob in Ihrem System bestimmte Formulare zur Verfügung stehen, können Sie mit der Option `-f` abrufen:

```
$ lpstat -f laser2,laser2 RETURN
```

Ausführliche Informationen über einige bzw. alle der verfügbaren Formulare erhalten Sie mit der Option `-l`:

```
$ lpstat -f all -l RETURN
```

Auf dem Bildschirm werden jetzt für jedes Formular Informationen über die Seitenlänge, die Seitenbreite, die Anzahl der Seiten, die Farbe des Farbbands usw. ausgegeben.

Verfügbare Zeichensätze und Typenräder

Einen Überblick über die Zeichensätze und/oder Typenräder, die Ihnen bei Ihrem Druck-Service LP zur Verfügung stehen, erhalten Sie mit dem Kommando `lpstat` in Kombination mit der Option `-S` sowie dem Argument `all`:

```
$ lpstat -S all RETURN
```

Auf dem Bildschirm wird jetzt eine Übersicht über sämtliche Zeichensätze und Typenräder angezeigt, die Ihnen auf dem Drucker Ihres Systems zur Verfügung stehen. Um festzustellen, ob bestimmte Zeichensätze bzw. Typenräder verfügbar sind, geben Sie diese Zeichensätze bzw. Typenräder in der Kommandozeile statt des Arguments `all` ein:

```
$ lpstat -S "zsatz_1 trad_3" RETURN
```

Die Anführungszeichen, durch die die beiden Argumente (`zsatz_1` und `trad_3`) der Option `-S` eingeschlossen sind, sind wegen des Leerzeichens zwischen den beiden Argumenten erforderlich (die Anführungszeichen können entfallen, wenn die Argumente durch Komma voneinander getrennt sind).

Eine ausführliche Übersicht erhalten Sie beim Kommando `lpstat`, indem Sie es mit der Option `-l` aufrufen. In der so erzeugten Übersicht sind dann zu jedem angeforderten Punkt die folgenden Informationen enthalten: Eine Auflistung der Drucker, auf denen jeder Zeichensatz bzw. jedes Typenrad verfügbar ist; Informationen darüber, ob der Zeichensatz bzw. das Typenrad installiert ist, und auf welchen eingebauten Zeichensatz er/es abgebildet werden kann.

Änderung einer Druckanforderung

Angenommen, Sie haben bei der eben abgeschickten Druckanforderung vergessen, eine andere Seitenlänge anzugeben. In einem solchen Fall können Sie an Ihrer Druckanforderung Änderungen vornehmen - selbstverständlich vorausgesetzt, daß der Ausdruck noch nicht erstellt wurde. Dafür rufen Sie einfach das Kommando `lp` erneut auf, wobei Sie zusätzlich die Option `-i` sowie die Auftragsnummer Ihrer Anforderung in die Kommandozeile eintragen. Die Option `-i` signalisiert Ihre Absicht zur Änderung der angegebenen Druckanforderung.

Wenn die ursprüngliche Anforderung beispielsweise mit der Seitenlänge 50, der Breite 70, ohne Titelseite und in dreifacher Ausfertigung ausgegeben werden soll, so geben Sie die folgende Kommandozeile ein:

```
$ lp -d drucker2-23 -o "length=50,width=70,nobanner"\
-n 3 july.report
 request id is printer2-23
```

(Die dritte Zeile im obigen Beispiel ist die Ausgabe des Systems auf Ihre Kommandozeile). Wenn Ihnen später einfällt, daß Sie eigentlich mit einer größeren Seitenlänge ausdrucken wollten, geben Sie folgendes Kommando ein:

```
$ lp -i drucker2-23 -o "length=60,width=70,\
nobanner" RETURN
```

Beachten Sie, daß die Änderungsanforderung von den ursprünglichen Optionen (`-o` und `-n`) nur noch eine einzige Option (`-o`) enthält. In der Änderungsanforderung sollten Sie nur diejenigen Optionen angeben, an denen Sie im Vergleich zur ursprünglichen Kommandozeile etwas ändern möchten.

Dieses Beispiel zeigt aber auch, daß Sie bei einer Änderung der Werte bei der Option `-o` nicht nur die neuen bzw. geänderten Argumente angeben müssen, sondern auch diejenigen Argumente, die Sie beibehalten möchten (dies gilt auch für die Option `-y`). Sehen Sie sich die Kommandozeilen im obigen Beispiel noch einmal an. Für die Option `-o` sind drei Argumente angegeben worden: `length`, `width` und `nobanner`. Obwohl nur ein Argument von `-o` geändert worden ist (von `"length=50"` in `"length=60"`), sind in der Änderungsanforderung alle Argumente aufgeführt. Die Wiederholung der Argumente `width` und `nobanner` ist unerlässlich; andernfalls werden sie nicht aus der ursprünglichen Kommandozeile übernommen.

Stornieren einer Anforderung: das Kommando `cancel`

Das Stornieren einer bereits abgeschickten Druckanforderung ist jederzeit möglich, vorausgesetzt, Sie haben die Anforderung selbst abgeschickt und fordern die Stornierung auf demselben System an, auf dem Sie die Anforderung zuvor abgeschickt hatten. Zur Stornierung einer Anforderung rufen Sie das Kommando `cancel` auf.

Beim Kommando `cancel` können Sie zwei verschiedene Argumente eingeben: Auftragsnummerns oder Druckernamen. Zur Stornierung einer einzigen Druckanforderung rufen Sie `cancel` *auftrags_nummer* auf. Um lediglich den aktuell laufenden Druckauftrag abzubrechen, rufen Sie `cancel` *drucker_name* auf; die übrigen Anforderungen, die in der Warteschlange des angegebenen Druckers enthalten sind, werden nicht storniert. In einer `cancel`- Kommandozeile können Argumente beiden Typs enthalten sein.

Um eine bestimmte Druckanforderung zu stornieren, rufen Sie das Kommando `cancel` mit einer Auftragsnummer auf. Bei der Datei `letters` (Auftragsnummer `laser-6885`) geben Sie also folgendes ein:

```
$ cancel laser-6885 RETURN
```

Sie können aber auch zwei oder mehr Druckanforderungen stornieren, indem Sie nach dem Kommando `cancel` die Option `-u` eingeben. Zur Stornierung sämtlicher Anforderungen, die Sie an einen bestimmten Drucker geschickt haben, geben Sie folgendes ein:

```
cancel -u benutzer_nummer drucker
```

Um sämtliche Anforderungen zu stornieren, die Sie an sämtliche Drucker geschickt haben, geben Sie folgendes ein:

```
cancel -u benutzer_nummer all
```

Nach der Durchführung des Kommandos `cancel` ist der angegebene Druckauftrag aus der Warteschlange entfernt worden.

Dieses Kommando können Sie jederzeit aufrufen, es sei denn, der Druckauftrag ist bereits abgeschlossen.

Aktivieren und Deaktivieren eines Druckers

Hinweis: Die Kommandos `enable` und `disable` stehen den Benutzern nicht in jedem Fall zur Verfügung. Da das Aktivieren und Deaktivieren der Drucker zu den Systemverwaltungsfunktionen gehört, bleibt es dem Systemverwalter überlassen, welche Benutzer Zugriff auf diese Kommandos haben.

Bevor ein Drucker mit dem Ausdrucken der beim Kommando `lp` angegebenen Dateien beginnen kann, muß er aktiviert werden. Hierzu wird das Kommando `enable` mit einem Argument (ein oder mehrere Drucker) aufgerufen:

```
$ enable druckerl drucker2 drucker3 RETURN
```

Informationen darüber, ob ein Drucker erfolgreich aktiviert worden ist, können Sie mit Hilfe einer Statusabfrage erfahren (siehe Abschnitt "Abfrage von Statusinformationen über die Drucker" an einer früheren Stelle dieses Kapitels).

Es kann vorkommen, daß Sie einen bereits laufenden Druckvorgang abbrechen möchten (z.B. wegen einer Drucker-Störung, eines Papierstaus, oder Papiermangels, oder weil das System abends heruntergefahren worden ist). Um in einem derartigen Fall den Druckvorgang abzubrechen, deaktivieren Sie den Drucker mit dem Kommando `disable`:

```
$ disable druckerl RETURN
```

Der Drucker unterbricht daraufhin den aktuellen Druckvorgang (nach der erneuten Aktivierung setzt er den Druckvorgang an der Stelle fort, an der er unterbrochen worden ist).

Beim Unterbrechen eines Druckvorgangs haben Sie noch zwei weitere Möglichkeiten. Vielleicht möchten Sie den Druckvorgang vor dem Deaktivieren des Druckers abschließen; dies ist mit der Option `-W` möglich.

In manchen Fällen spielt es jedoch keine Rolle, ob der Druckvorgang vor der Deaktivierung des Druckers abgeschlossen wird (z.B., wenn die Druckausgabe sehr viele Druckfehler wie unlesbaren Text aufgrund von Tonermangel enthält), da Sie den Druckvorgang
nach der Behebung der Störung ohnehin nochmals von vorn starten möchten. In einem solchen Fall sollte bei der Deaktivierung des Druckers gleichzeitig der Druckvorgang abgebrochen werden; hierzu geben Sie die Option `-c` ein, um den Druckauftrag mit der Deaktivierung des Druckers "hinauszuwerfen". Die Optionen `-W` und `-c` dürfen in keinem Fall zusammen in der Kommandozeile

enthalten sein.

Hinweis: Die Optionen `-c` und `- W` dürfen nicht für Druckaufträge angegeben werden, die an einen fernen Drucker gerichtet sind. Der Grund dafür liegt darin, daß die Kommandos `enable` und `disable` tatsächlich nicht die Drucker an fernen Systemen aktivieren oder deaktivieren, sondern die Dateiübertragungen zu einem fernen System. Daher werden die Optionen `-c` und `-W` ignoriert, wenn das Kommando `disable` für einen fernen Drucker aufgerufen wird.

Noch ein nützlicher Hinweis zum Schluß: Bei der Deaktivierung eines Druckers sollten Sie die Ursache auf jeden Fall protokollieren, damit die übrigen Benutzer den Grund für die Stillegung des Druckers in Erfahrung bringen können. Zur Protokollierung geben Sie die Option `-r` sowie den Grund in die Kommandozeile ein. Eine aus zwei oder mehreren Wörtern bestehende Angabe müssen Sie in Anführungszeichen (") einschließen, damit sie als ein einziges Argument interpretiert wird.

```
$ disable -r "disabling for reconfiguration"\
drucker4 RETURN
```

Der von Ihnen angegebene Grund wird immer dann beim Kommando `lpstat` angezeigt, wenn ein Benutzer Statusinformationen über diesen Drucker anfordert. Angenommen, Sie geben als Grund für die Deaktivierung des Druckers pdq10 `Papierstau` an; ruft ein Benutzer dann später zur Statusabfrage über pdq10 das Kommando `lpstat` auf, so erhält er die folgende Antwort:

```
$ lpstat -p pdq10
printer pdq10 disabled since July 18 10:15-
      papierstau
$
```

Bei der Deaktivierung eines Druckers ohne Angabe eines Grundes enthält die nachfolgende Ausgabe des Kommandos `disable` die Angabe `unknown reason`.

Formatierung der Druckausgabe mit dem Kommando lp

Beim Druck-Service LP können Sie das Aussehen Ihrer Druckausgabe über die Optionen des Kommandos `lp` beeinflussen. In diesem Abschnitt geht es um die Parameter, für die Sie Werte eingeben können:

- Dateiformat.
- Papierformat, Zeilenabstand und Zeichendichte
- Bei mehreren Dateien: Beginn einer neuen Seite mit jeder neuen Datei.
- Ausdrucken einer "Titelseite".
- Bedrucken von normalem Papier oder Formularen (z.B. Rechnungsvordrucke oder Briefbögen).
- Benutzung eines Nicht-Standard-Zeichensatzes bzw. -Typenrads.
- Verschiedene Parameter zur Formatierung des Druckauftrags (z.B. ein- und beidseitiges Drucken), die unter dem Begriff "Spezielle Druck-Modi" zusammengefaßt werden.
- Anzahl der Kopien

Angabe des Dateiformats

Beim Ausdrucken einer Datei muß ein Drucker den Inhalt der Datei ordnungsgemäß interpretieren können. Die diesbezüglichen Möglichkeiten unterscheiden sich von Drucker zu Drucker; nicht jeder Drucker kann jedes Dateiformat verarbeiten. Um sicherzustellen, daß der Druck-Service LP Ihre Anforderung an einen geeigneten Drucker übermittelt, rufen Sie das Kommando `lp` mit der Option `-T` auf:

Mit der Option `-T` können Sie das Format des auszudruckenden Dateiinhalts angeben. Angenommen, Sie möchten eine Datei ausdrucken, die die Bilanz für den Monat Juli enthält (`july.report`). Der Inhalt dieser Datei sei in das Format 455 gebracht worden (er kann also durch den Drucker AT&T Model 455 interpretiert werden).Sie wissen, daß es in Ihrem System mehrere Model 455 Drucker gibt, kennen aber ihre Namen nicht. Mit der Option `-T` können Sie dann einen Model 455 Drucker anfordern, ohne explizit einen Namen anzuge-

ben:

```
$ lp -T 455 -d any july.report RETURN
```

Die Option `-T` weist den Druck-Service an, einen beliebigen der Drucker auszuwählen, der eine Datei im Format 455 ausdrucken kann. Zur Angabe eines bestimmten Druckers können Sie —selbst wenn es sich dabei um den Standard-Drucker handelt—die Option `-d` benutzen.

Was tun Sie, wenn es keine Model 455 Drucker in Ihrem System gibt? Die Vorgehensweise hängt davon ab, ob für Ihr System Filter definiert worden sind (erkundigen Sie sich bei Ihrem Systemverwalter). Als Filter wird ein Programm bezeichnet, durch das Daten in ein anderes Format gebracht werden (in diesem Fall werden die Daten von dem Format, in dem sie in die Datei eingebracht worden sind, in ein Format umgesetzt, das vom Drucker "verstanden" wird). Wenn es überhaupt keine Drucker gibt, die das Format Ihrer Datei verarbeiten können und Filter für Ihr System definiert worden sind, so wird Ihre Druckanforderung an einen Filter übermittelt (bei Bedarf wird die Datei auch an mehrere Filter geschickt. So können Sie die Datei hintereinander an einen `troff`- Filter, einen `Postscript`- Filter, einen `Seitenauswahl`- Filter sowie an einen Filter zum Laden von Schriften schicken, um für die Datei verschiedene Funktionen durchzuführen). Der Inhalt der Datei wird durch den Filter in ein Format gebracht, das vom Drucker verarbeitet werden kann. Wenn es jedoch weder einen Drucker gibt, der das Format Ihrer Datei verarbeiten kann, noch einen Filter, der die Datei umwandeln könnte, so wird Ihre Druckanforderung abgewiesen.

Filter geben Ihnen die Möglichkeit, Ihre Dateien auf den unterschiedlichsten Druckern auszugeben. Es kann jedoch vorkommen, daß für einen bestimmten Druckauftrag ein bestimmtes Format obligatorisch ist, d.h., die Datei entweder im angeforderten Format ausgegeben werden muß, oder überhaupt nicht. Wenn Ihr System Filter unterstützt, Ihre Druckanforderung aber auf keinen Fall an einen Filter geschickt werden soll, so geben Sie in der `lp`-Kommandozeile nach der Option `-T` die Option `-r` ein:

```
$ lp -T 455 -r july.report RETURN
```

Die Option `-r` bewirkt, daß die Druckanforderung wegen ihres Formats abgewiesen wird, wenn sie von keinem Drucker im System bearbeitet werden kann.

Hinweis: Die Installation und Wartung der Filter für den Druck-Service LP ist Aufgabe des Systemverwalters. Erkundigen Sie sich beim Systemverwalter nach den Dateiformaten, die von den Druckern Ihres Systems verarbeitet werden können.

Angabe von Papierformat, Zeilenabstand und Zeichendichte

Das Papierformat wird über die Länge und die Breite angegeben. Mit dem Zeilenabstand wird die Anzahl der Zeichen pro Zoll in vertikaler Richtung angegeben, mit der Zeichendichte die Anzahl der Zeichen pro Zoll in horizontaler Richtung. Zur Einstellung dieser Druckparameter für eine Druckanforderung gibt es vier Möglichkeiten:

- Über die Standardparameter des Druckers.
- Über die Standardparameter, wie sie vom Systemverwalter festgelegt worden sind.
- Über die Parameter des speziellen Formulars, das Sie ausgewählt haben
- Über die Parameter, die Sie speziell für diesen Druckauftrag angegeben haben.

Wenn Sie für einen Druckauftrag spezielle Parameter definieren möchten, so geben Sie in der `lp`-Kommandozeile die Option `-o` ein und legen die gewünschten Werte in "skalierten Dezimalzahlen" fest.

Unter einer "skalierten Dezimalzahl" versteht man eine nicht-negative Zahl, mit der eine Maßeinheit angegeben wird (die jeweilige Maßeinheit wird durch einen der Zahl "nachgestellten" Buchstaben angegeben). Im folgenden werden drei Kategorien von skalierten Dezimalzahlen für den Druck-Service LP beschrieben: Zahlen zur Größenangabe in Zentimeter (gekennzeichnet durch ein nachgestelltes "c"), Zahlen zur Größenangabe in Zoll (gekennzeichnet durch ein nachgestelltes "i" für inch), außerdem Zahlen ohne nachgestellten Buchstaben, bei denen die Maßeinheit abhängig ist vom jeweiligen Druckparameter (z.B. der Anzahl der Zeilen und Spalten oder der Anzahl der Zeichen/Zeilen pro Zoll).

Die folgende Kommandozeile zeigt, wie Sie bei einem Druckauftrag die Druckparameter Papierformat und Zeilen-/Zeichenabstand einstellen können (die Werte werden in *sdz* bzw. skalierten Dezimalzahlen angegeben.

```
$ lp -d any -o "length=sdz width=sdz lpi=sdz\
cpi=sdz" dateiname RETURN
```

Ihr Dokument wird mit den Standard-Parametern des angegebenen Druckers ausgedruckt, wenn eine der folgenden Bedingungen erfüllt ist: (1) Sie haben für Ihre Druckanforderung keine anderen Druckparameter definiert oder (2) Sie benutzen keinen Drucker, für den vom Verwalter spezielle Parameter angegeben worden sind. Diese Standard-Parameter sind in einer Datenbank namens Terminfo enthalten, die vom Systemverwalter eingerichtet und verwaltet wird; wenden Sie sich bei Fragen an den Systemverwalter.

Benutzen Sie beispielsweise den AT&T-Drucker Model 455, so sind für Ihren Drucker die folgenden Standard-Parameter vorgegeben:

Seitenlänge: 66 Zeilen
Breite: 132 Spalten
Zeilenabstand: 6 Zeilen pro Zoll
Zeichenabstand: 12 Zeichen pro Zoll

Beim AT&T-Drucker Model 470 dagegen sehen die Standard-Parameter geringfügig anders aus:

Seitenlänge: 66 Zeilen
Breite: 80 Spalten
Zeilenabstand: 6 Zeilen pro Zoll
Zeichenabstand: 10 Zeichen pro Zoll

Seitenumbruch zwischen Dateien ausschalten

Ihre Druckanforderung kann aus zwei oder mehr Dateien bestehen. Der Druck-Service LP geht standardmäßig davon aus, daß Sie für jede neue Datei eine neue Seite anfangen möchten. Sollen die Dateien auf dem Ausdruck direkt aufeinanderfolgen, so geben Sie die Option `-o` in folgendem Format an:

```
$ lp -d any -o nofilebreak dateinamen RETURN
```

Ausdrucken ohne Titelseite

Der Druck-Service LP schaltet jedem Druckauftrag eine Titelseite ("banner page") vor. Um dies zu verhindern, geben Sie die Option `-o` in folgendem Format ein:

```
$ lp -d any -o nobanner dateiname (RETURN)
```

Der Systemverwalter kann diese Option für bestimmte Drucker ausschalten; jede Anforderung, die Sie an einen dieser Drucker schicken, wird dann ohne Titelseite ausgegeben.

Benutzung von vorgedruckten Formularen

In vielen Unternehmen müssen häufig spezielle Dokumente wie z.B. Gehaltszettel oder Rechnungen ausgedruckt und verschickt werden. Beim Druck-Service LP können Sie Ihre Dateien auf Formularen ausgeben, die der Verwalter in den Drucker eingelegt hat. Erkundigen Sie sich bei Ihrem Systemverwalter, welche Formulare auf Ihrem Drucker zur Verfügung stehen. Zum Bedrucken eines speziellen (verfügbaren) Formulars geben Sie in der `lp`- Kommandozeile die Option `-f` sowie den Namen des Formulars ein. Um beispielsweise die Datei `april.payroll` über den Drucker "drucker4." auf dem Formular `paycheck` auszugeben, geben Sie folgendes ein:

```
$ lp -d drucker4 -f paycheck april.payroll (RETURN)
```

Wird das angeforderte Formular vom Drucker nicht unterstützt, so wird Ihre Anforderung abgewiesen. Um sicherzustellen, daß Ihre Anforderung von jedem Drucker akzeptiert wird, in den das angeforderte Formular eingelegt werden kann, geben Sie in der Kommandozeile die Option `-d` sowie das Argument `any` ein:

```
$ lp -d any -f form_name dateiname (RETURN)
```

Der Druck-Service LP schickt Ihre Anforderung dann an jeden Drucker, der das angeforderte Formular bedrucken kann. Sind bei Ihrem Druck-Service LP sowohl lokale als auch ferne Drucker eingetragen, so versucht das Kommando zuerst, Ihren Druckauftrag an einen lokalen Drucker zu schicken; erst dann übermittelt es ihn an einen fernen Drucker.

Wahl des Zeichensatzes bzw. Typenrads

Beim Kommando `lp` können Sie den Zeichensatz bzw. das Typenrad angeben, mit dem Ihr Druckauftrag ausgedruckt werden soll. Eine Übersicht über die verfügbaren Typenräder und Zeichensätze erhalten Sie über das Kommando `lpstat -S`.

Zur Anforderung eines bestimmten Zeichensatzes bzw. Typenrads geben Sie in der `lp`- Kommandozeile die Option `-S` ein:

```
$ lp -d any -S z_satz dateiname (RETURN)
```

Wenn Sie weder einen bestimmten Zeichensatz bzw. ein bestimmtes Typenrad wünschen noch für Ihren Druckauftrag ein bestimmter Zeichensatz bzw. ein bestimmtes Typenrad erforderlich ist, so können Sie diese Option auslassen.

Spezielle Druck-Modi

Das letztendliche Erscheinungsbild des ausgedruckten Texts hängt nicht nur von seinem Inhalt ab, sondern auch von anderen Parametern, die das Format der Druckseite beeinflussen. So können Sie für Ihren Text beispielsweise eine weniger gängige Schrift verwenden. Die Bandbreite der verfügbaren speziellen Druck-Modi hängt von den verfügbaren Druckern ab.

Um für einen Druckauftrag spezielle Druck-Modi anzufordern, geben Sie in der Kommandozeile die Option `-y` ein:

```
$ lp -d any -y druck_modi dateiname (RETURN)
```

Die Druck-Modi sind dabei aus einem Wort bestehende Namen, die aus einer beliebigen Kombination von Buchstaben und Zahlen bestehen.

Der Drucker akzeptiert Ihre Anforderung unter der Voraussetzung, daß alle angeforderten Druck-Modi von dem "Filter" unterstützt werden, also der Schnittstelle zwischen Ihrer Druckanforderung und dem Drucker. Erkundigen Sie sich bei Ihrem Systemverwalter, welche Filter in Ihrem System verfügbar sind und welche -y -Optionen Sie benutzen können.

Anfordern von Mehrfach-Kopien

Wenn Sie von Ihrem Text mehr als eine Kopie benötigen, können Sie mit der Option `-n` (für "number") Mehrfach-Kopien anfordern. Für vier Kopien geben Sie beispielsweise eine Kommandozeile in folgendem Format ein:

```
$ lp -d any -n 4 dateiname RETURN
```

Ohne diese Option wird standardmäßig nur eine einzige Kopie erstellt.

Arbeiten mit PostScript-Druckern

PostScript ist eine universelle Programmiersprache wie C oder Pascal. Zusätzlich zu den üblichen Funktionen einer Programmiersprache bietet PostScript Ihnen Möglichkeiten zur Beeinflussung des Erscheinungsbilds von Text und Graphik auf der Druckseite, die über die Möglichkeiten von anderen Druckern hinausgehen. So können Sie beispielsweise geometrische Figuren erstellen und sie in jeder beliebigen Größe und Ausrichtung an eine beliebige Stelle der Druckseite setzen. Zum Ausdrucken von Text stehen Ihnen eine Vielzahl von Schriften zur Auswahl, die in jeder Größe und Ausrichtung an eine beliebige Stelle der Druckseite gesetzt werden können. Die Kombination von Graphik und Text ist kein Problem. Ein zusätzlicher Vorteil von PostScript besteht darin, daß die Dateien sowohl auf Druckern mit hoher Auflösung ausgegeben werden können, als auch auf Druckern mit niedriger Auflösung. Kurzum - mit PostScript-fähigen Druckern können Sie Ihre Texte wesentlich abwechslungsreicher und professioneller gestalten als mit anderen Druckern.

PostScript-Dateien können ausschließlich auf PostScript-fähigen Druckern ausgegeben werden. Bei diesen Druckern handelt es sich in Wirklichkeit um spezielle, auf einen bestimmten Zweck zugeschnittene Rechner, die PostScript-Dateien interpretieren können. Dateien, die an einen PostScript-Drucker übermittelt werden, müssen in PostScript geschrieben sein, es sei denn, der Drucker-Hersteller hat die Möglichkeit zum Umschalten in einen anderen Druck-Modus vorgesehen. Dennoch ist es zum Erstellen von PostScript-Dateien nicht notwendig, daß Sie sich in die PostScript-Kommandosprache einarbeiten.

Hinweis: PostScript wird von vielen gängigen Anwendungsprogrammen (z.B. Textverarbeitungs-, Tabellenkalkulations-, Desktop Publishing- und CAD-Programmen) unterstützt. Wenn auf Ihrem Rechner ein derartiges Programm installiert ist, können Sie Ihre Dateien ganz normal eingeben; das Programm setzt die Formatanweisungen in PostScript-Anweisungen um.

Erkundigen Sie sich bei Ihrem Systemverwalter, ob PostScript von Ihrem Programm unterstützt wird.

Mit installierten PostScript-Druckern und -Filtern behandelt der Druck-Service LP eine PostScript-Datei wie jede andere Datei. Um das Ausdrucken einer PostScript-Datei auf einem PostScript-Drucker anzufordern, müssen Sie lediglich in der Kommandozeile diesen Drucker sowie das Dateiformat angeben:

```
lp -dpsdrucker -Tpostscript psdatei
```

Wenn der Drucker (*psdrucker*) beim Druck-Service LP als PostScript-Drucker angemeldet worden ist, setzt der Druck-Service Ihre Anforderung in die Warteschlange und übermittelt sie an den Drucker.

Unterstützung von Nicht-PostScript-Druckanforderungen

Der Druck-Service LP bietet Ihnen einen "Übersetzungs- Service": Sie können eine Datei in einem bekannten Format eingeben; der Druck-Service übersetzt die Datei dann vor der Übermittlung an einen PostScript-Drucker nach PostScript. Hierzu durchläuft Ihre Datei einen Filter, durch den das "Format" Ihrer Datei nach PostScript umgesetzt wird. Stehen Ihnen derartige Filter zur Verfügung, so können Sie Ihre Dateien auf PostScript-Druckern ausdrucken, ohne daß Sie sich in ein neues Format einarbeiten müssen.

Da für jedes Dateiformat ein separater Filter erforderlich ist und die UNIX-Benutzer Dateien in einer Vielzahl von unterschiedlichen Formaten erstellen, stellt der Druck-Service zur Umsetzung nach PostScript eine große Anzahl von Filtern zur Verfügung. Wenn eine Datei umgesetzt werden soll, müssen Sie daher beim Abschicken der Druckanforderung (d.h. beim Aufrufen des Kommandos `lp`) die Umsetzung anfordern und das Format der Datei angeben. Im folgenden sind einige Dateiformate aufgeführt, die vor der Verarbeitung auf einem PostScript-Drucker umgesetzt werden müssen.

`troff`	Druckausgabe von `troff`- Dateien.
`simple`	Ausgabe einer ("reinen") ASCII- Textdatei.
`dmd`	Ausgabe der Bitmuster-Anzeige eines Terminal-Bildschirms (z.B. vom Typ AT&T 630).

`tek4014`	Ausgabe von Dateien, die für einen Drucker vom Typ Tektronix 4014 formatiert worden sind.
`daisy`	Ausgabe von Dateien, die für einen ("Typenrad-") Drucker vom Typ Diablo 630 formatiert worden sind.
`plot`	Ausgabe von Dateien im Plotter-Format

Um beispielsweise eine ASCII- Datei nach PostScript umzusetzen, liest der Filter den Text ein und durchsetzt ihn mit Programmcode, durch den Druckparameter wie die Schriften und das Format der Druckseite angegeben werden.

Die Filter, durch die diese Umsetzungen vorgenommen werden, werden automatisch von LP aufgerufen, wenn ein Benutzer in einer Druckanforderung eines der oben aufgeführten Formate bei der Option `-T` angibt. So wird mit der Kommandozeile

```
lp -dpsdrucker -Tsimple report2
```

die ASCII- Datei `report2` (eine Datei im "reinen" ASCII-Format) nach PostScript umgesetzt, vorausgesetzt, der Ziel-Drucker `psdrucker` ist beim System als PostScript- Drucker eingetragen worden. Das Standard-Dateiformat ist `simple`.

Zusätzliche PostScript-Möglichkeiten durch Filter

Die oben beschriebenen Filter nutzen die PostScript-Möglichkeiten auch für zusätzliche Gestaltungsmöglichkeiten der Druckseite, die zum größten Teil über die "speziellen Druck-Modi" (die mit der Option `-y` aufgerufen werden) des Kommandos `lp` aktiviert werden. Diese Filter eröffnen Ihnen eine Reihe von außergewöhnlichen Gestaltungsmöglichkeiten für Ihre Druckaufträge. Diese Möglichkeiten sind im folgenden zusammen mit den Optionen aufgeführt, durch die sie in der `lp`- Kommandozeile aufgerufen werden.

Format	Art der Druckanforderung
`-yreverse`	Die Seiten werden in umgekehrter Reihenfolge ausgedruckt.
`-ylandscape`	Die Ausrichtung der logischen Seite wird vom Hoch- ins Querformat geändert.
`-yx=`*xzahl*`,y=`*yzahl*	Die Standard-Position der logischen Seite auf einer physischen Seite wird geändert, indem die Koordinaten der obersten linken Ecke verschoben werden.
`-ygroup=`*zahl*	Mehrere logische Seiten werden auf eine einzige physische Seite gesetzt.
`-ymagnify=`*zahl*	Die logische Seite wird vergrößert oder verkleinert.
`-P`*zahl*	Das Schriftstück soll nur teilweise ausgedruckt werden; *zahl* steht für die auszudruckenden Seiten.
`-n`*zahl*	Von einem Schriftstück sollen mehrere Kopien erstellt werden.

Hinweis: Sollen diese Filter zusammen mit einem Anwendungsprogramm benutzt werden, durch das Dateien im PostScript-Format erstellt werden, so müssen Sie sicherstellen, daß das Anwendungsprogramm mit Befehlen im Format der PostScript-Strukturkommentare arbeitet. Achten Sie insbesondere darauf, daß der Beginn jeder PostScript-Seite mit dem Kommentar

```
%%Page:label zahl
```

gekennzeichnet ist, wobei *zahl* eine positive Ganzzahl ist, durch die die laufende Nummer der Seite innerhalb des Schriftstücks angegeben wird.

Angenommen, Sie möchten von der Datei `report2` mit dem Format `simple` (also einer ASCII- Datei) sechs Seiten (Seite 4 bis 9) ausdrucken, wobei auf jeder physischen Seite zwei logische Seiten enthalten sein sollen. Da einer der Drucker in Ihrem System (psdrucker) PostScript-fähig ist, können Sie dies mit dem folgenden Kommando erreichen:

```
lp -dpsprinter-Tsimple -P4-9 -ygroup=2 myfile
```

Der zuständige Filter versucht, den verfügbaren Platz beim Zusammenfassen von jeweils zwei logischen Seiten auf einer physische Seite möglichst effizient auszunutzen. Daher werden die Seiten in zwei parallelen Spalten ausgedruckt; die physische Seite wird also ins Querformat gedreht. Würde explizit der Modus Querformat gewählt, der sich nicht auf die Ausrichtung der physischen Seite auswirkt, sondern auf die der logischen Seite, so würden die logischen Seiten untereinandergesetzt.

Der Druck-Service LP bietet außerdem einen speziellen Filter, der die Halbton-Darstellung einer Matrix ermöglicht (darunter versteht man ein Bild, in dem jede "Zelle" ihrem Wert entsprechend in einer von sieben Halbtönen oder Graustufen getönt ist. Dunklere Graustufen entsprechen höheren Werten). Zum Ausdrucken einer Halbton-Darstellung geben Sie bei der Option `-T matrix` als Format Ihrer Ausgangsdatei an.

Die Abmessung der Matrix entspricht dabei standardmäßig der Quadratwurzel aus der Anzahl der Elemente in der Matrix, es sei denn, Sie geben die Anzahl der Zeilen und Spalten bei der Option - `ydimen=`*anzahl_zeilen*`x`*anzahl_spalten* an. Die Werte der Zellen, die in die verschiedenen Graustufen umgesetzt werden, können durch `-yinterval=`*liste* angegeben werden. `-1/0/1` ist die Standard-Liste. Dadurch werden die Elemente der Matrix in sieben Bereiche aufgeteilt: x<−1, x=− 1, −1<x<0, x=0, 0<x<1, x=1, 1<x. In *liste* können maximal drei durch Schrägstrich voneinander getrennte Zahlen enthalten sein.

Arbeiten mit PostScript-Schriften

Zu den wichtigsten Vorteilen von PostScript gehören die Möglichkeiten zur Schriftenverwaltung. Die Schriften sind im Konturen-Format entweder im Drucker oder im Rechner, über den der Drucker angesteuert wird, abgespeichert. Beim Ausdrucken eines Schriftstücks generiert der PostScript-Interpreter jedes Zeichen wie angefordert (in der entsprechenden Größe) aus seiner Konturen-Beschreibung. Ist die Schrift, die für ein Schriftstück angefordert worden ist, nicht im benutzten Drucker eingespeichert, so muß sie vor dem Beginn des Druckvorgangs an den Drucker übermittelt werden. Die Übertragung wird als "Laden von Schriften" bezeichnet. Häufig ist auch der englische Begriff "Downloading" anzutreffen.

Zur Speicherung und Benutzung von Schriften gibt es verschiedene Möglichkeiten.

- Die Schriften können im Drucker eingespeichert sein. Dabei können sie entweder permanent im Drucker eingespeichert sein oder bei jedem Einschalten des Druckers vom Verwalter in den Druckerspeicher geladen werden. Erkundigen Sie sich beim Druck-Service-Verwalter nach den Schriften, die auf Ihren Druckern verfügbar sind.
- Die Schriften können in Ihrem Home-Verzeichnis enthalten sein, so daß sie für Ihre Ausdrucke zur Verfügung stehen. Sobald Sie bei einer Druckanforderung eine Schrift aus Ihrem Verzeichnis benötigen, wird diese Schrift, zusammen mit der auszudruckenden Datei, an den Drucker übermittelt. Diese Konstellation ist besonders für weniger gängige Schriften empfehlenswert. Im allgemeinen wird die Schrift durch das Anwendungsprogramm, mit dem Sie die PostScript-Datei erstellt haben, unmittelbar vor der PostScript-Datei an den Druck-Service übermittelt.
- Auf einem System, an dem mehrere Benutzer arbeiten, können die Schriften in einem "öffentlichen" Verzeichnis gespeichert sein. Diese sogenannten "host-residenten" Schriften muß ein Benutzer über ein Anwendungsprogramm anfordern, mit dem PostScript-Dateien erstellt werden können. Bei der Erstellung der PostScript-Datei durch das Anwendungsprogramm müssen Anforderungen für alle benötigten Schriften eingebracht werden. Diese Methode ist immer dann empfehlenswert, wenn die Anzahl der Schriften die Kapazität des Druckers übersteigt.

Der Druck-Service LP unterstützt sämtliche aufgeführten Möglichkeiten zur Schriftenverwaltung.

Der Druck-Service LP weist einen speziellen Lade-Filter auf, mit dem die Schriften nach der letzten der aufgeführten Methoden verwaltet werden können. Die Prozedur wird durch den Druck-Service automatisch, ohne Ihr Zutun, erledigt.

Laden von host-residenten Schriften

Der Filter zum Laden von host-residenten Schriften erledigt folgende Aufgaben:

- Er durchsucht das PostScript-Schriftstück nach den angeforderten Schriften. Die Schrift-Anforderungen sind im Kopfteil im Format von PostScript-Strukturkommentaren enthalten:

  `%%DocumentFonts:` *schrift1 schrift2* `...`

- Er durchsucht die Schriftenliste auf dem Ziel-Drucker, um festzustellen, ob die angeforderte Schrift geladen werden muß. Ist die Schrift auf diesem Drucker nicht vorhanden, so durchsucht der Filter das Verzeichnis mit den host-residenten Schriften. Ist die angeforderte Schrift hier vorhanden, so lädt er die entsprechende Datei und übermittelt sie unmittelbar vor der auszudruckenden Datei an den Drucker.

Hinweis: Die zu ladenden Schriften ermittelt der Ladefilter anhand der PostScript-Strukturkommentare. Wenn Sie daher mit ladbaren Schriften arbeiten möchten, müssen Sie sicherstellen, daß die Schrift-Anforderungen in Ihrem Anwendungsprogramm den Konventionen der PostScript-Strukturkommentare entsprechen.

Die Kommandos des Druck-Service LP

Bild 8-2: Druckkommandos und ihre Funktionen

Kommando	Funktion
`lp`	Druckausgabe einer Datei anfordern.
`cancel`	Anforderung für die Druckausgabe einer Datei stornieren.
`lpstat`	Informationen über den aktuellen Status des Druck-Service LP ausgeben.
`enable`	Drucker aktivieren, so daß die Anforderungen des Kommandos `lp` bearbeitet werden können.
`disable`	Drucker deaktivieren, so daß die Anforderungen des Kommandos `lp` nicht mehr bearbeitet werden.

In Bild 8-3 bis 8-7 sind Syntax und Funktionen dieser Kommandos nochmals kurz zusammengefaßt.

Bild 8-3: `lp`**-Kurzübersicht**

Kurzbeschreibung		
	`lp` – Anforderung der Druckausgabe einer Datei	
kommando	*optionen*	*argumente*
`lp`	*(siehe liste)*	*datei(en)*
Funktion:	Mit dem Kommando `lp` kann die Druckausgabe einer Datei, und somit ihre Ausgabe in Papierform, angefordert werden.	
Optionen:	`-d` *ziel*	Hiermit können Sie *ziel* als zu benutzende(n) Drucker(klassen) angeben. Diese Option kann weggelassen werden, wenn der Verwalter ein Standard-Ziel definiert hat oder Sie die Umgebungsvariable LPDEST eingestellt haben.
	`-y` *modus*	Hiermit können Sie spezielle Druck-Modi wie z.B. das Ausdrucken im Hoch- oder Querformat anfordern (für diese Option ist ein spezieller Filter erforderlich; erkundigen Sie sich bei Ihrem Systemverwalter, ob es auf Ihrem System einen solchen Filter gibt).
	`-o` *option*	Hiermit geben Sie die Druckparameter (Länge und Breite, Zeilenabstand, Zeichendichte) an (`-o` hat noch weitere Funktionen; siehe hierzu Eintrag `lp`(1) im *Referenzhandbuch für Benutzer*.)
	`-P` *seiten-nr*	Hiermit können Sie die Nummern der Seiten angeben, die ausgedruckt werden sollen (für diese Option ist ein spezieller Filter erforderlich; erkundigen Sie sich bei Ihrem Systemverwalter, ob es auf Ihrem System einen solchen Filter gibt).
	`-n` *kopien*	Hiermit geben Sie die Anzahl der zu erstellenden Kopien an.
	`-f` *vordruck*	Hiermit geben Sie an, auf welchen Formularen die Dateien ausgedruckt werden sollen.

Bild 8-3: `lp`**-Kurzübersicht** (Fortsetzung)

Kurzbeschreibung		
`lp` – Anforderung der Druckausgabe einer Datei		
kommando	*optionen*	*argumente*
`lp`	*(siehe liste)*	*datei(en)*
Optionen:	`-S` *zeich_satz*	Hiermit geben Sie den Zeichensatz bzw. das Typenrad an.
	`-T` *format*	Hiermit geben Sie das Format der Druckanforderung an.
	`-w`	Hiermit geben Sie an, daß Sie von der Beendigung des Druckvorgangs mit einer Meldung auf dem Bildschirm benachrichtigt werden möchten.
	`-m`	Hiermit werden Sie von der Beendigung eines Druckvorgangs über die elektronische Post benachrichtigt.
	`-i` *auftrags_nr*	Hiermit können Sie eine bereits abgeschickte Druckanforderung (vor dem Beginn des Druckvorgangs) abändern.
	`-q` *stufe*	Hiermit können Sie für Ihre Druckanforderung eine Priorität angeben.
Hinweise:	Sie können eine Druckanforderung stornieren, indem Sie das Kommando `cancel` sowie die Auftragsnummer, die das System zur Bestätigung der Druckanforderung angezeigt hat, eingeben.	
	Erkundigen Sie sich bei Ihrem Systemverwalter, welche Kommandos für die Drucker in Ihrem System zusätzlich zur Verfügung stehen.	

Bild 8-4: `lpstat`**-Kurzübersicht**

Funktionsbeschreibung		
`lpstat` –Statusinformationen über den Druck-Service LP		
kommando	*optionen*	*argumente*
`lpstat`	*(siehe Liste)*	*datei(en)*
Funktion:	Das Kommando `lpstat` gibt Statusinformationen über die Druckanforderungen, die Drucker sowie den LP-Druckerverwalter aus, außerdem Informationen im Zusammenhang mit dem Status des Druck-Service.	
Optionen:	`-a` *liste*	Hiermit können Sie sich darüber informieren, ob Druckanforderungen von den angegebenen Druckerklassen angenommen werden.
	`-c` *liste*	Für jeden in der Liste enthaltenen Drucker werden die enthaltenen Klassen ausgegeben.
	`-d`	Das Standard-Ziel Ihres Druck-Service LP wird ausgegeben.
	`-f` *liste* [`-l`]	Hiermit können Sie überprüfen, ob die in *liste* enthaltenen Formulare vom Druck-Service LP akzeptiert werden. Mit der Option `-l` können Sie Beschreibungen der Formulare abrufen.
	`-o` *liste* [`-l`]	Der Status der Druckanforderungen wird ausgegeben. *liste* kann die Namen von Drucker(klasse)n oder Auftragsnummerns enthalten.
	`-p` *liste* [`-D`] [`-l`]	Der Status der in *liste* aufgeführten Drucker wird ausgegeben. Mit der Option `-D` wird zusätzlich eine Beschreibung jedes Druckers ausgegeben, und mit `-l` eine vollständige Beschreibung der Konfiguration jedes Druckers.

Bild 8-4: `lpstat`**-Kurzübersicht** (Fortsetzung)

Funktionsbeschreibung		
`lpstat` –Statusinformationen über den Druck-Service LP		
kommando	*optionen*	*argumente*
`lpstat`	(*siehe Liste*)	*datei(en)*
Optionen:	`-R`	Die Rangfolge der Druckanforderungen in der Warteschlange wird angezeigt.
	`-r`	Informationen über den Status des LP-Druckerverwalter werden ausgegeben.
	`-s`	Für den Druck-Service LP als Ganzes wird eine Statusübersicht ausgegeben.
	`-S` *liste* [`-l`]	Hiermit können Sie überprüfen, ob die Zeichensätze bzw. Typenräder in *liste* durch den Druck-Service LPakzeptiert werden. Die Option `-l` setzt eine Reihe von Druckern voraus, die jeden Zeichensatz und jedes Typenrad unterstützen.
	`-t`	Sämtliche Statusinformationen werden ausgegeben.
	`-u` *liste*	Der Status der Druckanforderungen der einzelnen Benutzer wird ausgegeben. *liste* enthält die Benutzernamen.
	`-v` *liste*	Die Drucker und Pfadnamen der zugehörigen Gerätedateien werden ausgegeben. *liste* enthält Druckernamen.
Hinweise:	*liste* kann entweder eine Liste von Druckernamen oder `all` sein. Ohne die Angabe von Optionen wird standardmäßig die Option `-o` eingesetzt.	

Bild 8-5: `cancel`**-Kurzübersicht**

Kurzbeschreibung		
`cancel` – Mit `lp` abgeschickte Druckanforderungen stornieren		
kommando	*optionen*	*argumente*
`lp`	*(siehe Liste)*	*datei(en)*
Funktion:	Das Kommando `cancel` storniert Druckanforderungen, die mit dem Kommando `lp` abgeschickt worden sind.	
Optionen	[*auftrags_nr...*]	Hiermit können Sie Ihre Druckanforderungen stornieren; geben Sie die Auftragsnummern der betreffenden Anforderungen ein.
	[*drucker*]	Hiermit können Sie jede Druckanforderung stornieren, die aktuell auf dem angegebenen Drucker ausgegeben wird.

Bild 8-6: `enable`**-Kurzübersicht**

Kurzbeschreibung		
`enable` – Drucker des Druck-Service LP aktivieren		
kommando	*optionen*	*argumente*
`enable`	*(siehe Liste)*	*datei(en)*
Funktion:	Mit dem Kommando `enable` können Sie einen Drucker aktivieren, der beim Druck-Service LP angemeldet ist. Das Aufrufen dieses Kommandos muß den Benutzern des Systems explizit vom Systemverwalter erlaubt werden.	
Optionen	keine	
Hinweise:	Mit `lpstat` können Sie den Status der Drucker abfragen.	

Bild 8-7: `disable`**-Kurzübersicht**

Kurzbeschreibung		
`disable` – Drucker des Druck-Service LP deaktivieren		
kommando	*optionen*	*argumente*
`disable`	*(siehe Liste)*	*datei(en)*
Funktion:	Mit dem Kommando `disable` können Sie einen Drucker deaktivieren, der beim Druck-Service LP angemeldet ist. Das Aufrufen dieses Kommandos muß den Benutzern des Systems explizit vom Systemverwalter erlaubt werden.	
Optionen	`-c`	Hiermit können Sie eine oder mehrere Druckanforderungen stornieren, die aktuell auf einem der angegebenen Drucker ausgedruckt werden (diese Option kann nicht mit der Option `-W` kombiniert werden.)
	`-r` *grund*	Hiermit können Sie einen *grund* für die Deaktivierung der Drucker angeben. Der von Ihnen angegebene Grund gilt für sämtliche Drucker, die bis zur nächsten Option `-r` in der Kommandozeile enthalten sind. Der *grund* wird von `lpstat` ausgegeben. Fehlt die Option `-r`, so wird der Standard- Grund ausgegeben.
	`-W`	Hiermit wird mit der Deaktivierung des Druckers gewartet, bis die aktuelle Druckanforderung beendet ist (diese Option kann nicht mit der Option `-c` kombiniert werden).

9 Die Shell

Einführung 9-1

Die Kommandosprache der Shell 9-2

Metazeichen 9-4

- Das Metazeichen, das zu allen Zeichen paßt: Der Stern (*) 9-4
- Das Metazeichen, das zu einem Einzelzeichen paßt: Das Fragezeichen (?) 9-7
- Die Metazeichen, die zu einem Zeichen aus einer Gruppe passen: Die eckigen Klammern ([ ]) 9-8

Sonderzeichen 9-10

- Kommandoausführung im Hintergrund: Das UND-Zeichen (&) 9-10
- Mehrere Kommandos hintereinander ausführen: Das Semikolon (;) 9-11
- Sonderbedeutung aufheben: Der Gegenschrägstrich (\) 9-12
- Sonderbedeutung aufheben: Hochkommata und Anführungszeichen 9-13
- Sonderbedeutung eines Leerzeichens mit Anführungszeichen und Hochkommata aufheben 9-13

Ein- und Ausgabeumleitung 9-15

- Umleitung der Eingabe: Das Kleiner-als-Zeichen (<) 9-16
- Umleitung der Ausgabe in eine Datei: Das Größer-als-Zeichen (>) 9-16
- Ausgabe an eine vorhandene Datei anhängen: Das doppelte Größer-als-Zeichen (>>) 9-17
- Anwendungsmöglichkeiten für die Ausgabeumleitung 9-18
- Verbindung von Hintergrundverarbeitung und Ausgabeumleitung 9-20
- Umleitung der Ausgabe zu einem Kommando: Das Pipe-Symbol (|) 9-21
- Pipeline mit den Kommandos `cut` und `date` 9-22
- Ausgabe eines Kommandos als Argument verwenden 9-27

Ausführung, Abbruch und Neustart von Prozessen 9-27
- Kommandos zu einem späteren Zeitpunkt verarbeiten - Die Kommandos batch und at 9-28
- Abfrage des Status von aktiven Prozessen 9-33
- Abbrechen von aktiven Prozessen 9-35
- Neustart eines angehaltenen Prozesses 9-36
- Das Kommando `nohup` 9-37

Übungen zur Kommandosprache 9-39

Shell-Programmierung 9-41
Shell-Programme 9-41
- Erstellung eines einfachen Shell-Programms 9-41
- Ausführung eines Shell-Programms 9-42
- Erstellung eines `bin`-Verzeichnisses für ausführbare Dateien 9-43
- Hinweise zur Benennung von Shell-Programmen 9-45

Variablen 9-46
- Stellungsparameter 9-46
- Spezielle Parameter 9-50
- Schlüsselwortparameter 9-53
- Wertzuweisung an eine Variable 9-55

Shell-Programmstrukturen 9-62
- Kommentare 9-63
- Das Here-Dokument 9-63
- Der Editor `ed` in einem Shell-Programm 9-65
- Rückkehr-Codes 9-68
- Schleifen 9-69
- Der Papierkorb der Shell: Die Datei `/dev/null` 9-75
- Bedingungsanweisungen 9-75
- Ablaufanweisungen ohne Abbruchbedingung: Die Kommandos `break` und `continue` 9-87

Fehlersuche in Programmen 9-88

Änderung Ihrer Benutzerumgebung 9-93
Einfügen von Kommandos in `.profile` 9-93
Einstellung von Terminaloptionen 9-94
Shell-Variablen 9-96

Übungen zur Shell-Programmierung 9-99

Einführung

In diesem Kapitel lernen Sie die Möglichkeiten der Shell kennen, Routineaufgaben leichter auszuführen. So wird beispielsweise die Verwaltung von Dateien mit Hilfe der Shell besprochen, die Manipulation von Dateiinhalten sowie die Zusammenfassung von Kommandos zu Programmen, die die Shell dann für Sie ausführt.

Dieses Kapitel enthält zwei Hauptabschnitte. Der erste Abschnitt "Die Kommandosprache der Shell" behandelt ausführlich die Shell als Kommandointerpreter. Im einzelnen geht es hier um die Verwaltung von Dateien mit Shell-Kommandos und Sonderzeichen, die Umleitung der Standard-Ein/Ausgabe sowie das Ausführen und Abbrechen von Prozessen. Der zweite Abschnitt "Shell-Programmierung" enthält eine ausführliche Beschreibung der Shell als Programmiersprache. hier geht es im einzelnen um die Erstellung und Ausführung von Programmen, die aus Kommandos, Variablen sowie Strukturen wie Schleifen und Bedingungsanweisungen bestehen, außerdem die Fehlersuche und -korrektur in diesen Programmen. Abschließend lernen Sie, wie Sie Ihre Benutzerumgebung an Ihre Anforderungen anpassen können.

Dieses Kapitel enthält viele Beispiele. Den besten Lerneffekt erzielen Sie, indem Sie sich bei UNIX anmelden und die Beispiele beim Durcharbeiten des Kapitels nachvollziehen. Wie in den anderen Kapiteln dieses Leitfadens werden auch in diesem Kapitel zur besseren Unterscheidung der Platzhalter von der tatsächlichen Ein- und Ausgabe verschiedene Schrifttypen verwendet (*kursiv* und `fester Zeichenabstand`). Weitere Erklärungen dazu finden Sie im "Vorwort" dieses Leitfadens im Abschnitt "Beschreibungsformat".

Zusätzlich zu den Beispielen finden Sie am Ende der Abschnitte "Die Kommandosprache der Shell" und "Shell-Programmierung" einen Abschnitt mit weiteren Übungen, die Ihnen das Verständnis der besprochenen Themen erleichtern. Die Auflösung der Übungen finden Sie am Ende des Kapitels.

Hinweis: Möglicherweise stellt Ihr UNIX-System nicht alle Kommandos zur Verfügung, die in diesem Kapitel behandelt werden. Wenn Sie auf eines der angegebenen Kommandos nicht zugreifen können, wenden Sie sich an Ihren Systemverwalter.

Wenn Sie sich vorab noch einmal einen Überblick über die Shell als Kommandointerpreter und Programmiersprache verschaffen möchten, lesen Sie vor diesem Kapitel die Kapitel 1 und 4 durch. Nähere Angaben dazu finden Sie im Anhang F "Die Kommandosprache der Shell - Kurzübersicht".

Die Kommandosprache der Shell

Dieser Abschnitt macht Sie mit Kommandos und - noch wichtiger - einer Reihe von Zeichen bekannt, die eine Sonderbedeutung haben. Mit diesen Kommandos und Sonderzeichen können Sie:

- Eine Gruppe von Dateien mit Hilfe von Suchmustern suchen und manipulieren.
- Ein Kommando im Hintergrund oder zu einem bestimmten Zeitpunkt ablaufen lassen.
- Mehrere Kommandos hintereinander ablaufen lassen.
- Die Standard-Ein- und Ausgabe von und in Dateien und zu anderen Kommandos umleiten.
- Die Ausführung von Programmen beenden.

In diesem Abschnitt werden zunächst die Zeichen behandelt, die für die Shell eine Sonderbedeutung haben, dann Kommandos und Eingabeformate, mit denen die oben aufgeführten Funktionen durchgeführt werden können. Tabelle 9-1 enthält einen Überblick über die Sonderzeichen, die in diesem Kapitel behandelt werden.

Tabelle 9-1: Sonderzeichen der Shell-Kommandosprache

Zeichen	Funktion
`* ? [ ]`	Metazeichen (Stern, Fragezeichen und eckige Klammern) ermöglichen die verkürzte Eingabe von Dateinamen in Suchmustern.
`&`	Das UND-Zeichen bewirkt die Ausführung von Kommandos im Hintergrund, so daß das Terminal für andere Aufgaben freigehalten wird.
`;`	Das Semikolon trennt verschiedene Kommandos in ein und derselben Kommandozeile.
`\`	Der Gegenschrägstrich hebt die Sonderbedeutung von Sonderzeichen wie `*, ?, [ ], &, ;, >, <` und `|` auf.
`'...'`	Durch Hochkommata wird die Bedeutung des Leerzeichens als Begrenzer und die Sonderbedeutung aller Sonderzeichen aufgehoben.
`"..."`	Durch Anführungszeichen wird die Bedeutung des Leerzeichens als Begrenzer und die Sonderbedeutung aller Sonderzeichen *mit Ausnahme von* `$` und `` ` `` aufgehoben.
`>`	Mit dem Größer-als-Zeichen wird die Ausgabe eines Kommandos in eine Datei umgeleitet (der vorherige Inhalt der Datei wird dabei gelöscht).
`<`	Mit dem Kleiner-als-Zeichen wird die Eingabe für ein Kommando umgeleitet und einer Datei entnommen (Eingabeumleitung).
`>>`	Mit dem doppelten Größer-als-Zeichen wird die Ausgabe eines Kommandos an das Ende einer vorhandenen Datei angehängt.
`|`	Das Symbol für die Pipe; die Ausgabe des einen Kommandos (vor dem Pipe-Symbol) wird zur Eingabe eines anderen Kommandos (hinter dem Pipe-Symbol).

Tabelle 9-1: Sonderzeichen der Shell-Kommandosprache (Fortsetzung)

Zeichen	Funktion
`...`	Durch zwei Gegenhochkommata kann die Ausgabe eines Kommandos in einer Kommandozeile als Argument verwendet werden.
$	Das Dollar-Zeichen wird in Stellungsparametern und benutzer-definierten Variablen benutzt; außerdem ist es die Standard-Eingabeaufforderung der Shell.

Metazeichen

Metazeichen, eine Untergruppe der Sonderzeichen, können stellvertretend für andere Zeichen benutzt werden. Man spricht hier auch von Jokerzeichen, analog zum Joker in einem Kartenspiel, der für jede beliebige Karte benutzt werden kann. In den folgenden Abschnitten werden die Metazeichen * (Stern), ? (Fragezeichen) und [ ] (eckige Klammern) besprochen.

Diese Zeichen können in Mustern zum Suchen von Dateinamen oder Teilen von Dateinamen benutzt werden und vereinfachen so die Angabe von Dateien oder Dateigruppen als Kommando-Argumente (die Dateien, die mit den Metazeichen gesucht werden, müssen bereits vorhanden sein). Man spricht hier von Dateinamenerweiterung. Sie können damit z.B. auf alle Dateinamen Bezug nehmen, die den Buchstaben "a" enthalten, auf alle fünfstelligen Dateinamen usw.

Das Metazeichen, das zu allen Zeichen paßt: Der Stern (*)

Der Stern (*) paßt zu einer beliebigen Zeichenkette einschließlich der leeren Zeichenkette. Er kann für einen ganzen Dateinamen oder einen Teil eines Dateinamens angegeben werden. Mit * allein wird auf alle Datei- und Verzeichnisnamen im aktuellen Verzeichnis Bezug genommen; hierzu gehören jedoch nicht die Datei- und Verzeichnisnamen, die mit einem Punkt (.) beginnen. Geben Sie den Stern (*) probehalber als Argument des Kommandos echo(1) ein:

```
echo *<CR>
```

Das Kommando echo zeigt seine Argumente auf dem Bildschirm an. Beachten Sie, daß das System bei der Eingabe von echo * eine Liste aller Dateinamen in Ihrem aktuellen Verzeichnis ausgibt.

Bild 9-2 faßt die Syntax und Funktionen des Kommandos echo zusammen.

Tabelle 9-2: echo**-Kurzübersicht**

Kommando echo Argumente auf die Ausgabe schreiben		
Kommando	*Optionen*	*Argumente*
echo	keine	beliebige(s) Zeichen
Funktion:	echo schreibt seine Argumente, die durch Leerzeichen voneinander getrennt sind und mit <CR> beendet werden, auf die Ausgabe.	
Hinweis:	In Shell-Programmen wird echo benutzt, um Anweisungen auf dem Bildschirm anzuzeigen, Wörter oder Daten in eine Datei umzuleiten und Daten über eine Pipe an ein Kommando weiterzuleiten. Diese Anwendungen werden an einer späteren Stelle dieses Kapitels behandelt.	

Vorsicht: Der Stern (*) allein paßt zu allen Datei- und Verzeichnisnamen. Wenn Sie z.B. rm * eingeben, werden alle Dateien in Ihrem aktuellen Verzeichnis gelöscht. Vorsicht also bei der Verwendung des Sterns!

Angenommen, Sie haben verschiedene Berichte eingegeben und sie report, report1, report1a, report1b.01, report25 und report316 genannt. Mit report1* können Sie auf alle Dateien Bezug nehmen, die zu report1 gehören. Um die Anzahl der von Ihnen eingegebenen Berichte zu ermitteln, können Sie

mit dem Kommando `ls` eine Übersicht über alle Dateien abrufen, deren Namen mit der Zeichenkette `report` beginnen:

```
$ ls report*<CR>
report
report1
report1a
report1b.01
report25
report316
$
```

Der Stern (*) steht für alle Zeichen, die auf die Zeichenkette "`report`" folgen (auch für die Zeichenkette "`report`" allein). Beachten Sie, daß der Stern (*) die Dateinamen in numerischer und alphanumerischer Reihenfolge überprüft. Mit folgendem Kommando können Sie den Inhalt der Berichtdateien schnell und einfach hintereinander auf dem Bildschirm abrufen:

```
pr report*<CR>
```

Ein weiteres Beispiel: In Ihrem aktuellen Verzeichnis mit dem Namen appraisals (Beurteilungen) befinden sich die Dateien Andrew_Adams, Paul_Lang, Jane_Peters und Fran_Smith. Suchen Sie nun ein Zeichen aus, das in allen Dateinamen in Ihrem Verzeichnis enthalten ist, z.B. den Kleinbuchstaben "a". Über dieses Zeichen können Sie dann eine Liste aller Dateinamen anfordern, die dieses Zeichen enthalten. Mit dem Kleinbuchstaben "a" sieht die Kommandozeile beispielsweise folgendermaßen aus:

```
ls *a*<CR>
```

Das System gibt daraufhin die Namen aller Dateien in Ihrem aktuellen Verzeichnis aus, die den Kleinbuchstaben "a" enthalten.

Der Stern (*) paßt zu Zeichen in jedem Teil des Dateinamens. Wenn Sie z.B. wissen, daß verschiedene Dateien denselben ersten und letzten Buchstaben haben, können Sie auf dieser Grundlage eine Liste erstellen. So können Sie z.B. in einem Verzeichnis, das die Dateien `FATE`, `FE`, `FADED_LINE`, `F123E`, `Fig3.4E`, `FIRE_LANE`, `FINE_LINE`, `FREE_ENTRY` und `FAST_LANE` enthält, mit Hilfe des Sterns eine Liste der Dateien abrufen, die mit F beginnen und mit E

enden. Die Kommandozeile muß dann folgendermaßen aussehen:

```
ls F*E<CR>
```

Das System gibt also eine Liste aller Dateinamen aus, die mit einem F beginnen und mit einem E enden. Die Dateinamen sind in der Liste folgendermaßen angeordnet:

```
F123E
FADED_LINE
FAST_LANE
FATE
FE
FINE_LINE
FIRE_LANE
Fig3.4E
```

Die Reihenfolge der Listeneinträge hängt von der Sortierreihenfolge der verwendeten Sprache ab, in diesem Fall Englisch. D.h., an erster Stelle stehen die Zahlen, gefolgt von den Großbuchstaben und den Kleinbuchstaben.

Der Stern (*) bietet aber noch wesentlich mehr Möglichkeiten: So können Sie damit beispielsweise alle Dateien mit dem Namen memo in einem beliebigen Systemverzeichnis suchen:

```
ls */memo
```

Das Metazeichen, das zu einem Einzelzeichen paßt: Das Fragezeichen (?)

Das Fragezeichen (?) paßt zu jedem beliebigen Zeichen eines Dateinamens mit Ausnahme des "führenden" Punkts. Angenommen, Sie haben mehrere Kapitel eines Buchs geschrieben, das insgesamt 12 Kapitel enthält, und wollen nun eine Liste aller bereits fertigen Kapitel bis Kapitel 9 erstellen. Ihr Verzeichnis enthalte folgende Dateien:

Chapter 1,
Chapter 2,
Chapter 5,
Chapter 9 und
Chapter 11,

Mit dem Kommando `ls` in Kombination mit dem Fragezeichen (?) können Sie

alle Kapitel abrufen, deren Namen mit der Zeichenkette "chapter" beginnen und ein einziges zusätzliches Zeichen enthalten. Ihr Bildschirm sieht dann folgendermaßen aus:

```
$ ls chapter?<CR>
chapter1
chapter2
chapter5
chapter9
$
```

Das System gibt alle passenden Dateinamen aus.

? paßt zu einem beliebigen Einzelzeichen, kann in einem Dateinamen jedoch auch mehrmals vorkommen. Mit dem folgenden Kommando können Sie die Dateien mit den übrigen Kapiteln Ihres Buchs auf dem Bildschirm abrufen:

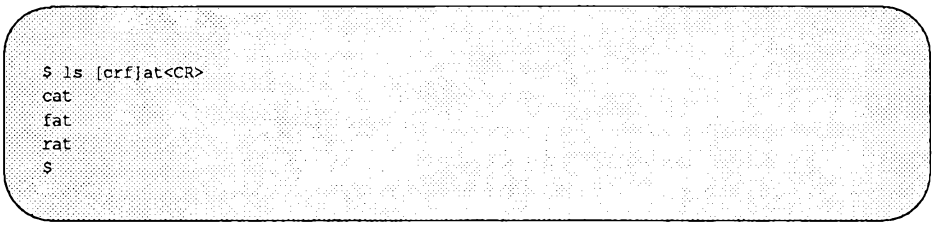
```
ls chapter??<CR>
```

Eine Übersicht über sämtliche Kapitel im aktuellen Verzeichnis erhalten Sie selbstverständlich am einfachsten mit Hilfe des Sterns (*):

```
ls chapter*
```

Die Metazeichen, die zu einem Zeichen aus einer Gruppe passen: Die eckigen Klammern ([ ])

Die eckigen Klammern ([ ]) verwenden Sie, wenn die Shell nach einem von mehreren möglichen Zeichen suchen soll, das an einer bestimmten Position im Dateinamen vorkommt. Angenommen, Ihr Verzeichnis enthält eine der folgenden Dateien: `cat`, `fat`, `mat` und `rat`. Wenn das Suchmuster die Zeichenkette `[crf]` enthält, sucht die Shell nach Dateinamen, die an der angegebenen Position den Buchstaben "c", "r" oder "f" enthalten:



Das Kommando zeigt alle Dateinamen an, die mit dem Buchstaben "c", "r" oder "f" beginnen und mit den Buchstaben "at" enden. Zeichen, die auf diese Weise in eckige Klammern eingeschlossen werden können, werden als "Zeichenklasse" bezeichnet.

Eckige Zeichen können auch zur Angabe eines Zeichen-Bereichs benutzt werden. Dieser Bereich kann sowohl aus Zahlen als auch aus Buchstaben bestehen. Angenommen, Ihr Verzeichnis enthält die folgenden Dateien: `chapter1`, `chapter2`, `chapter3`, `chapter4`, `chapter5` und `chapter6`. Bei der Angabe von

```
chapter[1-5]
```

sucht die Shell nach Dateien mit den Dateinamen `chapter1` bis `chapter5`. Dies stellt eine einfache Möglichkeit zur Selektion eines Teils Ihrer Dateien dar.

Geben Sie probehalber das Kommando `pr` mit einem in eckigen Klammern eingeschlossenen Argument ein:

```
$ pr chapter[2-4]<CR>
```

Mit diesem Kommando wird der Inhalt von `chapter2`, `chapter3` und `chapter4` (in dieser Reihenfolge) auf Ihrem Terminal angezeigt.

Eine Zeichenklasse kann auch einen Bereich von Buchstaben umfassen. Bei der Angabe von `[A-Z]` sucht die Shell nur nach Großbuchstaben, bei der Angabe von `[a-z]` nur nach Kleinbuchstaben.

Tabelle 9-3 enthält einen Überblick über die Funktion der Metazeichen. Probieren Sie die Metazeichen an Dateien in Ihrem aktuellen Verzeichnis aus.

Tabelle 9-3: Metazeichen - Kurzübersicht

Zeichen	Funktion
*	paßt zu einer beliebigen Zeichenkette einschließlich der leeren Zeichenkette; dazu gehört jedoch nicht der "führende" Punkt.
?	paßt zu einem beliebigen Einzelzeichen mit Ausnahme des "führenden" Punkts.
[ ]	paßt zu einem beliebigen der in eckigen Klammern eingeschlossenen Zeichen.
[-]	paßt zu einem der Zeichen aus dem in eckigen Klammern eingeschlossenen Zeichenbereich.

Sonderzeichen

Die Shell kennt noch weitere Sonderzeichen für eine Vielzahl nützlicher Funktionen. Einige dieser zusätzlichen Sonderzeichen werden in diesem Abschnitt beschrieben. Weitere Sonderzeichen werden im Abschnitt "Ein- und Ausgabeumleitung" beschrieben.

Kommandoausführung im Hintergrund: Das UND-Zeichen (&)

Einige Shell-Kommandos sind in der Durchführung sehr zeitaufwendig. Mit dem UND-Zeichen können Sie Kommandos als Hintergrundprozesse aufrufen, so daß Sie während der Kommandoausführung an Ihrem Terminal weiterarbeiten können. Das Aufrufen eines Kommandos als Hintergrundprozeß erfolgt in folgendem Format:

kommando &<CR>

Hinweis: Interaktive Shell-Kommandos wie z.B. das Kommando `read` (siehe Abschnitt "Das Kommando `read`" in diesem Kapitel) dürfen nicht als Hintergrundprozeß aufgerufen werden.

In folgendem Beispiel führt die Shell einen umfangreichen Suchvorgang als Hintergrundprozeß aus. Das Kommando `grep`(1) sucht nach der Zeichenkette "`delinquent`" in der Datei `accounts`. Beachten Sie, daß `&` das letzte Zeichen der Kommandozeile ist:

```
$ grep delinquent accounts &<CR>
21940
$
```

Wird ein Kommando als Hintergrundprozeß aufgerufen, zeigt UNIX eine Prozeßnummer an. In unserem Beispiel lautet die Prozeßnummer `21940`. Mit dieser Nummer kann die Ausführung eines Hintergrundprozesses abgebrochen werden (das Abbrechen von Prozessen wird im Abschnitt "Abbrechen von aktiven Prozessen" beschrieben). Die Eingabeaufforderung in der letzten Zeile zeigt Ihnen, daß das Terminal frei ist und auf Ihre Kommandos wartet. Die Ausführung des Kommandos `grep` im Hintergrund hat jetzt begonnen.

Die Ausführung eines Kommandos im Hintergrund bewirkt nur, daß Sie an Ihrem Terminal weiterarbeiten können; an der Ausgabe des Kommandos ändert sich nichts. Die Ausgabe wird in jedem Fall auf dem Bildschirm Ihres Terminals angezeigt, unabhängig davon, ob Sie das Kommando als Vorder- oder als Hintergrundprozeß aufgerufen haben; dies gilt jedoch nicht, wenn Sie die Ausgabe in eine Datei umleiten (weitere Angaben dazu finden Sie im Abschnitt "Umleitung der Ausgabe in eine Datei" in diesem Kapitel).

Wenn die Ausführung eines Hintergrundprozesses fortgesetzt werden soll, nachdem Sie sich von UNIX abgemeldet haben, kombinieren Sie das Kommando mit dem Kommando `nohup`(1) (siehe Abschnitt "Das Kommando `nohup`" in diesem Kapitel).

Mehrere Kommandos hintereinander ausführen: Das Semikolon (`;`)

Sie können in ein und derselben Kommandozeile auch zwei oder mehrere Kommandos eingeben, die durch ein Semikolon (`;`) oder ein UND-Zeichen (`&`)

voneinander getrennt werden:

kommando1; kommando2; kommando3<CR>

UNIX verarbeitet die Kommandos in der eingegebenen Reihenfolge und zeigt die Ausgabe auf dem Bildschirm an. Dieser Prozeß wird sequentielle Verarbeitung genannt.

Mit folgender Übung können Sie die Funktionsweise eines Semikolons (;) ausprobieren. Geben Sie folgendes ein:

```
cd; pwd; ls<CR>
```

Die Shell führt die Kommandos in der angegebenen Reihenfolge aus:

1. Mit `cd` wechseln Sie in Ihr Home-Verzeichnis.
2. `pwd` zeigt den absoluten Pfadnamen Ihres aktuellen Verzeichnisses an.
3. `ls` zeigt die Dateien in Ihrem aktuellen Verzeichnis an.

Wie Sie verhindern können, daß die Ausgabe des Systems auf Ihrem Bildschirm angezeigt wird, erfahren Sie im Abschnitt "Umleitung der Ausgabe in eine Datei" dieses Kapitels.

Sonderbedeutung aufheben: Der Gegenschrägstrich (\)

Die Shell interpretiert den Gegenschrägstrich (\) als Escape-Zeichen, mit dem die spezielle Bedeutung des unmittelbar darauffolgenden Zeichens aufgehoben wird. Die folgende Übung zeigt Ihnen, wie der Gegenschrägstrich (\) funktioniert. Erstellen Sie eine Datei mit dem Namen `trial`, die die folgenden zwei Zeilen enthält:

```
The all * game
was held in Summit.
```

Benutzen Sie das Kommando `grep`, um in der Datei nach dem Stern (*) zu suchen:

```
$ grep \* trial<CR>
The all * game
$
```

Das Kommando `grep` sucht den Stern (*) im Text und zeigt die Zeile an, in der dieser vorkommt. Ohne Gegenschrägstrich (\) würde der Stern (*) als Metazeichen verwendet werden und für alle Dateinamen des aktuellen

Verzeichnisses stehen.

Sonderbedeutung aufheben: Hochkommata und Anführungszeichen

Eine andere Möglichkeit, die Bedeutung eines Sonderzeichens aufzuheben, ist die Verwendung von Hochkommata (') und Anführungszeichen ("). Zwei Hochkommata ('...') heben die Sonderbedeutung eines beliebigen Zeichens mit Ausnahme des Hochkommas auf. Anführungszeichen ("...") heben die Sonderbedeutung aller Zeichen auf, mit Ausnahme des Anführungszeichens (") selbst sowie des Dollar-Zeichens ($) und des Gegenhochkommas (`). Die Verwendung von Hochkommata (') und Anführungszeichen (") hat den Vorteil, daß gleichzeitig mehrere Sonderzeichen angegeben werden können. Dadurch ist ein kompakteres und besser lesbares Format möglich, als wenn jedes Sonderzeichen durch einen Gegenschrägstrich (\) entwertet wird.

Angenommen, Ihre Datei `trial` enthält zusätzlich die folgende Zeile:

```
He really wondered why? Why???
```

Mit dem Kommando `grep` können Sie dann die Zeile mit den drei Fragezeichen suchen:

```
$ grep '???' trial<CR>
He really wondered why? Why???
$
```

Bei der Eingabe der Kommandozeile

```
grep ??? trial<CR>
```

werden die Fragezeichen als Shell-Metazeichen und somit als Suchmuster für alle Dateinamen interpretiert, die aus drei Zeichen bestehen.

Sonderbedeutung eines Leerzeichens mit Anführungszeichen und Hochkommata aufheben

Hochkommata (') und Anführungszeichen (") werden wie der Gegenschrägstrich (\) als Escape-Zeichen benutzt, um die Sonderbedeutung eines Leerzeichens aufzuheben. Die Shell interpretiert ein Leerzeichen in einer Kommandozeile als Begrenzer zwischen den Kommando-Argumenten. Diese Sonderbedeutung kann sowohl mit Hochkommata als auch mit Anführungszeichen aufgehoben werden.

Um z.B. in einem Text zwei oder mehrere Wörter zu suchen, müssen sie dem Kommando `grep` als ein einziges Argument übergeben werden. Dafür müssen sie in Hochkommata (') oder Anführungszeichen (") eingeschlossen werden. Mit der folgenden Kommandozeile können Sie beispielsweise die Wörter "The all" in Ihrer Datei `trial` suchen:

```
$ grep 'The all' trial<CR>
The all * game
$
```

`grep` sucht die Zeichenkette "`The all`" und zeigt auf dem Bildschirm die Zeile an, in der sie enthalten ist. Was wäre geschehen, wenn Sie die Zeichenkette nicht in Hochkommata (oder Anführungszeichen) eingeschlossen hätten?

Die Möglichkeit, die Bedeutung eines Leerzeichens als Begrenzer aufzuheben, kommt vor allem beim Kommando `banner`(1) zur Anwendung. Mit diesem Kommando wird eine Meldung auf dem Bildschirm in große Schrift angezeigt.

Das Kommando `banner` wird mit einer Meldung eingegeben, die aus ein oder mehreren, durch Leerzeichen voneinander getrennten Argumenten (normalerweise Wörter) besteht. Das Kommando `banner` interpretiert diese Leerzeichen als Begrenzer und zeigt jedes Argument in einer separaten Zeile an.

Um mehr als ein Argument in ein und derselben Zeile anzuzeigen, müssen die entsprechenden Wörter in Anführungszeichen eingeschlossen werden. Mit dem folgenden Kommando erreichen Sie beispielsweise die Anzeige eines Geburtstagsgrusses:

```
banner happy birthday to you<CR>
```

Das Kommando zeigt Ihre Nachricht in vier Zeilen an. Mit der folgenden Kommandozeile würde sie in drei Zeilen ausgegeben:

```
banner happy birthday "to you"<CR>
```

Beachten Sie, daß die Wörter "`to`" und "`you`" jetzt in derselben Zeile enthalten sind. Das dazwischenliegende Leerzeichen hat seine Bedeutung als Begrenzer verloren.

Tabelle 9-4 enthält eine Übersicht über Syntax und Funktion des Kommandos `banner`.

Tabelle 9-4: `banner`**-Kurzübersicht**

Kommando `banner` Meldungen in großer Schrift ausgeben.		
Kommando	*Optionen*	*Argumente*
`banner`	keine	Zeichen
Funktion:	`banner` zeigt bis zu zehn Zeichen in großer Schrift an.	
Hinweise:	An späterer Stelle dieses Kapitels erfahren Sie, wie man das Kommando `banner` in eine Datei umleitet, um an einen anderen Benutzer eine Nachricht in großer Schrift zu senden.	

Ein- und Ausgabeumleitung

Unter UNIX erwarten einige Kommandos, daß die Eingabe ausschließlich über die Tastatur erfolgt (Standard-Eingabe); ihre Ausgabe schreiben die meisten Kommandos auf das Terminal (die Standard-Ausgabe). Unter UNIX kann jedoch sowohl die Ein- als auch die Ausgabe zu anderen Dateien oder Kommandos umgeleitet werden. Mit einer Umleitung teilen Sie der Shell mit,

- daß sie die Eingabe aus einer Datei und nicht von der Tastatur einlesen soll;
- daß die Ausgabe in eine Datei und nicht an das Terminal geschickt werden soll;

- daß die Ausgabe eines Programms als Datenquelle für ein anderes Programm benutzt werden soll.

Die Operatoren für die Umleitung der Ein- und Ausgabe sind das Kleiner-als-Zeichen (<), das Größer-als-Zeichen (>), das doppelte Größer-als-Zeichen (>>), das Pipe-Symbol (|), sowie das doppelte Kleiner-als-Zeichen (<<) (siehe Abschnitt "Shell-Programmierung").

Umleitung der Eingabe: Das Kleiner-als-Zeichen (<)

Zur Umleitung der Eingabe geben Sie in der Kommandozeile hinter dem Kleiner-als-Zeichen (<) einen Dateinamen ein:

kommando < *datei*<CR>

Angenommen, Sie möchten mit dem Kommando `mail`(1) (siehe Kapitel 11 "Die elektronische Post") eine Nachricht an einen anderen Benutzer schicken, der den Benutzernamen `colleague` hat; die Nachricht sei bereits in der Datei `report` vorhanden. Wenn Sie die Nachricht nicht erneut eingeben möchten, verwenden Sie diese Datei als Eingabequelle:

```
mail colleague < report<CR>
```

Umleitung der Ausgabe in eine Datei: Das Größer-als-Zeichen (>)

Zur Umleitung der Ausgabe wird in der Kommandozeile hinter dem Größer-als-Zeichen (>) ein Dateiname eingegeben:

kommando > *datei*<CR>

Vorsicht: Wird die Ausgabe in eine bereits vorhandene Datei umgeleitet, so überschreibt die Ausgabe des Kommandos den Inhalt dieser Datei.

Bevor Sie die Ausgabe eines Kommandos in eine bestimmte Datei umleiten, sollten Sie sicherstellen, daß noch keine Datei mit dem angegebenen Namen vorhanden ist - es sei denn, diese Datei soll tatsächlich überschrieben werden. Die Shell überschreibt die ursprüngliche Datei ohne Vorwarnung.

Stellen Sie mit dem Kommando `ls` und dem Namen der Datei als Argument sicher, daß keine Datei gleichen Namens vorhanden ist. Ist bereits eine Datei gleichen Namens vorhanden, wird sie durch das Kommando `ls` angezeigt; andernfalls erhalten Sie eine Meldung, daß die Datei im aktuellen Verzeichnis nicht gefunden wurde. Mit den Dateinamen `temp` und `junk` erhalten Sie beispielsweise die folgende Ausgabe:

```
$ ls temp<CR>
temp
$ ls junk<CR>
junk:  no such file or directory
$
```

Sie können der neuen Ausgabedatei also den Namen `junk` zuordnen, nicht aber den Namen `temp` - es sei denn, Sie benötigen den Inhalt der Datei `temp` nicht mehr.

Ausgabe an eine vorhandene Datei anhängen: Das doppelte Größer-als-Zeichen (>>)

Wenn die vorhandene Datei nicht zerstört werden soll, benutzen Sie das doppelte Größer-als-Zeichen (>>):

kommando >> *datei*<CR>

Damit wird die Ausgabe eines Kommandos am Ende der *datei* angehängt. Ist die *datei* nicht vorhanden, wird sie bei der Verwendung des Symbols >> angelegt.

Das folgende Beispiel zeigt, wie die Ausgabe des Kommandos `cat` (siehe Abschnitt "Shell-Programmierung") an eine vorhandene Datei angehängt wird. Das Kommando `cat` schreibt den Inhalt der Dateien, die als Argumente angegeben worden sind, auf die Standard-Ausgabe. Fehlen Argumente, so zeigt es seine Standard-Eingabe auf der Standard-Ausgabe an. Zunächst wird das Kommando `cat` für beide Dateien ohne Umleitung der Ausgabe ausgeführt, um ihren jeweiligen Inhalt anzuzeigen. Danach wird der Inhalt von `trial2` am Ende der Datei `trial1` angehängt; dafür wird das Kommando `cat` für `trial2` ausgeführt und die Ausgabe nach `trial1` umgeleitet:

```
$ cat trial1<CR>
This is the first line of trial1.
Hello.
This is the last line of trial1.
$
$ cat trial2<CR>
This is the beginning of trial2.
Hello.
This is the end of trial2.
$
$ cat trial2 >> trial1<CR>
$ cat trial1<CR>
This is the first line of trial1.
Hello.
This is the last line of trial1.
This is the beginning of trial2.
Hello.
This is the end of trial2.
$
```

Anwendungsmöglichkeiten für die Ausgabeumleitung

Die Ausgabeumleitung wird vor allem dann verwendet, wenn die Ausgabe nicht unmittelbar auf dem Bildschirm erscheinen soll, oder wenn sie abgespeichert werden soll. Eine weitere wichtige Anwendung für die Ausgabeumleitung steht im Zusammenhang mit den Kommandos zur Manipulation von Textdateien wie z.B. `spell` und `sort`.

Das Kommando `spell`

Das Programm `spell` vergleicht jedes Wort einer Datei mit einer internen (englischen) Wortliste und zeigt die gefundenen Schreibfehler auf dem Bildschirm an. Ist ein Wort nicht in der Wortliste von `spell` enthalten (wie z.B. ein Name), wird dieser Name ebenfalls als fehlerhaft angezeigt.

Wird `spell` für einen langen Text eingegeben, kann die Ausführung lange dauern. Außerdem ist die erstellte Liste der Schreibfehler möglicherweise zu lang, um auf einen einzigen Bildschirm zu passen. `spell` gibt seine Ausgabe auf einmal aus. Paßt die Ausgabe nicht auf eine Bildschirmseite, wird sie vom Kommando mit schnellem Bildlauf bis zum Ende angezeigt. Die Ausgabe ist dann wegen des schnellen Bildlaufs sehr schwer zu lesen.

Um dieses Problem zu vermeiden, leiten Sie die Ausgabe von `spell` in eine Datei um. Im folgenden Beispiel sucht `spell` eine Datei mit dem Namen `memo` und setzt die fehlerhaften Wörter in die Datei `misspell`:

```
$ spell memo > misspell<CR>
```

Tabelle 9-5 enthält eine Übersicht über Syntax und Funktion des Kommandos `spell`.

Tabelle 9-5: `spell`-Kurzübersicht

Kommando `spell` Englische Rechtschreibprüfung.		
Kommando	*Optionen*	*Argumente*
`spell`	verfügbar*	*datei*
Funktion:	`spell` liest die Wörter einer bestimmten Datei und vergleicht sie mit den Wörtern einer Wortliste. Wörter, die nicht im Wörterverzeichnis enthalten sind, werden auf dem Terminal angezeigt.	
Optionen:	Für `spell` stehen mehrere Optionen zur Verfügung; so kann statt der amerikanischen Rechtschreibprüfung (Standard) auch eine britische durchgeführt werden.	
Hinweis:	Die Liste der fehlerhaften Wörter kann in eine Datei umgeleitet werden.	

* Eine vollständige Übersicht über die verfügbaren Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `spell`(1)

Das Kommando `sort`

Das Kommando `sort` sortiert die Zeilen einer angegebenen Datei und bringt sie in alphanumerische oder numerische Reihenfolge (mehr darüber finden Sie im Kapitel 3 "Das Dateisystem"). Da die Dateien, die von `sort` in alphabetische Reihenfolge gebracht worden sind, im Normalfall erhalten bleiben sollen, werden die Anwendungsmöglichkeiten des Kommandos `sort` durch die Ausgabeumleitung beträchtlich erhöht.

Bei der Wahl eines neuen Namens für die Datei, in die die Ausgabe des Kommandos `sort` (die nach dem Alphabet sortierte Liste) geschrieben werden soll, ist große Vorsicht geboten: Nach dem Aufrufen des Kommandos `sort` löscht die Shell zunächst den Inhalt der Datei, in die die umgeleitete Ausgabe geschrieben werden soll. Dann wird der Sortiervorgang durchgeführt und die Ausgabe in die leere Datei gesetzt. Bei der Eingabe von

```
sort list > list<CR>
```

löscht die Shell zunächst `list` und bringt in die Datei nichts ein.

Verbindung von Hintergrundverarbeitung und Ausgabeumleitung

Wird ein Kommando als Hintergrundprozeß ausgeführt, so hat das keine Auswirkung auf die Ausgabe des Kommandos. Die Ausgabe wird in jedem Fall auf dem Bildschirm angezeigt, es sei denn, sie wird umgeleitet. Wenn Sie während der Hintergrund-Ausführung eines Kommandos andere Aufgaben durchführen, werden diese Funktionen unterbrochen, wenn das im Hintergrund ablaufende Kommando seine Ausgabe auf Ihrem Bildschirm anzeigt. Wird die Ausgabe dagegen in eine Datei umgeleitet, können Sie ungestört weiterarbeiten, es sei denn, es tritt ein Fehler auf.

Im Abschnitt "Sonderzeichen" wurde beispielsweise gezeigt, wie das Kommando `grep` mit `&` als Hintergrundprozeß ausgeführt werden kann. Wenn Sie nun herausfinden möchten, ob das Wort "`test`" in der Datei `schedule` vorkommt, lassen Sie das Kommando `grep` als Hintergrundprozeß ablaufen und leiten Sie die Ausgabe in die Datei `testfile` um:

```
$ grep test schedule > testfile &<CR>
```

Sie können jetzt an Ihrem Terminal weiterarbeiten und sich danach das Ergebnis des Kommandos in der Datei `testfile` ansehen.

Umleitung der Ausgabe zu einem Kommando: Das Pipe-Symbol (|)

Das Zeichen | wird Pipe-Symbol genannt. Pipes sind leistungsfähige Werkzeuge, mit denen Sie die Ausgabe eines Kommandos als Eingabe eines anderen Kommandos verwenden können, ohne temporäre Zwischendateien anzulegen. Eine Kommandozeile, in der mehrere durch Pipes miteinander verknüpfte Kommandos enthalten sind, wird Pipeline genannt.

Eine Pipeline hat folgendes grundlegendes Format:

kommando1 | *kommando2* | *kommando3*...<CR>

Die Ausgabe von *kommando1* wird als Eingabe von *kommando2* verwendet, die Ausgabe von *kommando2* dann als Eingabe von *kommando3*.

Im folgenden werden für eine bestimmte Aufgabe zwei Lösungsmöglichkeiten gegenübergestellt, um die Leistungsfähigkeit und Effizienz einer Pipeline zu veranschaulichen:

- Benutzen Sie die Ein/Ausgabeumleitung; rufen Sie also ein Kommando auf, dessen Ausgabe Sie in eine temporäre Datei umleiten. Rufen Sie dann ein zweites Kommando auf, das den Inhalt der temporären Datei als Eingabe verwendet. Abschließend löschen Sie nach der Beendigung des zweiten Kommandos die temporäre Datei.
- Benutzen Sie eine Pipeline; rufen Sie also ein Kommando auf und leiten Sie seine Ausgabe direkt an ein zweites Kommando weiter.

Angenommen, Sie wollen eine Meldung mit Geburtstagsgrüßen in großer Schrift an den Benutzer mit dem Benutzernamen `david` schicken. Ohne Pipeline besteht diese Prozedur aus den folgenden drei Schritten:

1. Geben Sie das Kommando `banner` ein, und leiten Sie die Ausgabe in eine temporäre Datei um:

```
banner happy birthday > message.tmp
```

2. Geben Sie das Kommando `mail` ein, und verwenden Sie `message.tmp` als Eingabe:

```
mail david < message.tmp
```

3. Löschen Sie die temporäre Datei:

```
rm message.tmp
```

Mit einer Pipeline kann diese Prozedur in einem Schritt durchgeführt werden:

```
banner happy birthday | mail david<CR>
```

Pipeline mit den Kommandos `cut` und `date`

Die Kommandos `cut` und `date` sind ein gutes Beispiel dafür, wie die Flexibilität einzelner Kommandos durch Pipelines erhöht wird. Mit dem Kommando `cut` können Sie Teile aus jeder Zeile einer Datei extrahieren. Es sucht nach Zeichen in einem bestimmten Teil der Zeile und zeigt sie an. Den zu extrahierenden Zeilenbereich geben Sie mit der Option -c (für column) und die durch Bindestrich (-) voneinander getrennten Nummern der Zeichenpositionen an (dabei wird davon ausgegangen, daß die Zeichenpositionen innerhalb der Zeile von links nach rechts durchnumeriert sind).

Angenommen, Sie möchten aus einer Datei mit dem Namen `birthdays` ausschließlich die Datumsangaben extrahieren. Die Datei enthält folgende Liste:

```
Anne    12/26
Klaus   7/4
Mary    10/18
Peter   11/9
Nandy   4/23
Sam     8/12
```

Die Geburtstage stehen zwischen der 9. und 13. Zeichenposition jeder Zeile. Damit die Geburtstage angezeigt werden, geben Sie folgendes ein:

```
cut -c9-13 birthdays<CR>
```

Die Ausgabe sieht folgendermaßen aus:

```
12/26
7/4
10/18
11/9
4/23
8/12
```

Tabelle 9-6 enthält einen Überblick über die Syntax und Funktionen des Kommandos `cut`.

Tabelle 9-6: `cut`**-Kurzübersicht**

Kommando `cut` Bestimmte Felder aus allen Zeilen einer Datei extrahieren.		
Kommando	Optionen	Argumente
`cut`	`-`clist `-f`list [`-d`]	*datei*
Funktion:	Das Kommando `cut` extrahiert Spalten aus einer Tabelle bzw. Felder aus allen Zeilen einer Datei.	
Optionen:	Mit `-c` (für column) werden die Nummern der zu extrahierenden Zeichenpositionen angegeben (z.B. `-c1-9` für den Bereich 1–9). Dabei wird davon ausgegangen, daß die Zeichenpositionen von links nach rechts durchnumeriert sind. Mit `-f` (für field) werden die Feldnummern, vom linken Rand aus gezählt, angegeben. Nach jeder Feldnummer steht ein Begrenzer, der mit `-d` angegeben wird. `-d` (für delimiter) gibt den Begrenzer für `-f` an (Standard: ein Leerzeichen). Der Doppelpunkt als Begrenzer wird folgendermaßen angegeben: `-d :`	
Hinweis:	Außer dem Kommando `cut` sind für Sie in diesem Zusammenhang möglicherweise auch die Kommandos `paste` und `split` von Interesse.	

Das Kommando `cut` wird gewöhnlich für eine Datei aufgerufen. Sie können als Eingabe jedoch auch die Ausgabe von anderen Kommandos verwenden; hierfür bringen Sie die Kommandos in eine Pipeline ein. Dies ist immer dann sinnvoll, wenn Sie die von anderen Kommandos erzeugten Informationen nur teilweise benötigen. Angenommen, auf dem Bildschirm soll die Uhrzeit ausgegeben werden. Das Kommando `date` gibt Wochentag, Datum und Uhrzeit folgendermaßen aus:

```
$ date<CR>
Sat Dec 24 13:12:32 EST 1988
$
```

Die Uhrzeit steht zwischen Zeichenposition 12 und 19. Wenn Sie nur die Uhrzeit (ohne Datum) abrufen möchten, können Sie die Ausgabe von `date` als Eingabe für `cut` verwenden und mit der Option `-c` den Bereich `12-19` angeben. Kommandozeile und Ausgabe sehen dann folgendermaßen aus:

```
$ date | cut -c12-19<CR>
13:14:56
$
```

Bild 9-7 enthält einen Überblick über Syntax und Funktionen des Kommandos `date`.

Tabelle 9-7: `date`**-Kurzübersicht**

Kommando `date` Datum und Uhrzeit anzeigen.		
Kommando	Optionen	Argumente
`date`	`+%m%d%y*` `+%H%%M%S`	verfügbar*
Funktion:	`date` zeigt Datum und Uhrzeit auf Ihrem Terminal an.	
Optionen:	Mit dem Prozent-Zeichen `+%` , auf das ein `m` (für Monat), `d` (day - für Tag), `y` (year - für Jahr), `H` (hour - für Stunde), `M` (für Minute) und `S` (für Sekunde) folgt, werden die entsprechenden Angaben auf Ihrem Terminal angezeigt. Die Angaben können Sie z.B. folgendermaßen erläutern: `date '+%H:%M is the time'`	
Hinweise:	Wenn Sie an einem kleinen Rechnersystem arbeiten, auf dem Sie sowohl der Benutzer als auch der Systemverwalter sind, können Sie das Kommando `date` möglicherweise mit optionalen Argumenten kombinieren, um Datum und die Uhrzeit einzustellen. Weitere Angaben dazu befinden sich im Referenzhandbuch. Auf einem Mehrbenutzersystem stehen diese Argumente nur dem Systemverwalter zur Verfügung.	

* Ausführliche Informationen über sämtliche verfügbaren Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `date`(1)

Ausgabe eines Kommandos als Argument verwenden

Die Ausgabe der meisten Kommandos kann zwischengespeichert und als Argument in einer Kommandozeile verwendet werden. Dafür wird das Kommando in Gegenhochkommata (`...`) eingeschlossen und an die Stelle in der Kommandozeile gesetzt, an der die Ausgabe als Argument interpretiert werden soll. Dieser Vorgang wird Kommando-Substitution genannt.

Angenommen, Sie möchten als Argument des Kommandos `banner` die Ausgabe der Pipeline im letzten Abschnitt benutzen, die aus den Kommandos `date` und `cut` bestand. Dazu geben Sie die folgende Kommandozeile ein:

```
$ banner `date | cut -c12-19`<CR>
```

Das System gibt die aktuelle Uhrzeit in großer Schrift aus.

Im Abschnitt "Shell-Programmierung" in diesem Kapitel wird beschrieben, wie die Ausgabe einer Kommandozeile einer Variablen zugeordnet werden kann.

Ausführung, Abbruch und Neustart von Prozessen

Dieser Abschnitt behandelt folgende Themen:

- Aufrufen von Kommandos, die zu einem späteren Zeitpunkt ausgeführt werden sollen, mit den Kommandos `batch` und `at`.
- Abrufen des Status von gerade aktiven Prozessen.
- Abbruch aktiver Prozesse.
- Neustart eines abgebrochenen Prozesses.
- Fortsetzung eines Hintergrundprozesses nach Ihrer Abmeldung.
- Versetzen von Vordergrundprozessen in den Hintergrund und umgekehrt.

Kommandos zu einem späteren Zeitpunkt verarbeiten - Die Kommandos batch und at

Mit den Kommandos `batch` und `at` können Sie ein oder mehrere Kommandos aufrufen, die zu einem späteren Zeitpunkt verarbeitet werden sollen. Beim Kommando `batch` bestimmt das System den Zeitpunkt der Kommandoausführung, bei dem Kommando `at` der Benutzer selbst. Beide Kommandos lesen ihre Eingabe von der Standard-Eingabe (dem Terminal). Die Kommandoliste, die Sie über das Terminal eingeben, muß mit <^d> (Control-d) abgeschlossen werden.

Das Kommando `batch` wird vor allem für sehr rechenzeit-intensive Prozesse oder Shell-Programme benutzt. Das Kommando `batch` übermittelt dem System einen Stapelauftrag, der die auszuführenden Kommandos enthält. Der Auftrag wird in eine Warteschlange gesetzt und ausgeführt, sobald es die Systemauslastung zuläßt. Dies hat zwei Vorteile: Zum einen muß das System nicht mehr sofort auf Ihre Eingabe reagieren, zum anderen nehmen Sie Rücksicht auf die anderen Systembenutzer.

Das Kommando `batch` hat folgendes grundlegendes Format:

```
batch<CR>
erstes kommando<CR>
        .
        .
        .
letztes kommando<CR>
<^d>
```

Wenn mit `batch` nur ein Kommando verarbeitet werden soll, sieht die Eingabe folgendermaßen aus:

```
batch kommandozeile<CR>
<^d>
```

Im nächsten Beispiel wird `batch` verwendet, um das Kommando `grep` zu einem für das System günstigeren Zeitpunkt auszuführen. In diesem Beispiel durchsucht das Kommando `grep` alle Dateien im aktuellen Verzeichnis und leitet die Ausgabe in die Datei `dol.file` um.

```
$ batch<CR>
grep dollar * > dol-file<CR>
<^d>
job 155223141.b at Sun Dec 3 11:14:54 1989
$
```

Nachdem ein Auftrag mit `batch` abgesetzt worden ist, antwortet das System mit einer Auftragsnummer, dem Datum und der Uhrzeit. Die Auftragsnummer entspricht nicht der Prozeßnummer, die nach dem Starten eines Kommandos als Hintergrundprozeß ausgegeben wird.

Tabelle 9-8 enthält einen Überblick über die Syntax und Funktionen des Kommandos `batch` .

Tabelle 9-8: `batch`**-Kurzübersicht**

Kommando `batch` Kommandos zu einem späteren Zeitpunkt ausführen.		
Kommando	*Optionen*	*Eingabe*
`batch`	keine	*kommandozeilen*
Funktion:	`batch` setzt einen Stapelauftrag ab, der in eine Warteschlange eingereiht und erst dann ausgeführt wird, wenn es die Systemauslastung zuläßt.	
Hinweis:	Die Kommando-Liste muß mit <^d> (Control-d) beendet werden.	

Beim Kommando `at` können Sie den genauen Zeitpunkt angeben, zu dem das Kommando ausgeführt werden soll. Das Kommando `at` hat folgendes grundlegendes Format:

```
at zeit<CR>
erstes kommando<CR>
        .
        .
        .
letztes kommando<CR>
<^d>
```

Über das Argument *zeit* wird die Uhrzeit und, wenn es sich nicht um den gleichen Tag handelt, das Datum angegeben.

Das folgende Beispiel zeigt, wie man dem Benutzer mit dem Benutzernamen `emily` mit dem Kommando `at` einen Geburtstagsgruß in großer Schrift schickt:

```
$ at 8:15am Feb 27<CR>
banner happy birthday | mail emily<CR>
<^d>
job 453400603.a at Thurs Feb 23 08:15:00 1989
$
```

Das Kommando `at` gibt, wie das Kommando `batch`, eine Auftragsnummer, das Datum und die Uhrzeit aus.

Wenn die Kommandos in einer `batch`- oder `at`- Warteschlange doch nicht ausgeführt werden sollen, so können Sie die betreffenden Aufträge mit der `at`-Option `-r` in Kombination mit der Auftragsnummer stornieren, oder Sie speichern die Auftragsnummer ab, indem Sie sie umleiten. Die Stornierung erfolgt in folgendem grundlegenden Format:

```
at -r auftrags_nummer<CR>
```

Versuchen Sie nun, den zuvor eingegebenen `at`-Auftrag für den Geburtstagsgruß zu löschen. Geben Sie dazu folgendes ein:

```
at -r 453400603.a<CR>
```

Wenn Sie die Auftragsnummer vergessen haben, können Sie über das Kommando `at -l` eine Liste der aktuellen Aufträge in der `batch`- oder `at`-Warteschlange abrufen:

```
$ at -l<CR>CW
user = benutzer_name168302040.a at Sat Nov 25 13:00:00 1989
user = benutzer_name453400603.a at Fri Feb 24 08:15:00 1989
$
```

Das System zeigt die Auftragsnummer und den Zeitpunkt an, zu dem der Auftrag verarbeitet wird.

Versuchen Sie jetzt, mit dem Kommando `at`, sich selbst zur Mittagszeit eine Datei mit dem Namen `memo` zu senden, die Sie daran erinnert, daß es Zeit zum Mittagessen ist (leiten Sie die Datei nach `mail` um, oder verwenden Sie das "Here-Dokument", das im Abschnitt "Shell-Programmierung" beschrieben wird). Rufen Sie dann das Kommando `at` mit der Option `-l` auf.

```
$ at 12:00pm<CR>
mail mylogin < memo<CR>
<^d>
job 263131754.a at Jun 25 12:00:00 1989
$
$ at -l<CR>
user = benutzer_name 263131754.a at Jun 25 12:00:00 1989
$
```

Tabelle 9-9 enthält einen Überblick über Syntax und Funktionen des Kommandos `at`.

Tabelle 9-9: `at`-Kurzübersicht

Kommando		
`at` Kommandos zu einem bestimmten Zeitpunkt ausführen.		
Kommando	*Options*	*Argumente*
`at`	`-r` `-l`	*zeit (datum)* *auftrags_nummer*
Funktion:	Kommandoausführung zu einem angegebenen Zeitpunkt. Zur Angabe der Uhrzeit können ein bis vier Ziffern sowie die Angabe "`am`" oder "`pm`" verwendet werden. Das Datum wird über den Monatsnamen und eine Zahl (für den Tag im Monat) angegeben. Das Datum kann entfallen, wenn der Auftrag am gleichen Tag ausgeführt werden soll. Weitere Standard-Zeitangaben finden Sie *Referenzhandbuch für Benutzer* unter dem Eintrag `at`(1).	
Optionen:	Über die Option `-r` in Kombination mit der Auftragsnummer können Sie einen zuvor abgesetzten Auftrag stornieren. Die Option `-l` (ohne Argumente) zeigt die Auftragsnummer und den Status der Aufträge an, die mit den Kommandos `at` und `batch` abgesetzt worden sind.	

Hinweis:	Nachfolgend sind zwei Beispiele für mögliche Datums- und Uhrzeitangaben beim Kommando `at` aufgeführt. Weitere Möglichkeiten finden Sie im *Referenzhandbuch für Programmierer*.
	`at 08:15am Feb 27`
	`at 5:14pm Sept 24`	

Abfrage des Status von aktiven Prozessen

Das Kommando `ps` zeigt den Status aller Prozesse an, die aktuell verarbeitet werden. So können Sie z.B. den Status aller Prozesse abrufen, die Sie mit dem UND-Zeichen (`&`; siehe Abschnitt "Sonderzeichen") als Hintergrundprozeß aufgerufen haben.

Im nächsten Abschnitt "Abbrechen von aktiven Prozessen" wird beschrieben, wie man mit der Prozeßnummer (PID) ein Kommando abbrechen kann. Die Prozeßnummer ist eine eindeutige Nummer, die UNIX jedem aktiven Auftrag zuteilt.

Im folgenden Beispiel wird zunächst das Kommando `grep` als Hintergrundprozeß aufgerufen, dann das Kommando `ps`. Das System antwortet mit der Anzeige der Prozeßnummer (`PID`) und der Terminalnummer (`TTY`). Außerdem wird die insgesamt benötigte Ausführungszeit (`TIME`) und der Name des ausgeführten Kommandos (`COMMAND`) angezeigt.

```
$ grep word * > temp &<CR>
28223
$
$ ps<CR>
PID             TTY   TIME   COMD
28124           tty10 0:00   sh
28223           tty10 0:04   grep
28224           tty10 0:04   ps
$
```

Beachten Sie, daß das System sowohl für das Kommando `grep` als auch für die anderen aktiven Prozesse eine Prozeßnummer ausgibt: Für das Kommando `ps` selbst und das Kommando `sh` (für shell), das während Ihrer gesamten UNIX-Sitzung, bis zu Ihrer Abmeldung, aktiv ist. Das Shell-Programm `sh`, das die Shell-Kommandos interpretiert, wird in den Kapiteln 1 und 4 beschrieben.

Tabelle 9-10 enthält eine Kurzübersicht über die Syntax und Funktion des Kommandos `ps`.

Tabelle 9-10: `ps`**-Kurzübersicht**

Kommando `ps` Prozeßstatus ausgeben		
Kommando	*Optionen*	*Argumente*
`ps`	verfügbar*	keine
Funktion:	`ps` zeigt Informationen über aktive Prozesse an.	
Optionen:	Verschiedene Optionen sind verfügbar. Werden keine Optionen angezeigt, gibt `ps` den Status aller aktiven Prozesse aus.	
Hinweis:	Die ausgegebene Prozeßnummer (PID) benötigen Sie, wenn Sie den Prozeß über das Kommando `kill` abbrechen möchten.	

* Eine vollständige Übersicht über die verfügbaren Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `ps`(1).

Sie können Programme anhalten und neu starten, wenn in der Konfiguration Ihrer Benutzerumgebung die Auftragssteuerung vorgesehen ist (wenden Sie sich gegebenenfalls an Ihren Systemverwalter). Mit dem Kommando `jobs` erhalten Sie ebenfalls eine Auflistung der aktuell aktiven und angehaltenen Hintergrundprozesse. Zusätzlich zur Prozeßnummer (PID) wird beim Kommando `jobs` jedoch eine Auftragskennung (JID) sowie der Name des Kommandos (*auftragsname*) angezeigt, mit dem der Auftrag ursprünglich gestartet worden ist. Ohne die Auftragskennung können Sie weder einen angehaltenen Auftrag neu starten noch Hintergrundprozesse in den Vordergrund holen. Die Auftragskennung wird jedesmal auf dem Bildschirm angezeigt, wenn ein Prozeß durch ein Kommando gestartet oder gestoppt wird. Geben Sie folgendes ein, wenn Sie Informationen über Ihre gestoppten oder im Hintergrund aktiven Aufträge

benötigen:

```
jobs <CR>
```

Darauf werden in etwa die folgenden Informationen angezeigt:

[auftragskennung] - `Stopped` (signal) *<auftragsname>*
oder
[auftragskennung] + `Running` *<auftragsname>*

Abbrechen von aktiven Prozessen

Mit dem Kommando `kill` werden aktive Shell-Prozesse abgebrochen, die im Hintergrund ablaufen. Mit dem Kommando `stop` kann - bei aktiver Auftragssteuerung - ein Prozeß zeitweise angehalten werden. Diese Kommandos haben folgendes allgemeines Format:

`kill` *prozeß_nummer*`<CR>`
oder
`stop` *auftragskennung*`<CR>`

Beachten Sie, daß Sie Hintergrundprozesse nicht über die Tasten `BREAK` oder `DELETE` abbrechen können. Das folgende Beispiel zeigt, wie das Kommando `grep`, das im letzten Beispiel als Hintergrundprozeß aufgerufen wurde, abgebrochen wird:

```
$ kill 28223<CR>
28223 Terminated
$
```

Beachten Sie, daß das System eine Meldung und die Eingabeaufforderung `$` ausgibt. Dies zeigt Ihnen, daß der Prozeß abgebrochen worden ist. Wenn das System die angegebene Prozeßnummer nicht finden kann, gibt es folgende Fehlermeldung aus:

```
kill:28223:No such process
```

Um einen Vordergrundprozeß zeitweise anzuhalten, geben Sie folgendes ein (nur bei aktiver Auftragssteuerung):

```
ctrl Z
```

Auf dem Bildschirm erscheint jetzt in etwa die folgende Meldung:

<auftragskennung> Stopped (user) *<auftragsname>*

Tabelle 9-11 enthält einen Überblick über Syntax und Funktion des Kommandos `kill`.

Tabelle 9-11: `kill`-Kurzübersicht

Kommando `kill` Prozeß abbrechen.		
Kommando	*Optionen*	*Argumente*
`kill`	verfügbar*	*PID*
Funktion:	`kill` beendet den über die Prozeßnummer (PID) angegebenen Prozeß.	

* Einen Überblick über sämtliche verfügbaren Optionen und ihre Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `kill`(1).

Neustart eines angehaltenen Prozesses

Bei aktiver Auftragssteuerung kann ein angehaltener Prozeß neu gestartet werden. Um einen mit dem Kommando `stop` angehaltenen Prozeß neu zu starten, muß zunächst mit dem Kommando `jobs` die Auftragskennung ermittelt werden. Die Auftragskennung können Sie dann mit den folgenden Kommandos eingeben:

fg <auftragskennung>(foreground)
Die Ausführung eines angehaltenen oder Hintergrund-Auftrags wird im Vordergrund fortgesetzt.

bg <auftragskennung>(background)
Die Ausführung eines angehaltenen Auftrags wird im Hintergrund fortgesetzt.

Das Kommando nohup

Bei der Abmeldung des Benutzers werden sämtliche Prozesse mit Ausnahme der mit den Kommandos at und batch abgesetzten Aufträge abgebrochen. Damit ein Hintergrundprozeß auch nach dem Abmelden weiterverarbeitet wird, muß er mit dem Kommando nohup eingegeben werden.

Das Kommando nohup hat folgendes grundlegendes Format:

nohup *kommando* &<CR>

Das Kommando nohup wird also vor das Kommando gesetzt, das als Hintergrundprozeß ablaufen soll.

Angenommen, Sie möchten alle Dateien in Ihrem aktuellen Verzeichnis mit dem Kommando grep nach der Zeichenkette "word" durchsuchen und die Ausgabe in die Datei word.list umleiten; unmittelbar nach der Eingabe des Kommandos möchten Sie sich abmelden. Dazu geben Sie folgende Kommandozeile ein:

```
nohup grep word * > word.list & <CR>
```

Das Kommando nohup kann mit dem Kommando kill abgebrochen werden. Tabelle 9-12 enthält einen Überblick über Syntax und Funktion des Kommandos nohup.

Tabelle 9-12: `nohup`**-Kurzübersicht**

Kommando		
`nohup` Abbruch eines Kommandos beim Abmelden oder Ausschalten des Terminals verhindern.		
Kommando	*Optionen*	*Argumente*
`nohup`	*keine*	*kommandozeile*
Funktion:	Dieses Kommando bewirkt die Weiterverarbeitung eines Kommandos, nachdem Sie sich abgemeldet oder das Terminal ausgeschaltet haben.	

Sie beherrschen jetzt die grundlegenden Shell-Kommandos und ihr Eingabeformat. Verwenden Sie sie nun in Ihren Shell-Programmen! Im folgenden finden Sie einige Anwendungsbeispiele, mit denen Sie die Kommandosprache der Shell üben können. Die Auflösung der Übungen finden Sie am Ende des Kapitels.

Übungen zur Kommandosprache

1-1. Was geschieht, wenn Sie einen Stern (*) am Anfang eines Dateinamens verwenden? Versuchen Sie einige Dateien Ihres Verzeichnisses aufzulisten, indem Sie * zusammen mit dem letzten Buchstaben eines Dateinamens angeben. Was geschieht?

1-2. Versuchen Sie die folgenden beiden Kommandos einzugeben:

```
cat[0-9]*<CR>
echo *<CR>
```

1-3. Kann das Fragezeichen (?) am Anfang oder in der Mitte einer Zeichenkette eingegeben werden? Probieren Sie es aus.

1-4. Beginnen einige Ihrer Dateien mit einer Zahl? Können Sie sie ohne die übrigen Dateien Ihres Verzeichnisses auflisten? Können Sie nur die Dateien auflisten, die mit den Kleinbuchstaben a und m beginnen? (Tip: Benutzen Sie die eckigen Klammern ([ ]) mit einem Zahlen- oder Buchstabenbereich).

1-5. Kann ein Kommando als Hintergrundprozeß durchgeführt werden, das in einer Kommandozeile mit mehreren hintereinander ausgeführten Kommandos enthalten ist? Was geschieht? (Tip: Verwenden Sie ; und &.). Kann das Kommando, das als Hintergrundprozeß abläuft, an jeder beliebigen Stelle der Kommandozeile stehen? Geben Sie das Kommando probehalber an verschiedenen Stellen ein. Experimentieren Sie mit jedem neu gelernten Zeichen, um seine Leistungsfähigkeit kennenzulernen.

1-6. Leiten Sie die Ausgabe von `pwd` und `ls` in eine Datei um. Verwenden Sie dazu folgende Kommandozeile:

```
cd; pwd; trial; ls >> trial <CR>
```

Beachten Sie, daß Sie für die zweite Umleitung das Zeichen >> (anhängen) verwenden müssen, wenn Sie beide Kommandos in die gleiche Datei umleiten; andernfalls wird die Ausgabe des Kommandos `pwd` gelöscht.

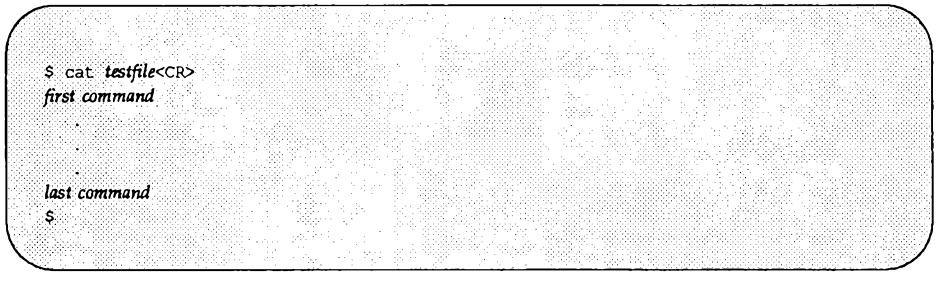
1-7. Extrahieren Sie aus der Ausgabe von `date` nicht die Uhrzeit, sondern versuchen Sie, nur das Datum (ohne Uhrzeit) an das Kommando `banner` weiterzuleiten. Welchen Teil der folgenden Kommandozeile müssen Sie dafür als einziges verändern?

```
banner `date | cut -c12-19`<CR>
```

Shell-Programmierung

Mit der Shell können Sie Programme bzw. neue Kommandos erstellen. Derartige Programme werden "Shell-Prozeduren" genannt. In diesem Abschnitt erfahren Sie, wie Sie Shell-Programme mit Kommandos, Variablen, Stellungsparametern, Rückkehr-Codes und einfachen Ablaufsteuerungen erstellen und ausführen können.

Die Shell-Programme werden in diesem Abschnitt auf zweierlei Weise präsentiert. Zum einen wird der Inhalt der Datei, die ein Shell-Programm enthält, mit Hilfe des Kommandos `cat` auf dem Bildschirm angezeigt:



Zum anderen wird das Ergebnis eines Shell-Programms nach einer Kommandozeile gezeigt:

```
$ testfile<CR>
programm_ausgabe
$
```

Die Erstellung von Shell-Programmen setzt Kenntnisse über einen Editor voraus (siehe Kapitel 6 über den Editor `ed`) und Kapitel 7 über den Editor `vi`).

Shell-Programme

Erstellung eines einfachen Shell-Programms

Zunächst erstellen wir ein einfaches Shell-Programm, das hintereinander die folgenden Aufgaben ausführt:

- Namen des aktuellen Verzeichnisses ausgeben.

- Inhalt dieses Verzeichnisses ausgeben.
- Folgende Meldung auf Ihrem Terminal anzeigen:

  "This is the end of the shell program."

Legen Sie mit einem Editor die Datei dl (für directory list) an und geben Sie folgendes ein:

```
pwd<CR>
ls<CR>
echo This is the end of the shell program.<CR>
```

Speichern Sie diese Zeilen ab, und verlassen Sie die Datei. Sie haben soeben ein Shell-Programm geschrieben! Mit dem Kommando cat können Sie den Inhalt der Datei abrufen; die Bildschirmanzeige sieht dann folgendermaßen aus:

```
$ cat dl<CR>
pwd
ls
echo This is the end of the shell program.
$
```

Ausführung eines Shell-Programms

Ein Shell-Programm können Sie beispielsweise mit dem Kommando sh ausführen. Dafür geben Sie folgendes ein:

```
sh dl<CR>
```

Das Kommando dl wird durch sh aufgerufen; zunächst wird der Pfadname des aktuellen Verzeichnisses angezeigt, dann die Liste der Dateien im aktuellen Verzeichnis, und schließlich der Kommentar This is the end of the shell program. Mit dem Kommando sh können Sie die Funktionsfähigkeit des Shell-Programms austesten.

Wenn die Datei dl ein Kommando ist, mit dem Sie öfter arbeiten möchten, können Sie sie mit dem Kommando chmod (siehe *Referenzhandbuch für Programmierer*) zu einer ausführbaren Datei machen. Danach brauchen Sie zur Ausführung der Kommandos in dieser Datei nur noch dl einzugeben. Das folgende Beispiel zeigt, wie man eine Datei mit dem Kommando chmod zu einer

ausführbaren Datei macht und dann das Kommando `ls -l` aufruft, um die neuen Zugriffsberechtigungen zu überprüfen (eine Beschreibung des Kommandos `ls -l` finden Sie im *Referenzhandbuch für Programmierer*).

```
$ chmod u+x dl<CR>
$  ls -l<CR>
total 2
-rw-------          1   name    name      3661    Nov  2      10:28 mbox
-rwx------          1   name    name        48    Nov 15      10:50 dl
$
```

Mit dem Kommando `chmod` wurde dem Benutzer (`u`) die Ausführungsberechtigung (`+x`) erteilt. `dl` ist jetzt ein ausführbares Programm. Versuchen Sie es folgendermaßen aufzurufen:

```
dl<CR>
```

Auf dem Bildschirm wird jetzt dieselbe Ausgabe wie oben bei der Eingabe von `sh dl` angezeigt. Weitere Informationen über das Kommando `chmod` finden Sie in Kapitel 3.

Erstellung eines `bin`-Verzeichnisses für ausführbare Dateien

Um von jedem Ihrer Verzeichnisse aus auf Ihre Shell-Programme zugreifen zu können, können Sie in Ihrem Home-Verzeichnis ein `bin`-Verzeichnis anlegen und alle Shell-Dateien in dieses Verzeichnis versetzen.

Das Verzeichnis `bin` muß dann auch in den Zugriffspfad Ihrer Umgebungsvariablen `PATH` eingebracht werden:

```
PATH=$PATH:$HOME/bin
```

Weitere Informationen über `PATH` finden Sie in den Abschnitten "Variablen" und "Shell-Variablen" in diesem Kapitel.

Das folgende Beispiel zeigt Ihnen nochmals, welche Kommandos zum Anlegen eines Verzeichnisses benötigt werden. In diesem Beispiel ist `dl` im Home-Verzeichnis enthalten. Geben Sie folgende Kommandozeilen ein:

```
cd<CR>
mkdir bin<CR>
mv dl bin/dl<CR>
```

Wechseln Sie in das Verzeichnis `bin`, und geben Sie das Kommando `ls -l` ein. Ist `dl` immer noch ausführbar?

Wechseln Sie nun in ein Verzeichnis, das nicht das Home-Verzeichnis ist, und geben Sie folgendes Kommando ein:

```
dl<CR>
```

Was geschieht?

Tabelle 9-13 zeigt einen Überblick über Ihr neues Shell-Programm `dl`.

Tabelle 9-13: Kurzübersicht über das Shell-Programm `dl`

Shell-Programm `dl` Anzeige des Verzeichnispfades und Ausgabe des Verzeichnisinhalts (benutzer-definiert).	
Kommando	*Argumente*
`dl`	keine
Funktion:	`dl` zeigt den Namen und den Inhalt des aktuellen Verzeichnisses an.

Sie können dem Verzeichnis `bin` auch einen anderen Namen geben; in diesem Fall müssen Sie die Umgebungsvariable `PATH` erneut ändern.

Hinweise zur Benennung von Shell-Programmen

Sie können Ihrem Shell-Programm jeden beliebigen Dateinamen geben. Achten Sie jedoch darauf, daß der Name sich nicht mit dem Namen eines Shell-Kommandos überschneidet; andernfalls führt das System Ihr Kommando und nicht das Systemkommando aus. Wenn Sie z.B. Ihrem `dl`-Programm den Namen `mv` geben, führt das System jedesmal Ihr Programm zur Ausgabe des Verzeichnisinhalts aus, wenn Sie `mv` zum Versetzen einer Datei aufrufen.

Ein anderes Problem kann auftreten, wenn Sie der Datei `dl` den Namen `ls` geben und dann versuchen die Datei auszuführen. Sie erzeugen dadurch eine Endlosschleife, da Ihr Programm versucht, das Kommando `ls` auszuführen. Nach einiger Zeit gibt das System die folgende Fehlermeldung aus:

```
Too many processes, cannot fork
```

Was ist geschehen? Nachdem Sie Ihr neues Kommando (`ls`) eingegeben haben, las die Shell das Kommando `pwd` und führte es aus. Dann las es in Ihrem Programm das Kommando `ls` und versuchte, Ihr `ls`-Kommando auszuführen. Somit entstand eine Endlosschleife. Aus diesem Grund ist die Anzahl der Wiederholungen von Endlosschleifen unter UNIX eingeschränkt. Endlosschleifen können Sie beispielsweise verhindern, indem Sie in Ihrem Shell-Programm für das System-Kommando `ls` den Pfadnamen `/usr/bin/ls` angeben.

Das Shell-Programm `ls` in folgendem Beispiel ist funktionsfähig:

```
$ cat ls<CR>
pwd
/bin/ls
echo This is the end of the shell program
```

Wenn Sie Ihrem Kommando den Namen `ls` zuordnen, müssen Sie das System-Kommando `ls` auf jeden Fall mit seinem absoluten Pfadnamen `/usr/bin/ls` aufrufen.

Variablen

Die Variablen sind neben den Dateien die grundlegenden Datenobjekte, die in Shell-Programmen manipuliert werden. Im folgenden werden drei Variablen-Typen und Ihre Anwendung beschrieben:

- Stellungsparameter
- Spezielle Parameter
- Schlüsselwortparameter

Stellungsparameter

Ein Stellungsparameter ist eine Variable innerhalb eines Shell-Programms. Ihr Wert wird einem Argument entnommen, das in der Aufrufzeile des Programms angegeben wird. Stellungsparameter bestehen aus Zahlen, die auf das Dollar-Zeichen (`$`) folgen (`$1`, `$2`, `$3` usw.).

Ein Shell-Programm kann auf bis zu neun Stellungsparameter Bezug nehmen. Wenn ein Shell-Progamm mit einer Kommandozeile wie

```
shell.prog pp1 pp2 pp3 pp4 pp5 pp6 pp7 pp8 pp9<CR>
```

aufgerufen wird, so wird beim Aufrufen des Shell-Programms dem Stellungsparameter `$1` innerhalb des Programms der Wert `pp1` zugeordnet, dem Stellungsparameter `$2` der Wert `pp2` usw.

Legen Sie zur Übung der Stellungsparameter-Substitution eine Datei namens `pp` (für positional parameters) an. Tragen Sie die `echo`-Kommandos ein, die auf dem folgenden Bildschirm gezeigt werden. Tragen Sie die Kommandozeilen so ein, daß die Durchführung des Kommandos `cat` für die fertige Datei die folgende Ausgabe ergibt:

```
$ cat pp<CR>
echo  The first positional parameter is: $1<CR>
echo  The second positional parameter is: $2<CR>
echo  The third positional parameter is: $3<CR>
echo  The fourth positional parameter is: $4<CR>
$
```

Wenn Sie nun das Shell-Programm mit den Argumenten `one`, `two`, `three` und `four` aufrufen, erhalten Sie die folgenden Ergebnisse (zunächst müssen Sie jedoch das Shell-Programm `pp` mit dem Kommando `chmod` zu einem ausführbaren Programm machen):

```
$ chmod u+x pp<CR>
$
$ pp one two three four<CR>
The first positional parameter is: one
The second positional parameter is: two
The third positional parameter is: three
The fourth positional parameter is: four
$
```

Der folgende Bildschirm zeigt das Shell-Programm `bbday` zur Übermittlung von Grüßen an den in der Kommandozeile angegebenen Benutzernamen:

```
$ cat bbday<CR>
banner happy birthday | mail $1
```

Versuchen Sie nun, sich selbst einen Geburtstagsgruß zu senden. Wenn Sie den Benutzernamen `sue` haben, sieht Ihre Kommandozeile folgendermaßen aus:

```
bbday sue<CR>
```

Tabelle 9-14 enthält einen Überblick über die Syntax und die Funktion des Shell-Programms `bbday`.

Tabelle 9-14: `bbday`**-Kurzübersicht**

Shell-Programm `bbday` Geburtstagsgruß in großer Schrift senden.	
Kommando	*Argumente*
`bbday`	*login*
Funktion:	`bbday` sendet die Nachricht : 'happy birthday' in großer Schrift an den angegebenen Benutzer.

Mit dem Kommando `who` werden alle Benutzer angezeigt, die aktuell am System angemeldet sind. Wie sieht ein einfaches Shell-Programm aus, mit dem Sie feststellen können, ob der Benutzer mit einem bestimmten Benutzernamen aktuell am System angemeldet ist?

Geben Sie in einer Datei mit dem Namen `whoson` folgende Kommandozeile ein:

```
who | grep $1<CR>
```

Das Kommando `who` gibt alle Benutzer aus, die aktuell angemeldet sind; das Kommando `grep` sucht in dieser Ausgabe nach einer Zeile, die die für den Stellungsparameter `$1` angegebene Zeichenkette enthält.

Versuchen Sie jetzt Ihren eigenen Benutzernamen als Argument für das neue Programm `whoson` anzugeben. Angenommen, Ihr Benutzername ist `sue`. Wenn Sie das Kommando `whoson` aufrufen, ersetzt das Shell-Programm den Parameter `$1` in Ihrem Programm durch `sue`. Das Programm wird ausgeführt, als hätten Sie folgendes eingegeben:

```
who | grep sue <CR>
```

Der folgende Bildschirm zeigt die Ausgabe:

```
$ whoson sue<CR>
sue   tty26      Jan 24 13:35
$
```

Wenn der Benutzer mit dem angegebenen Benutzernamen aktuell nicht am System angemeldet ist, kann das Kommando `grep` nicht durchgeführt werden; in diesem Fall erzeugt das Kommando `whoson` keine Ausgabe.

Bild 9-15 faßt die Syntax und die Funktion des Kommandos `whoson` zusammen.

Tabelle 9-15: `whoson`**-Kurzübersicht**

Shell-Programm `whoson` Informationen über einen Benutzer anzeigen, falls dieser Benutzer angemeldet ist (benutzer-definiert).	
Kommando	*Argumente*
`whoson`	*benutzername*
Funktion:	Wenn der Benutzer angemeldet ist, zeigt `whoson` den Benutzernamen des Benutzers, die Terminal-Nummer sowie Datum und Zeitpunkt seiner Anmeldung an.

Die Kommandozeile der Shell kann maximal 128 Argumente enthalten. Ein Shell-Programm kann jedoch nur auf bis zu neun Stellungsparameter (`$1` bis `$9`) gleichzeitig Bezug nehmen. Diese Einschränkung kann mit dem Kommando `shift` umgangen werden, das im Handbuch *Shell Commands and Programming* beschrieben wird. Um auf die Werte aller in der Kommandozeile angegebenen

Argumente Bezug zu nehmen, kann auch der spezielle Parameter `$*` (siehe nächster Abschnitt) verwendet werden.

Spezielle Parameter

`$#` Diesem Parameter ist innerhalb eines Shell-Programms die Anzahl der Argumente zugeordnet, mit der das Shell-Programm aufgerufen worden ist. Dieser Wert kann an einer beliebigen Stelle des Shell-Programms verwendet werden.

Tragen Sie die Kommandozeile, die auf dem folgenden Bildschirm gezeigt wird, in das ausführbare Shell-Programm `get.num` ein. Führen Sie dann für das Programm das Kommando `cat` durch:

```
$ cat get.num<CR>
echo The number of arguments is: $#
$
```

Das Programm gibt ausschließlich die Anzahl der Argumente aus, mit denen es aufgerufen worden ist:

```
$ get.num test out this program<CR>
The number of arguments is: 4
$
```

Tabelle 9-16 enthält einen Überblick über das Shell-Programm `get.num`.

Tabelle 9-16: Kurzübersicht über das Shell-Programm `get.num`

Shell-Programm `get.num` Anzahl der Argumente zählen und anzeigen (benutzer-definiert).	
Kommando	*Argumente*
`get.num`	*(zeichenkette)*
Funktion:	`get.num` zählt die Anzahl der Argumente in der Aufrufzeile des Kommandos und zeigt die Gesamtzahl an.
Hinweis:	Dieses Kommando ist ein Beispiel für die Funktionsweise des speziellen Parameters `$#`.

`$*` Diesem speziellen Parameter ist eine Zeichenkette zugeordnet, die aus allen in der Aufrufzeile des Shell-Programms enthaltenen Argumente besteht. Für diesen Parameter gibt es - im Gegensatz zu den Stellungsparametern `$1` bis `$9` - keine Einschränkung hinsichtlich der Anzahl der Argumente.

Im folgenden wird die Funktionsweise von `$*` anhand eines einfachen Shell-Programms gezeigt. Erstellen Sie ein Shell-Programm mit dem Namen `show.param`, das mit dem Kommando `echo` alle Parameter anzeigt. Tragen Sie dazu die folgende `echo`-Kommandozeile in die Datei ein:

```
$ cat show.param<CR>
echo The parameters for this command are: $*
$
```

Das Programm `show.param` gibt alle Argumente aus, die in der Aufrufzeile des Kommandos enthalten sind. Machen Sie `show.param` zu einer ausführbaren Datei. Probieren Sie das Programm mit den folgenden Parametern aus:

```
Hello. How are you?
```

```
$ show.param Hello. How are you?<CR>
The parameters for this command are: Hello.  How are you?
$
```

`show.param` gibt also `Hello. How are you?` aus. Versuchen Sie jetzt `show.param` mit mehr als neun Argumenten aufzurufen:

```
$ show.param one two 3 4 5 six 7 8 9 10 11<CR>
The parameters for this command are: one two 3 4 5 six 7 8 9 10 11
$
```

`show.param` zeigt auch in diesem Fall alle in der Aufrufzeile enthaltenen Argumente an. Der Parameter `$*` ist vor allem dann nützlich, wenn die Argumente eines Shell-Kommandos Metazeichen zur Dateinamenerweiterung enthalten.

Probieren Sie die Dateinamenerweiterung an Ihrem Kommando `show.param` aus. Angenommen, Ihr Verzeichnis enthält mehrere Dateien, die jeweils ein Kapitel eines Buches enthalten. Die Dateien haben die Namen `chap1`, `chap2` ... `chapF`. Mit dem Kommando `show.param` wird eine Liste dieser Dateien angezeigt.

```
$ show.param chap?<CR>
The parameters for this command are: chap1 chap2 chap3
chap4 chap5 chapB chapF
$
```

Tabelle 9-17 enthält einen Kurzüberblick über das Shell-Programm `show.param`.

Tabelle 9-17: Kurzübersicht über das Shell-Programm `show.param`

Shell-Programm `show.param` Alle Stellungsparameter anzeigen (benutzer-definiert).	
Kommando	*Argumente*
`show.param`	(beliebige stellungsparameter)
Funktion:	`show.param` zeigt alle Parameter an.
Hinweis:	Enthalten die Parameter Metazeichen zur Dateinamenerweiterung, so werden alle passenden Dateinamen ausgegeben.

Schlüsselwortparameter

Eine andere Variablenart, die in Shell-Programmen verwendet wird, ist der Schlüsselwortparameter. Die Zuweisung eines Werts an einen Schlüsselwortparameter erfolgt in folgendem Format:

schlüsselwortparameter=wert<CR>

Achten Sie darauf, daß sich vor und nach dem Gleichheitszeichen (=) kein Leerzeichen befindet.

Im folgenden Beispiel ist `var1` ein Schlüsselwortparameter und `myname` der Wert, der dieser Variablen zugewiesen wird:

```
var1=myname<CR>
```

Mit dem Dollar-Symbol (`$`), auf das ein Variablenname folgt, wird in einem Shell-Programm Bezug auf den Wert dieser Variablen genommen. Im letzten Beispiel weist die Variable `$var1` die Shell an, die Zeichenkette `$var1` bei jedem Auftreten durch den Wert `myname` (also den Wert der Variablen `var1`) zu ersetzen.

Beim ersten Zeichen eines Variablennamens muß es sich um einen Buchstaben oder einen Unterstrich handeln. Der übrige Teil des Namens kann aus Buchstaben, Unterstrichen und Ziffern bestehen. Wie bei Dateinamen von Shell-Programmen sollte es keine Überschneidung zwischen Variablen- und Shell-Programmnamen geben. Außerdem gibt es einige vorbelegte Shell-Variablennamen, die Sie nicht für Ihre Variablen verwenden dürfen. Dabei handelt es sich um folgende Shell-Variablennamen:

- `CDPATH` definiert den Suchpfad für das Kommando `cd`.
- `HOME` ist die Standardvariable für das Kommando `cd` (Home-Verzeichnis).
- `IFS` definiert die internen Feldbegrenzer (normalerweise das Leerzeichen, das Tabulatorzeichen sowie das Wagenrücklaufzeichen).
- `LOGNAME` ist Ihr Benutzername.
- `MAIL` gibt den Namen Ihres elektronischen Briefkastens an.
- `PATH` definiert den Suchpfad, den die Shell nach Kommandos durchsucht.
- `PS1` definiert die primäre Eingabeaufforderung (Standard: `$`).
- `PS2` definiert die sekundäre Eingabeaufforderung (Standard: >).
- `TERM` gibt den Terminaltyp an. Dieser Wert ist für das Editieren mit `vi` unerläßlich.
- `TERMINFO` gibt das Verzeichnis an, das nach Informationen über Ihr Terminal (z.B. die Bildschirmgröße) durchsucht werden soll.
- `TZ` definiert die Zeitzone (Standard: `EST5EDT`).

Viele dieser Variablen werden im Abschnitt "Änderung Ihrer Benutzerumgebung" in diesem Kapitel beschrieben. Weitere Informationen finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `sh`(1).

Wenn Sie sich über die Belegung Ihrer Shell-Variablen informieren möchten, haben Sie zwei Möglichkeiten. Mit der folgenden Kommandozeile können Sie

den Wert einer einzelnen Variablen abrufen:

```
echo $variablen_name
```

Das System zeigt den Wert von *variablen_name* an. Einen Überblick über die Werte sämtlicher Shell-Variablen erhalten Sie mit dem Kommando env(1). Dazu geben Sie einfach env in einer separaten Kommandozeile ein. Das System zeigt eine Liste der Variablennamen und der zugehörigen Werte an.

Wertzuweisung an eine Variable

In einer vi-Sitzung können Sie die Variable TERM mit der folgenden Kommandozeile einstellen:

```
TERM=terminal_name<CR>
EXPORT TERM
```

Dies ist die einfachste Methode der Wertzuweisung an eine Variable. Es gibt jedoch noch zahlreiche andere Möglichkeiten:

- Weisen Sie der Variablen die Eingabe des Kommandos read zu.
- Setzen Sie die Ausgabe eines Kommandos als Wert der in Gegenhochkommata (` ... `) eingeschlossenen Variablen ein (Kommandosubstitution).
- Weisen Sie der Variablen einen Stellungsparameter zu.

In den folgenden Abschnitten werden diese Möglichkeiten ausführlich beschrieben.

Das Kommando read

Das Kommando read kann innerhalb eines Shell-Programms benutzt werden, um den Benutzer zur Eingabe von Werten für Variablen aufzufordern. Das Kommando read wird in folgendem Format eingegeben:

```
read variable<CR>
```

$*variable* wird bei jedem Auftreten im Programm durch den Wert ersetzt, der *variable* mit dem Kommando read zugeordnet worden ist. Wenn das Kommando echo in einem Programm vor dem Kommando read steht, kann man damit das Programm zur Anzeige von Anweisungen veranlassen (in folgendem Beispiel Type in ...). Das Kommando read wartet, bis Sie die Zeichenkette eingeben und die Taste RETURN betätigen. Die eingegebene Zeichenkette wird dann zum Wert der Variablen.

Das folgende Beispiel zeigt ein einfaches Shell-Programm namens `num.please`, mit dem man Telefonnummern aus einem Telefonverzeichnis auf dem Bildschirm abrufen kann. Das Programm verwendet dafür die folgenden Kommandos:

`echo`	zeigt eine Aufforderung zur Eingabe des Nachnamens der betreffenden Person an.
`read`	ordnet den eingegebenen Wert der Variablen `name` zu.
`grep`	sucht in der Datei `list` nach dieser Variablen.

Ihr fertiges Programm sollte jetzt folgendermaßen aussehen:

```
$ cat num.please<CR>
echo Type in the last name:
read name
grep $name home/list
$
```

Erstellen Sie eine Datei mit dem Namen `list`, die verschiedene Nachnamen und Telefonnummern enthält. Versuchen Sie dann, das Programm `num.please` aufzurufen.

Das nächste Beispiel stellt ein Programm mit dem Namen `mknum` vor, mit dem ein Telefonverzeichnis erstellt werden kann. Für diesen Zweck enthält `mknum` die folgenden Kommandos:

- `echo` fordert Sie zur Eingabe des Namens der betreffenden Person auf.
- `read` ordnet den Namen der Person der Variablen *name* zu.
- `echo` fordert Sie zur Eingabe der Telefonnummer der Person auf.
- `read` ordnet die Telefonnummer der Variablen *num* zu.
- `echo` trägt die Werte der Variablen *name* und *num* in die Datei `list` ein.

Um die Ausgabe des Kommandos `echo` am Ende des Telefonverzeichnisses `list` einzufügen, müssen Sie sie mit >> umleiten. Mit dem Umleitungssymbol > enthält `list` nur die zuletzt eingefügte Telefonnummer.

Bei der Eingabe des Kommandos `cat` für `mknum` wird der Inhalt des Programms angezeigt. Mit einem Programm, wie es nachfolgend gezeigt wird, können Sie es (mit dem Kommando `chmod`) zu einem ausführbaren Programm machen:

```
$ cat mknum<CR>
echo Type in name
read name
echo Type in number
read num
echo $name $num >> list
$ chmod u+x mknum<CR>
$
```

Probieren das neue Programm an Ihrem Telephonverzeichnis aus. Im nächsten Beispiel erstellt `mknum` einen neuen Eintrag für Mr. Niceguy. `num.please` gibt dann die Telephonnummer von Mr. Niceguy aus:

```
$ mknum<CR>
Type in name
Mr. Niceguy<CR>
Type in number
668-0007<CR>
$ num.please<CR>
Type in last name
Niceguy<CR>
Mr. Niceguy 668-0007
$
```

Beachten Sie, daß als Wert der Variablen `name` sowohl `Mr.` als auch `Niceguy` angegeben werden kann.

Die Tabellen 9-18 und 9-19 enthalten einen Überblick über die Shell-Programme `mknum` bzw. `num.please`.

Tabelle 9-18: Kurzübersicht über das Shell-Programm `mknum`

Shell-Programm `mknum` Namen und Telephonnummer in ein Telephonverzeichnis schreiben.	
Kommando	*Argumente*
`mknum`	(interaktiv)
Funktion:	Dieses Programm fordert Sie zur Eingabe des Namens und der Telephonnummer einer Person auf und trägt den Namen und die Telephonnummer in ein Telephonverzeichnis ein.
Hinweis:	Das Kommando ist interaktiv.

Tabelle 9-19: Kurzübersicht über das Shell-Programm `num.please`

Shell-Programm `num.please` Namen und Telephonnummer einer Person anzeigen.	
Kommando	*Argument*
`num.please`	(interaktiv)
Funktion:	Dieses Programm fordert Sie zur Eingabe des Nachnamens einer Person auf und zeigt dann den vollen Namen und die Telephonnummer der Person an.
Hinweis:	Das Kommando ist interaktiv.

Ausgabe eines Kommandos als Wert einer Variablen einsetzen

Die Ausgabe eines Kommandos kann mit Hilfe der sogenannten *Kommandosubstitution* als Wert einer Variablen eingesetzt werden. Eine Kommandosubstitution erfolgt in folgendem Format:

variable=`*kommando*`<CR>

Die Ausgabe von *kommando* wird als Wert der Variablen *variable* verwendet.

In einem der letzten Beispiele wurde das Kommando `date` über eine Pipe zum Kommando `cut` weitergeleitet, um die aktuelle Uhrzeit zu ermitteln. Die Kommandozeile sah folgendermaßen aus:

```
date | cut -c12-19<CR>
```

Diese Kommandofolge kann auch in ein einfaches Shell-Programm namens `t` eingebracht werden, das Ihnen die Uhrzeit anzeigt.

```
$ cat t<CR>
time=`date | cut -c12-19`
echo The time is: $time
$
```

Beachten Sie stets, daß vor und hinter dem Gleichheitszeichen (=) kein Leerzeichen stehen darf. Machen Sie die Datei zu einer ausführbaren Datei. Sie haben nun ein Programm, das Sie über die Uhrzeit informiert:

```
$ chmod u+x t<CR>
$ t<CR>
The time is: 10:36
$
```

Tabelle 9-20 enthält eine Übersicht über Ihr Shell-Programm `t`.

Tabelle 9-20: Kurzübersicht über das Shell-Programm `t`

Shell-Programm `t` Aktuelle Uhrzeit anzeigen	
Kommando	*Argumente*
`t`	keine
Funktion:	`t` zeigt die Uhrzeit in Stunden und Minuten an.

Wertzuweisung mit Stellungsparametern

Mit einer Kommandozeile in folgendem Format können Sie einem Schlüsselwortparameter einen Stellungsparameter zuordnen:

var1=$1<CR>

Das nächste Beispiel ist ein einfaches Programm namens `simp.p`, mit dem Sie einer Variablen einen Stellungsparameter zuordnen können. Der folgende Bildschirm zeigt die Kommandos in `simp.p`:

```
$ cat simp.p<CR>
var1=$1
echo $var1
$
```

Sie können einer Variablen natürlich auch die Ausgabe eines Kommandos zuordnen, das Stellungsparameter verwendet:

```
person=`who | grep $1`<CR>
```

Das Programm `log.time` im nächsten Beispiel protokolliert das Ergebnis des Programms `whoson`. Die Ausgabe des Programms `whoson` wird der Variablen `person` zugeordnet und mit dem Kommando `echo` in die Datei `login.file` eingebracht. Das letzte `echo`-Kommando zeigt den Wert von `$person` an, der der Ausgabe des Kommandos `whoson` entspricht:

```
$ cat log.time<CR>
person=`who | grep $1`
echo $person >> $home/login.file
echo $person
$
```

Folgender Bildschirm zeigt die Ausgabe des Systems beim Programm `log.time`:

```
$ log.time maryann<CR>
maryann     tty61          Apr 11 10:26
$
```

Tabelle 9-21 enthält einen Überblick über das Shell-Programm `log.time`.

Tabelle 9-21: Kurzübersicht über das Shell-Programm `log.time`

Shell-Programm `log.time` Benutzernamen protokollieren und anzeigen (benutzer-definiert).	
Kommando	*Argument*
`log.time`	`benutzer_name`
Funktion:	Wenn der betreffende Benutzer aktuell am System angemeldet ist, bringt das Programm `log.time` die von `who` erzeugte Ausgabe in die Datei `login.file` ein und zeigt sie dann auf Ihrem Terminal an.

Shell-Programmstrukturen

Die Programmiersprache der Shell unterstützt verschiedene Strukturen, die Ihren Programmen zusätzliche Flexibilität verleihen:

- Mit Kommentaren können Sie die einzelnen Funktionen des Programms erläutern.
- Mit dem "Here-Dokument" können Sie in das Shell-Programm selbst Zeilen einfügen, die von einem Kommando im Shell-Programm als Eingabe benutzt werden sollen.
- Mit dem Kommando `exit` können Sie ein Programmm an einer beliebigen Stelle, außer am Programmende, beenden und Rückkehr-Codes festlegen.
- Mit einer `for`- und `while`-Schleife können Sie ein Programm veranlassen, Kommandos mehrmals zu wiederholen.

- Mit den Ablaufanweisungen `if` und `case` können Sie das Programm anweisen, eine Kommandogruppe nur dann auszuführen, wenn bestimmte Bedingungen erfüllt sind.
- Mit dem Kommando `break` können Sie eine Schleife sofort beenden, ohne daß eine Abbruchbedingung angegeben ist.

Kommentare

Zur Kommentierung von Shell-Programmzeilen gibt es zwei Möglichkeiten. Die Shell ignoriert alle Zeichen, die in einer Zeile hinter einem Nummern-Zeichen (#) stehen. Steht das Zeichen # ganz am Anfang der Zeile, wird die ganze Zeile als Kommentar betrachtet. Folgt das Zeichen # auf ein Kommando, so wird das Kommando ausgeführt; der übrige Teil der Zeile wird ignoriert. Ein Kommentar endet auf jeden Fall mit dem Zeilenendezeichen. Eine Kommentarzeile hat folgendes grundlegendes Format:

#kommentar<CR>

Die folgenden Zeilen eines Programms werden bei der Ausführung beispielsweise ignoriert:

```
# This program sends a generic birthday greeting.<CR>
# This program needs a login as<CR>
# the positional parameter.<CR>
```

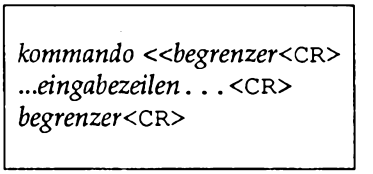
Mit Kommentaren können Sie die Funktionen eines Programms erläutern. Sie sollten in jedes Ihrer Programme Kommentare einbringen.

Das Here-Dokument

Mit dem "Here-Dokument" können in ein Shell-Programm Zeilen eingefügt werden, die umgeleitet und von einem Kommando in diesem Programm als Eingabe verwendet werden sollen. Auf diese Weise kann in einem Shell-Programm eine Eingabe für ein Kommando angegeben werden, ohne daß dafür eine separate Datei verwendet werden muß. Ein Here-Dokument besteht aus dem Umleitungssymbol ≪ sowie einem Begrenzer, der den Anfang und das Ende der Eingabezeilen angibt. Der Begrenzer kann aus einem Zeichen oder einer Zeichenkette bestehen; oft wird dafür das Ausrufezeichen (!) verwendet.

Bild 9-1 zeigt das allgemeine Format eines Here-Dokuments.

Bild 9-1: Format eines Here-Dokuments



Im nächsten Beispiel verwendet das Programm `gbday` ein Here-Dokument, um einen vorformulierten Geburtstagsgruß abzuschicken; die Eingabezeilen werden zum Kommando `mail` umgeleitet:

```
$ cat gbday<CR>
mail $1 <<!
Best wishes to you on your birthday.
!
$
```

Bei diesem Kommando muß der Benutzername des Empfängers als Argument angegeben werden. Das Here-Dokument fügt dann die folgende Eingabe in das Programm ein:

```
Best wishes to you on your birthday
```

Um diese Grüße an den Benutzer mit dem Benutzernamen `mary` zu schicken, muß folgende Kommandozeile eingegeben werden:

```
$ gbday mary<CR>
```

Wenn `mary` das nächste Mal ihre Post liest, erhält sie folgende Grüße:

```
$ mail<CR>
From mylogin Wed May 14 14:31 CDT 1986
Best wishes to you on your birthday
$
```

Tabelle 9-22 enthält einen Überblick über Format und Funktion des Kommandos `gbday` .

Tabelle 9-22: `gbday`**-Kurzübersicht**

Shell-Programm `gbday` Vorformulierten Geburtstagsgruß senden (benutzer-definiert).	
kommando	*argumente*
`gbday`	*benutzername*
Funktion:	`gbday` sendet einen vorformulierten Geburtstagsgruß an den als Argument angegebenen Benutzer.

Der Editor `ed` in einem Shell-Programm

Here-Dokumente bieten eine bequeme und gute Möglichkeit zur Benutzung des Editors `ed` in einer Shell-Prozedur. Angenommen, Sie möchten ein Shell-Programm erstellen, das den Editor `ed` aufruft, in einer Datei einen globalen Ersetzungs-Vorgang durchführt, die Datei abspeichert und dann den Editor `ed` beendet. Der folgende Bildschirm zeigt den Inhalt der Programmdatei `ch.text`, mit der diese Funktionen ausgeführt werden können.

```
$ cat ch.text<CR>
echo Type in the filename.
read file1
echo Type in the exact text to be changed.
read old_text
echo Type in the exact new text to replace the above.
read new_text
ed - $file1 <<!
g/$old_text/s//$new_text/g
w
q
!
$
```

Beachten Sie die Option - (Minus) des Kommandos `ed`. Diese Option verhindert, daß die Anzahl der Zeichen auf dem Bildschirm angezeigt wird. Beachten Sie auch das Format des globalen Ersetzen-Kommandos von `ed`:

`g/`*text_alt*`/s//`*text_neu*`/g<CR>`

Das Programm verwendet die drei Variablen *datei1*, *text_alt* und *text_neu*. Während des Programmlaufs werden die Werte für diese Variablen mit dem Kommando `read` eingelesen. Die Variablen stellen folgende Informationen zur Verfügung:

datei	Den Namen der zu editierenden Datei.
text_alt	Den exakten Wortlaut des zu ändernden Textes.
text_neu	Den neuen Text.

Sobald die Variablen an das Programm übergeben worden sind, leitet das Here-Dokument das globale Ersetzen-Kommando sowie die Kommandos zum Abspeichern und Beenden zum Kommando `ed` um. Probieren Sie das neue Kommando `ch.text` aus. Der folgende Bildschirm zeigt, wie Ihre Eingaben bei den Eingabeaufforderungen der Kommandos aussehen können:

```
$ ch.text<CR>
Type in the filename.
memo<CR>
Type in the exact text to be changed.
Dear John:<CR>
Type in the exact new text to replace the above.
To whom it may concern:<CR>
$ cat memo<CR>
To whom it may concern:
$
```

Beachten Sie, daß Sie das Ergebnis des globalen Ersetzungs- Vorgangs mit dem Kommando `cat` überprüfen können.

Tabelle 9-23 enthält eine Übersicht über Format und Funktion des Kommandos `ch.text`.

Tabelle 9-23: `ch.text`**-Kurzübersicht**

Shell-Programm `ch.text` Text einer Datei ändern	
Kommando	*Argumente*
`ch.text`	(interaktiv)
Funktion:	In einer Datei wird Text durch neuen Text ersetzt.
Hinweis:	Dieses Shell-Programm ist interaktiv. Es fordert Sie zur Eingabe von Argumenten auf.

Eine ausführliche Beschreibung des Editors `ed` finden Sie im Kapitel 6, "Der Zeileneditor `ed`". Der Zeichenstromeditor `sed` kann ebenfalls in einem Shell-Programm verwendet werden.

Rückkehr-Codes

Die meisten Shell-Kommandos geben Rückkehr-Codes aus, die Sie darüber informieren, ob das Kommando ordnungsgemäß ausgeführt worden ist. Der Rückkehr-Code 0 (Null) zeigt normalerweise an, daß die Ausführung des Kommandos ordnungsgemäß verlaufen ist. Jeder andere Wert gibt an, daß Fehler bei der Verarbeitung aufgetreten sind. Der Rückkehr-Code wird nicht automatisch angezeigt, sondern muß explizit mit dem speziellen Parameter `$?` angefordert werden.

Überprüfung des Rückkehr-Codes

Nach der interaktiven Ausführung eines Kommandos können Sie seinen Rückkehr-Code mit dem folgenden Kommando abrufen:

```
echo $?
```

Betrachten Sie dazu folgendes Beispiel:

```
$ cat hi
This is file hi.
$ echo $?
0
$ cat hello
cat: cannot open hello
$ echo $?
2
$
```

Im ersten Fall ist die Datei `hi` in Ihrem Verzeichnis vorhanden, und Sie haben dafür die Leseberechtigung. Das Kommando `cat` wird ordnungsgemäß ausgeführt und führt zur Anzeige des Dateiinhalts auf dem Bildschirm. Das Kommando wird mit dem Rückkehr-Code 0 beendet, den Sie über den Parameter `$?` abrufen können. Im zweiten Fall ist die Datei entweder nicht vorhanden, oder Sie haben dafür keine Leseberechtigung. Das Kommando `cat` informiert Sie in einer Meldung über die Ursache des Fehlers und gibt den Rückkehr-Code 2 aus.

Angabe von Rückkehr-Codes beim Kommando `exit`

Ein Shell-Programm ist normalerweise beendet, wenn das letzte Kommando einer Datei ausgeführt ist. Mit dem Kommando `exit` kann man ein Programm jedoch auch an einer anderen Stelle abbrechen, und, was vielleicht noch wichtiger ist, die Anzeige des Rückkehr-Codes veranlassen. Weitere Informationen über das Kommando `exit` finden Sie im *Referenzhandbuch für Programmierer* unter dem Eintrag `sh`.

Schleifen

In den bisherigen Beispielen des Kapitels wurden die Kommandos in den Shell-Programmen der Reihe nach verarbeitet. Mit `for`- und `while`-Schleifen erreichen Sie, daß ein oder mehrere Kommandos mehrmals ausgeführt werden.

Die `for`-Schleife

In einer `for`-Schleife wird eine Folge von Kommandos für jeden Eintrag einer Liste ausgeführt. Bild 9-2 zeigt das Format einer `for`-Schleife.

Bild 9-2: Format der `for`-Schleife

```
for variable<CR>
      in eine_werteliste<CR>
do<CR>
     kommando 1<CR>
     kommando 2<CR>
          .
          .
          .
     letztes kommando<CR>
done<CR>
```

Bei jedem neuen Schleifendurchlauf wird der nächste Listeneintrag der im `for`-Teil angegebenen Variablen zugeordnet. Auf diese Variable kann in den Kommandos innerhalb des `do`-Teils beliebig Bezug genommen werden.

Die Struktur von Schleifen sollte der Übersichtlichkeit halber auch optisch deutlichgemacht werden. Da die Shell Leerzeilen am Zeilenanfang ignoriert, kann jeder Kommandoabschnitt wie in obigem Bild durch Einrücken sichtbar gemacht werden. Dadurch können Sie außerdem einfacher überprüfen, ob jedem `do` am Schleifen-Ende ein `done` zugeordnet ist.

Der Variablen können Sie einen beliebigen Namen zuordnen. Wenn Sie sie beispielsweise `var` nennen, werden die Werte, die in der Liste nach dem Schlüsselwort `in` stehen, nacheinander `var` zugeordnet. Der Wert wird durch Bezugnahmen auf `$var` in der Kommandoliste verfügbar. Fehlt der `in`-Teil, werden dem Wert `var` alle Argumente zugeordnet, die in der Aufrufzeile des Kommandos angegeben und dem speziellen Parameter `$*` zugeordnet worden sind. Die Kommandoliste zwischen den Schlüsselwörtern `do` und `done` wird für jeden Wert einmal durchlaufen.

Nachdem die Kommandos für den letzen Wert der Liste ausgeführt worden sind, führt das Programm die auf `done` folgende Zeile aus. Ist keine weitere Zeile vorhanden, wird das Programm beendet.

Am einfachsten versteht man eine Shell-Programmstruktur anhand eines Beispiels. Erstellen Sie ein Programm, das Dateien in ein anderes Verzeichnis versetzt. Dafür muß es Kommandos für folgende Funktionen enthalten:

`echo`	Der Benutzer wird zur Eingabe des Pfadnamens des neuen Verzeichnisses aufgefordert.
`read`	Der Pfadname wird der Variablen `path` zugeordnet.
`for` *variable*	Der Variablen wird der Name `file` zugeordnet; auf diese Variable kann in der Kommandofolge mit `$file` Bezug genommen werden.
`in` *werte_liste*	Eine Liste mit Werten wird angegeben. Fehlt der `in`-Teil, entspricht die Werteliste dem Wert von `$*` (alle Argumente der Aufrufzeile).
`do` *kommandofolge*	Eine Kommandofolge wird angegeben. In diesem Programm wird dafür folgende Anweisung verwendet:

```
do
    mv $file $path/$file<CR>
done
```

Der folgende Bildschirm zeigt den Inhalt des Shell-Programms `mv.file`:

```
$ cat mv.file<CR>
echo Please type in the directory path
read path
for file
   in memo1 memo2 memo3
do
   mv $file $path/$file
done
$
```

In diesem Programm sind die Werte für die Variable `file` bereits im Programm enthalten. Sollen bei jedem Programmaufruf andere Dateien benutzt werden, so können Sie die Werte mit Hilfe von Stellungsparametern oder dem Kommando `read` zuordnen. Bei der Verwendung von Stellungsparametern kann das Schlüsselwort `in` entfallen, wie der nächste Bildschirm zeigt:

```
$ cat mv.file<CR>
echo type in the directory path
read path
for file
do
   mv $file $path/$file
done
$
```

Sie können mehrere Dateien gleichzeitig in ein anderes Verzeichnis versetzen, indem Sie eine Liste mit Dateinamen als Kommando-Argument angeben (dies ist sehr einfach mit der bereits beschriebenen Dateinamenerweiterung möglich).

Tabelle 9-24 enthält eine Übersicht über das Shell-Programm `mv.file`.

Tabelle 9-24: Kurzübersicht über das Shell-Programm `mv.file`

Shell-Programm `mv.file` Dateien in ein anderes Verzeichnis versetzen (benutzer-definiert)	
Kommando	*Argumente*
`mv.file`	*dateinamen* (interaktiv)
Funktion:	Dateien werden in ein neues Verzeichnis versetzt.
Hinweise:	Für dieses Programm müssen Dateinamen als Argumente angegeben werden. Das Programm fordert Sie zur Angabe des Zugriffspfads für das neue Verzeichnis auf.

Die `while`-Schleife

Eine andere Schleifenanweisung, die `while`-Schleife, verwendet zwei Kommandogruppen. Die Kommandofolge der zweiten Gruppe (im `do...done`-Teil) wird solange ausgeführt, wie das letzte Kommando der ersten Gruppe (im `while`-Teil) den Status "wahr" ausgibt.

Bild 9-3 zeigt das allgemeine Format der `while`-Schleife.

Bild 9-3: Format der `while`-Schleife

```
while<CR>
      kommando 1<CR>
          .
          .
          .
      letztes kommando<CR>
do<CR>
      kommando 1<CR>
          .
          .
          .
      letztes kommando<CR>
done<CR>
```

Angenommen, ein Programm namens `enter.name` schreibt einige Namen mit Hilfe einer `while`-Schleife in eine Datei. Das Programm enthält folgende Kommandozeilen:

```
$ cat enter.name<CR>
while
   read x
do
   echo $x>>xfile
done
$
```

Nach einigen Ergänzungen sieht das Programm folgendermaßen aus:

```
$ cat enter.name<CR>
echo Please type in each person's name and then a <CR>
echo Please end the list of names with a <^d>
while read x
do
   echo $x>>xfile
done
echo xfile contains the following names:
cat xfile
$
```

Nach der Beendigung der Schleife führt das Programm die Kommandos unterhalb von `done` aus.

Da Sie in den ersten beiden `echo`-Kommandozeilen Sonderzeichen verwendet haben, müssen Sie ihre Sonderbedeutung mit Hochkommata (') oder Anführungszeichen (") aufheben. Der nächste Bildschirm zeigt das Ergebnis des Programms `enter.name`:

```
$  enter.name<CR>
Please type in each person's name and then a <CR>
Please end the list of names with a <^d>
Mary Lou<CR>
Janice<CR>
<^d>
xfile contains the following names:
Mary Lou
Janice
$
```

Nach Beendigung der Schleife zeigt das Programm alle Namen an, die in der Datei `xfile` enthalten sind.

Der Papierkorb der Shell: Die Datei `/dev/null`

Im Dateisystem gibt es eine Datei mit dem Namen `/dev/null`. Sie können die Shell veranlassen, jede unerwünschte Ausgabe in diese Datei zu schreiben.

Mit `/dev/null` können Sie beispielsweise das Ergebnis des Kommandos `who` "wegwerfen". Geben Sie zunächst das Kommando `who` ein. Das System gibt eine Übersicht über die angemeldeten Benutzer aus. Rufen Sie dann erneut das Kommando `who` auf und leiten Sie die Ausgabe in die Datei `/dev/null` um:

```
who > /dev/null<CR>
```

Das System gibt jetzt lediglich eine Eingabeaufforderung aus. Das Ergebnis wird in die Datei `/dev/null` geschrieben und damit weggeworfen.

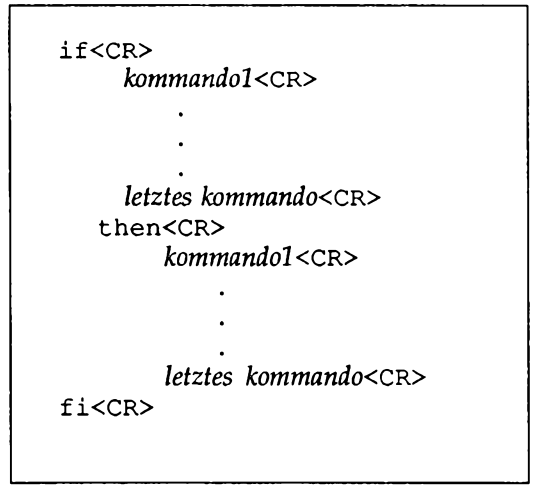
Bedingungsanweisungen

`if...then`

Die `if`-Anweisung weist das Shell-Programm an, die nach `then` angegebene Kommandofolge nur dann auszuführen, wenn das letzte Kommando in der `if`-Kommandoliste den Status "wahr" ausgibt. Eine `if`-Anweisung wird mit dem Schlüsselwort `fi` beendet.

Bild 9-4 zeigt das Grundformat der `if`-Anweisung.

Bild 9-4: Format der `if...then`-Bedingungsanweisung



Die `if...then`- Anweisung wird im folgenden am Shell-Programm `search` erklärt. Im Programm `search` wird mit dem Kommando `grep` ein Wort in einer Datei gesucht. Nachdem `grep` das Wort gefunden hat, gibt das Programm `echos` eine entsprechende Meldung aus. Geben Sie das folgende Programm `search` an Ihrem Terminal ein, und probieren Sie es selbst aus.

```
$ cat search<CR>
echo Type in the word and the file name.
read word file
if grep $word $file
   then echo $word is in $file
fi
$
```

Mit dem Kommando `read` werden zwei Variablen Werte zugeordnet: Die ersten der eingegebenen Zeichen (bis zum Leerzeichen) werden der Variablen `word` zugeordnet, die übrigen Zeichen (einschließlich der Leerzeichen) der Variablen `file`.

Ein Problem dieses Programms ist die unerwünschte Ausgabe des Kommandos `grep`. Um zu verhindern, daß die Ausgabe des Kommandos `grep` vom System angezeigt wird, leiten Sie sie in die Datei `/dev/null` um. Dafür ändern Sie die `if`-Kommandozeile wie folgt ab:

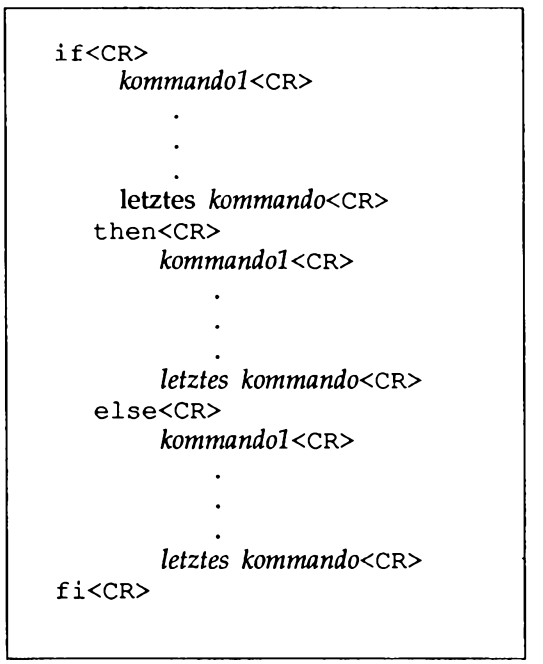
```
if grep $wort $datei > /dev/null<CR>
```

Führen Sie jetzt Ihr Programm `search` aus. Die Ausgabe des Programms sollte jetzt nur aus der Meldung bestehen, die beim Kommando `echo` eingegeben worden ist.

`if...then...else`

In einer `if...then`-Anweisung kann mit `else` eine Ersatz-Kommandofolge für den Fall angegeben werden, daß die `if`-Kommandofolge den Status "falsch" ausgibt. Bild 9-5 zeigt das allgemeine Format dieser Anweisung.

Bild 9-5: Format der `if...then...else`-Bedingungsanweisung



Ihr `search`-Kommando kann nun so verbessert werden, daß auf jeden Fall eine Meldung ausgegeben wird, unabhängig davon, ob ein Wort gefunden wird oder nicht. Der folgende Bildschirm zeigt den Inhalt des verbesserten Programms:

```
$ cat search<CR>
echo Type in the word and the file name.
read word file
if
   grep $word $file >/dev/null
then
   echo $word is in $file
else
   echo $word is NOT in $file
fi
$
```

Tabelle 9-25 enthält eine Übersicht über das verbesserte Programm `search`.

Tabelle 9-25: Kurzübersicht über das Shell-Programm `search`

Shell-Programm `search` Wort in einer Datei suchen und anzeigen, ob es vorhanden ist (benutzer-definiert).	
Kommando	*Argumente*
`search`	interaktiv
Funktion:	Wort in einer Datei suchen und anzeigen, ob es vorhanden ist.
Hinweis:	Das Kommando fordert Sie zur Eingabe der Argumente (des Worts und der Datei) auf.

Überprüfung von Bedingungen mit `test`

Das Kommando `test`, mit dem überprüft werden kann, ob bestimmte Bedingungen erfüllt sind, ist ein nützliches Kommando für Bedingungsanweisungen. Ist eine bestimmte Bedingung erfüllt, wird die Schleife fortgesetzt; andernfalls wird die Schleife beendet und das nächste Kommando ausgeführt. Für das Kommando `test` gibt es u.a. die folgenden Optionen:

`test -r` *datei*`<CR>`	Wahr, wenn die Datei vorhanden ist und gelesen werden kann.
`test -w` *datei*`<CR>`	Wahr, wenn die Datei vorhanden ist und geändert werden kann.
`test -x` *datei*`<CR>`	Wahr, wenn die Datei vorhanden ist und ausführbar ist.
`test -s` *datei*`<CR>`	Wahr, wenn die Datei vorhanden ist und aus mindestens einem Zeichen besteht.
`test` *var1* `-eq` *var2*`<CR>`	Wahr, wenn *var1* gleich *var2 ist.*
`test` *var1* `-ne` *var2*`<CR>`	Wahr, wenn *var1* nicht gleich *var2* ist.

Angenommen, mit einem Shell-Programm sollen alle ausführbaren Dateien aus dem aktuellen Verzeichnis in Ihr Verzeichnis `bin` versetzt werden. Mit dem Kommando `test -x` können Sie die ausführbaren Dateien selektieren. Sehen Sie sich auf folgendem Bildschirm noch einmal das Beispiel für eine `for`-Anweisung an, wie sie im Programm `mv.file` verwendet wird:

```
$ cat mv.file<CR>
echo type in the directory path
read path
for file
do
  mv $file $path/$file
done
$
```

Erstellen Sie ein Programm mit dem Namen `mv.ex`, dessen `do...done`-Schleife die Anweisung `if test -x` zur Selektion der ausführbaren Dateien enthält. Ihr Programm sieht dann folgendermaßen aus:

```
$ cat mv.ex<CR>
echo type in the directory path
read path
for file
  do
    if test -x $file
      then
        mv $file $path/$file
    fi
  done
$
```

Der Verzeichnispfad ist der Pfad vom aktuellen Verzeichnis zum Verzeichnis `bin`. Die Verwendung des Werts der Shell-Variablen `HOME` erspart Ihnen jedoch die erneute Eingabe des Zugriffspfads. `$HOME` definiert den Zugriffspfad zum Home-Verzeichnis. `$HOME/bin` definiert den Zugriffspfad zu Ihrem Verzeichnis `bin`.

Im folgenden Beispiel fordert `mv.ex` Sie nicht zur Eingabe des Verzeichnisnamens auf. Daher entfällt die Variable `path`:

```
$ cat mv.ex<CR>
for file
  do
    if test -x $file
      then
        mv $file $HOME/bin/$file
    fi
  done
$
```

Probieren Sie das Kommando an allen Dateien im aktuellen Verzeichnis aus (geben Sie als Kommando-Argument das Metazeichen `*` an). Die Kommandozeile in folgendem Beispiel führt das Kommando im aktuellen Verzeichnis aus, wechselt in das Verzeichnis `bin` und listet dann die Dateien in diesem

Verzeichnis auf. Alle ausführbaren Dateien sollten jetzt in diesem Verzeichnis enthalten sein.

```
$ mv.ex *<CR>
$ cd; cd bin; ls<CR>
liste_aller_ausführbaren_dateien
$
```

Bild 9-26 faßt Format und Funktion des Shell-Programms `mv.ex` zusammen.

Tabelle 9-26: Kurzübersicht über das Shell-Programm `mv.ex`

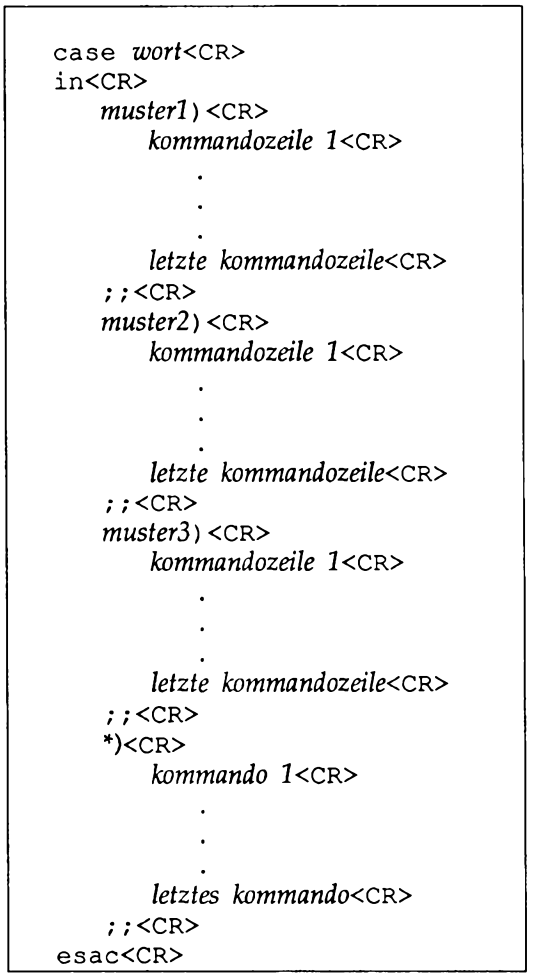
Shell-Programm `mv.ex` Alle ausführbaren Dateien im aktuellen Verzeichnis in das Verzeichnis `bin` versetzen	
Kommando	*Argumente*
`mv.ex`	Auflistung der Dateien im aktuellen Verzeichnis
Funktion:	Alle ausführbaren Dateien im aktuellen Verzeichnis werden in das Verzeichnis `bin` versetzt.
Hinweis:	Die ausführbaren Dateien im Verzeichnis `bin` (bzw. einem anderen der bei der Variablen `PATH` angegebenen Verzeichnisse) können von jedem beliebigen Verzeichnis aus aufgerufen werden.

`case..esac`

Mit der `case...esac`-Anweisung können Sie eine Auswahl unter mehreren Mustern treffen und für dieses Muster dann mehrere Kommandos ausführen. Der Muster-Teil muß mit dem Schlüsselwort `in` beginnen; hinter dem letzten Zeichen jedes Musters muß eine schließende Klammer `())` stehen. Die für jedes Muster angegebene Kommandofolge wird mit zwei Semikola `(;;)` beendet. Die `case`-Anweisung wird mit dem Schlüsselwort `esac` (case umgedreht) abgeschlossen.

Bild 9-6 zeigt das allgemeine Format der `case`-Anweisung:

Bild 9-6: Die `case...esac`-Bedingungsanweisung



Die `case`-Anweisung vergleicht zunächst *wort* nach dem Wort *case* mit dem *muster* im ersten Muster-Teil. Liegt eine Übereinstimmung vor, werden die Kommandozeilen zwischen dem ersten Muster und den zugehörigen Semikola (`;;`) ausgeführt.

Liegt keine Übereinstimmung mit dem ersten Muster vor, fährt das Programm beim zweiten Muster fort. Wird ein Muster gefunden, das mit `wort` übereinstimmt, bricht das Programm den Mustervergleich ab und fährt sofort bei `esac` fort.

Das Muster `*` paßt zu jedem beliebigen *wort*; bei diesem Muster können Sie Kommandos angeben, die ausgeführt werden sollen, falls mit keinem der übrigen Muster eine Übereinstimmung vorhanden ist. Damit zuvor ein Vergleich mit den anderen Mustern durchgeführt wird, muß es in der `case`-Anweisung als letztes Muster angegeben werden. Damit können falsche oder unerwartete Eingaben festgestellt werden.

Die Muster im *muster*-Teil jedes Abschnitts können auch die Metazeichen `*`, `?` und `[]` enthalten (siehe Abschnitt über die Dateinamenerweiterung), was die Flexibilität dieser Anweisung wesentlich erhöht.

Das Programm `set.term` enthält ein gutes Beispiel für eine `case...esac`-Anweisung. Mit diesem Programm wird die Shell-Variable `TERM` dem Typ Ihres Terminals entsprechend eingestellt. Dazu wird folgende Kommandozeile verwendet:

`TERM=`*terminal_name*`<CR>`

Eine Erklärung der verwendeten Kommandos finden Sie in Kapitel 7 "Der Bildschirmeditor `vi`". Im folgenden Beispiel handelt es sich um ein Terminal vom Typ Teletype 4420, 5410 oder 5420.

Das Programm `set.term` überprüft zunächst, ob für `term` der Wert 4420 eingegeben worden ist. Ist dies der Fall, ordnet das Programm der Variablen `TERM` den Wert T4 zu und wird beendet; andernfalls überprüft das Programm die beiden anderen Werte (5410 und 5420). Nachdem das Programm die Kommandos unter dem ersten passenden Muster ausgeführt hat, fährt es beim ersten Kommando unterhalb von `esac` fort.

Das Muster `*` , das für jede beliebige Eingabe steht, wurde als letztes Muster eingegeben. Es informiert Sie in einer Meldung, daß für das angegebene Terminal kein Muster vorhanden ist und ermöglicht die Beendigung der `case`-Anweisung:

```
$ cat set.term<CR>
echo If you have a TTY 4420 type in 4420
echo If you have a TTY 5410 type in 5410
echo If you have a TTY 5420 type in 5420
read term
case $term
        in
             4420)
                  TERM=T4
             ;;
             5410)
                  TERM=T5
             ;;
             5420)
                  TERM=T7
             ;;
             *)
             echo not a correct terminal type
             ;;
esac
export TERM
echo end of program
$
```

Beachten Sie die Verwendung des Kommandos `export` auf dem letzten Bildschirm. Dieses Kommando wird benutzt, um eine Variable innerhalb Ihrer Umgebung für andere Shell-Prozeduren verfügbar zu machen. Was wäre geschehen, wenn Sie * als erstes Muster angegeben hätten? Das Programm `set.term` würde der Variablen `TERM` nie einen Wert zuordnen, da das erste Muster (also jedes beliebige Terminal) in jedem Fall paßt.

Tabelle 9-27 enthält einen Überblick über Format und Funktion des Shell-Programms `set.term`.

Tabelle 9-27: Kurzübersicht über das Shell-Programm `set.term`

Shell-Programm `set.term` Wertzuweisung an TERM (benutzer-definiert).	
Kommando	*Argumente*
`set.term`	interaktiv
Funktion:	Der Shell-Variablen TERM wird ein Wert zugeordnet, der dann exportiert und somit auch anderen Shell-Prozeduren zur Verfügung gestellt werden soll.
Hinweis:	Dieses Kommando fordert Sie zur Eingabe eines bestimmten Terminaltyps auf, der in der `case`-Anweisung als Suchmuster verwendet wird.

Ablaufanweisungen ohne Abbruchbedingung: Die Kommandos `break` und `continue`

Das Kommando `break` bricht die Ausführung der aktuellen Schleife in jedem Fall ab; es ist dazu keine Abbruchbedingung notwendig. Das Programm fährt dann mit den Kommandos fort, die auf die `done-`, `fi-` oder `esac`-Anweisung folgen. Folgen auf diese Anweisung keine Kommandos, wird das Programm beendet.

Im Beispielprogramm `set.term` hätten Sie zum Beenden des Programms anstelle des Kommandos `echo` auch das Kommando `break` verwenden können:

```
$ cat set.term<CR>
echo If you have a TTY 4420 type in 4420
echo If you have a TTY 5410 type in 5410
echo If you have a TTY 5420 type in 5420
read term
case $term
        in
             4420)
             TERM=T4
        ;;
        5410)
             TERM=T5
        ;;
        5420)
             TERM=T7
        ;;
        *)
             break
        ;;
esac
export TERM
echo end of program
$
```

Das Kommando `continue` bewirkt, daß das Programm sofort mit dem nächsten Durchgang der `while`- oder `for`-Schleife fortfährt, ohne die übrigen Kommandos in der Schleife auszuführen.

Fehlersuche in Programmen

Für das Kommando `sh` gibt zwei Optionen, die Ihnen die Fehlersuche in einem Programm erleichtern:

`sh -v` *shell_programm_name*	Ausgabe der Shell-Eingabezeilen, wie sie vom System eingelesen worden sind.
`sh -x` *shell_programm_name*	Ausgabe der Kommandos und ihrer Argumente, wie sie vom System interpretiert worden sind.

Probieren Sie diese beiden Optionen an einem fehlerhaften Shell-Programm aus. Erstellen Sie beispielsweise eine Datei mit dem Namen bug, die die folgende Kommandoliste enthält:

```
$ cat bug<CR>
today=`date`
echo enter person
read person
mail $1
$person
When you log off come into my office please.
$today.
MLH
$
```

Beachten Sie, daß today die Ausgabe des Kommandos date zugeordnet worden ist; für diese Kommandosubstitution muß date in Gegenhochkommata (`) eingeschlossen sein.

Die Nachricht, die dem Benutzer Tom ($1) mit dem Benutzernamen tommy ($1) geschickt wird, sollte folgendermaßen aussehen:

```
$ mail<CR>
From mlh  Mon  Apr 10  11:36  CST  1989
Tom
When you log off come into my office please.
Mon  Apr 10  11:36:32  CST  1989
MLH
?
.
```

Um bug auszuführen, müssen Sie das Programm mit der Taste BREAK oder DELETE beenden.

Versuchen Sie die Fehlersuche zu starten, indem Sie die Datei bug mit dem Kommando sh -v ausführen. Damit werden die Zeilen der Datei angezeigt, wie sie vom System eingelesen worden sind:

```
$ sh -v bug tommy<CR>
today=`date`
echo enter person
enter person
read person
tom
mail $1
```

Beachten Sie, daß die Ausgabe beim Kommando `mail` abbricht; das Kommando `mail` ist also fehlerhaft. Die Eingabe muß mit einem Here-Dokument zu `mail` umgeleitet werden.

Rufen Sie jedoch zunächst das Kommando `sh -x` auf, bevor Sie den Fehler im Programm `bug -x` korrigieren. Damit werden die Kommandos und ihre Argumente angezeigt, wie sie vom System interpretiert worden sind.

```
$ sh -x bug tommy<CR>
+date
today=Mon Apr 10 11:07:23 CST 1989
+ echo enter person
enter person
+ read person
tom
+ mail tom
$
```

Wieder wird die Anzeige des Programms beim Kommando `mail` abgebrochen. Allerdings sind die Wertzuweisungen der Variablen jetzt erfolgt; die Werte werden ausgegeben.

Das korrigierte Programm `bug` sieht folgendermaßen aus:

```
$ cat bug<CR>
today=`date`
echo enter person
read person
mail $1 <<!
$person
When you log off come into my office please.
$today
MLH
!
$
```

Mit dem Kommando `tee` kann eine Pipeline auf Fehler untersucht werden. Dieses Kommando, das einfach seine Standard-Eingabe auf seine Standard-Ausgabe schreibt, speichert gleichzeitig eine Kopie seiner Eingabe in der Datei ab, die als Argument angegeben worden ist.

Das Kommando `tee` hat folgendes allgemeines Format:

kommando1 | `tee` *sicherungsdatei* | *kommando2*<CR

In der Datei `sicherungsdatei` wird die Ausgabe von *kommando1* abgespeichert, so daß sie von Ihnen überprüft werden kann.

Angenommen, Sie möchten die Ausgabe des Kommandos `grep` in der folgenden Kommandozeile überprüfen:

```
who | grep $1 | cut -c1-9<CR>
```

Mit dem Kommando `tee` können Sie die Ausgabe von `grep` in die Datei `check` ohne Auswirkungen auf den Rest der Pipeline kopieren.

```
who | grep $1 | tee check | cut -c1-9<CR>
```

Die Datei `check` auf dem folgenden Bildschirm enthält eine Kopie der Ausgabe des Kommandos `grep`:

```
$ who | grep mlhmo | tee check | cut -c1-9<CR>
mlhmo
$ cat check<CR>
mlhmo    tty61    Apr 10   11:30
$
```

Änderung Ihrer Benutzerumgebung

Unter UNIX gibt es verschiedene Möglichkeiten, um Ihre Benutzerumgebung Ihren Anforderungen anzupassen. Viele Benutzer wollen z.B. die Standardwerte für das Zeichenlösch-Zeichen (^H) und das Zeilenlösch-Zeichen (@) ändern.

Wenn Sie sich anmelden, durchsucht die Shell zunächst eine Datei, die in Ihrem Home-Verzeichnis enthalten ist und den Namen `.profile` hat (die sogenannte Login-Prozedur). Diese Datei enthält Kommandos, die Ihre Shell-Umgebung steuern.

Die Datei `.profile` ist eine Shell-Prozedur, die Sie Ihren Bedürfnissen entsprechend editieren können. Auf einigen Systemen können Sie diese Datei selbst editieren, auf anderen Systemen muß das der Systemverwalter für Sie erledigen. Mit der folgenden Kommandozeile können Sie überprüfen, ob in Ihrem Home-Verzeichnis eine Datei namens `.profile` enthalten ist:

```
ls -al $HOME
```

Wenn Sie selbst die Datei editieren können, sollten Sie bei Ihren ersten Versuchen sehr vorsichtig sein. Kopieren Sie die Datei `.profile` vor der Durchführung der Änderungen in eine Datei mit dem Namen `safe.profile`:

```
cp .profile safe.profile<CR>
```

Sie können in Ihre Login-Prozedur `.profile` wie in jedes andere Shell-Programm Kommandos einfügen. Außerdem können Sie mit dem Kommando `stty` verschiedene Terminaloptionen konfigurieren und einige Shell-Variablen einstellen.

Einfügen von Kommandos in `.profile`

Das folgende Beispiel zeigt Ihnen, wie Sie Kommandos in Ihre Datei `.profile` einfügen können. Rufen Sie die Datei mit einem Editor auf und fügen Sie ganz am Ende der Datei das folgende `echo`-Kommando ein:

```
echo Good Morning! I am ready to work for you.
```

Speichern Sie den Text ab und beenden Sie den Editor.

Wenn die Änderungen, die Sie an Ihrer Login-Prozedur `.profile` vorgenommen haben, bereits während der aktuellen UNIX-Sitzung gültig sein sollen, so müssen Sie Ihre Login-Prozedur `-profile` mit dem Shell-Kommando Punkt (`.`) aufrufen. Die Shell liest nun die Kommandos in Ihrer Login-Prozedur

`.profile` ein und initialisiert die Umgebung entsprechend. Geben Sie probehalber folgendes ein:

```
. .profile<CR>
```

Das System sollte die folgende Meldung ausgeben:

```
Good Morning! I am ready to work for you.
$
```

Einstellung von Terminaloptionen

Mit dem Kommando `stty` können Sie Ihre Shell-Umgebung Ihren Anforderungen entsprechend einrichten. Folgende Optionen können bei den Kommandos `stty` und `echoe` angegeben werden:

`stty -tabs`	Mit dieser Option werden die Tabulatorzeichen unverändert ausgegeben. Die Tabulatorzeichen werden standardmäßig in acht Leerzeichen umgesetzt. Die Anzahl der Leerzeichen kann jedoch für jedes Tabulatorzeichen geändert werden (weitere Informationen finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `stty`(1)).
`stty echoe`	Auf einem Datensichtgerät bewirkt diese Option, daß Zeichen bei Betätigung der Taste `BACKSPACE` vom Bildschirm gelöscht werden.

Wenn Sie mit diesen Optionen des Kommandos `stty` arbeiten möchten, können Sie die entsprechenden Kommandozeilen in Ihre Login-Prozedur `.profile` wie in ein Shell-Programm einfügen. Mit dem Kommando `tail` können Sie die letzten Zeilen einer Datei auf dem Bildschirm abrufen. Der folgende Bildschirm zeigt die letzten drei Zeilen der geänderten Datei `.profile`:

```
$ tail -3 .profile<CR>
echo Good Morning! I am ready to work for you
stty -tabs
stty echoe
$
```

Tabelle 9-28 enthält eine Übersicht über Format und Funktion des Kommandos `tail`.

Tabelle 9-28: `tail`**-Kurzübersicht**

Kommando `tail` Letzten Teil einer Datei ausgeben		
Kommando	*Optionen*	*Argumente*
`tail`	*-n*	*dateiname*
Funktion:	Die letzten Zeilen einer Datei werden angezeigt.	
Optionen:	Mit *-n* geben Sie die Anzahl der auszugebenden Zeilen an (Standard: 10 Zeilen). Statt der Anzahl der Zeilen kann auch die Anzahl der Blöcke (*-n*b) oder Zeichen (*-n*c) angegeben werden.	

Shell-Variablen

In der Login-Prozedur `.profile` können zahlreiche reservierte Shell-Variablen verwendet werden. Mit folgendem Kommando können Sie den aktuellen Wert jeder Shell-Variablen abrufen:

`echo` *$variablen_name*

Im folgenden werden vier der wichtigsten Variablen beschrieben:

`HOME` Dieser Variablen ist der Pfadname Ihres Home-Verzeichnisses zugeordnet. Wechseln Sie mit dem Kommando `cd` *in Ihr Home-Verzeichnis und geben Sie folgendes ein:*

```
pwd<CR>
```

Wie sieht die Ausgabe des Systems aus? Geben Sie nun folgendes ein:

```
echo $HOME<CR>
```

Entspricht die Ausgabe des Systems der bei `pwd`?

`$HOME` ist das Standard-Argument des Kommandos `cd`. Wird kein anderes Verzeichnis angegeben, versetzt `cd` Sie in das Verzeichnis `$HOME`.

`PATH` Dieser Variablen ist der Zugriffspfad zugeordnet, in dem auszuführende Kommandos gesucht werden. Die aktuellen Werte für `PATH` können Sie folgendermaßen abrufen:

```
echo $PATH<CR>
```

Das System gibt daraufhin den aktuellen Wert Ihrer Variablen PATH an.

```
$ echo $PATH<CR>
:/mylogin/bin:/bin:/usr/bin
$
```

Der Doppelpunkt (:) hat die Funktion eines Begrenzers, durch den die verschiedenen Pfadnamen in der bei $PATH angegebenen Zeichenkette voneinander abgegrenzt werden. Der Doppelpunkt (:) allein steht für das aktuelle Verzeichnis.

Beachten Sie, daß das System im letzten Beispiel zunächst im aktuellen Verzeichnis nach Kommandos suchte, dann in den Verzeichnissen `/mylogin/bin`, `/usr/bin`, `/usr/bin` und schließlich in `/usr/lib`.

Wenn Sie mit mehreren Mitarbeitern an einem Projekt arbeiten, können Sie für die Gruppe ein Verzeichnis namens `bin` anlegen; in diesem Verzeichnis können dann spezielle Shell-Programme enthalten sein, die nur von den am Projekt Beteiligten genutzt werden. Der Pfadname kann beispielsweise `/project1/bin` sein. Editieren Sie Ihre Datei `.profile` und fügen Sie `:/project1/bin` am Ende des Zugriffspfads bei der Variablen `$PATH` an:

```
PATH="$PATH:/project1/bin"<CR>
```

`TERM` Mit dieser Variablen teilen Sie der Shell den Typ Ihres Terminals mit. Die Wertzuweisung dieses Kommandos besteht aus den folgenden drei Kommandos, die hintereinander aufgerufen werden müssen:

```
TERM=terminal_name<CR>
export TERM<CR>
tput init
```

Mit den beiden ersten Zeilen teilen Sie dem Rechner den Terminaltyp mit. Die letzte Zeile (mit dem Kommando `tput`) teilt dem Terminal mit, daß der Rechner den bei der Variablen TERM angegebenen Terminaltyp erwartet. Dieses Kommando muß nach der Exportierung der Variablen unbedingt angegeben werden.

Wenn Sie die Variable TERM nicht bei jeder Anmeldung einstellen möchten, so fügen Sie diese drei Kommandozeilen in Ihre Login-Prozedur `.profile` ein. Im Anhang G, "Einstellung des Terminals", finden Sie Informationen über die zulässigen Terminalnamen, die Sie der Variablen TERM zuordnen können. Dieser Anhang enthält auch weitere Informationen über das Kommando `tput`.

Wenn Sie an mehreren Terminals unterschiedlichen Typs arbeiten, sollte in Ihrer Login-Prozedur `.profile` außerdem das Kommando `set.term` enthalten sein.

`PS1` Dieser Variablen ist die primäre Eingabeaufforderung zugeordnet (Standard: Das Dollar-Symbol `$`). Durch die Änderung der Variablen `PS1` in Ihrer Login-Prozedur `.profile` können Sie eine neue Eingabeaufforderung festlegen.

Eine Eingabeaufforderung, die wie das folgende Beispiel aus mehreren Wörtern besteht, muß in Anführungszeichen (") eingeschlossen werden. Ordnen Sie der Variablen in Ihrer Login-Prozedur `.profile` den folgenden Wert zu:

```
PS1="Your command is my wish<CR>"
```

Führen Sie jetzt Ihre Login-Prozedurs `.profile` (mit dem Kommando `.`) aus, und achten Sie auf Ihre neue Eingabeaufforderung.

```
$ . .profile<CR>
Your command is my wish
```

Das Zeichen `$` wird erst dann wieder als Eingabeaufforderung angezeigt, nachdem Sie die Variable `PS1` in Ihrer Login-Prozedur `.profile` entsprechend geändert haben.

Übungen zur Shell-Programmierung

2-1. Erstellen Sie aus der folgenden Kommandozeile ein Shell-Programm mit dem Namen `time`:

```
banner `date | cut -c12-19`<CR>
```

2-2. Schreiben Sie ein Shell-Programm, das nur das Datum in einer Meldung in großer Schrift anzeigt. Achten Sie darauf, daß Sie Ihrem Programm nicht den Namen eines UNIX-Kommandos geben.

2-3. Schreiben Sie ein Shell-Programm, mit dem Sie verschiedenen Benutzern an Ihrem System eine Meldung senden können.

2-4. Leiten Sie das Kommando `date` ohne Anzeige der Uhrzeit in eine Datei um.

2-5. Schreiben Sie den Satz 'dear collegue' mit dem Kommando `echo` in die obige Datei, ohne das Datum zu löschen.

2-6. Schreiben Sie anhand der oben aufgeführten Übungen ein Shell-Programm, das an die gleichen Benutzer wie in Übung 2-3 ein Memo sendet. Schreiben Sie folgendes in Ihr Memo:

10 Die Programmiersprache awk

Einführung 10-1

Grundlagen der Programmiersprache awk 10-2
Programmaufbau 10-2
Aufrufen von awk 10-3
Felder 10-4
Bildschirmausgabe 10-5
Formatierte Ausgabe 10-7
Einfache Muster 10-8
Einfache Aktionen 10-9
- Vordefinierte Variablen 10-9
- Benutzer-definierte Variablen 10-10
- Funktionen 10-10

Einige nützliche Einzeiler 10-11
Fehlermeldungen 10-12

Muster 10-13
BEGIN und END 10-13
Vergleichsausdrücke 10-14
Reguläre Ausdrücke 10-15
Muster-Kombinationen 10-20
Musterbereiche 10-20

Aktionen 10-22
Vordefinierte Variablen 10-22
Arithmetische Berechnungen 10-23

Zeichenketten und Zeichenketten-Funktionen 10-26
Vordefinierte Variablen 10-32
Zahl oder Zeichenkette? 10-33
Ablaufsteuerung 10-34
Felder (Arrays) 10-37
Benutzer-definierte Funktionen 10-40
Zum Format von awk-Programmen 10-41

Die Ausgabe 10-43
Die Funktion `print` 10-43
Trennzeichen für die Ausgabe 10-43
Die Funktion `printf` 10-44
Ausgabe in Dateien 10-46
Ausgabe an Pipes 10-47

Die Eingabe 10-49
Dateien und Pipes 10-49
Trennzeichen für die Eingabe 10-49
Mehrzeilige Sätze 10-50
Die Funktion `getline` 10-51
Die Argumente in der Aufrufzeile 10-54

`awk` mit anderen Kommandos und der Shell 10-55
Die Funktion `system` 10-55
awk und die Shell 10-55

Anwendungsbeispiele 10-58
Generierung von Reports 10-58
Weitere Beispiele 10-60
- Worthäufigkeiten 10-60
- Aufsummierung 10-61
- Auswahl von Zufallselementen 10-61
- Shell-ähnliche Funktionen 10-62
- Generierung von Formbriefen 10-62

`awk` **- Kurzübersicht** 10-64
Die Kommandozeile 10-64
Muster 10-64
Steuerungsanweisungen 10-64
Ein/-Ausgabe 10-65
Funktionen 10-65
Zeichenketten-Funktionen 10-66
Arithmetische Funktionen 10-67
Operatoren (in aufsteigender Priorität) 10-67
Reguläre Ausdrücke (in aufsteigender Priorität) 10-68
Vordefinierte Variablen 10-68
Einschränkungen 10-69
Initialisierung, Vergleich und Typumwandlung 10-69

Einführung

In diesem Kapitel wird eine Programmiersprache beschrieben, mit der Sie Datenmanipulationen und Informationsabfragen auf elegante und einfache Weise durchführen können. Mit awk können Sie Ergebnisse, die in einer Datei gespeichert sind, in Tabellenform ausgeben, diese Ergebnisse in verschiedenen Reports zusammenfassen, Formulare erstellen, eine Datei nach einer Zeichenkette durchsuchen oder eine Datei, die mit einem Anwendungsprogramm erstellt worden ist, zur Weiterverarbeitung mit einem anderen Anwendungsprogramm aufbereiten.

Der Name awk setzt sich aus den Anfangsbuchstaben der Namen seiner Entwickler zusammen. Der Name awk bezieht sich sowohl auf die Programmiersprache selbst als auch auf das UNIX-Kommando zum Aufrufen eines awk-Programms.

Hinweis: Dieses Kapitel erklärt die neue Version der Programmiersprache awk, wie sie im *Referenzhandbuch für Benutzer* unter dem Eintrag nawk(1) beschrieben wird. Die ältere Version, die in demselben Handbuch unter awk(1) beschrieben wird, wird in einer späteren UNIX-Version durch nawk abgelöst.

awk ist eine einfach zu erlernende Programmiersprache, mit der Sie viele Funktionen durchführen können, für die Sie in anderen Sprachen spezielle Programme schreiben müssen. Daher bestehen sehr viele awk- Programme aus nur ein oder zwei Zeilen. Da die awk- Programme in der Regel kürzer sind als die entsprechenden Programme, die in anderen Programmiersprachen realisiert worden sind, und sie außerdem interpretiert, und nicht übersetzt werden, eignet sich awk auch hervorragend zum Erstellen von Software-Prototypen ("Prototyping").

Der erste Teil dieses Kapitels, der Sie in die Grundlagen von awk einführt, soll Ihnen das Schreiben und Durchführen Ihrer ersten awk- Programme erleichtern. Anschließend werden sämtliche Elemente der Programmiersprache besprochen. Speziell für erfahrene awk- Benutzer wurde am Ende dieses Kapitels eine Kurzübersicht über die Programmiersprache angefügt.

Dieses Kapitel setzt Kenntnisse über das UNIX-System und die Shell-Programmierung voraus. Erfahrungen mit anderen Programmiersprachen sind nicht erforderlich; allerdings sind Grundkenntnisse über die Programmiersprache C wegen ihrer Ähnlichkeiten mit awk von Vorteil.

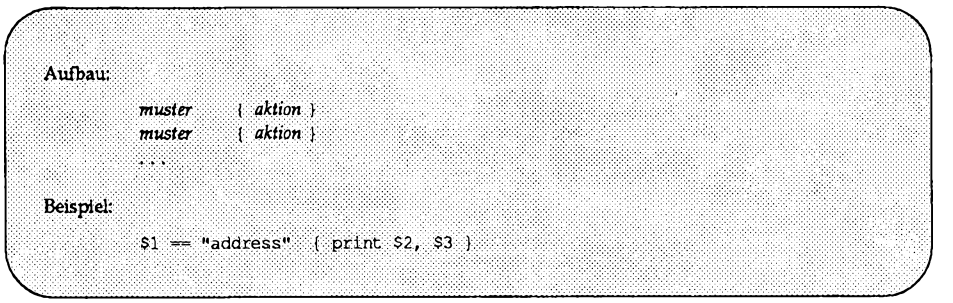
Grundlagen der Programmiersprache awk

Dieser Abschnitt enthält sämtliche Informationen, die Sie zum Erstellen und Ausführen Ihrer ersten Programme benötigen. Die einzelnen Themen werden in späteren Abschnitten ausführlicher beschrieben.

Programmaufbau

Die grundlegende Funktion von awk besteht darin, eine Reihe von Eingabezeilen hintereinander nach Zeilen zu durchsuchen, in denen bestimmte, von Ihnen angegebene Muster auftreten bzw. Bedingungen zutreffen. Sie können für jedes Muster eine Aktion angeben; diese Aktion wird für jede Zeile durchgeführt, die zu dem Muster paßt. Ein awk- Programm besteht also aus einer Aneinanderreihung von Anweisungen in Form von Muster-Aktions-Paaren (siehe Bild 10-1).

Bild 10-1: awk-Programm - Aufbau und Beispiel



Das oben gezeigte Beispiel ist ein typisches awk- Programm, das aus einem einzigen Muster-Aktions-Paar besteht. Das Programm gibt in diesem Fall das zweite und dritte Feld aller Eingabezeilen aus, in deren erstem Feld der Eintrag address enthalten ist. Der grundsätzliche Ablauf eines awk- Programms sieht folgendermaßen aus: Jede Eingabezeile wird mit jedem angegebenen Muster verglichen. Paßt ein Muster zu einer Eingabezeile, so wird die zugehörige (ein- oder mehrteilige) Aktion durchgeführt. Dann wird die nächste Zeile abgearbeitet und ein weiterer Mustervergleich durchgeführt. Auf diese Weise wird fortgefahren, bis alle Eingabezeilen abgearbeitet worden sind.

In einem Muster-Aktions-Paar kann entweder das Muster oder die Aktion weggelassen werden. Wird für ein Muster wie in

```
$1 == "name"
```

keine Aktion angegeben, so wird die gefundene Zeile ausgegeben. Wird dagegen für eine Aktion wie in

```
{ print $1, $2 }
```

kein Muster angegeben, so wird die Aktion für jede Eingabezeile durchgeführt. Da sowohl die Muster als auch die Aktionen optionale Elemente sind, werden die Aktionen zur Abgrenzung gegenüber den Mustern in geschweifte Klammern eingeschlossen.

Aufrufen von awk

Zum Aufrufen eines `awk`- Programms gibt es zwei Möglichkeiten. Zum einen können Sie die Kommandozeile

`awk` *'muster-aktions-paare' optionale liste von eingabedateien*

eingeben, um die Muster-Aktions-Paare für die angegebenen Eingabedateien durchzuführen. Eine solche Kommandozeile sieht beispielsweise folgendermaßen aus:

```
awk '{ print $1, $2 }' datei1 datei2
```

Beachten Sie, daß die Muster-Aktions- Paare in Hochkommata (') eingeschlossen sind, um Zeichen wie das Dollar-Zeichen ($) vor einer Auswertung durch die Shell zu schützen; außerdem macht dies zwei- oder mehrzeilige Programme möglich.

Werden in der Kommandozeile keine Dateinamen angegeben, liest `awk` seine Eingabedaten von der Standard-Eingabe. Sie können aber auch explizit angeben, daß `awk` die Standard-Eingabe lesen soll; dafür geben Sie einen Bindestrich (-) als eine der Eingabedateien ein. So bewirkt beispielsweise das folgende Kommando, daß als Eingabe zunächst die Datei `file1` und dann die Standard-Eingabe benutzt werden soll:

```
awk '{ print $3, $4 }' file1 -
```

Die oben beschriebene Konstellation ist vollkommen praktikabel, wenn das `awk`-Programm nur aus wenigen Zeilen besteht. Bei langen Programmen dagegen sollten die Anweisungen in der Regel in eine separate Datei eingebracht und mit der Option `-f` aufgerufen werden:

`awk -f` *programm_datei optionale liste von eingabedateien*

So bewirkt beispielsweise die folgende Kommandozeile, daß das Programm `myprogram` mit `file1` als Eingabedatei aufgerufen wird:

```
awk  -f myprogram file1
```

Felder

Im Normalfall arbeitet `awk` die Zeilen bzw. Sätze seiner Eingabe hintereinander ab; als Satz wird eine Folge von Zeichen bezeichnet, die durch ein Zeilenendezeichen (newline) abgeschlossen wird. Dann teilt `awk` jeden Satz in Felder auf; als Feld wird standardmäßig eine Folge von Zeichen bezeichnet, bei denen es sich weder um Leer- noch um Tabulatorzeichen handelt.

In diesem Kapitel wird als Eingabe für die `awk`- Programme häufig die Datei `countries` benutzt, in der Informationen über die zehn größten Länder der Welt enthalten sind (siehe Bild 10-2). Jeder Satz enthält den Namen eines Landes, seine Größe in Tausend Quadratmeilen, seine Bevölkerungszahl in Millionen sowie den Kontinent, auf dem es sich befindet (die Daten stammen aus dem Jahr 1978; die UdSSR (U.S.S.R.) wurde absichtlich unter Asien eingeordnet). Der Leerraum zwischen den Feldern ist in der ursprünglichen Eingabe ein Tabulatorzeichen; `North` und `South` werden durch ein Leerzeichen von `America` getrennt.

Bild 10-2: Die Beispieldatei `countries`

```
USSR            8650        262         Asia
Canada          3852        24          North America
China           3692        866         Asia
USA             3615        219         North America
Brazil          3286        116         South America
Australia       2968        14          Australia
India           1269        637         Asia
Argentina       1072        26          South America
Sudan           968         19          Africa
Algeria         920         18          Africa
```

Anhand dieser Datei mit ihrer Kombination aus Wörtern und Zahlen, die durch Leer- und Tabulatorzeichen in Felder aufgeteilt wird, lassen sich die Vorteile von `awk` bei der Verarbeitung von Daten sehr gut darstellen.

Die Anzahl von Feldern in einem Satz hängt von der Anzahl der Feldtrennzeichen ab. Die Felder werden normalerweise durch beliebig lange Folgen von Leer- und/oder Tabulatorzeichen voneinander getrennt; der erste Satz der Datei `countries` besteht also aus vier Feldern, der zweite aus fünf usw. Als Feldtrennzeichen kann auch nur das Tabulatorzeichen verwendet werden, so daß jede Zeile der Bedeutung der Daten entsprechend vier Felder enthalten würde; darauf kommen wir später zurück. Vorerst werden die Standard-Feldtrennzeichen (eine Folge von Leer- und/oder Tabulatorzeichen) benutzt. Dem ersten Feld innerhalb einer Zeile ist die Variable `$1` zugeordnet, dem zweiten die Variable `$2` usw. Dem Satz als Ganzes ist die Variable `$0` zugeordnet.

Bildschirmausgabe

Fehlt in einem Muster-Aktions-Paar das Muster, so wird die Aktion für alle Eingabezeilen durchgeführt. Die einfachste Aktion besteht aus der Ausgabe jeder Zeile; hierzu geben Sie ein `awk`-Programm ein, das aus einer einzigen `print`-Anweisung besteht:

```
{ print }
```

Mit der Kommandozeile

```
awk '{ print }' countries
```

wird also jede Zeile der Datei `countries` ausgegeben, indem die Datei auf die Standard-Ausgabe kopiert wird. Mit der Funktion `print` können auch Teile eines Satzes ausgegeben werden. So gibt beispielsweise das Programm

```
{ print $1, $3 }
```

das erste und dritte Feld jedes Satzes aus. Die Anweisung

```
awk '{ print $1, $3 }' countries
```

erzeugt also die folgende Ausgabe (achten Sie auf die Reihenfolge):

```
USSR 262
Canada 24
China 866
USA 219
Brazil 116
Australia 14
India 637
Argentina 26
Sudan 19
Algeria 18
```

Bei der Ausgabe werden diejenigen Elemente, die in der `print`- Answeisung durch ein Komma voneinander getrennt sind, durch das Ausgabe- Feldtrennzeichen (Standard: ein einzelnes Leerzeichen) voneinander getrennt. Jede ausgegebene Zeile wird durch das Ausgabe-Satztrennzeichen (Standard: das Zeilenendezeichen) abgeschlossen.

Hinweis: In den nachfolgenden Abschnitten dieses Kapitels werden nur die `awk`-Programme selbst gezeigt, ohne die Kommandozeile, durch die sie aufgerufen werden. Zum Aufrufen des vollständigen Programms schließen Sie es entweder als erstes Argument des Kommandos `awk` in Hochkommata ein, oder Sie bringen es in eine Datei ein und rufen `awk` mit der Option `-f` auf (siehe Abschnitt "Aufrufen von awk" an einer früheren Stelle dieses Kapitels). Fehlt beispielsweise die Angabe einer Eingabedatei, wird auf die Datei `countries` bezuggenommen.

Formatierte Ausgabe

Eine etwas weitergehend formatierte Ausgabe können Sie bei awk mit der Funktion printf erzeugen, die wie die gleichnamige C-Funktion funktioniert:

printf *format*, *ausdr*$_1$, *ausdr*$_2$, . . . , *ausdr*$_n$

Hiermit wird *ausdr*$_1$, *ausdr*$_2$ usw. dem Ausgabeformat entsprechend formatiert, das über die Zeichenkette *format* angegeben wird. So gibt beispielsweise das awk- Programm

```
{ printf "%10s %6d\n", $1, $3 }
```

das erste Feld ($1) als (rechtsbündig justierte) Folge von zehn Zeichen aus, gefolgt von einem Leerzeichen, dem dritten Feld ($3) als Dezimalzahl in einem sechsstelligen Feld sowie einem Zeilenendezeichen (\n). Wenn Sie als Eingabedatei dieses Programms dann die Datei countries angeben, erstellt das Programm eine formatierte Tabelle:

```
      USSR    262
    Canada     24
     China    866
       USA    219
    Brazil    116
 Australia     14
     India    637
 Argentina     26
     Sudan     19
   Algeria     18
```

Bei der Funktion printf werden die Ausgabe-Satztrennzeichen bzw. Zeilenendezeichen nicht automatisch erzeugt; daher müssen sie explizit in der Formatangabe (Sonderzeichen \n) angegeben werden. Im Abschnitt "Die Funktion printf" in diesem Kapitel finden Sie ausführliche Informationen über die Funktion printf.

Einfache Muster

Die Angabe der auszugebenden Sätze und andere Datenmanipulationen sind bereits über einfache Muster möglich. awk kennt drei Muster-Typen. Zum einen können Sie mit sogenannten Vergleichs- oder relationalen Ausdrücken Vergleiche durchführen. So überprüft z.B. der Operator == die Übereinstimmung der Operanden. Damit alle Zeilen ausgegeben werden, deren viertes Feld zur Zeichenkette Asia paßt, genügt das folgende, aus einem einzigen Muster bestehende Programm:

```
$4 == "Asia"
```

Mit der Datei countries als Eingabedatei wird durch das Programm die folgende Ausgabe erzeugt:

```
USSR   8650   262    Asia
China  3692   866    Asia
India  1269   637    Asia
```

awk unterstützt die Vergleichsoperatoren >, >=, <, <=, == (gleich) und != (ungleich). Diese Vergleichsoperatoren können zum Vergleich sowohl von Zahlen als auch von Zeichenketten verwendet werden. Angenommen, die Ausgabe soll sich auf die Länder mit einer Bevölkerungszahl größer 100 Millionen beschränken. Dafür genügt das folgende Programm:

```
$3 > 100
```

Dieses Programm gibt alle Zeilen aus, deren drittes Feld den Wert 100 übersteigt (zur Erinnerung: das dritte Feld in der Datei countries enthält die Bevölkerungszahl in Millionen).

Beim zweiten Muster-Typ handelt es sich um sogenannte reguläre Ausdrücke, mit denen die Auswahl von Sätzen mit Hilfe von Mustervergleichen vorgenommen werden kann. In seiner einfachsten Form ist ein regulärer Ausdruck eine Folge von Zeichen, die in Schrägstriche (/) eingeschlossen ist:

```
/US/
```

Dieses Programm gibt jede Zeile aus, in denen die Zeichenkette US enthalten ist; mit der Datei countries als Eingabe erzeugt es folgende Ausgabe:

```
USSR   8650   262    Asia
USA    3615   219    North America
```

Auf die regulären Ausdrücke wird an einer späteren Stelle dieses Kapitels ausführlich eingegangen.

Beim dritten Muster-Typ handelt es sich um die beiden speziellen Muster BEGIN und END. Die bei BEGIN eingegebene Aktion wird vor dem Einlesen des ersten Eingabesatzes abgearbeitet, die Aktion bei END nach dem Einlesen des letzten Eingabesatzes. Mit dem folgenden Programm wird anhand des Musters BEGIN vorab ein Titel ausgegeben:

```
BEGIN   { print "Countries of Asia:" }
/Asia/  { print "      ", $1 }
```

Das Programm erzeugt die folgende Ausgabe:

```
Countries of Asia:
      USSR
      China
      India
```

Einfache Aktionen

Die einfachste Aktion, die durch ein awk- Programm durchgeführt werden kann, haben Sie bereits kennengelernt: Die Ausgabe jeder Eingabezeile. Im folgenden werden Sie in die Programmierung weiterer einfacher Aktionen mit Hilfe von vordefinierten und benutzer-definierten Variablen eingeführt.

Vordefinierte Variablen

Außer dem Einlesen der Eingabezeilen und ihrer Aufteilung in Felder hat awk die Funktion, die Anzahl der eingelesenen Sätze sowie die Anzahl der Felder im aktuellen Satz zu ermitteln. Die so ermittelten Werte können Sie in Ihren awk-Programmen auswerten. Die Variable NR steht für die laufende Nummer des aktuellen Satzes in der Eingabe, NF für die Anzahl der Felder im Satz. So gibt das Programm

```
{ print NR, NF }
```

die laufende Nummer jedes Satzes sowie seine Größe (Anzahl der Felder) aus, während

```
{ print NR, $0 }
```

jeden Satz mit vorangestellter Satz-Nummer ausgibt.

Benutzer-definierte Variablen

Außer vordefinierten Variablen wie NF und NR können Sie unter awk Ihre eigenen, "benutzer-definierten" Variablen definieren, die Sie zur Speicherung von Daten, Durchführung von Berechnungen u.ä. benutzen können. Angenommen, aus dem Inhalt der Datei countries soll die gesamte Bevölkerungszahl sowie der Durchschnitt aus den Bevölkerungszahlen ermittelt werden:

```
 { sum = sum + $3 }
END { print "Total population is", sum, "million"
      print "Average population of" ,NR, "countries is",
        sum/NR }
```

Hinweis: awk initialisiert sum zu Beginn mit dem numerischen Wert Null.

Bei der ersten Aktion werden die Werte im dritten Feld (gesamte Bevölkerungszahl) aufsummiert; bei der zweiten Aktion, die nach dem Einlesen der letzten Eingabezeile durchgeführt wird, werden Summe und Mittelwert ausgegeben:

```
Total population is 2201 million
Average population of 10 countries is 220.1
```

Funktionen

Mit den vordefinierten awk- Funktionen können Sie sowohl gängige arithmetische Berechnungen durchführen als auch Zeichenketten (alphanumerische Werte) verarbeiten. So gibt es zur Bildung der Quadratwurzel eine spezielle arithmetische Funktion; mit einer der Zeichenketten- Funktionen wird eine Zeichenkette durch eine andere ersetzt. Unter awk können Sie auch Ihre eigenen Funktionen definieren. Nähere Informationen zu Funktionen finden Sie im Abschnitt "Aktionen" in diesem Kapitel.

Einige nützliche Einzeiler

Obwohl awk sich auch zur Erstellung von umfangreichen, komplexeren Programmen eignet, bleiben viele Programme im Rahmen der bisher vorgestellten Programme. Im folgenden lernen Sie weitere kurze Programme kennen, die Sie häufiger auch als Grundlage für Ihre eigenen Programme benutzen werden. Die Programme werden hier nicht erklärt; neue Strukturen werden gegebenenfalls später in diesem Kapitel beschrieben.

Von jeder Eingabezeile wird das letzte Feld ausgegeben:

```
{ print $NF }
```

Die zehnte Eingabezeile wird ausgegeben:

```
NR == 10
```

Die letzte Eingabezeile wird ausgegeben:

```
      { line = $0}
END   { print line }
```

Diejenigen Eingabezeilen werden ausgegeben, die mehr oder weniger als vier Felder enthalten:

```
NF != 4    { print $0, "does not have 4 fields" }
```

Die Eingabezeilen mit mehr als vier Feldern werden ausgegeben:

```
NF > 4
```

Diejenigen Eingabezeilen werden ausgegeben, in denen die Nummer des letzten Felds höher ist als 4:

```
$NF > 4
```

Die Gesamtzahl der Eingabezeilen wird ausgegeben:

```
END   { print NR }
```

Die Gesamtzahl der Felder wird ausgegeben:

```
      { nf = nf + NF }
END   { print nf }
```

Die Gesamtzahl der Eingabezeichen wird ausgegeben:

```
        { nc = nc + length($0) }
END     { print nc + NR }
```

(durch die vordefinierte Variable NR wird zusätzlich die Gesamtzahl der Zeilenendezeichen ausgegeben).

Die Gesamtzahl der Zeilen wird ausgegeben, in denen die Zeichenkette Asia enthalten ist:

```
/Asia/      { nlines++ }
END     { print nlines }
```

(nlines++ ist identisch mit nlines = nlines + 1).

Fehlermeldungen

Bei einem Fehler in einem awk -Programm wird normalerweise eine Fehlermeldung ausgegeben. So führt das Programm

```
$3 < 200 { print ( $1 }
```

zu den folgenden Fehlermeldungen:

```
awk: syntax error at source line 1
 context is
        $3 < 200 { print ( >>> $1 } <<<
awk: illegal statement at source line 1
        1 extra (
```

Einige Fehler (z.B. die Division durch Null) werden erst während des Programmlaufs entdeckt. awk bricht dann den Programmlauf ab und gibt die laufende Nummer des betreffenden Eingabesatzes (NR) sowie die Zeilennummer aus.

Muster

In einem Muster-Aktions-Paar ist das Muster ein Ausdruck zur Auswahl der Sätze, für die die zugehörige Aktion durchgeführt werden soll. In diesem Abschnitt geht es um die Ausdrücke, die als Muster verwendet werden können.

BEGIN und END

BEGIN und END sind zwei spezielle Muster, über die Sie die am Anfang bzw. am Ende der Programmausführung auszuführenden Aufgaben festlegen können. Die zu BEGIN gehörige Aktion wird vor dem Einlesen des ersten Eingabesatzes durchgeführt; die Anweisungen im Aktions-Teil von BEGIN werden also unmittelbar nach dem Aufrufen des Programms durchgeführt, noch bevor awk seinen ersten Eingabesatz einliest. Das Muster END paßt zum Ende der Eingabe; die zugehörige Aktion wird nach der Verarbeitung des letzten Eingabesatzes durchgeführt.

Im folgenden awk- Programm werden im BEGIN -Teil das Feldtrennzeichen auf das Tabulatorzeichen (\t) gesetzt und Spaltentitel auf die Ausgabe geschrieben. Das Feldtrennzeichen ist in der vordefinierten Variablen FS abgelegt. Obwohl FS jederzeit neu gesetzt werden kann, wird dies im Normalfall im BEGIN- Teil erledigt, also noch vor dem Einlesen der ersten Eingabezeile. Die zweite printf -Anweisung im Programm, die für jede Eingabezeile ausgeführt wird, formatiert die Ausgabe zu einer Tabelle, deren Einträge sauber unter den Spalten-Titeln angeordnet sind. Die Aktion END gibt die Gesamtsummen aus (beachten Sie, daß eine überlange Zeile hinter einem Komma fortgesetzt werden kann).

```
BEGIN { FS = "\t"
        printf "%10s %6s %5s   %s\n",
               "COUNTRY", "AREA", "POP","CONTINENT" }
      { printf "%10s %6d %5d   %s\n", $1, $2, $3, $4
        area = area + $2; pop = pop + $3 }
END   { printf "\n%10s %6d %5d\n", "TOTAL", area, pop }
```

Mit der Eingabedatei countries erzeugt dieses Programm die folgende Ausgabe:

```
COUNTRY   AREA   POP   CONTINENT
   USSR   8650   262   Asia
 Canada   3852    24   North America
  China   3692   866   Asia
    USA   3615   219   North America
 Brazil   3286   116   South America
Australia 2968    14   Australia
  India   1269   637   Asia
Argentina 1072    26   South America
  Sudan    968    19   Africa
Algeria    920    18   Africa
  TOTAL  30292  2201
```

Vergleichsausdrücke

Ein `awk`- Muster kann ein beliebiger Ausdruck sein, mit dem Zeichenketten oder Zahlen miteinander verglichen werden. `awk` kennt sechs Vergleichsoperatoren sowie die zwei Mustervergleichsoperatoren Tilde (~) und Ausrufezeichen-Tilde (!~); diese Operatoren werden im nächsten Abschnitt beschrieben, in dem es um die Durchführung von Vergleichen geht. Tabelle 10-1 enthält einen Überblick über diese Operatoren und ihre Funktion.

Tabelle 10-1: `awk`-Vergleichsoperatoren

Operator	Funktion
<	Kleiner als
<=	Kleiner gleich
==	Gleich
!=	Ungleich
>=	Größer gleich
>	Größer als
~	Übereinstimmung
!~	Keine Übereinstimmung

Bei einem Vergleich von zwei numerischen Operanden wird ein numerischer Vergleich durchgeführt; andernfalls werden die Operanden als Zeichenketten behandelt und ein alphanumerischer Vergleich durchgeführt (bei beiden Operanden kann es sich entweder um eine Zahl oder eine Zeichenkette handeln; der Datentyp wird von `awk` in der Regel automatisch ermittelt. Nähere Informationen zu diesem Thema finden Sie im Abschnitt "Zahl oder Zeichenkette?"). Das Muster `$3>100` bewirkt beispielsweise die Auswahl derjenigen Zeilen, deren drittes Feld einen Wert größer 100 enthält; das Programm

```
$1 >= "S"
```

dagegen wählt diejenigen Zeilen aus, die mit einem Buchstaben im Bereich S bis Z beginnen:

```
USSR      8650  262   Asia
USA       3615  219   North America
Sudan     968   19    Africa
```

Beim Fehlen von sonstigen Informationen behandelt `awk` ein Feld als Zeichenkette; das Programm

```
$1 == $4
```

bewirkt also einen alphanumerischen Vergleich zwischen dem ersten und vierten Feld. Bei der Eingabedatei `countries` trifft diese Bedingung demnach nur auf eine einzige Zeile zu:

```
Australia   2968   14   Australia
```

Bei zwei eindeutig numerischen Feldern wird ein numerischer Vergleich durchgeführt.

Reguläre Ausdrücke

Die Möglichkeiten von `awk` zum Suchen von Zeichenketten beschränken sich nicht auf die Vergleiche, die im letzten Abschnitt vorgestellt wurden. Komplexere Suchvorgänge können mit leistungsfähigen Mustern, sogenannten regulären Ausdrücken, durchgeführt werden. Die regulären Ausdrücke von `awk` haben dasselbe Format wie diejenigen von `egrep`(1) und `lex`(1). Der einfachste reguläre Ausdruck ist eine in Schrägstriche eingeschlossene Zeichenkette wie

z.B.

```
/Asia/
```

Mit diesem Programm werden sämtliche Eingabesätze ausgegeben, die die Teilzeichenkette `Asia` enthalten (hierzu gehören auch die Datensätze, die `Asia` als Teil einer größeren Zeichenkette enthalten wie z.B. `Asian` oder `Pan-Asiatic`). Wenn *ra* für einen regulären Ausdruck steht, so paßt das Muster

/ra/

zu jeder Zeile, die eine der im regulären Ausdruck *ra* angegebenen Teilzeichenketten enthält.

Den Mustervergleich können Sie auch mit den Mustervergleichsoperatoren ~ (Übereinstimmung) und `!~` (Keine Übereinstimmung) auf ein bestimmtes Feld beschränken. So gibt das Programm

```
$4 ~ /Asia/ { print $1 }
```

das erste Feld aller Zeilen aus, deren viertes Feld zu `Asia` paßt; das Programm

```
$4 !~ /Asia/ { print $1 }
```

dagegen gibt das erste Feld aller Zeilen aus, deren viertes Feld NICHT zu `Asia` paßt.

Reguläre Ausdrücke können die Metazeichen

```
\ ^ $ . [] * + ? () |
```

enthalten, die dieselbe Sonderbedeutung wie die entsprechenden Shell-Metazeichen haben. So passen die Metazeichen Zirkumflex (`^`) und Dollar-Symbol (`$`) zum Anfang bzw. Ende einer Zeichenkette, während der Punkt (`.`) zu einem beliebigen Einzelzeichen paßt. Der reguläre Ausdruck

```
/^.$/
```

paßt somit zu sämtlichen Sätzen, die nur ein einziges Zeichen enthalten.

Eine in eckige Klammern eingeschlossene Zeichengruppe paßt zu jedem der eingeschlossenen Zeichen; so paßt `/ [ABC] /` zu allen Sätzen, die eines der Zeichen `A, B` oder `C` enthalten. Buchstaben- oder Zahlenbereiche können innerhalb von eckigen Klammern verkürzt dargestellt werden: `/ [a-zA-Z] /` paßt zu einem beliebigen Einzelbuchstaben.

Handelt es sich beim ersten Zeichen hinter der eckigen Klammer ([ ) um einen Zirkumflex (^), so wird das "Komplement" zu den nachfolgenden Zeichen gesucht; d.h., es werden nur diejenigen Zeichen gesucht, die NICHT aufgeführt sind: / [^a-zA-Z] / paßt zu jedem Zeichen, bei dem es sich nicht um einen Buchstaben handelt. Das Zeichen + steht für "Ein- oder mehrmaliges Auftreten." So gibt das Programm

```
$2 !~ /^[0-9]+$/
```

sämtliche Sätze aus, deren zweites Feld keine Zeichenkette mit ein oder mehreren Ziffern enthält (^ steht für den Anfang der Zeichenkette, [0- 9]+ für ein oder mehrere Ziffern und $ für das Ende der Zeichenkette). Programme dieser Art werden häufig zur Datenauswertung benutzt.

In regulären Ausdrücken kann außerdem die Klammerung ( ) zur Gruppierung von Elementen sowie das logische ODER (|) vorkommen. Das Programm

```
/(apple|cherry) (pie|tart)/
```

sucht nach allen Zeilen, in denen eine der vier Teilzeichenketten `apple pie`, `apple tart`, `cherry pie` oder `cherry tart` enthalten ist.

Die Sonderbedeutung eines Metazeichens können Sie durch einen vorangestellten Gegenschrägstrich (\ ) aufheben. So gibt das Programm

```
/b\$/
```

sämtliche Zeilen aus, in denen der Kleinbuchstabe `b` enthalten ist, gefolgt von einem Dollar-Zeichen ($).

Außer den Metazeichen kennt `awk` in regulären Ausdrücken und Zeichenketten die folgenden aus der C- Sprache bekannten Escape-Sequenzen:

`\b`	(backspace) Rücksetzzeichen
`\f`	Seitenvorschub
`\n`	Zeilenendezeichen
`\r`	(carriage return) Wagenrücklauf
`\t`	Tabulatorzeichen
`\`*ddd*	Oktaler Wert *ddd*
`\"`	Anführungszeichen
`\`*c*	ein beliebiges anderes Zeichen *c* als darstellbares Zeichen

So können Sie beispielsweise mit dem folgenden Programm alle Zeilen auf dem

Bildschirm abrufen, in denen ein Tabulatorzeichen enthalten ist:

```
/\t/
```

`awk` interpretiert jede Zeichenkette oder Variable, die auf eine Tilde (~) oder die Kombination aus einem Ausrufezeichen und einer Tilde (!~) folgt, als regulären Ausdruck. Das Programm

```
$2 !~ /^[0-9]+$/
```

hätte also auch in folgendem Format eingegeben werden können:

```
BEGIN      { digits = "^[0-9]+$" }
$2 !~ digits
```

Angenommen, Sie möchten nach einer Zeichenkette wie `^[0-9]+$` suchen. Wenn als regulärer Ausdruck dann eine darstellbare, in Anführungszeichen eingeschlossene Zeichenkette wie `"^[0-9]+$"` angegeben wird, müssen die Metazeichen im regulären Ausdruck durch zusätzliche Gegenschrägstriche (\) vor einer Auswertung durch die Shell geschützt werden. Der Grund dafür liegt darin, daß der jeweils erste Gegenschrägstrich beim ersten Abarbeiten einer Zeichenkette gelöscht wird. Wenn die Sonderbedeutung eines Zeichens in einem regulären Ausdruck durch einen Gegenschrägstrich aufgehoben werden muß, so muß der Gegenschrägstrich durch einen weiteren vorangestellten Gegenschrägstrich selbst geschützt werden.

Angenommen, Sie möchten nach Zeichenketten suchen, die den Kleinbuchstaben `b`, gefolgt von einem Dollar-Zeichen ($), enthalten. Der reguläre Ausdruck für dieses Muster sieht dann folgendermaßen aus: `b\$`. Die Zeichenkette für diesen regulären Ausdruck muß einen weiteren Gegenschrägstrich enthalten: `"b\\$"`. Die folgenden Zeilen enthalten jeweils funktionsgleiche reguläre Ausdrücke:

```
x ~ "b\\$"          x ~ /b\$/
x ~ "b\$"           x ~ /b$/
x ~ "b$"            x ~ /b$/
x ~ "\\t"           x ~ /\t/
```

Das exakte Format der regulären Ausdrücke und die Teilzeichenketten, zu denen sie passen, sind in Tabelle 10-2 aufgeführt. Die einstelligen Operatoren *, +, und ? haben die höchste Priorität, gefolgt von der Konkatenation sowie dem logischen ODER (|). Die Operatoren werden von links nach rechts abgearbeitet. *r* steht für einen beliebigen regulären Ausdruck.

Tabelle 10-2: `awk` **- Reguläre Ausdrücke**

Ausdruck	Bedeutung
c	Das Zeichen *c*, wobei *c* kein Metazeichen sein darf.
\c	Das darstellbare Zeichen *c*.
^	Anfang einer Zeichenkette.
$	Ende einer Zeichenkette.
.	Ein beliebiges Zeichen mit Ausnahme des Zeilenendezeichens.
[*s*]	Ein beliebiges Zeichen aus der Gruppe *s*.
[^*s*]	Ein beliebiges Zeichen, das nicht in der Gruppe *s* enthalten sein darf.
*r**	Null, ein- oder mehrmals der reguläre Ausdruck *r*.
r+	Ein- oder mehrmals der reguläre Ausdruck *r*.
r?	Null- oder einmal der reguläre Ausdruck *r*.
(*r*)	*r*
r_1r_2	r_1 gefolgt von r_2 (Konkatenation).
$r_1 \mid r_2$	r_1 oder r_2 (logisches ODER).

Muster-Kombinationen

In komplexeren Mustern sind einfachere Muster durch Klammern und die logischen Operatoren `||` (logisches ODER), `&&` (logisches UND) und `!` (Negation) miteinander verknüpft. Angenommen, Sie möchten eine Übersicht über alle Länder in Asien auf dem Bildschirm abrufen, in denen mehr als 500 Millionen Menschen leben. Sie erreichen dies mit dem folgenden Programm zur Auswahl aller Zeilen, in denen das vierte Feld die Zeichenkette `Asia` enthält und das dritte Feld einen Wert größer 500:

```
$4 == "Asia" && $3 > 500
```

Das Programm

```
$4 == "Asia" || $4 == "Africa"
```

sucht nach allen Zeilen, deren viertes Feld eine der Zeichenketten `Asia` oder `Africa` enthält. Die letzte Abfrage hätte auch mit einem regulären Ausdruck realisiert werden können, in dem eine logische ODER-Verknüpfung (Operator: `|`) verwendet wird:

```
$4 ~ /^(Asia|Africa)$/
```

Der Negations-Operator (`!`) hat die höchste Priorität, gefolgt von `&&` und `||`. Die Operatoren `&&` und `||` werten ihre Operanden entsprechend ihrer Reihenfolge von links nach rechts aus; die Auswertung ist beendet, sobald die Bedingung sich als "wahr" oder "falsch" herausgestellt hat.

Musterbereiche

Ein Musterbereich besteht aus zwei durch Komma voneinander getrennte Muster. Beispiel:

pat$_1$, *pat*$_2$ { . . . }

In diesem Fall wird die Aktion für alle Zeilen durchgeführt, die zwischen dem ersten Auftreten von *pat*$_1$ und dem darauffolgenden Auftreten von *pat*$_2$ (einschließlich) liegen. So paßt das Muster

```
/Canada/, /Brazil/
```

zu dem Zeilenbereich, der bei der ersten Zeile mit der Zeichenkette Canada

beginnt und der darauffolgenden Zeile mit der Zeichenkette Brazil endet:

```
Canada      3852     24       North America
China       3692     866      Asia
USA         3615     219      North America
Brazil      3286     116      South America
```

Ähnlich gibt das Programm

```
FNR == 1, FNR == 5 { print FILENAME, $0 }
```

die ersten fünf Sätze jeder Eingabedatei aus, wobei den Sätzen der Name der aktuellen Eingabedatei vorangestellt ist (`FNR` steht für die laufende Nummer des aktuellen Satzes in der aktuellen Eingabedatei und `FILENAME` für den Namen der aktuellen Eingabedatei).

Aktionen

In einem Muster-Aktions-Paar bestimmt die Aktion darüber, wie die über das Muster ausgewählten Eingabesätze verarbeitet werden. In vielen Fällen handelt es sich bei der Aktion um eine einfache Ausgabeanweisung oder Zuweisung; es ist jedoch auch die Kombination aus zwei oder mehreren Anweisungen möglich. In diesem Abschnitt geht es um die Anweisungen, aus denen sich Aktionen zusammensetzen.

Vordefinierte Variablen

Tabelle 10-3 enthält einen Überblick über die vordefinierten Variablen, die in `awk` eingebaut sind. Einige dieser Variablen haben Sie bereits kennengelernt; in den nachfolgenden Abschnitten dieses Kapitels werden Sie weitere kennenlernen.

Tabelle 10-3: `awk` - Vordefinierte Variablen

Variable	Bedeutung	Standardwert
`ARGC`	Anzahl der Argumente in der Kommandozeile	-
`ARGV`	Feld, das die Argumente der Kommandozeile enthält	-
`FILENAME`	Name der aktuellen Eingabedatei	-
`FNR`	Laufende Nummer des Satzes in der aktuellen Eingabedatei	-
`FS`	Feldtrennzeichen für die Eingabe	Leer- & Tabulatorzeichen
`NF`	Anzahl der Felder im aktuellen Satz	-
`NR`	Laufende Nummer des aktuellen Satzes in der Eingabe	-
`OFMT`	Ausgabeformat für Gleitkommazahlen	`%.6g`
`OFS`	Feldtrennzeichen für die Ausgabe	Leerzeichen
`ORS`	Satztrennzeichen für die Ausgabe	Zeilenendezeichen
`RS`	Satztrennzeichen für die Eingabe	Zeilenendezeichen

lw(5.1i) _ **Tabelle 10-3:** `awk` **- Vordefinierte Variablen** (Fortsetzung)

Variable	Bedeutung	Standardwert
`RSTART`	Anfangsposition der Zeichenkette, die die `match()`-Funktion als passend erkannt hat.	-
`RLENGTH`	Länge der Zeichenkette, die die `match()`-Funktion als passend erkannt hat.	-
`SUBSEP`	Trennzeichen für Indizes	"\034"

Arithmetische Berechnungen

In einer Aktion können zur Errechnung von numerischen Werten arithmetische Standard-Ausdrücke benutzt werden. Angenommen, Sie möchten für jedes Land, das in der Datei `countries` aufgeführt ist, die Bevölkerungsdichte ermitteln. Da das zweite Feld die Fläche in Tausend Quadratmeilen enthält, das dritte Feld die Bevölkerungszahl in Millionen, können Sie über die Anweisung `1000 * $3 / $2` die Bevölkerungsdichte pro Quadratmeile errechnen. Das Programm

```
{ printf "%10s %6.1f\n", $1, 1000 * $3 / $2 }
```

gibt mit der Eingabedatei `countries` den Namen jedes Landes mit seiner Bevölkerungsdichte aus:

```
      USSR   30.3
    Canada    6.2
     China  234.6
       USA   60.6
    Brazil   35.3
 Australia    4.7
     India  502.0
 Argentina   24.3
     Sudan   19.6
   Algeria   19.6
```

Arithmetische Berechnungen werden intern als Gleitkomma-Arithmetik ausgeführt. `awk` kennt die folgenden arithmetischen Operatoren: `+, -, *, /, %` (Modulo) und `^` (Exponent; gleichbedeutend mit `**`). Diese Operatoren können in arithmetischen Ausdrücken auf Konstante, Variablen, Feldnamen, Feldelemente, Funktionen und andere Ausdrücke angewandt werden, die an einer späteren Stelle dieses Kapitels beschrieben werden. Beachten Sie, daß `awk` die wissenschaftliche Exponential-Schreibweise interpretieren und in seiner Ausgabe verwenden kann: `1e6, 1E6, 10e5,` und `1000000` sind funktionsgleich.

Zuweisungen werden unter `awk` wie in der Programmiersprache C eingegeben. Die einfachste Form einer Zuweisung ist die Anweisung

v = *a*

mit *v* als Variable oder Feldname und *a* als Ausdruck. So könnten Sie die Anzahl der Länder in Asien und ihre Gesamtbevölkerung mit dem folgenden Programm ermitteln:

```
$4 == "Asia"   { pop = pop + $3; n = n + 1 }
END            { print "population of", n,
                   "Asian countries in millions is", pop }
```

Mit der Eingabedatei `countries` erzeugt dieses Programm die folgende Ausgabe:

```
population of 3 Asian countries in millions is 1765
```

Die zu dem Muster `$4 == "Asia"` gehörige Aktion enthält zwei Zuweisungen zur Errechnung der Gesamt-Bevölkerungszahl und der Anzahl der Länder. Obwohl die Variablen nicht explizit initialisiert werden, läuft das Programm wie gewünscht ab, da `awk` jede Variable mit dem Wert `""` (leere Zeichenkette) und dem numerischen Wert `0` initialisiert.

Die Zuweisungen im letzten Programm können mit den Operatoren += und ++ in eine kompaktere Form gebracht werden:

```
$4 == "Asia"    { pop += $3; ++n }
```

Der Operator += hat dieselbe Funktion wie der gleichlautende C-Operator. Daher führt

```
pop += $3
```

zum selben Ergebnis wie

```
pop = pop + $3
```

Allerdings hat der Operator += den Vorteil eines kompakteren Formats und einer höheren Durchführungsgeschwindigkeit. Dasselbe gilt für den Operator ++ durch den eine Variable um dem Wert 1 hochgezählt wird.

Die verkürzten Varianten der Zuweisungs-Operatoren sind: `+=`, `-=`, `*=`, `/=`, `%=` und `^=`. Die beiden Varianten haben eine ähnliche Bedeutung:

v op= a

führt zu demselben Ergebnis wie

v = v op a.

Als Inkrement-Operatoren werden ++ und -- verwendet. Wie in der C-Sprache kann man sie sowohl in Präfix- (`++x`) als auch in Postfix-Position (`x++`) benutzen. Ist `x` gleich 1, so bewirkt `i=++x` zunächst die Erhöhung von `x`, dann die Zuweisung des Werts 2 an `i`. `i=x++` dagegen setzt zunächst `i` auf 1 und zählt dann `x` hoch. Auf die gleiche Weise wird der Operator -- in Präfix- und Postfix-Position interpretiert.

In arithmetischen Ausdrücken können alle Zuweisungs-, Inkrement- und Dekrement-Operatoren verwendet werden.

Die Standard-Initialisierung ist im folgenden Programm hilfreich, in dem das Land mit der höchsten Bevölkerungszahl ermittelt wird:

```
maxpop < $3   { maxpop = $3; country = $1 }
END           { print country, maxpop }
```

Beachten Sie jedoch, daß dieses Programm nicht das korrekte Ergebnis bringen würde, wenn alle Werte im Feld `$3` negativ wären.

Tabelle 10-4 enthält einen Überblick über die vordefinierten arithmetischen Funktionen von `awk`.

Tabelle 10-4: `awk` - Vordefinierte arithmetische Funktionen

Funktion	Bedeutung
`atan2(`*y*,*x*`)`	Arcustangens von *y*/*x* im Bereich *–pi* bis π.
`cos(`*x*`)`	Cosinus von *x* (*x* wird in Rad angegeben).
`exp(`*x*`)`	Exponentialfunktion von *x*.
`int(`*x*`)`	Ganzzahliger Anteil von *x*.
`log(`*x*`)`	Natürlicher Logarithmus von *x*.
`rand()`	Zufallszahl zwischen 0 und 1.
`sin(`*x*`)`	Sinus von *x* (*x* wird in Rad angegeben).
`sqrt(`*x*`)`	Quadratwurzel von *x*.
`srand(`*x*`)`	*x* ist der Anfangs-Berechnungswert für rand ().

x und *y* sind beliebige Ausdrücke. Die Funktion `rand()` gibt eine Pseudo-Zufallszahl (eine Gleitkommazahl im Bereich (0,1)) aus; `srand(`*x*`)` kann zur Angabe des Anfangs- Berechnungswerts für den Zufallsgenerator benutzt werden. Wird bei `srand()` kein Argument angegeben, wird der Anfangswert aus der Uhrzeit abgeleitet.

Zeichenketten und Zeichenketten-Funktionen

Eine Zeichenkette (alphanumerische Konstante) ist eine Folge von Zeichen, die in Anführungszeichen (") eingeschlossen ist (z.B. `"abc"` oder `"hello, everyone"`). In einer alphanumerischen Konstanten können die aus der C-Sprache bekannten Escape-Sequenzen für Steuerzeichen enthalten sein (siehe Abschnitt "Reguläre Ausdrücke" in diesem Kapitel).

Zeichenketten (alphanumerische Konstanten) entstehen durch Aneinanderreihung (Konkatenation) von Konstanten, Variablen, Feldnamen, Feldelementen, Funktionen und anderen Ausdrücken. Das Programm

```
{ print NR ":" $0 }
```

gibt jeden Satz mit vorangestellter Satznummer sowie einem Doppelpunkt (:) ohne Leerzeichen aus. Die drei Zeichenketten, die für die Satznummer, den Doppelpunkt sowie den Satz selbst stehen, werden aneinandergereiht; die so

entstandene Zeichenkette wurde ausgegeben. Für die Konkatenation gibt es kein explizites Symbol; es genügt, die Operanden einfach hintereinanderzuschreiben.

Tabelle 10-5 enthält einen Überblick über die vordefinierten Zeichenketten-Funktionen von `awk`. In dieser Tabelle stehen *r* für einen regulären Ausdruck (entweder in Form einer Zeichenkette oder im Format */r/*), *s* und *t* für alphanumerische Ausdrücke sowie *n* und *p* für Ganzzahlen.

Tabelle 10-5: `awk` **Vordefinierte Zeichenketten-Funktionen**

Funktion	Bedeutung
`gsub(`*r*`,`*s*`)`	Globale Substitution von *r* durch *s* im aktuellen Satz; gibt die Anzahl der Substitutionen aus.
`gsub(`*r*`,`*s*`,`*t*`)`	Globale Substitution von *r* durch *s* in der Zeichenkette *t*; gibt die Anzahl der Substitutionen aus.
`index(`*s*`,`*t*`)`	Anfangsposition der Teilzeichenkette *t* in *s*; gibt 0 aus, wenn die Teilzeichenkette nicht enthalten ist.
`length(`*s*`)`	Länge der Zeichenkette *s*.
`match(`*s*`,`*r*`)`	Position in *s*, an der die zu *r* passende Zeichenkette anfängt; wird keine Zeichenkette gefunden, wird der Wert 0 ausgegeben.
`split(`*s*`,`*a*`)`	Aufteilung der Zeichenkette *s* in Feldelemente entsprechend FS und Ablegen der Feldelemente im Feld *a*; gibt die Anzahl der Felder aus.
`split(`*s*`,`*a*`,`*r*`)`	Aufteilung der Zeichenkette *s* in Feldelemente entsprechend *r* und Ablegen der Feldelemente im Feld *a*; gibt die Anzahl der Feldelemente aus.
`sprintf(`*fmt*`,`*ausdr_liste*`)`	*ausdr_liste* wird dem Format *fmt* entsprechend formatiert und ausgegeben.
`sub(`*r*`,`*s*`)`	Substitution des regulären Ausdrucks *r* bei seinem ersten Auftreten durch die Zeichenkette *s*; gibt die Anzahl der Substitutionen aus.
`sub(`*r*`,`*s*`,`*t*`)`	Substitution des regulären Ausdrucks *r* bei seinem ersten Auftreten in *t* durch *s*; gibt die Anzahl der Substitutionen aus.
`substr(`*s*`,`*p*`)`	Teilzeichenkette am Ende von *s*, die bei Position *p* `substr(`*s*`,`*p*`,`*n*`); beginnt.`
`substr(`*s*`,`*p*`,`*n*`)`	Teilzeichenkette aus *s* mit der Länge *n*, die bei Position *p* beginnt.

Die Funktionen `sub` und `gsub` funktionieren nach dem Muster des Ersetzen-Kommandos des Texteditors `ed`(1). Die Funktion `gsub (`*r*`,`*s*`,`*t*`)` ersetzt in der Zeichenkette *t* alle Teilzeichenketten, die zum regulären Ausdruck *r* passen, durch die Ersatzkette *s* (wie unter `ed` wird das erste der aufgeführten Muster so lange benutzt, wie es zu einer Teilzeichenkette paßt). Die Funktion gibt die Anzahl der durchgeführten Substitutionen aus. Die Funktion `gsub (`*r*`,`*s*`)`

entspricht `gsub (r,s,$0)`. So ändert das Programm

```
{ gsub(/USA/, "United States"); print }
```

seine Eingabe, indem es `USA` bei jedem ersten Auftreten durch `United States` ersetzt. Die Funktion `sub` läuft ähnlich ab, ersetzt die passende Teilzeichenkette aber nur bei ihrem Auftreten in der Zeichenkette.

Die Funktion `index(s,t)` gibt die äußerst linke Position aus, an der die Teilzeichenkette *t* in *s* beginnt; kommt *t* nicht in *s* vor, gibt sie den Wert 0 aus. Das erste Zeichen einer Zeichenkette befindet sich an Position 1. So wird bei der Funktion

```
index("banana", "an")
```

der Wert 2 ausgegeben.

Die Funktion `length` gibt die Anzahl der Zeichen in ihrem alphanumerischen Argument aus; so gibt

```
{ print length($0), $0 }
```

jeden Satz mit vorangestellter Längenangabe aus (`$0` schließt das Satztrennzeichen für die Eingabe nicht ein). Mit der Eingabedatei `countries` gibt das Programm

```
length($1) > max   { max = length($1); name = $1 }
END                { print name }
```

den längsten Ländernamen aus:
`Australia`.

Die Funktion `match(s,r)` gibt die Position innerhalb der Zeichenkette *s* aus, an der das erste Zeichen der zum regulären Ausdruck *r* passenden Zeichenkette vorkommt; ist er nicht enthalten, gibt sie den Wert 0 aus. Über diese Funktion werden auch die beiden vordefinierten Variablen `RSTART` und `RLENGTH` eingestellt. `RSTART` wird auf die Anfangsposition der gefundenen Teilzeichenkette innerhalb der Zeichenkette (den Rückkehr-Wert von `match`) gesetzt. `RLENGTH` wird auf die Länge der gefundenen Teilzeichenkette gesetzt. Wenn keine Zeichenkette paßt, wird `RSTART` auf 0 und `RLENGTH` auf −1 gesetzt. So sucht das folgende Programm nach dem ersten Auftreten des Kleinbuchstabens `i`, gefolgt von maximal einem Zeichen, auf das wiederum der Kleinbuchstabe `a` folgt:

```
{ if (match($0, /i.?a/))
      print RSTART, RLENGTH, $0 }
```

Mit der Eingabedatei `countries` erzeugt das Programm die folgende Ausgabe:

```
17 2 USSR 8650    262     Asia
26 3 Canada        3852    24      North America
3 3 China 3692    866     Asia
24 3 USA  3615     219     North America
27 3 Brazil        3286    116     South America
8 2 Australia      2968    14      Australia
4 2 India 1269    637     Asia
7 3 Argentina      1072    26      South America
17 3 Sudan         968     19      Africa
6 2 Algeria        920     18      Africa
```

Hinweis: `match ()` paßt zur äußerst linken, längsten der passenden Zeichenketten. So sucht das Programm

```
{ if (match($0, /a+/))  print RSTART, RLENGTH, $0 }
```

mit dem Eingabesatz

```
AsiaaaAsiaaaaan
```

nach der ersten Folge aus den Kleinbuchstaben `a` und setzt `RSTART` auf 4 und `RLENGTH` auf 3.

Die Funktion `sprintf(`*format, ausdr*$_1$, *ausdr*$_2$, . . ., *ausdr*$_n$`)` formatiert eine Zeichenkette aus den Ausdrücken *ausdr*$_1$, *ausdr*$_2$, . . ., *ausdr*$_n$ entsprechend den `printf`- Formatangaben in der Zeichenkette *format* (die Zeichenkette wird jedoch nicht ausgegeben). Im Abschnitt "Die Funktion `printf`" finden Sie eine vollständige Übersicht über die unterstützten Formatangaben., Die Anweisung

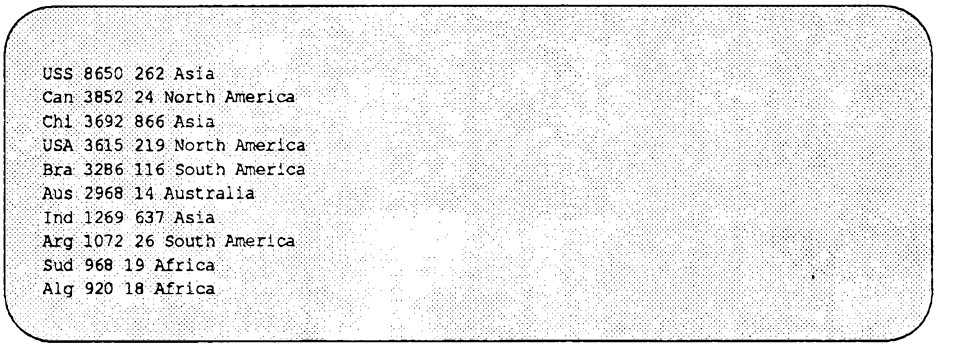
```
x = sprintf("%10s %6d", $1, $2)
```

weist `x` die Zeichenkette zu, die durch die Formatierung der Werte von `$1` und `$2` als zehnstellige Zeichenkette und Dezimalzahl in einem Feld mit einer Breite von mindestens sechs Zeichen erzeugt worden ist; `x` kann später weiterverarbeitet werden.

Die Funktion `substr(s,p,n)` liefert diejenige Teilzeichenkette in *s*, die bei Position *p* beginnt und maximal *n* Zeichen lang ist. Bei Verwendung der Variante `substr(s,p)` muß das Ende der Teilzeichenkette mit dem von *s* zusammenfallen; sie besteht dann also aus dem Suffix von *s*, das an Position *p* beginnt. So können Sie beispielsweise die Ländernamen in der Eingabedatei `countries` mit dem folgenden Programm auf ihre ersten drei Zeichen verkürzen:

```
{ $1 = substr($1, 1, 3); print }
```

Die Ausgabe sieht dann folgendermaßen aus:



```
USS 8650 262 Asia
Can 3852 24 North America
Chi 3692 866 Asia
USA 3615 219 North America
Bra 3286 116 South America
Aus 2968 14 Australia
Ind 1269 637 Asia
Arg 1072 26 South America
Sud 968 19 Africa
Alg 920 18 Africa
```

Beachten Sie, daß `awk $0` (also den ganzen Satz) wegen der Einstellung von `$1` neu berechnen muß; daher sind die Felder nicht durch Tabulatorzeichen voneinander getrennt, sondern durch Leerzeichen (Standardeinstellung der vordefinierten Variablen `OFS`).

Sollen die Zeichenketten bei der Ausgabe unmittelbar aufeinanderfolgen (Konkatenation), müssen Sie in einem Ausdruck einfach ohne Operator nebeneinander geschrieben werden. Wenn Sie beispielsweise das Programm

```
{ s = s substr($1, 1, 3) " " } END  { print s }
```

mit der Eingabedatei `countries` aufrufen, sieht die Ausgabe folgendermaßen aus:

```
USS Can Chi USA Bra Aus Ind Arg Sud Alg
```

Ausgehend von einer leeren Zeichenkette, wird `s` Stück für Stück aufgebaut.

Vordefinierte Variablen

Auf die Felder des aktuellen Satzes können Sie über die vordefinierten Variablen `$1, $2, . . ., $NF` zugreifen. Die vordefinierten Variablen haben dieselben Eigenschaften wie die übrigen Variablen —Sie können sie sowohl bei arithmetischen Berechnungen als auch bei der Verarbeitung von Zeichenketten verwenden und ihnen auch Werte zuweisen. So können Sie beispielsweise das zweite Feld der Datei `countries` durch 1000 dividieren, um die Fläche statt in Tausend Quadratmeilen in Millionen Quadratmeilen anzugeben:

```
{ $2 /= 1000; print }
```

Ebenso können Sie einem Feld eine neue Zeichenkette zuweisen:

```
BEGIN                       { FS = OFS = "\t" }
$4 == "North America"       { $4 = "NA" }
$4 == "South America"       { $4 = "SA" }
                            { print }
```

In diesem Programm wird durch die Aktion bei `BEGIN` das Eingabe-Feldtrennzeichen `FS` sowie das Ausgabe-Feldtrennzeichen `OFS` auf das Tabulatorzeichen gesetzt. Beachten Sie, daß die Funktion `print` in der vierten Programmzeile den Wert von `$0` erst dann ausgibt, nachdem er zuvor durch Zuweisungen geändert worden ist.

Auf die Felder können Sie über Ausdrücke zugreifen. So greift z.B. `$(NF-1)` auf das vorletzte Feld des aktuellen Satzes zu. Die Klammern dürfen nicht fehlen, da der Wert von `$NF-1` um 1 unter dem Wert des letzten Feldes liegt.

Eine vordefinierte Variable, die auf ein nicht vorhandenes Feld zugreift (z.B. `$(NF+1)`), wird mit der leeren Zeichenkette initialisiert. Allerdings können Sie ein neues Feld anlegen, indem Sie ihm einfach einen Wert zuweisen. Wird das folgende Programm beispielsweise mit der Eingabedatei `countries` aufgerufen, so legt es ein fünftes Feld an, in dem die Bevölkerungsdichte angegeben wird:

```
BEGIN   { FS = OFS = "\t" }
        { $5 = 1000 * $3 / $2; print }
```

Die Sätze können unterschiedlich lang sein; bei den meisten Implementierungen liegt die Grenze jedoch bei 100 Feldern pro Datensatz.

Zahl oder Zeichenkette?

Variablen, Felder und Ausdrücke können sowohl numerisch als auch alphanumerisch (Zeichenketten) sein. Der zugeordnete Datentyp ist kontextabhängig. In einem Kontext wie

```
pop += $3
```

müssen `pop` und `$3` als numerische Werte behandelt werden; sie werden also gegebenenfalls in numerische Werte umgewandelt.

Bei einer Funktion zur Verarbeitung von Zeichenketten wie

```
print $1 ":" $2
```

dagegen müssen `$1` und `$2` aneinanderzureihende Zeichenketten sein; auch hier wird gegebenenfalls die notwendige Typumwandlung vorgenommen.

In einer Zuweisung im Format $v = a$ oder $v\ op= a$, paßt sich v im Datentyp an a an. In einem mehrdeutigen Kontext wie

```
$1 == $2
```

hängt die Art des durchgeführten Vergleichs davon ab, ob es sich um numerische oder alphanumerische Felder handelt; dies kann erst während des Programmlaufs ermittelt werden, da die unterschiedlichen Sätze unterschiedliche Datentypen haben können.

Bei einem Vergleich von zwei numerischen Operanden wird ein numerischer Vergleich durchgeführt; andernfalls werden die Operanden in Zeichenketten umgewandelt und ein alphanumerischer Vergleich durchgeführt. Vordefinierte Variablen sind auf jeden Fall Zeichenketten; außerdem wird jedes Feld, in dem nur eine Zahl enthalten ist, als numerisch betrachtet. Der Datentyp wird während des Programmlaufs ermittelt. So kann z.B. der Vergleich `"$1 == $2"` für die Eingabe

```
1     1.0     +1     0.1e+1     10E-1    001
```

durchgeführt werden, nicht aber für die Eingabezeilen

```
(Null)  0
(Null)  0.0
0a      0
1e50    1.0e50
```

Für die Typumwandlung gibt es zwei Formate:

zahl ""	Eine *zahl* wird mit einer leeren Zeichenkette verknüpft, um sie in eine Zeichenkette umzuwandeln.
zeichenkette + 0	Zu einer *zeichenkette* wird der Wert Null addiert, um sie in einen numerischen Wert umzuwandeln.

Soll zwischen zwei Feldern auf jeden Fall ein alphanumerischer Vergleich durchgeführt werden, so geben Sie also eine Anweisung in folgendem Format ein:

```
$1 "" == $2 ""
```

Als numerischer Wert einer Zeichenkette wird der Wert eines beliebigen Präfixes der Zeichenkette benutzt, der numerisch aussieht; so hat `12.34x` den Wert 12.34, `x12.34` dagegen den Wert Null. Der alphanumerische Wert eines arithmetischen Ausdrucks wird ermittelt, indem die Zeichenkette dem Ausgabeformat `OFMT` entsprechend formatiert wird.

Nicht initialisierten Variablen wird der numerische Wert 0 und der alphanumerische Wert "" (leere Zeichenkette) zugeordnet. Felder, die in der Aufrufzeile nicht existieren oder explizit leer sind, haben lediglich den alphanumerischen Wert "" (leere Zeichenkette); sie sind nicht numerisch.

Ablaufsteuerung

Zur Ablaufsteuerung von `awk`-Programmen gibt es die Anweisungen `if-else`, `while`, `do-while` und `for`; außerdem können die Anweisungen wie in der C-Sprache durch geschweifte Klammern gruppiert werden.

Die Anweisung `if` hat folgendes Format:

`if` (*ausdruck*) *anweisung*$_1$ `else` *anweisung*$_2$

ausdruck ist dabei eine Bedingung ohne EInschränkungen; er kann die Vergleichsoperatoren <, <=, >, >=, == und != enthalten, außerdem die Mustervergleichsoperatoren ~ und !~, die logischen Operatoren `||`, `&&` sowie `!`. Außerdem ist die Konkatenation durch Aneinanderreihung sowie die Klammerung zulässig.

In einer `if`- Anweisung wertet `awk` zunächst *ausdruck* aus. Ist *ausdruck* ungleich Null, wird *anweisung*$_1$ ausgeführt, andernfalls *anweisung*$_2$. Der `else`- Teil ist optional.

Statt einer einzelnen Anweisung können auch mehrere in geschweifte Klammern eingeschlossene Anweisungen angegeben werden; die einzelnen Anweisungen müssen dann durch Zeilenendezeichen oder Semikola voneinander getrennt werden.

Das Programm zur Ermittlung der höchsten Bevölkerungszahl, das im Abschnitt "Arithmetische Berechnungen" beschrieben wurde, kann mit Hilfe einer `if`- Anweisung folgendermaßen umformuliert werden:

```
{    if (maxpop < $3) {
         maxpop = $3
         country = $1
     }
}
END  { print country, maxpop }
```

Die `while`- Anweisung entspricht der gleichnamigen C-Anweisung:

`while` (*ausdruck*) *anweisung*

Solange *ausdruck* ungleich Null ist und nicht die leere Zeichenkette, wird *anweisung* ausgeführt und *ausdruck* erneut ausgewertet. Dieser Zyklus wird fortgesetzt, bis *ausdruck* Null wird. Mit der folgenden Anweisung werden beispielsweise sämtliche Eingabefelder in separaten Zeilen ausgegeben:

```
{  i = 1
   while (i <= NF) {
      print $i
      i++
   }
}
```

Die Anweisung `for` entspricht der gleichnamigen C- Anweisung:

`for` (*ausdruck*$_1$`;` *ausdruck*`;` *ausdruck*$_2$) *anweisung*

Ihre Funktion entspricht

```
ausdruck1
while (ausdruck) {
        anweisung
        ausdruck2
}
```

Somit hat

```
{ for (i = 1; i <= NF; i++)  print $i }
```

dasselbe Ergebnis wie das obige Beispiel mit der `while`- Anweisung. Im nächsten Abschnitt wird eine Variante der `for`- Anweisung vorgestellt.

Die `do`-Anweisung hat folgendes Format:

`do` *anweisung* `while` (*ausdruck*)

anweisung wird so oft ausgeführt, bis *ausdruck* Null wird. Da die Überprüfung nach der Durchführung von *anweisung* erfolgt (am Ende der Schleife), findet sie in jedem Fall mindestens einmal statt. Daher wird die `do`-Anweisung wesentlich seltener benutzt als `while` oder `for`, bei denen die Überprüfung zu Beginn der Schleife durchgeführt wird.

Mit der folgenden `do`-Anweisung werden sämtliche Zeilen mit Ausnahme derjenigen zwischen `start` und `stop` ausgegeben:

```
/start/ {
        do {
            getline x
        } while (x !~ /stop/)
    }
    { print }
```

Die `break`- Anweisung bewirkt den sofortigen Abbruch einer übergeordneten `while`- oder `for`- Anweisung; die `continue`- Anweisung bewirkt die Fortsetzung des Programms beim nächsten Schleifendurchgang. Bei der Anweisung `next` springt `awk` sofort zum nächsten Satz und beginnt den Mustervergleich wieder beim ersten Muster-Aktions-Paar.

Bei der `exit`- Anweisung verhält sich das Programm so, als wäre das Ende der Eingabe erreicht; es werden keine weiteren Eingabezeilen mehr eingelesen; die Aktion bei `END` wird gegebenenfalls durchgeführt (falls vorhanden). Enthält die Aktion bei `END` die Anweisung

```
exit ausdr
```

so gibt das Programm den Wert von *ausdr* als seinen Ende-Status aus. Fehlt *ausdr*, wird als Ende-Status Null ausgegeben.

Felder (Arrays)

`awk` kennt eindimensionale Felder (Arrays). Felder und Feldelemente müssen nicht deklariert werden; wie Variablen werden sie bereits durch Erwähnung ins Leben gerufen. Ein Feldindex kann eine Zahl oder eine Zeichenkette sein.

Das folgende Beispiel enthält einen gängigen numerischen Index: In der Anweisung

```
x[NR] = $0
```

wird die aktuelle Eingabezeile dem `NR`-ten Element des Felds `x` zugewiesen. Prinzipiell ist es sogar möglich (wenn auch etwas zeitaufwendig), die gesamte Eingabe in ein Feld einzulesen; hierzu eignet sich das folgende `awk`- Programm:

```
      { x[NR] = $0 }
END   { . . . Verarbeitung . . . }
```

Die erste Aktion speichert lediglich jede Eingabezeile in dem Feld `x` ab, das über einen Index in Form einer Zeilennummer angesprochen wird; die Verarbeitung findet dann im `END`- Teil statt.

Feldelemente können auch über nicht- numerische Werte angesprochen werden. So summiert das folgende Programm die Bevölkerungszahlen von Asien und Afrika im assoziativen Feld `pop` auf. Die Aktion `END` gibt dann die Summe aus den Bevölkerungszahlen beider Kontinente aus:

```
/Asia/      { pop["Asia"] += $3 }
/Africa/    { pop["Africa"] += $3 }
END         { print "Asian population in millions is", pop["Asia"]
              print "African population in millions is", pop["Africa"]
            }
```

Mit der Eingabedatei `countries` erzeugt dieses Programm die folgende Ausgabe:

```
Asian population in millions is 1765
African population in millions is 37
```

Enthielte dieses Programm `pop[Asia]` statt `pop["Asia"]` , so hätte der Ausdruck den Wert der Variablen `Asia` als Feldindex verwendet; da die Variable nicht initialisiert worden ist, wären diese Werte zu `pop[""]` aufsummiert worden.

Angenommen, Sie möchten anhand der Eingabedatei `countries` die Gesamtfläche jedes Kontinents ermittelt. In einer Bezugnahme auf ein Feld können Sie jeden beliebigen Ausdruck als Index verwenden. So benutzt

```
area[$4] += $2
```

die Zeichenkette im vierten Feld des aktuellen Eingabesatzes zum Zugriff auf das Feld `area` und summiert in diesem Eintrag den Wert des zweiten Felds auf:

```
BEGIN { FS = "\t" }
      { area[$4] += $2 }
END   { for (name in area)
                print name, area[name]
}
```

Mit der Eingabedatei `countries` erzeugt dieses Programm die folgende Ausgabe:

```
Africa 1888
North America 7467
South America 4358
Asia 13611
Australia 2968
```

In diesem Programm wird eine Variante der `for`- Anweisung benutzt, die sämtliche definierten Indizes eines Felds durchläuft: Die Anweisung

`for (`*i* `in` *feld*`)` *anweisung*

führt *anweisung* aus, wobei die Variable *i* hintereinander auf jeden Wert von *i* gesetzt wird, für den *feld[i]* definiert worden ist. Die Schleife wird für jeden definierten Index einmal durchlaufen; die Reihenfolge wird nach dem Zufallsprinzip ermittelt. Bei einer Änderung von *i* oder *feld* während der Schleife ist das Ergebnis nicht vorhersehbar.

`awk` unterstützt zwar keine mehrdimensionalen Felder, läßt aber mehrere Indizes zu. Die Indizes werden zu einem einzigen Index verknüpft, wobei die Werte durch eine nur selten vorkommende Zeichenkette (die in der Variablen `SUBSEP` abgelegt ist) voneinander getrennt werden. So erzeugt

```
for (i = 1; i <= 10; i++)
      for (j = 1; j <= 10; j++)
                  arr[i,j] = ...
```

ein Feld, das sich wie ein zweidimensionales Feld verhält. Der Index entsteht durch Konkatenation von *i*, `SUBSEP` und *j*.

Ob in einem Feld *feld* ein bestimmter Index *i* vorkommt, können Sie wie im folgenden Beispiel über die Bedingung *i* in *feld* überprüfen:

```
if ("Africa" in area) ...
```

Diese Bedingung wird überprüft, ohne daß zusätzlich `area["Africa"]` angelegt wird; dies hat die folgende Anweisung zur Folge:

```
if (area["Africa"] != "") ...
```

Beachten Sie jedoch, daß in keinem der beiden Fälle überprüft wird, ob das Feld `area` ein Element mit dem Wert `"Africa"` enthält.

Mit der vordefinierten Funktion `split` können Sie jede beliebige Zeichenkette in Feldelemente aufteilen und in Feldern ablegen. So teilt die Funktion

```
split("s1:s2:s3", a, ":")
```

die Zeichenkette `s1:s2:s3` in drei Feldelemente auf, die durch das Feldtrennzeichen : voneinander getrennt sind, und legt `s1` in `a[1]`, `s2` in `a[2]` und `s3` in `a[3]` ab. Die Anzahl der gefundenen Feldelemente (in diesem Fall drei) wird von `split` als Funktionswert ausgegeben. Beim dritten Argument von `split` handelt es sich um einen regulären Ausdruck, der als Feldtrennzeichen verwendet werden soll. Fehlt das dritte Argument, so wird `FS` als Feldtrennzeichen benutzt.

Ein Feldelement kann mit der `delete`- Anweisung gelöscht werden:

`delete` *feld_name*`[`*index*`]`

Benutzer-definierte Funktionen

`awk` bietet dem Benutzer die Möglichkeit, eigene Funktionen zu definieren. Eine Funktion ist folgendermaßen definiert:

```
function name (argument_liste)  {
        anweisungen
}
```

Die Definition kann an jeder Stelle erfolgen, an der auch ein Muster-Aktions-Paar vorkommen kann. Die Argumenten-Liste besteht aus einer Reihe von Variablennamen, die durch Komma voneinander getrennt sind; sind diese Variablen im Funktions-Rumpf angegeben, so nehmen sie auf die realen Parameter im Funktionsaufruf Bezug. Zwischen dem Funktionsnamen und der öffnenden Klammer der Argumentenliste darf im Funktionsaufruf kein Leerzeichen stehen; andernfalls interpretiert `awk` sie als Konkatenation. Das folgende Programm definiert und überprüft beispielsweise die rekursive Fakultätsfunktion (als Eingabe wird hier selbstverständlich nicht die Datei `countries` benutzt):

```
function fact(n) {
    if (n <= 1)
        return 1
    else
        return n * fact(n-1)
}
{ print $1 "! is " fact($1) }
```

Feld-Argumente werden wie in C durch Referenz übergeben; dadurch kann die Funktion vorhandene Feldelemente ändern oder neue erstellen. Nicht strukturierte Argumente (Skalare) werden dagegen als Wert übergeben, so daß die Werte außerhalb der Funktion unverändert bleiben. Innerhalb einer Funktion sind die Funktionsparameter lokale Variablen, alle übrigen Variablen jedoch global (für die Anzahl der zusätzlichen Funktionsparameter, die ausschließlich als lokale Variablen benutzt werden, gibt es keine Einschränkung). Die Anweisung `return` ist optional; allerdings ist der zurückgegebene Wert undefiniert, wenn diese Anweisung fehlt.

Zum Format von awk-Programmen

Ein `awk`- Programm kann auch Kommentare enthalten; ein Kommentar beginnt mit dem Zeichen # und endet am Zeilenende:

```
print x, y      # Kommentar
```

Eine Anweisung ist in einem `awk`- Programm normalerweise in einer einzigen Zeile enthalten. Sind in einer Zeile mehrere Anweisungen enthalten, so müssen sie durch Semikolon voneinander getrennt werden. Eine überlange Anweisung kann auf zwei oder mehr Zeilen verteilt werden, indem am Ende der einzelnen Zeilen ein Gegenschrägstrich (\) eingegeben wird (eine in Anführungszeichen eingeschlossene Zeichenkette ("...") kann nicht unterbrochen und in der nächsten Zeile fortgesetzt werden). Eine derartige explizite Fortsetzung der Anweisung ist allerdings nur selten notwendig, da eine Anweisung automatisch durch ein Komma am Zeilenende fortgesetzt wird (dies kann beispielsweise in einer `print`- oder `printf`- Anweisung oder nach den Operatoren `&&` und `||` der Fall sein).

In einer Zeile können mehrere, durch Semikolon voneinander getrennte Muster-Aktions-Paare vorkommen.

Die Ausgabe

Die Generierung einer Ausgabe erfolgt in der Regel über die Anweisungen `print` und `printf`. Mit der `print`- Anweisung können Sie eine einfache Ausgabe erzeugen, mit der `printf`- Anweisung eine etwas weitergehend formatierte Ausgabe. Wie die Shell erlaubt `awk` die Umleitung der Ausgabe, so daß die Ausgabe von `print` und `printf` an Dateien und Pipes weitergeleitet werden kann. In diesem Abschnitt geht es um das Arbeiten mit diesen beiden Funktionen.

Die Funktion `print`

Die Funktion

`print` *ausdr*$_1$, *ausdr*$_2$, . . . , *ausdr*$_n$

gibt den alphanumerischen Wert (Zeichenkette) aller Ausdrücke, getrennt durch das Ausgabe-Feldtrennzeichen, aus. Die Funktion

```
print
```

ist die verkürzte Form für

```
print $0
```

Die Ausgabe einer Leerzeile erreichen Sie mit

```
print "" .
```

Trennzeichen für die Ausgabe

Das Feld- und Satztrennzeichen für die Ausgabe sind in den vordefinierten Variablen `OFS` und `ORS` abgelegt. `OFS` ist standardmäßig auf ein einzelnes Leerzeichen gesetzt, `ORS` auf ein einzelnes Zeilenendezeichen; allerdings ist die Änderung dieser Variablen jederzeit möglich. So gibt das folgende Programmbeispiel das erste und zweite Feld jedes Satzes aus, wobei die Felder durch einen Doppelpunkt (:) voneinander getrennt sind und auf das zweite Feld zwei Zeilenendezeichen folgen:

```
BEGIN   { OFS = ":"; ORS = "\n\n" }
        { print $1, $2 }
```

Beachten Sie, daß

```
{ print $1 $2 }
```

das erste und zweite Feld ohne Feldtrennzeichen ausgibt, da `$1 $2` eine Zeichenkette ist, die durch Konkatenation der ersten beiden Felder entstanden ist.

Die Funktion `printf`

Die Funktion `printf` von `awk` ist mit der gleichnamigen C-Funktion identisch, nur wird die Formatangabe Stern `(*)` nicht unterstützt. Die `printf`-Anweisung hat folgendes allgemeines Format:

`printf` *format*, $ausdr_1$, $ausdr_2$, . . . , $ausdr_n$

Dabei ist *format* eine Zeichenkette, die sowohl die auszugebenden Informationen als auch Angaben über die Umwandlungen enthält, die mit den Ausdrücken in der Argumenten-Liste durchgeführt werden sollen (siehe Bild 10-9). Jede Formatangabe beginnt mit dem Symbol `%` und endet mit einem Buchstaben, der für die Art der Umwandlung steht. Folgende Formatangaben sind möglich:

- `-` Linksbündige Ausrichtung des Ausdrucks in seinem Feld.
- *fb* (Minimale Feldbreite). Das Feld soll gegebenenfalls mit Leerzeichen aufgefüllt werden; ist *fb* die Zahl 0 vorangestellt, so wird das Feld mit Nullen aufgefüllt.
- *.g* Maximale Länge der Zeichenkette bzw. Anzahl der Dezimalstellen.

Tabelle 10-6 enthält einen Überblick über die Zeichen zur Zeichenformatierung unter `printf`.

Tabelle 10-6: `awk printf` **Konvertierungszeichen**

Zeichen	Formatierung des Ausdrucks als:
`c`	(character) Einzelzeichen
`d`	Dezimalzeichen
`e`	[−]*d.dddddd*`E`[+−]*dd*
`f`	[−]*ddd.dddddd*
`g`	Gleitpunktzahl im Format `e` oder `f`, je nachdem, welches kürzer ist; nicht signifikante Nullen werden unterdrückt.
`o`	Oktalzahl ohne Vorzeichen.
`s`	(string) Zeichenkette.
`x`	Hex-Zahl ohne Vorzeichen.
`%`	Ausgabe des Zeichens `%`; es findet keine Argument-Umwandlung statt.

Nachfolgend finden Sie einige Beispiele für `printf`- Anweisungen mit der jeweils erzeugten Ausgabe:

```
printf "%d", 99/2                    49
printf "%e", 99/2                    4.950000e+01
printf "%f", 99/2                    49.500000
printf "%6.2f", 99/2                 49.50
printf "%g", 99/2                    49.5
printf "%o", 99                      143
printf "%06o", 99                    000143
printf "%x", 99                      63
printf "|%s|", "January"             |January|
printf "|%10s|", "January"           |   January|
printf "|%-10s|", "January"          |January   |
printf "|%.3s|", "January"           |Jan|
printf "|%10.3s|", "January"        |       Jan|
printf "|%-10.3s|", "January"       |Jan       |
printf "%%"                          %
```

Das Standard-Ausgabeformat für Zahlen ist `%.6g`, es sei denn, der Variablen `OFMT` wird ein neuer Wert zugeordnet. `OFMT` steuert zusätzlich die Umwandlung von numerischen Werten in Zeichenketten für eine Konkatenation sowie die Erzeugung von Feldindizes.

Ausgabe in Dateien

Die Ausgabe kann nicht nur über die Standard-Ausgabe erfolgen, sondern auch in Dateien; dafür können Sie die Umleitungs-Operatoren > und >> benutzen. Wird das folgende Programm beispielsweise mit der Eingabedatei `countries` aufgerufen, so werden alle Zeilen, deren drittes Feld (die Bevölkerungszahl) den Wert 100 übersteigt, in die Datei `bigpop` geschrieben; die übrigen Zeilen werden zur Datei `smallpop` umgeleitet:

```
$3 > 100    { print $1, $3 >"bigpop" }
$3 <= 100   { print $1, $3 >"smallpop" }
```

Beachten Sie, daß die Dateinamen in Anführungszeichen (") eingeschlossen werden müssen; andernfalls werden `bigpop` und `smallpop` lediglich wie nicht initialisierte Variablen behandelt. Sind die Namen der Ausgabedateien das Ergebnis eines Ausdrucks, so müssen sie zusätzlich in Klammern eingeschlossen sein, da der Umleitungs-Operator > eine höhere Priorität hat als die Konkatenation:

```
$4 ~ /North America/ { print $1 > ("tmp"
FILENAME) }
```

Ohne die Klammern wäre keine Konkatenation von `tmp` und `FILENAME` möglich.

Hinweis: Dateien werden während eines `awk`- Programmlaufs nur einmal eröffnet. Wird eine Datei über den Operator > eröffnet, so wird ihr ursprünglicher Inhalt überschrieben. Bei der Eröffnung der Datei mit dem Operator >> dagegen bleibt ihr Inhalt erhalten; die Ausgabe wird dann am Dateiende eingefügt. Bei einer eröffneten Datei haben die beiden Operatoren dieselbe Auswirkung.

Ausgabe an Pipes

Die Ausgabe können Sie nicht nur in eine Datei, sondern auch zu einer Pipe umleiten, wenn nach dem Pipe-Symbol ein Kommando steht. Die Anweisung

```
print | "kommando_zeile"
```

übergibt die Ausgabe der `print`- Funktion über eine Pipe an *kommando_zeile.*

Bei *kommando_zeile* und den Dateinamen kann es sich, abweichend von den hier gezeigten Anweisungen, nicht nur um in Anführungszeichen (") eingeschlossene Zeichenketten handeln, sondern auch um Variablen und die Rückkehr-Werte von Funktionen.

Angenommen, Sie benötigen einen Überblick über die Kontinente mit ihren jeweiligen Bevölkerungszahlen, der nach den Anfangsbuchstaben der Kontinente sortiert ist. Mit dem folgenden `awk`- Programm werden die Bevölkerungszahlen im dritten Feld für jeden der Kontinent-Namen im vierten Feld in einem Feld namens `pop` aufsummiert. Die Kontinente werden mit ihren Bevölkerungszahlen ausgegeben, wobei die Ausgabe an das Kommando `sort` übergeben wird:

```
BEGIN    { FS = "\t" }
         { pop[$4] += $3 }
END      { for (c in pop)
               print c ":" pop[c] | "sort"
}
```

Mit der Eingabedatei `countries` erzeugt dieses Programm die folgende Ausgabe:

```
Africa:37
Asia:1765
Australia:14
North America:243
South America:142
```

In allen `print`- Anweisungen mit einer Umleitung der Ausgabe werden die Dateien bzw. Pipes über ihre Namen angegeben (d.h., die obengenannte Pipe hat den Namen `sort`); sie werden jedoch während des gesamten Programmlaufs nur einmal angelegt und eröffnet. So ist im letzten Beispiel für alle `c` in `pop` nur eine einzige sort-Pipe eröffnet worden.

In einem `awk` -Programm kann nur eine begrenzte Anzahl von Dateien gleichzeitig eröffnet sein. Die Anweisung `close` (*datei*) schließt eine Datei bzw. Pipe; *datei* ist dabei die Zeichenkette, mit der die Datei ursprünglich eröffnet worden ist Beispiel:

```
close("sort")
```

Beim Eröffnen oder Schließen einer Datei werden unterschiedliche Zeichenketten als unterschiedliche Kommandos behandelt.

Die Eingabe

Die Übergabe der Eingabe an ein `awk`- Programm erfolgt in der Regel durch die Angabe der Datei(en) im Programmaufruf. Nach dieser Methode wird in diesem Kapitel verfahren; allerdings gibt es noch eine Reihe weiterer Möglichkeiten. Um die verschiedenen Übergabe-Möglichkeiten geht es in diesem Abschnitt.

Dateien und Pipes

Die Übergabe der Eingabe an ein `awk`- Programm kann durchgeführt werden, indem die Eingabedaten in eine Datei (z.B. `awkdata`) eingebracht werden und dann die Datei abgearbeitet wird:

```
awk 'programm' awkdata
```

Ohne die Angabe von Dateinamen liest `awk` seine Standard-Eingabe (siehe Abschnitt "Aufrufen von awk " in diesem Kapitel); bei einer zweiten sehr häufig benutzten Möglichkeit übergibt ein anderes Programm seine Ausgabe über eine Pipe an `awk`. So kann auch `egrep`(1) zur Suche nach Zeilen benutzt werden, in denen ein bestimmter regulärer Ausdruck enthalten ist (im übrigen mit einer höheren Durchführungsgeschwindigkeit, da dies seine einzige Funktion ist). Somit könnte die Pipe

```
egrep 'Asia' countries | awk '...'
```

`egrep` zur schnellen Auswahl der Zeilen mit der Zeichenkette `Asia` benutzt werden, die dann zur Weiterverarbeitung an das `awk`- Programm übergeben werden.

Trennzeichen für die Eingabe

Nach der Standardeinstellung von `FS`, des Feldtrennzeichens für die Eingabe, werden die Eingabefelder durch Leer- oder Tabulatorzeichen voneinander getrennt; da führende Leerzeichen unterdrückt werden, ist das erste Feld der folgenden Zeilen identisch:

```
   field1   field2
  field1
field1
```

Dies gilt jedoch nicht, wenn als Feldtrennzeichen ein Tabulatorzeichen benutzt wird.

Das Feldtrennzeichen kann auf einen beliebigen regulären Ausdruck gesetzt werden, indem der vordefinierten Variablen `FS` ein Wert zugeordnet wird. So wird durch die Anweisung

```
BEGIN { FS = ",[ \t]*|([ \t]+" }
```

jede Zeichenkette zu einem Feldtrennzeichen, die entweder aus einem Komma in Kombination mit Leer- oder Tabulatorzeichen besteht, oder aus aufeinanderfolgenden Leer- oder Tabulatorzeichen ohne Komma. `FS` kann auch in der Kommandozeile mit dem Argument `-F` eingestellt werden:

```
awk -F'(,[
\t]*)|([ \t]+)' '. . .'
```

hat dieselbe Auswirkung wie das letzte Beispiel. Reguläre Ausdrücke als Feldtrennzeichen passen zur äußerst linken und längsten der passenden Teilzeichenketten (wie in `sub()`), jedoch nicht zu leeren Zeichenketten.

Mehrzeilige Sätze

Die einzelnen Sätze werden im Normalfall durch Zeilenendezeichen voneinander getrennt; somit ist jede Zeile ein Satz. Dies kann, wenn auch nur in beschränktem Maße, geändert werden. Ist `RS` die vordefinierte Variable für das Satztrennzeichen wie in

```
BEGIN   { RS = "" }
```

auf eine leere Zeichenkette gesetzt, so können die Eingabesätze auch mehrere Zeilen lang sein; die Sätze werden dann durch mehrere Leerzeilen voneinander getrennt. Sollen mehrzeilige Sätze verarbeitet werden, so wird häufig die Anweisung

```
BEGIN   { RS = ""; FS = "\n" }
```

benutzt, mit der das Satztrennzeichen auf eine Leerzeile und das Feldtrennzeichen auf das Zeilenendezeichen gesetzt wird. Jede Zeile ist dann ein Feld. Allerdings kann ein Satz nicht beliebig lang sein. Die maximale Länge liegt im Normalfall bei 2500 Zeichen. Die Abschnitte "Die Funktion `getline`" sowie "awk und die Shell" in diesem Kapitel enthalten weitere Beispiele zur Verarbeitung von mehrzeiligen Sätzen.

Die Funktion `getline`

Die Möglichkeiten von `awk` zur automatischen Aufteilung seiner Eingabe in mehrzeilige Sätze reichen für manche Funktionen nicht aus. Sind die Sätze beispielsweise nicht durch Leerzeilen voneinander getrennt, sondern durch komplexere Satztrennzeichen, reicht es nicht aus, nur `RS` auf die leere Zeichenkette zu setzen. In einem derartigen Fall muß die Aufteilung jedes Satzes in Felder durch das Programm vorgenommen werden. Im folgenden werden dafür einige Anregungen gegeben.

Mit der Funktion `getline` kann die Eingabe entweder aus der aktuellen Eingabe, aus einer Datei oder einer Pipe eingelesen werden; in den letzteren beiden Fällen wird sie wie bei `printf` umgeleitet. Beim Aufrufen der Funktion `getline` allein wird der nächste Eingabesatz eingelesen und normal in Felder aufgeteilt. Gleichzeitig werden die Variablen `NF`, `NR`, und `FNR` neu gesetzt. `getline` gibt den Wert 1 aus, wenn ein Satz vorhanden ist, 0 , wenn das Ende der Eingabe erreicht ist und −1 beim Auftreten eines Fehlers (z.B. wenn eine Datei nicht eröffnet werden kann).

Angenommen, Ihre Eingabedaten bestehen aus mehrzeiligen Sätzen, deren erste Zeile jeweils mit `START` beginnt, während die letzte Zeile jeweils mit `STOP` beginnt. Das folgende `awk`- Programm verarbeitet diese mehrzeiligen Sätze Zeile für Zeile und bringt die Zeilen der Sätze in aufeinanderfolgende Einträge des Felds

```
f[1] f[2] ... f[nf]
```

ein. Sobald die Zeile mit dem Eintrag `STOP` erreicht ist, kann der Satz aufgrund der Daten im Feld `f` verarbeitet werden:

```
/^START/ {
        f[nf=1] = $0
        while (getline && $0 !~ /^STOP/)
            f[++nf] = $0
        # Verarbeitung der Daten in f[1]...f[nf]
        ...
}
```

Da && seine Operanden entsprechend ihrer Anordnung von links nach rechts auswertet, endet der Programmteil, sobald eine der Bedingungen erfüllt ist.

Dieselbe Aufgabe kann mit dem folgenden Programm durchgeführt werden:

```
/^START/ && nf==0          { f[nf=1] = $0 }
nf > 1                     { f[++nf] = $0 }
/^STOP/                    { # Verarbeitung der Daten in f[1]...f[nf]
                             ...
                             nf = 0
}
```

Die Anweisung

```
getline x
```

liest den nächsten Datensatz in die Variable x ein. Es wird keine Aufteilung vorgenommen; NF wird nicht neu gesetzt. Die Anweisung

```
getline <"file"
```

liest ihre Eingabe aus der Datei file ein. An der Einstellung von NR oder FNR ändert sich nichts; allerdings findet eine Aufteilung in Felder statt, und die Variable NF wird neu gesetzt. Die Anweisung

```
getline x <"file"
```

liest den nächsten Satz in file in die Variable x ab; es findet weder eine Aufteilung noch eine Änderung von NF, NR oder FNR statt.

Wird als Dateiname ein Ausdruck angegeben, so sollte er für die Auswertung in Klammern eingeschlossen werden, da < gegenüber der Konkatenation die Priorität hat:

```
while ( getline x < (ARGV[1] ARGV[2]) ) {  ... }
```

Ohne Klammern würde in einer Anweisung wie

```
getline x < "tmp" FILENAME
```

die Variable x so gesetzt, daß statt tmp *<wert von FILENAME>* die Datei tmp eingelesen wird. Ein weiterer Nachteil dieser Variante der getline-Anweisung

besteht darin, daß Sie mit einer Anweisung wie

```
while ( getline x < file ) { ... }
```

eine Endlosschleife erzeugen, wenn die Datei nicht eingelesen werden kann, da `getline` beim Auftreten eines Fehlers den Rückkehrwert -1 und nicht 0 ausgibt. Eine derartige Überprüfung wird besser folgendermaßen realisiert:

```
while ( getline x < file > 0) { ... }
```

An `getline` können Sie auch die Ausgabe eines anderen Kommandos direkt über eine Pipe übergeben. So wird mit der Anweisung

```
while ("who" | getline)
     n++
```

das Kommando `who` aufgerufen und seine Ausgabe über eine Pipe an `getline` übergeben. Bei jedem Durchgang der `while`- Schleife wird eine weitere Zeile eingelesen und die Variable `n` um den Wert 1 hochgezählt, so daß `n` nach Beendigung der `while`- Schleife die Anzahl der Benutzer enthält. Ähnlich wird bei der Anweisung

```
"date" | getline d
```

die Ausgabe des Kommandos `date` an die Variable `d` übergeben, so daß `d` auf das aktuelle Datum gesetzt wird. Tabelle 10-7 enthält einen Überblick über die Funktion `getline`.

Tabelle 10-7: Die Funktion `getline`

Format	Einstellung von
`getline`	`$0, NF, NR, FNR`
`getline` *var*	*var*, `NR, FNR`
`getline <`*datei*	`$0, NF`
`getline` *var* `<`*datei*	*var*
kommando `| getline`	`$0, NF`
kommando `| getline` *var*	*var*

Die Argumente in der Aufrufzeile

Die Argumente in der Aufrufzeile werden einem `awk`- Programm folgendermaßen zur Verfügung gestellt: Das Feld `ARGV` enthält die Elemente `ARGV[0], ... ARGV[ARGC-1]` der `ARGC` enthält wie in der C-Sprache die Anzahl der Elemente im Feld `ARGV`. `ARGV[0]` ist der Name des Programms (im Normalfall `awk`); hinzu kommen die übrigen Argumente in der Aufrufzeile mit Ausnahme von *programm_datei* und Optionen. Die folgende Kommandozeile ruft ein `awk`- Programm auf, durch das die Argumente hinter dem Programmnamen ausgegeben werden:

```
awk '
BEGIN {
   for (i = 1; i < ARGC; i++)
        printf "%s ", ARGV[i]
   printf "\n"
}' $*
```

Die Argumente können geändert oder ergänzt werden; `ARGC` kann geändert werden. Am Ende jeder Eingabedatei interpretiert `awk` das nächste Element ungleich Null in `ARGV` (bis zum aktuellen Wert von `ARGC-1`) als Namen der nächsten Eingabedatei.

Beim folgenden Format ist das Argument ausnahmsweise kein Dateiname:

var=wert

In diesem Fall wurde die Variable *var* wie bei einer Zuweisung auf *wert* gesetzt. Ein derartiges Argument wird nicht wie ein Dateiname behandelt. Ist *wert* eine Zeichenkette, so sind keine Anführungszeichen (") erforderlich.

awk mit anderen Kommandos und der Shell

Die Leistungsfähigkeit von awk läßt sich am besten ausnutzen, wenn es zusammen mit anderen Programmen verwendet wird. In diesem Kapitel werden hierfür einige Möglichkeiten beschrieben.

Die Funktion system

Die vordefinierte Funktion system(*kommando_zeile*) führt das Kommando *kommando_zeile* aus; hierbei kann es sich um eine Zeichenkette handeln, die beispielsweise durch die vordefinierte Funktion sprintf formatiert worden ist. Der Funktionswert von system ist der Rückkehr-Status des ausgegebenen Kommandos.

So ruft beispielsweise das Programm

```
$1 == "#include"  { gsub(/[<>"]/, "", $2);
  system("cat " $2) }
```

das Kommando cat zur Ausgabe der Datei im zweiten Feld aller Eingabesätze auf, deren erstes Feld #include enthält; <, >, oder " werden zuvor entfernt.

awk und die Shell

In den meisten der bisherigen Beispiele war das awk- Programm in einer Datei enthalten, die entweder mit der Option -f geladen worden ist oder wie im folgenden Beispiel, in Hochkommata (') eingeschlossen, im Programmaufruf enthalten war:

```
awk '{ print $1 }' . . .
```

Wegen der Ähnlichkeiten der awk- und Shell-Metazeichen (z.B. $ und ") muß das awk- Programm in Hochkommata (') eingeschlossen werden, damit die Shell das ganze Programm unverändert an den awk- Interpreter übergibt.

Angenommen, mit einem Kommando addr soll eine Datei namens addresslist nach Namen, Adressen und Telephonnummern durchsucht werden. Bei den typischen Einträgen von addresslist handelt es sich dabei um mehrzeilige Datensätze wie

```
G. R. Emlin
600 Mountain Avenue
Murray Hill, NJ 07974
201-555-1234
```

Die Sätze werden durch eine einzelne Leerzeile voneinander getrennt.

Die Adreßliste soll mit Kommandos wie

```
addr Emlin
```

durchsucht werden. Dies ist leicht möglich mit einem Programm wie

```
awk '
BEGIN { RS = "" }
/Emlin/
' addresslist
```

Wie sieht das Programm aber aus, wenn bei jedem Programmlauf mit einem anderen Suchmuster gearbeitet werden muß?

Zur Lösung dieses Problems gibt es verschiedene Möglichkeiten. So können Sie eine Datei `addr` folgenden Inhalts erstellen:

```
awk '
BEGIN { RS = "" }
/'$1'/
' addresslist
```

Der entscheidende Punkt sind in diesem Fall die Hochkommata (`'`). Trotz der zwei Hochkomma-Paare ist das `awk`- Programm nur ein einziges Argument (Hochkommata können nicht ineinandergeschachtelt werden). Da die Variable `$1` sich außerhalb der Hochkommata befindet, kann sie durch die Shell ausgewertet werden; somit ersetzt die Shell die Variable durch das Muster `Emlin`, sobald das Kommando `addr Emlin` aufgerufen wird. Unter UNIX kann `addr` zu einem ausführbaren Kommando gemacht werden, indem seine Zugriffsrechte mit dem folgenden Kommando geändert werden:

```
chmod +x addr
```

Eine weitere Möglichkeit zur Realisierung des Programms `addr` hängt damit zusammen, daß die Shell `$-` Parameter auswertet und ersetzt, wenn sie zwischen Anführungszeichen (`'`) stehen:

```
awk "
BEGIN { RS = \"\" }
/$1/
" addresslist
```

Daher müssen Sie die Anführungszeichen, (") die der Variablen RS zugeordnet sind, durch Gegenschrägstriche (\) entwerten, damit die Shell sie ohne vorherige Interpretation an awk weiterleitet. Da $1 jedoch als Parameter erkannt wird, wird es von der Shell beim Aufrufen des Programms addr *muster* durch das Suchmuster ersetzt.

Eine dritte Möglichkeit zur Realisierung des Kommandos addr besteht darin, den regulären Ausdruck mit Hilfe der Variablen ARGV an ein awk- Programm zu übergeben, das die Adreßliste explizit mit der Funktion getline durchsucht:

```
awk 'b
BEGIN   { RS = ""
          while (getline <"addresslist")
             if ($0 ~ ARGV[1])
                print $0
} ' $*
```

Die Verarbeitung erfolgt vollständig im BEGIN-Teil.

Beachten Sie, daß jeder reguläre Ausdruck an addr übergeben werden kann; die Informationssuche ist sowohl über Teile einer Adresse als auch über die Telephonnummern oder die Namen möglich.

Anwendungsbeispiele

awk ist bereits in den unterschiedlichsten Bereichen zur Anwendung gekommen, angefangen bei den traditionelleren Aufgaben wie Informationsabfragen, Datenmanipulationen und Report-Generierungen bis hin zur Realisierung von Datenbanksystemen sowie verschiedenen Compilern und Assemblern. Die awk-Programme sind in jedem Fall wesentlich kürzer als die entsprechenden Programme, die in einer der Standard- Programmiersprachen wie Pascal oder C realisiert worden sind. In diesem Abschnitt werden weitere Beispiele für awk-Programme vorgestellt.

Generierung von Reports

Eines der wichtigsten Einsatzgebiete von awk ist die Generierung von Reports, in denen Informationen formatiert zusammengefaßt sind. So können Sie anhand der Datei countries einen Report generieren, in dem die Kontinente nach ihrem Namen in alphabetischer Reihenfolge und die Länder auf jedem Kontinent in absteigender Folge nach ihrer Bevölkerungszahl aufgeführt sind:

```
Africa:
        Sudan          19
        Algeria        18

Asia:
        China         866
        India         637
        USSR          262
Australia:
        Australia      14

North America:
        USA           219
        Canada         24

South America:
        Brazil        116
        Argentina      26
```

Wie häufig in der Datenverarbeitung, ist es auch bei der Generierung eines Reports wesentlich einfacher, in mehreren Schritten vorzugehen. Zunächst erstellen Sie eine Liste mit den Spalten Kontinent, Land und Bevölkerungszahl, wobei die Felder durch Doppelpunkt (:) voneinander getrennt sind. Hierfür

erstellen Sie das Programm `triples`, in dem die Bevölkerungszahl eines bestimmten Landes in einem Feld `pop` abgelegt wird, das durch Indizes im Format *kontinent*`:`*land* angesprochen wird. Die print-Anweisung im `END`- Teil des Programms gibt die drei Spalten aus, die dann über eine Pipe an das Programm `sort` übergeben werden:

```
BEGIN { FS = "\t" }
      { pop[$4 ":" $1] += $3 }
  END { for (cc in pop)
          print cc ":" pop[cc] | "sort -t: +0 -1 +2nr" }
```

Die Argumente von `sort` verdienen eine nähere Betrachtung. Das Argument `-t:` bewirkt, daß `sort` den Doppelpunkt `(:)` als sein Feldtrennzeichen benutzt. Durch das Argument `+0 -1` wird das erste Feld zum primären Sortierschlüssel. Im allgemeinen macht *+i −j* die Felder *i+1*, *i+2*, . . ., *j* zum Sortierschlüssel. Fehlt *−j*, so werden die Felder zwischen *i+1* und dem Satz-Ende benutzt. Durch das Argument `+2nr` wird das dritte Feld zum sekundären Sortierschlüssel (Sortieren in fallender Folge). Dabei steht `n` für numerisches Sortieren und `r` (für reverse) für die fallende Sortierfolge. Mit der Eingabedatei `countries` erzeugt dieses Programm die folgende Ausgabe:

```
Africa:Sudan:19
Africa:Algeria:18
Asia:China:866
Asia:India:637
Asia:USSR:262
Australia:Australia:14
North America:USA:219
North America:Canada:24
South America:Brazil:116
South America:Argentina:26
```

Diese Ausgabe ist zwar in der richtigen Reihenfolge sortiert, hat aber das falsche Format. Um die Ausgabe in das gewünschte Format zu bringen, übergeben Sie es an ein zweites `awk`- Programm namens `format`:

```
BEGIN   { FS = ":" }
{       if ($1 != prev) {
            print "\n" $1 ":"
            prev = $1
        }
        printf "\t%-10s %6d\n", $2, $3
}
```

Hierbei handelt es sich um ein Formatierprogramm, das einen Ländernamen lediglich beim ersten Auftreten ausgibt und die zugehörigen Zeilen mit den Informationen über den Ländernamen und die Bevölkerungszahl in das gewünschte Format bringt. Mit der Kommandozeile

```
awk -f triples countries | awk -f format
```

können Sie dann den gewünschten Report erzeugen. Dieses Beispiel zeigt, daß komplexe Datentransformations- und Formatierungsaufgaben häufig bereits mit wenigen einfachen `awk`s- und `sort`-Kommandos realisiert werden können.

Weitere Beispiele

Worthäufigkeiten

Das erste der nachfolgenden Beispiele stellt Ihnen das Zählen über assoziative Felder vor. Angenommen, Sie möchten die Häufigkeit jedes Wortes in der Eingabe ermitteln, wobei ein Wort eine ununterbrochene Folge von Zeichen (keine Leer- oder Tabulatorzeichen) ist. Das folgende Programm gibt die Worthäufigkeiten, in fallender Folge sortiert, aus:

```
     { for (w = 1; w <= NF; w++) count[$w]++ }
END  { for (w in count) print count[w],w | "sort -nr" }
```

In der ersten Anweisung wird anhand des Felds `count` die Häufigkeit jedes Worts ermittelt. Nach dem Einlesen der Eingabe übergibt die zweite `for`-Schleife das zugehörige Wort zusammen mit dem ermittelten Wert über eine Pipe an das Kommando `sort`.

Aufsummierung

Angenommen, die beiden Dateien `deposits` und `withdrawals` enthalten jeweils ein Namens- und ein Betrag-Feld. Für jeden Namen soll der Reinertrag (net balance) ermittelt werden, indem die Abzüge (withdrawals) von den Gesamtbeträgen (total deposits) subtrahiert werden. Der Reinertrag kann dann mit dem folgenden Programm errechnet werden:

```
awk '
FILENAME == "deposits"     { balance[$1] += $2 }
FILENAME == "withdrawals"  { balance[$1] -= $2 }
END                        { for (name in balance)
                                 print name, balance[name]
} ' deposits withdrawals
```

In der ersten Anweisung wird im Feld `balance` für jeden Namen in der Datei `deposits` der Gesamtbetrag ermittelt. In der zweiten Anweisung werden die Abzüge von jedem Gesamtbetrag subtrahiert. Gehören zu einem Namen nur Abzüge, so wird für diesen Namen in der zweiten Anweisung ein Eintrag angelegt. Die `END`- Aktion gibt jeden Namen mit zugehörigem Reinertrag aus.

Auswahl von Zufallselementen

Die folgende Funktion gibt hintereinander `k` Zufallselemente aus den ersten `n` Elementen des Felds `A` aus. Dabei ist `k` die Anzahl der Einträge, die noch ausgegeben werden müssen, und `n` die Anzahl der noch zu überprüfenden Elemente. Ob das *i*- te Element ausgegeben wird, wird durch die folgende Testfunktion entschieden: `rand() < k/n`.

```
function choose(A, k, n, i) {
        for (i = 1; n > 0; i++)
                if (rand() < k/n--) {
                        print A[i]
                        k--
                }
        }
}
```

Shell-ähnliche Funktionen

Das folgende awk- Programm entspricht in etwa dem History- Mechanismus der UNIX-Shell. Eine Zeile, die nur = enthält, führt erneut das zuletzt ausgeführte Kommando aus. Eine Zeile, die mit = *kommando* beginnt, führt das letzte Kommando erneut aus, in dessen Aufruf die Zeichenkette *kommando* enthalten war. Andernfalls wird die aktuelle Zeile abgearbeitet.

```
$1 == "=" { if (NF == 1)
              system(x[NR] = x[NR-1])
            else
              for (i = NR-1; i > 0; i—)
                if (x[i] ~ $2) {
                  system(x[NR] = x[i])
                  break
                }
            next }

/./         { system(x[NR] = $0) }
```

Generierung von Formbriefen

Das folgende Programm erzeugt Formbriefe mit Hilfe einer Schablone, die in der Datei form.letter enthalten ist:

```
This is a form letter.
The first field is $1, the second $2, the third $3.
The third is $3, second is $2, and first is $1.
```

Der zu ersetzende Text wird folgendermaßen angegeben:

```
field 1|field 2|field 3
one|two|three
a|b|c
```

Die BEGIN- Aktion speichert die Schablone im Feld template ab; die übrigen Aktionen arbeiten die Eingabedaten ab, wobei mit Hilfe der Funktion gsub die Schablonen-Felder im Format $*n* durch die zugehörigen Datenfelder ersetzt werden.

```
BEGIN {   FS = "|"
                  while (getline <"form.letter")
                line[++n] = $0
}
{       for (i = 1; i <= n; i++) {
                s = line[i]
                for (j = 1; j <= NF; j++)

gsub("\\$"j, $j, s)
                print s
        }
}
```

Bei der Erstellung eines derartigen Programms sollten Sie auf jeden Fall mit einer Kurzversion starten, die Sie nach und nach erweitern; jede Erweiterung sollte auf ihre Funktion überprüft werden, bevor Sie die nächste Erweiterung vornehmen.

awk - Kurzübersicht

Die Kommandozeile

awk		*programm dateinamen*
awk	-f	*programm_datei dateinamen*
awk	-F*s*	setzt das Feldtrennzeichen auf die Zeichenkette *s;*
	-Ft	setzt das Feldtrennzeichen auf das Tabulatorzeichen.

Muster

BEGIN
END
/re_ausdruck/
vgl_ausdruck
muster && *muster*
muster || *muster*
(*muster*)
!*muster*
muster, *muster*

Steuerungsanweisungen

if (*ausdr*) *anweisung* [else *anweisung*]
if (*index* in *feld*)
anweisung [else *anweisung*]
while (*ausdr*) *anweisung*
for (*ausdr; ausdr; ausdr*) *anweisung*
for (*var* in *feld*) *anweisung*
do *anweisung* while (*ausdr*)
break
continue
next
exit [*ausdr*]
return [*ausdr*]

Ein/-Ausgabe

`close (`*dateiname*`)`	Datei schließen
`getline`	Nächsten Eingabesatz in `$0` einlesen; Setzen von `NF, NR, FNR.`
`getline <`*datei*	`Nächsten Eingabesatz aus` *datei* in `$0` einlesen; Setzen von `NF.`
`getline` *var*	Nächsten Eingabesatz in *var* einlesen; Setzen von `NR, FNR.`
`getline` *var* `<`*datei*	Nächsten Eingabesatz von *datei* in *var* einlesen.
`print`	Ausgabe des aktuellen Satzes.
`print` *ausdr_liste*	Ausgabe von Ausdrücken
`print` *ausdr_liste* `>`*datei*	Ausgabe von Ausdrücken nach *datei.*
`printf` *fmt, ausdr_liste*	Formatierte Ausgabe.
`printf` *fmt, ausdr_liste*`>`*datei*	Formatierte Ausgabe nach *datei.*
`system (`*kmdo_zeile*`)`	Durchführung des Kommandos *kmdo_zeile,* Ausgabe des Rückkehrwerts.

Bei den Funktionen `print` und `printf` wird die Ausgabe durch `>>`*datei* am Ende von *datei* eingefügt und durch `|` *kommando* an eine Pipe übergeben. Ähnlich wird die Eingabe mit *kommando* `| getline` über eine Pipe an `getline` übergeben. `getline` gibt beim Dateiende den Wert 0 aus, bei einem Fehler den Wert −1.

Funktionen

`func` *name* `(`*parameter_liste*`) {`*anweisung* `}`
`funktion` *name* `(`*parameter_liste*`) {`*anweisung* `}`
funktions_name `(`*ausdr*`,`*ausdr*`, . . . )`

Zeichenketten-Funktionen

gsub(*r,s,t*)	Die Teilzeichenkette, die in der Zeichenkette *t* zu dem regulären Ausdruck *r* paßt, wird durch die Zeichenkette *s* ersetzt; die Anzahl der durchgeführten Substitutionen wird ausgegeben. Fehlt *t*, wird die Funktion auf $0 angewandt.
index(*s,t*)	Die Anfangsposition der Teilzeichenkette *t* in der Zeichenkette *s* wird ausgegeben; ist sie nicht enthalten, wird der Wert 0 ausgegeben.
length(*s*)	Die Länge der Zeichenkette *s*
match(*s,r*)	Die Position des ersten Zeichens des regulären Ausdrucks *r* in der Zeichenkette *s*; kommt *r* nicht vor, wird der Wert 0 ausgegeben.
split(*s,a,r*)	Die Zeichenkette *s* wird dem regulären Ausdruck *r* entsprechend in Feldelemente aufgeteilt und im Feld *a* abgelegt; die Anzahl der Felder wird ausgegeben. Fehlt *r*, wird das bei FS angegebene Feldtrennzeichen benutzt.
sprintf(*fmt, ausdr_liste*)	*ausdr_liste* wird der Formatangabe *fmt* entsprechend formatiert und ausgegeben.
sub(*r,s,t*)	Wie gsub, nur wird lediglich die erste gefundene Teilzeichenkette ersetzt.
substr(*s,i,n*)	Die *n*-stellige Teilzeichenkette aus *s*, die bei Position *i* beginnt, wird ausgegeben; fehlt *n*, werden die Zeichen bis zum Ende der Zeichenkette *s* ausgegeben.

Arithmetische Funktionen

`atan2(`*y,x*`)`	Der Arcustangens von *y/x* in Rad.
`cos(`*ausdr*`)`	Der Cosinus (Winkel in Rad).
`exp(`*ausdr*`)`	Exponentialfunktion.
`int(`*ausdr*`)`	Der ganzzahlige Anteil eines numerischen Werts.
`log(`*ausdr*`)`	Der natürliche Logarithmus.
`rand()`	Eine Zufallszahl im Bereich 0 bis 1.
`sin(`*ausdr*`)`	Der Sinus (Winkel in Rad).
`sqrt(`*ausdr*`)`	Die Quadratwurzel.
`srand(`*ausdr*`)`	Neuer Anfangswert für den Zufallszahlen-Generator; fehlt *ausdr*, wird ein Wert aus der Uhrzeit berechnet.

Operatoren (in aufsteigender Priorität)

`= += -= *= /= %= ^=`	Zuweisung
`?:`	Bedingung
`||`	Logisches ODER
`&&`	Logisches UND
`~ !~`	Mustervergleichsoperator, Negation
`< <= > >= != ==`	Vergleichsoperatoren
`Leerzeichen`	Konkatenation von Zeichenketten
`+ -`	Addition, Subtraktion
`* / %`	Multiplikation, Division, Modulo
`+ - !`	Einstelliges Plus-Zeichen, Einstelliges Minuszeichen, logische Negation
`^`	Exponent (identisch mit `**`)
`++ --`	Inkrement, Dekrement (Präfix und Postfix)
`$`	Feld

Reguläre Ausdrücke (in aufsteigender Priorität)

c	Ein beliebiges Zeichen *c*, bei dem es sich nicht um ein Metazeichen handelt.
c	Das darstellbare Zeichen *c*.
.	Ein beliebiges Zeichen mit Ausnahme des Zeilenendezeichens.
^	Anfang einer Zeile oder Zeichenkette.
$	Ende einer Zeile oder Zeichenkette.
[*abc...*]	Ein beliebiges Zeichen aus *abc...*.
[^*abc...*]	Ein beliebiges Zeichen mit Ausnahme des Zeilenendezeichens und der Zeichen aus der Gruppe *abc...*.
r1 | *r2*	Entweder *r1* oder *r2*.
r1r2	Konkatenation: f2r1, dann *r2*.
r+	Ein- oder mehrmals der reguläre Ausdruck *r*.
*r**	Null-, ein- oder mehrmals der reguläre Ausdruck *r*.
r?	Null- oder einmal der reguläre *r*.
(*r*)	Klammerung: Zeichenketten, die zu *r* passen.

Vordefinierte Variablen

ARGC	Anzahl der Argumente in der Kommandozeile.
ARGV	Feld, das die Argumente der Kommandozeile enthält (0..ARGC-1).
FILENAME	Name der aktuellen Eingabedatei.
FNR	Laufende Nummer des Satzes in der aktuellen Datei.
FS	Feldtrennzeichen für die Eingabe (Standard: Leerzeichen).
NF	Anzahl der Felder im aktuellen Eingabesatz.
NR	Laufende Nummer des aktuellen Satzes in der gesamten Eingabe.
OFMT	Ausgabeformat für Gleitkommazahlen (Standard: %.6g).
OFS	Feldtrennzeichen für die Ausgabe (Standard: Leerzeichen).
ORS	Satztrennzeichen für die Ausgabe (Standard: Zeilenendezeichen).
RS	Satztrennzeichen für die Eingabe (Standard: Zeilenendezeichen).
RSTART	Anfangsposition des ersten Zeichens, das von der Funktion match() als passend erkannt worden ist; 0, wenn kein Zeichen gefunden worden ist.
RLENGTH	Länge der Zeichenkette, die von der Funktion match() als passend erkannt worden ist; −1, wenn keine Zeichen gefunden worden ist.

SUBSEP Index-Zeichenketten-Trenner für mehrdimensionale Felder; Standard: \034

Einschränkungen

In den verschiedenen Implementierungen von awk gelten bestimmte Einschränkungen. Einige typische Werte sind im folgenden aufgeführt:

100 Felder
2500 Zeichen pro Eingabesatz
2500 Zeichen pro Ausgabesatz
1024 Zeichen pro Einzelfeld
1024 Zeichen pro printf-Zeichenkette
400 Zeichen zwischen zwei Anführungszeichen
400 Zeichen in Zeichen-Klasse
15 eröffnete Dateien
1 Pipe
Die Zahlendarstellung hängt von den Darstellungsmöglichkeiten des lokalen Rechners ab, z.B. 1e−38..1e+38.

Initialisierung, Vergleich und Typumwandlung

Jede Variable und jedes Feld kann eine Zeichenkette oder eine Zahl sein, oder beides gleichzeitig. Wird eine Variable durch die Zuweisung

var = ausdr

eingestellt, so wird ihr Typ an den des Ausdrucks angepaßt (die Zuweisungs-Operatoren sind +=, -= usw). Ein arithmetischer Ausdruck besitzt den numerischen Datentyp, eine Konkatenation den alphanumerischen (Zeichenkette) usw. Handelt es sich bei der Zuweisung um einen einfachen Kopiervorgang wie in

```
v1 = v2 ,
```

so richtet sich der Typ von v1 nach dem von v2.

Bei einem Vergleich von zwei numerischen Operanden wird ein numerischer Vergleich durchgeführt; andernfalls werden die Operanden bei Bedarf in Zeichenketten umgewandelt

und ein alphanumerischer Vergleich durchgeführt. Ein Ausdruck kann über Zusätze wie

```
ausdr + 0
```

in einen numerischen Ausdruck umgewandelt werden und über

```
ausdr ""
```

in einen alphanumerischen Ausdruck (dies entspricht einer Konkatenation mit einer leeren Zeichenkette).

Nicht initialisierten Variablen wird der numerische Wert `0` und der alphanumerische Wert `""` zugeordnet. Wenn `x` also nicht initialisiert ist, so ist die Bedingung

```
if (x) ...
```

nicht erfüllt ("falsch"), während

```
if (!x) ...
if (x == 0) ...
if (x == "") ...
```

sämtlich wahr sind. Die folgende Bedingung dagegen hat das Ergebnis "falsch":

```
if (x == "0") ...
```

Der Typ eines Felds wird nach Möglichkeit nach dem Kontext bestimmt. So ist bei

```
$1++
```

die Variable `$1` auf jeden Fall numerisch, während bei

```
$1 = $1 "," $2
```

sowohl `$1` als auch `$2` alphanumerisch sind. Die Typumwandlung erfolgt nach Bedarf.

In einem Kontext, in dem der Datentyp nicht eindeutig ist (z.B. in

```
if ($1 == $2) ... )
```

wird der Typ jedes Felds anhand der Eingabe bestimmt. Alle Felder sind alphanumerisch; zusätzlich wird jedes Feld, das nur eine einzige Zahl enthält, als numerisch interpretiert.

Felder, die explizit leer sind, haben den alphanumerischen Wert "" und keinen numerischen Wert. Nicht vorhandene Felder (d.h. Felder größer NF) werden ebenfalls nach diesem Verfahren behandelt.

Für Feldelemente, die mit split() erzeugt worden sind, gilt dasselbe wie für Felder.

Eine Variable wird schon dadurch angelegt, daß sie in einem Ausdruck angegeben wird; ihr wird dann wie oben beschrieben der Wert "" zugeordnet. Ist feld[i] aktuell also nicht vorhanden, so wird sie durch eine Anweisung wie

```
if (feld[i] == "") ...
```

angelegt und mit dem Wert "" belegt; die Bedingung bei if wird dadurch also erfüllt. Die spezielle Anweisung

```
if (i in feld) ...
```

ermittelt die Existenz von feld[i] und legt es nicht an, wenn es nicht vorhanden ist.

11 Die elektronische Post

Einleitung 11-1

Austauschen von Nachrichten 11-2

Das Kommando `mail` 11-3
Senden von Nachrichten 11-3
- Unzustellbare Nachrichten 11-4
- Nachrichten an eine bestimmte Person senden 11-5
- Nachrichten an mehrere Personen gleichzeitig senden 11-7

Nachrichten an ferne Systeme senden: Die Kommandos `uname` und `uuname` 11-7
Empfangen von Nachrichten 11-12
- Die Kommandos `vacation` und `notify` 11-16

Das Kommando `mailx` 11-17

Überblick über das Kommando `mailx` 11-18

Kommando-Optionen 11-20

Senden von Nachrichten: Die Tilde-Kommandos 11-21
Editieren einer Nachricht 11-23
Integration von vorhandenem Text in Ihre Nachricht 11-25
- Einlesen einer Datei in Ihre Nachricht 11-26
- Nachricht aus dem Briefkasten in eine Antwort integrieren 11-27

Teile des Nachrichtenkopfes ändern 11-28
Die elektronische Unterschrift 11-29
Aufzeichnen der abgeschickten Nachrichten 11-30
Beenden von `mailx` 11-31
Zusammenfassung 11-32

Empfangen von Nachrichten 11-33
Das Argument `nach_liste` 11-33
Kommandos zum Lesen und Löschen von Post 11-34
- Post lesen 11-34
- Inhalt des elektronischen Briefkastens 11-36
- Wechsel zu anderen Briefkastendateien 11-37
- Post löschen 11-38

Post abspeichern 11-39
Beantworten von Post 11-40
Beenden von `mailx` 11-41
`mailx`-Kommandoübersicht 11-41

Die Datei `.mailrc` 11-42

Einleitung

Unter UNIX stehen Ihnen eine Reihe von Kommandos zur Verfügung, die Ihnen die Kommunikation mit anderen UNIX-Benutzern ermöglichen. Diese Kommandos bieten verschiedene Möglichkeiten wie die Übermittlung von Nachrichten an andere Benutzer bzw. den Empfang von Nachrichten von anderen Benutzern (die an Ihrem oder einem anderen UNIX-System arbeiten), außerdem den Austausch von Dateien sowie die Kommunikation mit anderen UNIX-Rechnern, die an Ihr Netzwerk angeschlossen sind. Über ein Netzwerk kann der Benutzer an einem System Nachrichten und Dateien mit einem anderen Rechner austauschen und Kommandos auf fernen Rechnern ausführen.

In diesem Kapitel geht es um die Kommandos, mit denen Sie diese Möglichkeiten nutzen können:

Kommandos zum Austauschen von Nachrichten:
`mail`, `mailx`, `uname` und `uuname`

Informationen zur Übermittlung von Dateien und zu Netzwerken allgemein finden Sie in Kapitel 12.

Austauschen von Nachrichten

Zur Übermittlung von Nachrichten gibt es die Kommandos `mail` und `mailx`. Diese Kommandos setzen Ihre Nachricht in eine Datei, die dem Empfänger gehört. Der Empfänger erhält dann bei bzw. nach seiner Anmeldung die folgende Meldung: `you have mail` (Sie haben Post). Der Empfänger kann nun Ihre Nachricht mit `mail` oder `mailx` lesen und gegebenenfalls beantworten.

`mail` und `mailx` haben im wesentlichen dieselbe Funktion, doch bietet `mailx` einen im Vergleich zu `mail` erweiterten Funktionsumfang:

- Bearbeitung der ein- und abgehenden Nachrichten mit verschiedenen Editoren (`ed` oder `vi`).
- Verschiedene Optionen zur Sicherung von Dateien.
- Kommandos, mit denen Nachrichten beantwortet und Kopien (sowohl der ein- als auch der abgehenden Nachrichten) an andere Benutzer geschickt werden können.

Mit den Kommandos `mail` und `mailx` können auch kurze Dateien mit Memos, Berichten usw. abgeschickt werden. Für die Übermittlung von Dateien, die länger als eine Seite sind, sollten Sie jedoch ein spezielles Dateiübertragungs-Kommando wie `uuto` oder `uucp` benutzen (eine Beschreibung dieser Kommandos finden Sie in Kapitel 12 im Abschnitt "Übermittlung von Dateien").

Das Kommando `mail`

In diesem Abschnitt wird das Kommando `mail` vorgestellt. Im einzelnen informiert er Sie darüber, wie Sie Nachrichten an ein oder mehrere Personen am lokalen (also Ihrem) oder einem fernen System senden und mit eingehenden Nachrichten umgehen.

Senden von Nachrichten

Die Kommandozeile, mit der Sie eine Nachricht abschicken können, hat folgendes grundlegendes Format:

`mail` *empfänger*<CR>

wobei *empfänger* der UNIX-Benutzername des Empfängers ist. *empfänger* kann sein:

- Ein Benutzername, wenn der Empfänger an Ihrem System arbeitet (z.B. `bob`)
- Eine Kombination aus System- und Benutzernamen, wenn der Empfänger an einem anderen UNIX-System arbeitet, das mit Ihrem System Daten austauschen kann (z.B. `sys2!bob`).

Vorerst wollen wir davon ausgehen, daß sich der Empfänger der Nachricht auf Ihrem lokalen System befindet (die Übermittlung von Nachrichten an die Benutzer eines fernen Systems wird an einer späteren Stelle dieses Kapitels behandelt). Zum Aufrufen des Kommandos geben Sie bei der Eingabeaufforderung `mail` ein, dann den Benutzernamen des Empfängers und betätigen abschließend die Taste `RETURN`. Nun können Sie beginnen, die Nachricht in der nächsten Zeile einzugeben. Nachdem Sie die Nachricht vollständig eingegeben haben, schicken Sie sie ab, indem Sie am Anfang einer neuen Zeile einen Punkt (`.`) oder das Steuerzeichen <cntrl-d> eingeben.

Das folgende Beispiel zeigt, wie sich diese Prozedur auf dem Bildschirm darstellt:

```
$ mail phyllis<CR>
My meeting with Smith's<CR>
group tomorrow has been moved<CR>
up to 3:00 so I won't be able to<CR>
see you then.  Could we meet<CR>
in the morning instead?<CR>
.<CR>
$
```

Die UNIX-Eingabeaufforderung in letzten Zeile zeigt Ihnen, daß Ihre Nachricht in die Nachrichten-Warteschlange gesetzt und abgeschickt worden ist.

Unzustellbare Nachrichten

Wenn Sie bei der Eingabe des Benutzernamens des Empfängers einen Fehler machen, kann das Kommando `mail` Ihre Nachricht nicht zustellen. Stattdessen werden Sie in einer zweizeiligen Meldung darüber informiert, daß das Kommando nicht ausgeführt werden konnte und Ihre Nachricht daher zurückgegeben werden muß. Aus der Meldung geht außer dem System- und Benutzernamen von Sender und Empfänger auch der Grund für den Abbruch des Kommandos hervor.

Angenommen, Sie haben den Benutzernamen `kol` und möchten an den Benutzer mit dem Benutzernamen `chris` eine Nachricht senden. Die Nachricht lautet folgendermaßen: `The meeting has been changed to 2:00`. Sie haben nicht gemerkt, daß Sie aus Versehen den Benutzernamen mit `cris` angegeben haben und versuchen die Nachricht abzuschicken.

```
$ mail cris<CR>
The meeting has been changed to 2:00.
.<CR>
mail: Can't send to cris
mail: Return to kol
you have mail
$
```

Die Meldung you have mail wird von der Shell ausgegeben. Der Wortlaut der Meldungen kann bei den verschiedenen Shells unterschiedlich ausfallen.

Die Post, die Sie in einem solchen Fall von der Shell erhalten, wird in die Datei /var/mail gesetzt. Anhand dieser Datei können Sie den Grund für den Abbruch des Kommandos mail ermitteln oder die Nachricht laden und erneut abschicken, ohne sie noch einmal eingeben zu müssen. Die Datei enthält im einzelnen folgende Angaben:

```
$ mail<CR>
From kol Mon Jan 23 16:00 EST 1989
Date: Mon Jan 23 11:00:01 GMT 1989
Original-Date:  Mon Jan 23 15:59 EST 1989
Not-Delivered-To: marmaduk!cris  due  to 02  Ambiguous Originator/Recipient Name
     ORIGINAL MESSAGE ATTACHED
     (mail: Error # 8 'Invalid recipient')
Content-Length: 77

Content-Type: text
Content-Length: 38
The meeting has been changed to 2:00.

?
```

Informationen darüber, wie Sie diese Nachricht auf dem Bildschirm abrufen und verarbeiten können, finden Sie im Abschnitt "Empfangen von Nachrichten" in diesem Kapitel.

Nachrichten an eine bestimmte Person senden

Der folgende Bildschirm zeigt eine typische Nachricht:

```
$ mail tommy<CR>
Tom,<CR>
There's a meeting of the review committee<CR>
at 3:00 this afternoon.  D.F. wants your<CR>
comments and an idea of how long you think<CR>
the project will take to complete.<CR>
B.K.<CR>
.<CR>
$
```

Wenn Tom sich an seinem Terminal anmeldet (oder bereits angemeldet ist), wird er durch eine Meldung darüber informiert, daß Post für ihn angekommen ist:

```
you have mail
```

Die Vorgehensweise beim Lesen dieser Nachricht wird im Abschnitt "Empfangen von Nachrichten" in diesem Kapitel beschrieben.

Rufen Sie probehalber das Kommando `mail` mit einer Nachricht an sich selbst auf. Geben Sie beim Kommando `mail` Ihren eigenen Benutzernamen ein und richten Sie eine kurze Nachricht an sich selbst. Nach der Eingabe des abschließenden Punkts (.) bzw. des Steuerzeichens <cntrl-d> wird die Nachricht an eine Datei im Verzeichnis `/var/mail` geschickt, deren Name mit Ihrem Benutzernamen übereinstimmt. Sie erhalten daraufhin eine Meldung, daß eine Nachricht für Sie vorliegt.

Die Möglichkeit, an sich selbst eine Nachricht zu senden, kann man auch als Gedächtnisstütze benutzen. Haben Sie beispielsweise den Benutzernamen `bob` und möchten am nächsten Morgen jemanden anrufen, können Sie Ihrem Gedächtnis mit einer Nachricht nachhelfen:

```
$ mail bob<CR>
Call Accounting and find out<CR>
why they haven't returned my 1988 figures!<CR>
.<CR>
$
```

Wenn Sie sich dann am nächsten Morgen anmelden, werden Sie in einer Meldung darüber informiert, daß eine Nachricht für Sie vorliegt.

Nachrichten an mehrere Personen gleichzeitig senden

Um eine Nachricht an mehrere Personen gleichzeitig zu senden, geben Sie deren Benutzernamen in der mail-Kommandozeile ein:

```
$ mail tommy jane wombat dave<CR>
Diamond cutters,<CR>
The game is on for tonight at diamond three.<CR>
Don't forget your gloves!<CR>
Your Manager<CR>
.<CR>
$
```

Nachrichten an ferne Systeme senden: Die Kommandos uname und uuname

Bisher haben sind wir davon ansgegangen, daß die Nachrichten an andere Benutzer des lokalen UNIX-Systems geschickt wurden. In manchen Unternehmen sind jedoch mehrere voneinander unabhängige Rechnersysteme installiert, die sich in verschiedenen Teilen des Gebäudes befinden; in anderen Fällen sind in den verschiedenen Geschäftstellen verschiedene Systeme installiert.

Wenn auf Ihrem System das "Softwarepaket Basic Networking" Utilities installiert ist, können Sie Nachrichten an einen Benutzer eines anderen Systems senden, indem Sie einfach den Namen des Systems, an dem der Empfänger arbeitet, vor dem Benutzernamen des Empfängers in die Kommandozeile eingeben:

```
mail sys2!bob<CR>
```

Beachten Sie, daß der System- und Benutzername des Empfängers durch ein Ausrufezeichen (!) voneinander getrennt sind.

Zum Aufrufen dieses Kommandos benötigen Sie jedoch drei Informationen:

- Den Namen des fernen Systems.
- Informationen darüber, ob Ihr System und das ferne System miteinander kommunizieren können.
- Den Benutzernamen des Empfängers.

Diese Informationen erhalten Sie über die Kommandos `uname` und `uuname`

Versuchen Sie beim Empfänger den Namen des fernen Systems und den Benutzernamen des Empfängers in Erfahrung zu bringen. Falls der Empfänger den Namen seines Systems nicht kennt, kann er ihn über das folgende Kommando ermitteln:

```
uname -n<CR>
```

Das Kommando gibt den Namen des Systems aus:

```
$ uname -n<CR>
dumbo
$
```

Wenn Sie nun den Namen des fernen Systems kennen, können Sie mit dem Kommando `uuname` überprüfen, ob zwischen Ihrem und dem fernen System eine Verbindung besteht. Geben Sie bei der Eingabeaufforderung folgendes ein:

```
uuname<CR>
```

Auf dem Bildschirm wird jetzt eine Liste der fernen Systeme ausgegeben, mit denen Ihr System Daten austauschen kann. Wenn das System des Empfängers auf dieser Liste aufgeführt ist, können Sie ihm mit `mail` Nachrichten senden.

Einfacher geht die Ermittlung des Namens, wenn Sie das Kommando `uuname` über eine Pipe mit dem Kommando `grep` verbinden, um die Ausgabe von `uuname` nach dem angegebenen System zu durchsuchen. Dazu geben Sie folgendes ein:

```
uuname | grep system<CR>
```

(*system* ist hier der Systemname des Empfängers). Wenn `grep` den angegebenen Systemnamen findet, wird er auf dem Bildschirm angezeigt:

```
$ uuname | grep dumbo<CR>
dumbo
$
```

Diese Ausgabe zeigt Ihnen, daß dumbo mit Ihrem System Daten austauschen kann; andernfalls wird nur eine Eingabeaufforderung zurückgegeben.

```
$ uuname | grep dumbo<CR>
$
```

Zum Abschluß der Erläuterung von uname und uuname1 hier noch ein Beispiel: Angenommen, Sie möchten an die Benutzerin sarah auf dem fernen System dumbo eine Nachricht senden. Überprüfen Sie zunächst, ob dumbo mit Ihrem System kommunizieren kann, und schicken Sie dann die Nachricht ab. Auf folgendem Bildschirm sind beide Schritte zu sehen:

```
$ uuname | grep dumbo<CR>
dumbo
$ mail dumbo!sarah<CR>
Sarah,<CR>
The final counts for the writing seminar<CR>
are as follows:<CR>
<CR>
Our department - 18<CR>
Your department - 20<CR>
<CR>
Tom<CR>
.<CR>
$
```

Bild 11-1 und 11-2 enthält einen Überblick über Syntax und Funktion der Kommandosuname und uuname .

Bild 11-1: `uname`**-Kurzübersicht**

Kurzbeschreibung `uname` – Systemnamen anzeigen		
Kommando	*Optionen*	*Argumente*
`uname`	`-n` und andere*	keine
Funktion:	`uname -n` gibt den Namen des Systems aus, auf dem Sie angemeldet sind.	

* Eine vollständige Beschreibung sämtlicher verfügbarer Optionen und ihrer Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `uname`(1).

Bild 11-2: `uuname`**-Kurzübersicht**

Kurzbeschreibung `uuname` – Liste der ans Netzwerk angeschlossenen Systeme anzeigen.		
Kommando	*Optionen*	*Argumente*
`uuname`	keine	keine
Funktion:	`uuname` gibt eine Liste der fernen Systeme aus, die mit Ihrem System kommunizieren können.	

Zur Angabe des Empfängers gibt es außer der oben beschriebenen Adressierungsmöglichkeit ein weiteres Format, das als bereichsbezogene Adressierung bezeichnet wird. Die Adresse wird dann in folgendem Format angegeben:

empfänger@fernes_system

oder

empfänger@fernes_system.bereich_info

Diese beiden Adressen sind mit den folgenden Adressen identisch:

fernes_system!*empfänger*

oder

fernes_system.bereich_info!*empfänger*

Möglicherweise hat Ihr Systemverwalter andere Syntaxregeln für die Angabe der Adresse festgelegt. Vielleicht müssen Sie auf Ihrem System auch nicht überprüfen, ob Ihr System direkt mit dem fernen System kommunizieren kann (erkundigen Sie sich bei Ihrem lokalen Systemverwalter). Falls keine direkte Verbindungsaufnahme mit dem fernen System möglich ist, wird die Nachricht an ein anderes System geschickt, das mit dem gewünschten fernen System verbunden ist.

Bild 11-3 enthält einen Überblick über Syntax und Funktion des Kommandos `mail`.

Bild 11-3: `mail`-Kurzüberblick (Senden von Nachrichten)

Kurzbeschreibung `mail` – Nachricht an einen anderen Benutzer senden.		
Kommando	*Optionen**	*Argumente*
`mail`	keine	[*system_name!*]*empfänger*
Funktion:	Beim Aufrufen von `mail` mit einem oder mehreren Benutzernamen (möglicherweise auch einem Systemnamen) wird die in den nächsten Zeilen enthaltene Nachricht an die angegebenen `empfänger` geschickt.	
Hinweis:	Die Nachricht wird abgeschickt, sobald Sie entweder am Anfang einer neuen Zeile einen Punkt (`.`) eingeben und die Taste `RETURN` betätigen oder das Steuerzeichen <cntrl-d> eingeben.	

* Eine vollständige Beschreibung sämtlicher verfügbarer Optionen und ihrer Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mail`(1).

Empfangen von Nachrichten

Wie bereits erwähnt, können Sie mit dem Kommando `mail` auch Nachrichten, die Sie von anderen Benutzern erhalten haben, auf dem Bildschirm abrufen. Wenn Sie angemeldet sind, während Ihnen jemand eine Nachricht sendet, wird die folgende Meldung auf dem Bildschirm angezeigt:

```
you have mail
```

Dies bedeutet, daß ein oder mehrere Nachrichten für Sie in der Datei

`/var/mail/`*benutzer_name* (Ihrem sogenannten Briefkasten) vorhanden sind. Um diese Nachrichten nun auf Ihrem Bildschirm abzurufen, geben Sie das Kommando `mail` ohne Argumente ein:

```
mail<CR>
```

Die Nachrichten werden hintereinander angezeigt. Die Nachricht, die als letzte eingetroffen ist, wird als erste angezeigt. Eine typische mit `mail` übermittelte Nachricht sieht folgendermaßen aus:

```
$ mail
>From tommy Wed May 21 15:33 CST 1989
Content-Length: 104

Bob,
Looks like the meeting has been canceled.
Do you still want the material for the technical review?
Tom

?
```

Die ersten Zeilen, die Nachrichtenkopf genannt werden, enthalten Informationen über die Nachricht, z.B. den Benutzernamen des Senders, Datum und Uhrzeit der Absendung der Nachricht sowie den Umfang der Nachricht (Anzahl der Zeichen). In den Zeilen zwischen der ersten Leerzeile und der Zeile, die ein Fragezeichen (?) enthält, steht der Nachrichtentext.

Wenn auf Ihrem Bildschirm eine lange Nachricht angezeigt wird, die nicht auf eine Bildschirmseite paßt, können Sie die Ausgabe der Nachricht durch die Eingabe des Steuerzeichens <cntrl-s> anhalten; die Fortsetzung der Ausgabe erreichen Sie dann über das Steuerzeichen <cntrl-q>.

Nach jeder Nachricht zeigt das Kommando `mail` eine Eingabeaufforderung in Form eines Fragezeichens (`?`) an und wartet auf eine Eingabe. Sie haben hier verschiedene Möglichkeiten: Sie können die aktuelle Nachricht im Briefkasten lassen und die nächste Nachricht lesen, die aktuelle Nachricht löschen, oder sie abspeichern, damit Sie später darauf zugreifen können. Eine weitere Möglichkeit ist, alle verfügbaren Optionen abzurufen, indem Sie bei der Eingabeaufforderung `?` des Kommandos `mail` ein Fragezeichen (`?`) eingeben.

Wenn Sie mit der nächsten Nachricht fortfahren möchten, ohne die aktuelle Nachricht zu löschen, betätigen Sie nach dem Fragenzeichen (`?`) die Taste `RETURN`.

```
?<CR>
```

Dadurch bleibt die aktuelle Nachricht in Ihrem Briefkasten, und die nächste Nachricht wird angezeigt. Nachdem alle Nachrichten in Ihrem Briefkasten angezeigt worden sind, wird wieder die Eingabeaufforderung der Shell ausgegeben.

Um eine Nachricht zu löschen, geben Sie nach dem Fragezeichen (`?`) ein `d` ein:

```
? d<CR>
```

Dadurch wird die Nachricht aus Ihrem Briefkasten gelöscht. Wenn dort eine weitere Nachricht vorhanden ist, wird sie als nächste angezeigt.

Um eine Nachricht abzuspeichern, auf die Sie später nochmals zugreifen möchten, geben Sie nach dem Fragezeichen (`?`) ein `s` ein:

```
? s<CR>
```

Die Nachricht wird dadurch standardmäßig in die Datei `mbox` in Ihrem Home-Verzeichnis eingebracht. Um die Nachricht in eine andere Datei zu schreiben,geben Sie den Namen dieser Datei nach dem Kommando `s` ein.

Um z.B. einer Nachricht in einer Datei mit dem Namen `mailsave` (in Ihrem aktuellen Verzeichnis) zu abzuspeichern, geben Sie folgendes ein:

```
? s mailsave<CR>
```

Ist `mailsave` eine bereits vorhandene Datei, wird sie um diese Nachricht erweitert. Ist keine Datei mit diesem Namen vorhanden, wird sie angelegt und die Nachricht in ihr abgespeichert. Mit dem Kommando `ls` können Sie später überprüfen, ob die neue Datei abgelegt worden ist (`ls` gibt den Inhalt Ihres aktuellen Verzeichnisses aus).

Sie können die Nachricht auch in einer Datei in einem anderen Verzeichnis abspeichern. Dazu müssen Sie wie den Pfadnamen im folgenden Beispiel angeben:

```
? s project1/memo<CR>
```

Dieser relative Pfadname steht für die Datei `memo` (in die Ihre Nachricht geschrieben wird), die in einem Unterverzeichnis (`project`) Ihres aktuellen Verzeichnisses enthalten ist. Beim Abspeichern von Nachrichten können sowohl

relative als auch absolute Pfadnamen benutzt werden (Informationen über die Bildung von Pfadnamen finden Sie im Kapitel 3, "Das Dateisystem").

Wenn Sie keine Nachrichten mehr lesen möchten, geben Sie nach dem Fragezeichen das folgende Kommando ein:

```
? q<CR>
```

Alle noch nicht gelesenen Nachrichten verbleiben bis zum nächsten Aufrufen des Kommandos `mail` im Briefkasten.

Die Ausgabe einer Nachricht kann mit der Taste BREAK abgebrochen werden. Das Kommando `mail` beendet dann die Ausgabe, gibt die Eingabeaufforderung (?) aus und wartet auf Ihre Eingabe.

Bild 11-4 enthält einen Überblick über Syntax und Funktion des Kommandos `mail` zum Lesen von Nachrichten.

Bild 11-4: `mail`-Kurzübersicht (Lesen von Nachrichten)

Kurzbeschreibung `mail` – An Ihren Benutzernamen gerichtete Nachrichten lesen		
Kommando	*Optionen*	*Argumente*
`mail`	vorhanden*	keine
Beschreibung:	Ohne Optionen zeigt das Kommando `mail` die in Ihrem Briefkasten (der Systemdatei `/var/mail/`*benutzer_name*) enthaltenen Nachrichten an.	
Hinweis:	Ein Fragezeichen (`?`) am Ende der Nachricht bedeutet, daß eine Eingabe erwartet wird. Eine Übersicht über sämtliche zulässigen Eingaben finden Sie im *Referenzhandbuch für Benutzer*.	

* Eine vollständige Beschreibung sämtlicher verfügbarer Optionen und ihrer Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mail`(1).

Die Kommandos `vacation` und `notify`

Zwei weitere Programme zum Empfangen von Nachrichten sind *notify*(1) und *vacation*(1). Das Kommando *notify* sorgt dafür, daß der Empfänger der Nachricht - sofern er aktuell angemeldet ist - über eine neu eingegangene Nachricht unterrichtet wird. Mit dem Kommando *vacation* kann man für eingehende Nachrichten automatisch eine vordefinierte Antwort zurückschicken und die Nachricht für den späteren Zugriff abspeichern. Weitere Informationen über diese Kommandos finden Sie im *Referenzhandbuch für Benutzer.*

Das Kommando `mailx`

Dieser Abschnitt führt Sie in das Kommando `mailx` ein. Sie finden hier Informationen darüber, wie Sie Ihre `mailx`-Umgebung einrichten, mit dem Kommando `mailx` Nachrichten abschicken und eingegangene Nachrichten bearbeiten können. Der Abschnitt ist in vier Teile untergliedert:

- Überblick über das Kommando `mailx`
- Senden von Nachrichten
- Empfangen von Nachrichten
- Die Datei `.mailrc`

Überblick über das Kommando `mailx`

Das Kommando `mailx`(1) ist eine erweiterte Version des Kommandos `mail`(1). Für das Kommando `mailx` gibt es zahlreiche Optionen zum Senden und Empfangen von Nachrichten, die über den Funktionsumfang des Kommandos `mail`(1) hinausgehen. So können Sie z.B. für einen oder mehrere Benutzer einen Alias-Namen (d.h. eine Abkürzung) angeben. Mit diesem Alias-Namen können Sie Nachrichten an einen anderen Benutzer senden, ohne seinen Benutzernamen zu verwenden, oder eine Nachricht durch die Angabe eines einzigen Worts bzw. Namens an mehrere Benutzer gleichzeitig senden. Eine für Sie eingegangene Nachricht, die Sie mit `mailx` lesen, können Sie u.a. in verschiedenen Dateien abspeichern, editieren, an einen weiteren Benutzer senden oder beantworten. Mit den Umgebungsvariablen von `mailx` können Sie für sich eine Umgebung einrichten, die exakt auf Ihre Anforderungen zugeschnitten ist.

Werden für das Kommando `mailx` ein oder mehrere Benutzernamen als Argumente angegeben, weiß `mailx`, daß Sie eine Nachricht an die angegebenen Benutzer senden möchten. `mailx` fordert Sie jetzt zur Eingabe eines Titels für die Nachricht auf und wartet darauf, daß Sie Ihre Nachricht eingeben oder ein Kommando aufrufen. Im Abschnitt "Senden von Nachrichten" werden u.a. die verschiedenen Möglichkeiten zum Editieren von Nachrichten, zur Integration von anderen Dateien sowie zur Erweiterung der Verteilerlisten beschrieben.

Wird `mailx` ohne Argumente angegeben, sucht `mailx` in der Datei `/var/mail/`*benutzer_name* nach eingegangenen Nachrichten. Ist hier Post für Sie vorhanden, wird eine Liste der einzelnen Nachrichten angezeigt; die Nachrichten können Sie jetzt lesen, abspeichern, löschen oder in eine andere Datei kopieren. Im Abschnitt "Empfangen von Nachrichten" finden Sie hierfür einige Beispiele, außerdem eine Beschreibung der verfügbaren Optionen.

Wenn Sie das Kommando `mailx` auf Ihre Anforderungen zuschneiden möchten, sollten Sie in Ihrem Home-Verzeichnis eine Startdatei mit dem Namen `.mailrc` erstellen. Der Abschnitt "Die Datei `.mailrc`" beschreibt die Variablen, die in der Startdatei verwendet werden können.

`mailx` hat zwei Funktionsmodi: den Eingabe- und den Kommandomodus. Zum Erstellen und Senden von Nachrichten müssen Sie sich im Eingabemodus befinden. Im Kommandomodus können Sie eingegangene Nachrichten lesen. Zur Steuerung der Funktionsweise von `mailx` gibt es die folgenden Möglichkeiten:

- Die Eingabe von Optionen in der Kommandozeile (siehe Eintrag `mailx`(1) im *Referenzhandbuch für Benutzer*).
- Die Eingabe von Kommandos im Eingabemodus (z.B. Eingabe einer Nachricht, die abgeschickt werden soll). Diesen Kommandos ist in jedem Fall das Fluchtsymbol Tilde (~) vorangestellt; daher werden sie Tilde-Kommandos (tilde escapes) genannt (siehe Eintrag `mailx`(1) im *Referenzhandbuch für Benutzer*).
- Die Eingabe von Kommandos im Kommandomodus (z.B. um eingegangene Post zu lesen).
- Die Abspeicherung von Kommandos und Umgebungsvariablen in der Startdatei `$HOME/.mailrc` in Ihrem Home-Verzeichnis.

Die Tilde-Kommandos werden im Abschnitt "Senden von Nachrichten" besprochen. Eine Beschreibung der Kommandos im Kommandomodus finden Sie im Abschnitt "Empfangen von Nachrichten". Die Datei `.mailrc` wird im Abschnitt "Die Datei `.mailrc`" behandelt.

Kommando-Optionen

In diesem Abschnitt werden die Optionen behandelt, die in der `mailx`-Kommandozeile eingegeben werden können.

Das Kommando `mailx` hat die folgende Syntax:

`mailx` [*optionen*] [*name...*]

Die *optionen* steuern die Funktionsweise des Kommandos; *name...* steht für den bzw. die Empfänger.

Den in der Kommandozeile angegebenen Optionen muß unmittelbar ein Bindestrich vorangestellt werden. Alle anderen Angaben auf der Kommandozeile werden vom Kommando `mailx` als *name...* interpretiert, d.h. als Benutzer- bzw. Alias-Name des Empfängers.

Eine wichtige Kommandooption, die auch für das Kommando `mail` zur Verfügung steht, ist `-f` :

`-f` [*dateiname*]: Hiermit können Sie Nachrichten lesen, die in der Datei *dateiname* und nicht in Ihrem Briefkasten enthalten sind.

Da die Nachrichten mit dem Kommando `mailx` in einer beliebigen Datei gespeichert werden können, müssen Sie zum Abrufen der Nachrichten die Option `-f` benutzen. Die Nachrichten werden standardmäßig in der Datei `$HOME/mbox` (Ihrem Briefkasten) abgespeichert; bei der Eingabe von `mailx -f` ohne Angabe eines Briefkastens wird der Inhalt dieser Datei ausgegeben.

Senden von Nachrichten: Die Tilde-Kommandos

Um eine Nachricht an andere UNIX-Benutzer zu senden, geben Sie folgendes Kommando ein:

```
$ mailx empfänger<CR>
Subject:
```

Den Empfänger der Nachricht geben Sie über seinen Benutzernamen an. Das System schaltet in den Eingabemodus um und fordert Sie durch die Eingabeaufforderung `Subject:` zur Eingabe des Nachrichten-Titels auf (auf einem stark ausgelasteten System kann es ein paar Sekunden dauern, bis diese Eingabeaufforderung erscheint). Dies ist die einfachste Möglichkeit zum Aufrufen des Kommandos `mailx`. Sie unterscheidet sich nur wenig vom Aufrufformat des Kommandos `mail`.

Die folgenden Beispiele zeigen Ihnen, wie man Nachrichten editiert, vorhandenen Text in Nachrichten integriert, den Nachrichtenkopf ändert und weitere Funktionen mit dem Kommando `mailx` ausführt. Jedem Beispiel folgt eine Erklärung der jeweils wichtigsten Punkte.

```
$ mailx sms<CR>
Subject:
```

Ob Sie für Ihre Nachricht einen Titel eingeben, bleibt Ihnen überlassen. Wenn Sie keinen Titel eingeben möchten, betätigen Sie einfach die Taste `RETURN`. Daraufhin springt der Cursor in die nächste Zeile, und das Programm wartet auf Ihre Eingabe des Nachrichtentexts.

```
$ mailx sms<CR>
Subject: meeting notice<CR>
We're having a meeting for novice mailx users in<CR>
the auditorium at 9:00 tomorrow.<CR>
Would you be willing to give ademonstration?<CR>
Bob<CR>
~. <CR>
EOT
$
```

An diesem Beispiel sind zwei wichtige Punkte zu beachten:

- Untergliedern Sie Ihre Nachricht, indem Sie die einzelnen Zeilen über die Taste RETURN abschließen. Dies erhöht die Lesbarkeit Ihrer Nachricht für den Empfänger und verhindert das Überlaufen des Zeilenpuffers.
- Mit der Eingabe des Tilde-Zeichens in Kombination mit einem Punkt (~.) oder des Steuerzeichens cntrl-d am Anfang einer Zeile wird die Texteingabe beendet und die Nachricht abgeschickt. Das System antwortet mit der Zeichenfolge EOT (Textende) und einer Eingabeaufforderung.

Im Eingabemodus (der auch in unserem Beispiel aktiv war) stehen Ihnen verschiedene Kommandos zur Verfügung. Diese Kommandos, die am Zeilenanfang eingegeben werden, bestehen aus dem Tilde-Symbol (~), gefolgt von einem Buchstaben. Die Kombination aus Tilde und Buchstaben wird als Tilde- Kommando (tilde escape) bezeichnet (siehe Eintrag mailx(1) im *Referenzhandbuch für Benutzer*). In den folgenden Beispielen werden die meisten Tilde-Kommandos verwendet.

Den Titel Ihrer Nachricht können Sie mit der Option -s (für subject) in die Kommandozeile einfügen. So entspricht die Kommandozeile

```
$ mailx -s "meeting notice" sms<CR>
```

der folgenden Eingabe:

```
$ mailx sms<CR>
Subject: meeting notice<CR>
```

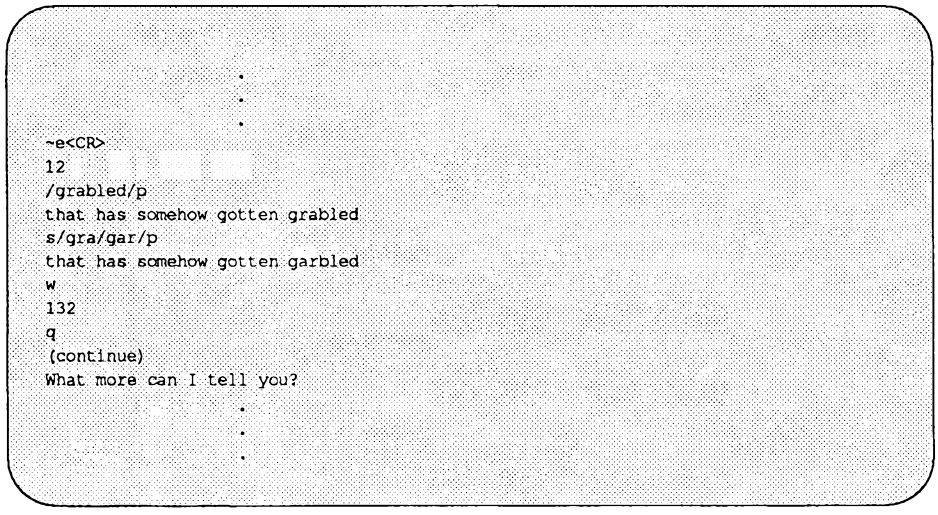
Die Nachricht hat für den Empfänger bei beiden Eingabeformaten dieselbe Subject: (also Titel-)Zeile. Beachten Sie, daß ein Titel, der aus mehr als einem Wort besteht, in Anführungszeichen gesetzt werden muß.

Editieren einer Nachricht

Im Eingabemodus von `mailx` kann durch die Eingabe des Kommandos <Tilde e> am Anfang einer Zeile ein Editor aufgerufen werden. Das folgende Beispiel zeigt eine Anwendungsmöglichkeit für dieses Tilde-Kommando:

```
$ mailx sms<CR>
Subject: Testing my tilde<CR>
When entering the text of a message<CR>
that has somehow gotten grabled<CR>
you may invoke your favorite editor<CR>
by means of a <tilde e> (~e).
                .
                .
                .
```

Die obenstehende Nachricht enthält einen Eingabefehler. Zur Korrektur des Fehlers kann man mit dem Tilde-Kommando `~e` einen Editor aufrufen, in diesem Fall den Standard-Editor `ed`:



```
.
.
.
~e<CR>
12
/grabled/p
that has somehow gotten grabled
s/gra/gar/p
that has somehow gotten garbled
w
132
q
(continue)
What more can I tell you?
.
.
.
```

In diesem Beispiel wurde der Editor `ed` benutzt. Der Editor, der bei der Eingabe des Tilde-Kommandos `~e` aufgerufen wird, hängt vom Inhalt Ihrer Login-Prozedur `.profile` oder der Startdatei `.mailrc` ab. Mit dem Tilde-Kommando `~v` können Sie einen anderen verfügbaren Editor (im Normalfall `vi`) aufrufen.

Durch die Eingabe von `q` können Sie den Editor verlassen; Sie kehren dann in den Eingabemodus zurück und werden von `mailx` zur Fortsetzung der Nachricht aufgefordert. Sie können nun Ihre korrigierte Nachricht überprüfen, indem Sie das Tilde-Kommando `~p` eingeben. Das Tilde- Kommando `~p` zeigt daraufhin die gesamte Nachricht bis zu der Stelle an, an der `~p` eingegeben wurde. Auf diese Weise können Sie den bisher eingegebenen Text Ihrer Nachricht jederzeit überprüfen.

```
                .
                .
                .
~p
Message contains:
To: sms
Subject: Testing my tilde

When entering the text of a message
that has somehow gotten garbled
you may invoke your favorite editor
by means of a <tilde e> (~e).
What more can I tell you?
(continue)
~.
EOT
$
```

Integration von vorhandenem Text in Ihre Nachricht

Beim Kommando `mailx` gibt es vier verschiedene Möglichkeiten, um Text aus einer anderen Quelle in die Nachricht einzufügen, die Sie gerade erstellen:

- Einlesen einer Datei in Ihre Nachricht.
- Einlesen einer empfangenen Nachricht in eine Antwort.
- Integration des Werts einer bestimmten Umgebungsvariablen in die Nachricht.
- Ausführung eines Shell-Kommandos und Integration der Kommandoausgabe in die Nachricht.

Für die ersten zwei (am häufigsten benutzten) Funktionen werden im folgenden Beispiele gezeigt. Informationen über die beiden anderen Funktionen finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mailx`(1).

Einlesen einer Datei in Ihre Nachricht

```
$ mailx sms<CR>
Subject: Work Schedule<CR>
As you can see from the following<CR>
~r letters/file1
"letters/file1"   10/725
we have our work cut out for us.
Please give me your thoughts on this.
- Bob
~.
EOT
$
```

Wie dieses Beispiel zeigt, folgt auf das Tilde-Kommando `~r` der Name der einzufügenden Datei. Das System zeigt den Dateinamen und die Anzahl der darin enthaltenen Zeilen und Zeichen an. Da Sie sich immer noch im Eingabemodus befinden, können Sie mit der Eingabe der Nachricht fortfahren. Wenn die Nachricht beim Empfänger ankommt, enthält sie den Text der Datei `letters/file1` (selbstverständlich können Sie den Inhalt der Nachricht mit dem Tilde-Kommando `~p` überprüfen, bevor Sie sie abschicken).

Nachricht aus dem Briefkasten in eine Antwort integrieren

```
$ mailx<CR>
mailx version 4.0  Type ? for help.
"/var/mail/roberts": 2 messages 1 new
>N   1 abc         Tue May 1  08:09 8/155  Meeting Notice
     2 hqtrs       Mon Apr 30 16:57 4/127  Schedule
? m jones<CR>
Subject: Hq Schedule<CR>
Here is a copy of the schedule from headquarters...<CR>
~f 2<CR>
Interpolating: 2
(continue)
As you can see, the boss will be visiting our district on<CR>
the 14th and 15th.<CR>
- Robert
~.
EOT
?
```

Dieses Beispiel zeigt eine Reihe von wichtigen Punkten:

- Die Anzeige setzt im Kommandomodus ein, in dem Sie Ihre Post lesen und beantworten. Dann schalten Sie mit dem Kommando `m jones` (Funktion: eine Datei an `jones` senden) in den Eingabemodus um.
- Das Tilde-Kommando `~f` wird im Eingabemodus benutzt, um eine Nachricht aus dem Briefkasten zu holen und sie zu einem Bestandteil der abgehenden Nachricht zu machen. Die Zahl `2` hinter dem Tilde-Kommando `~f` bedeutet, daß Nachricht 2 integriert (eingelesen) wird.
- `mailx` zeigt, daß Nachricht 2 eingefügt worden ist, und fordert Sie auf, mit Ihrer Eingabe fortzufahren.
- Nach dem Erstellen und Abschicken der Nachricht kehren Sie in den Kommandomodus zurück (auf dem Bildschirm wird die Eingabeaufforderung in Form eines Fragezeichens (`?`) angezeigt). Sie können jetzt im Kommandomodus weitere Kommandos aufrufen oder `mailx` durch die Eingabe von `q` verlassen.

Ein Tilde-Kommando mit ähnlicher Funktion wie `~f` ist das `~m`, mit dem die eingefügte Nachricht um eine Tabulatorposition eingerückt wird. Sowohl für `~m` als auch für `~f` müssen Sie im Kommandomodus beginnen und dann über die Eingabe eines Kommandos in den Eingabemodus wechseln. Andere Kommandos, die nach diesem Prinzip funktionieren, werden im Abschnitt "Empfangen von Nachrichten" behandelt.

Teile des Nachrichtenkopfes ändern

Der Nachrichtenkopf einer `mailx`- Nachricht besteht aus vier Komponenten:

- Subject: (der Titel der Nachricht).
- To: (die Empfänger-Liste).
- Cc: (carbon copy, offener Verteiler). Eine Liste, in der sämtliche Empfänger der Nachricht aufgeführt sind.
- Bcc: (blind carbon copy, verdeckter Verteiler). Ebenfalls eine Empfänger-Liste, nur erfahren die einzelnen Empfänger nicht die Namen der übrigen Empfänger.

Wenn Sie das Kommando `mailx` zusammen mit einem Benutzer- oder Alias-Namen eingeben, wechseln Sie in den Eingabemodus und werden zur Eingabe des Titels der Nachricht (Zeile Subject: im Nachrichtenkopf) aufgefordert. Nach der Eingabe der Titelzeile und der Betätigung von `RETURN` wartet `mailx` darauf, daß Sie den Text der Nachricht eingeben. Die Informationen im Nachrichtenkopf können Sie im Eingabemodus jederzeit ergänzen oder ändern; dafür stehen Ihnen vier Tilde-Kommandos zur Verfügung: `~h`, `~t`, `~c`, und `~b`.

`~h` Alle Felder des Nachrichtenkopfes werden mit ihrem aktuellen Inhalt angezeigt: Titel, Empfänger, Cc-Liste (offener Verteiler) und Bcc-Liste (verdeckter Verteiler). Sie können die aktuellen Werte ändern, ergänzen oder durch Betätigung von `RETURN` bestätigen.

`~t` Hiermit können Sie die Empfänger-Liste für die Nachricht (To-Liste) um Namen erweitern. Es können entweder Benutzer- oder Alias-Namen angegeben werden.

`~c`	Hiermit können Sie die Cc-Liste für die Nachricht erstellen oder ergänzen. Geben Sie entweder die Benutzer- oder die Alias-Namen der Nachrichtenempfänger an.
`~b`	Hiermit können Sie die Bcc-Liste für die Nachricht erstellen oder ergänzen.

Alle Tilde-Kommandos müssen ganz am Anfang einer Zeile eingegeben werden. Bei den Kommandos `~t`, `~c` oder `~b` werden alle weiteren Angaben in der Zeile als Einträge für die betreffende Liste gewertet. Wird ein Tilde-Kommando ohne weitere Angaben eingegeben, wird die betreffende Kopfzeile angezeigt. Sie können jetzt über die Korrekturtaste Änderungen daran vornehmen. Alle Angaben, die sich in der Zeile mit dem Tilde-Kommando `~h` befinden, werden ignoriert.

Die elektronische Unterschrift

Bei den Umgebungsvariablen `sign` und `Sign` können Sie zwei verschiedene elektronische Unterschriften definieren. Zum Aufrufen der Unterschriften geben Sie das Tilde-Kommando `~a` bzw. `~A` ein. Wenn Sie z.B. dem Tilde-Kommando `~A` den Wert "Supreme Commander" zugeordnet haben, sieht Ihr Bildschirm bei der Eingabe dieses Kommandos folgendermaßen aus:

```
$ mailx -s orders bll<CR>
Be ready to move out at 0400 hours.<CR>
~A<CR>
Supreme Commander
~.<CR>
EOT
$
```

Aufgrund der zwei verfügbaren Kommando-Varianten `~a` und `~A` können Sie zwei verschiedene elektronische Unterschriften einrichten. Allerdings müssen Sie Ihre Nachricht nicht unbedingt mit einer Unterschrift versehen - der Benutzername des Senders erscheint automatisch im Nachrichtenkopf.

Aufzeichnen der abgeschickten Nachrichten

Beim Kommando `mailx` gibt es verschiedene Möglichkeiten, um Kopien der abgeschickten Nachrichten aufzubewahren. Zwei Möglichkeiten, bei denen Sie keine speziellen Umgebungsvariablen einstellen müssen, sind das Tilde-Kommando `~w` sowie die Option `-F` in der Kommandozeile.

Mit dem Tilde-Kommando `~w` und einem Dateinamen wird der Nachrichtentext in die angegebene Datei geschrieben, wenn es diese Datei nicht bereits gibt:

```
$ mailx bdr<CR>
Subject: Saving Copies<CR>
When you want to save a copy of<CR>
the text of a message, use the tilde w.<CR>
~w savemail
"savemail" 2/71
~.
EOT
$
```

Wenn Sie nun den Inhalt von `savemail` abrufen, sieht Ihr Bildschirm folgendermaßen aus:

```
$ cat savemail<CR>
When you want to save a copy of
the text of a message, use the tilde w.
$
```

Der Nachteil dieser Methode liegt auf der Hand: Die Informationen im Nachrichtenkopf werden nicht gespeichert.

Die Option `-F` fügt den Nachrichtentext am Ende einer Datei ein, die nach dem Namen des ersten Empfängers benannt wird. Wenn Sie für den bzw. die Empfänger einen Alias-Namen angegeben haben, wird der Alias-Name zunächst in den bzw. die entsprechenden Benutzernamen umgewandelt; der erste Benutzername wird als Dateiname verwendet. Gibt es bereits eine Datei dieses Namens in Ihrem aktuellen Verzeichnis, wird die Nachricht am Ende dieser Datei eingefügt.

Durch die Verwendung der Option - `F` in der Kommandozeile bleibt der Nachrichtenkopf erhalten. Die Eingabe erfolgt in folgenden Schritten:

```
$ mailx -F bdr<CR>
Subject:  Savings
This method appends this message to a
file in my current directory named bdr.
~.
EOT
$
```

Um das Ergebnis Ihrer Eingabe zu überprüfen, können Sie den Inhalt der Datei `bdr` abrufen.

```
$ cat bdr<CR>
From: kol  Fri May 2  11:14:45  1989
To: bdr
Subject: Savings

This method appends this message to a
file in my current directory named
bdr.
$
```

Beenden von `mailx`

Wenn Sie die Erstellung Ihrer Nachricht beendet haben, können Sie `mailx` mit einem der folgenden drei Kommandos verlassen:

Mit dem Tilde-Kommando ~. (Tilde Punkt) oder dem Steuerzeichen `cntrl-d` (hierbei handelt es sich um die Standard-Methoden zum Verlassen des Eingabemodus). Dadurch wird die Nachricht zusätzlich abgeschickt. Wenn Sie sich vor dem Einschalten des Eingabemodus im Kommandomodus von `mailx` befunden haben, kehren Sie nun in den Kommandomodus zurück (Sie erkennen dies an der Eingabeaufforderung

(?), die nach der Eingabe dieses Kommandos angezeigt wird). Wenn Sie sich dagegen von Anfang an im Eingabemodus befunden haben, kehren Sie jetzt zur Shell zurück (auf dem Bildschirm wird also die Eingabeaufforderung der Shell angezeigt).

`~q` Mit dem Tilde-Kommando `~q`. Dieses Kommando bricht Ihre Texteingabe ab. Hiermit können Sie den Eingabemodus von `mailx` verlassen. Wenn Sie einen Nachrichtentext eingegeben haben, wird er in eine Datei in Ihrem Home- Verzeichnis mit dem Namen `dead.letter` eingefügt.

`~x` Mit dem Tilde-Kommando x (`~x`). Dieses Kommando bricht die Texteingabe ab. Sie verlassen den Eingabemodus von `mailx`, ohne Ihren Text zuvor abzuspeichern.

Zusammenfassung

In den vorausgegangenen Abschnitten wurde ein Teil der Tilde-Kommandos beschrieben, die beim Abschicken von Nachrichten mit dem Kommando `mailx` zur Verfügung stehen (siehe auch Eintrag `mailx`(1) im *Referenzhandbuch für Benutzer.*)

Empfangen von Nachrichten

`mailx` stellt über fünfzig Kommandos zur Verwaltung und Bearbeitung Ihrer Post zur Verfügung. Eine vollständige Liste dieser Kommandos und ihrer Synonyme finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mailx`(1) . In den folgenden Abschnitten werden die wichtigsten Kommandos (und Argumente) beschrieben:

- Das Argument *nach_liste.*
- Kommandos zum Lesen und Löschen von Post.
- Kommandos zum Abspeichern von Post
- Kommandos zum Beantworten von Post.
- Kommandos zum Verlassen von `mailx`.

Das Argument `nach_liste`

Bei vielen `mailx`-Kommandos wird ein Argument in Form von *nach_liste* eingegeben. Mit diesem Argument erhält das Kommando eine Liste der zu verarbeitenden Nachrichten. Wird von Ihnen keine *nach_liste* eingegeben, obwohl sie vom Kommando erwartet wird, so wird das Kommando für die aktuelle Nachricht ausgeführt. *nach_liste* kann in einem der folgenden Formate eingegeben werden:

n	Die Nachricht mit der Nummer *n* soll als aktuelle Nachricht benutzt werden.
^	Die erste der nicht gelöschten Nachrichten.
$	Die letzte der nicht gelöschten Nachrichten.
*	Alle Nachrichten.
n-m	Alle Nachrichten mit Nummern im Bereich n bis m (einschließlich).
benutzer	Alle Nachrichten von *benutzer.*
/zeichenkette	Alle Nachrichten mit *zeichenkette* in der Subject-Zeile (Groß- und Kleinbuchstaben werden als identisch interpretiert).

`:c` Alle Nachrichten vom Typ *c*. c kann sein:

`d` - (delete) Gelöschte Nachrichten
`n` - (new) Neue Nachrichten
`o` - (old) Alte Nachrichten
`r` - (read) Gelesene Nachrichten
`u` - (unread) Ungelesene Nachrichten

Achten Sie darauf, daß das verwendete Argument zu Ihrer aktuellen Situation paßt.

Der folgende Bildschirm enthält dazu zwei Beispiele (das Fragezeichen (`?`) ist die Eingabeaufforderung des Kommandos):

```
? d 1-3              [Nachricht 1, 2 und 3 löschen ]
? s bdr bdrmail  [ Alle Nachrichten von Benutzer bdr in Datei bdrmail abspeichern. ]
?
```

Weitere Beispiele finden Sie in den nächsten drei Unterabschnitten.

Kommandos zum Lesen und Löschen von Post

Wenn in Ihrem Briefkasten eine Nachricht eintrifft, wird auf Ihrem Bildschirm die folgende Meldung dargestellt:

```
you have mail
```

Diese Meldung (Sie haben Post) wird angezeigt, wenn Sie sich anmelden oder aus einer anderen Prozedur zur Shell zurückkehren.

Post lesen

Um Ihre Post zu lesen, geben Sie das Kommando `mailx` mit oder ohne Argumente ein. Nach dem Aufrufen des Kommandos befinden Sie sich im Kommandomodus von `mailx`. Die Bildschirmanzeige sieht jetzt in etwa wie folgt aus:

```
mailx version 4.0  Type ? for help.
"/var/mail/bdr":   3 messages  3 new
> N 1 rbt          Thur Apr 30 14:20   8/190  Review Session
  N 2 admin        Thur Apr 30 15:56   5/84   New printer
  N 3 sms          Fri  May  1 08:39  64/1574 Reorganization
?
```

Die erste Zeile gibt die auf Ihrem System verwendete Version des Kommandos `mailx` an und erinnert Sie daran, daß Sie Hilfe mit einem Fragezeichen (`?`) abrufen können. Die zweite Zeile enthält den Pfadnamen der Datei, auf die sich Angaben in der Anzeige beziehen (der Dateiname entspricht normalerweise Ihrem Benutzernamen), außerdem Angaben über die Gesamtzahl der Nachrichten sowie ihren Status. Der Rest der Anzeige enthält die Kopfzeilen der eingegangenen Nachrichten. Den Nachrichten sind der Reihenfolge ihres Eingangs entsprechend laufende Nummern zugeordnet; die zuletzt eingegangene Nachricht ist als letzte aufgeführt. Links neben der Nummer befindet sich in vielen Fällen eine Statusanzeige: N (new) für neu, U (unread) für nicht gelesen. Das Größer-als-Zeichen (>) zeigt auf die aktuelle Nachricht. Andere Felder in den Kopfzeilen enthalten den Benutzernamen des Absenders der Nachricht, Datum und Uhrzeit ihrer Absendung, die Anzahl der Zeilen und Zeichen in der Nachricht sowie den Inhalt den Nachrichten-Titel. Das letzte Feld kann auch leer sein.

Bei der Anzeige der Kopfzeilen auf Ihrem Bildschirm können Sie die Nachrichten entweder über die Taste `RETURN` abrufen oder ein Kommando eingeben, auf das eine *nach_liste* folgt. Wenn Sie ein Kommando ohne *nach_liste* eingeben, wird das Kommando für die Nachricht ausgeführt, auf die das Größer-als-Zeichen (>) zeigt. Die Betätigung von `RETURN` entspricht der Eingabe des Kommandos `p` (für print) ohne *nach_liste;* in diesem Fall wird die Nachricht angezeigt, auf die das Größer-als-Zeichen (>) zeigt. Um andere Nachrichten zu lesen, müssen Sie das Kommando `p` (für print) oder `t` (für type) eingeben, gefolgt von den den Nummern der Nachrichten. Der folgende Bildschirm enthält weitere Beispiele:

```
? <CR>              [Aktuelle Nachricht anzeigen. ]
? p 2<CR>           [Nachricht Nr. 2 anzeigen.    ]
? p sms<CR>     [ Alle Nachrichten von Benutzer sms anzeigen. ]
```

Das Kommando `t` (für type) ist funktionsgleich mit `p` (für print).

Inhalt des elektronischen Briefkastens

Mit dem Kommando `mailx` können Sie sich die Nachrichten in Ihrem Briefkasten ansehen, um die aktuell wichtigste auszusuchen.

Unmittelbar nach dem Aufrufen des Kommandos `mailx` werden Sie zunächst in einer Übersichtszeile darüber informiert, wieviele Nachrichten vorhanden sind. Darauf folgen die Kopfzeilen von 20 Nachrichten (wenn Sie mit dem Rechner über eine langsame Übertragungsleitung verbunden sind, werden nur 10 Nachrichten angezeigt). Passen nicht die Kopfzeilen sämtlicher Nachrichten auf eine Bildschirmseite, können Sie mit dem Kommando `z` mit der nächsten Bildschirmseite fortfahren. Mit der Eingabe von `z-` wird die vorhergehende Bildschirmseite angezeigt, wenn Sie die Anzeige zuvor bereits in Vorwärtsrichtung gerollt haben. Wenn Sie die Kopfzeilen von bestimmten Nachrichten abrufen möchten, geben Sie beim Kommando `f` (für from) die entsprechende *nach_liste* an.

Der folgende Bildschirm zeigt einige Beispiele für diese Kommandos:

```
? z       [ Nächste Bildschirmseite mit Kopfzeilen anzeigen. ]
? z-      [ Vorhergehende Bildschirmseite anzeigen. ]
? f sms   [ Kopfzeilen sämtlicher Nachrichten des Benutzers sms anzeigen. ]
```

Wechsel zu anderen Briefkastendateien

Wenn Sie das Kommando `mailx` mit

```
$ mailx<CR>
```

aufrufen, wird auf dem Bildschirm der Inhalt der Datei `/var/mail/`*benutzer_name* ist dargestellt (f2benutzer_name ist dabei Ihr eigener Benutzername).

Unter `mailx` können Sie zu anderen Briefkastendateien umschalten und ihren Inhalt mit einem beliebigen `mailx`-Kommando manipulieren (Sie können sogar eine Datei aufrufen, die keine Briefkastendatei ist; allerdings wird dann beim Versuch, `mailx`-Kommandos aufzurufen, die Meldung `No applicable messages` (keine Nachrichten vorhanden) angezeigt). Der Wechsel zu einer anderen Datei erfolgt mit dem Kommando `fi` bzw. `fold` (die beiden Kommandos sind funktionsgleich), gefolgt von *dateiname*. Anstelle des Arguments *dateiname* können folgende Sonderzeichen verwendet werden:

`%` Standard-Briefkasten des Benutzers (`/var/mail/`*benutzer_name*).

`%`*benutzer_name*
: Der Briefkasten des Benutzers *benutzer_name* (wenn Sie die erforderlichen Zugriffsberechtigungen haben).

`#` Die vorher bearbeitete Datei.

`&` Der aktuelle Briefkasten.

Die Bildschirmanzeige kann jetzt beispielsweise folgendermaßen aussehen:

```
$ mailx<CR>
mailx version 4.0  Type ? for help.
"/var/mail/sms":  3 messages 2 new 3 unread
  U 1 jaf          Sat May 9 07:557/137   test25
> N 2 todd         Sat May 9 08:599/377   UNITS requirements
  N 3 has          Sat May 9 11:08 29/1214  access to bailey

? fi &                [ Wechseln Sie zu Ihrem Briefkasten.
]

Held 3 messages in /var/mail/sms
"+mbox":  74 messages 10 unread
   .                  [ Geben Sie Kommandos für Ihren Briefkasten ein. ]
   .
   .
? q<CR>
$
```

Post löschen

Sie können eine Nachricht löschen, indem Sie das Kommando `d` mit dem Argument *nach_liste* eingeben. Ohne die Angabe von *nach_liste* wird die aktuelle Nachricht gelöscht. Endgültig gelöscht wird die Nachricht jedoch erst, wenn Sie die Briefkastendatei verlassen, die Sie gerade bearbeiten. Bis zu diesem Zeitpunkt gibt Ihnen das Kommando `u` (für undelete) die Möglichkeit, das Löschen einer Nachricht rückgängig zu machen. Nach der Eingabe des Kommandos `q` (für quit) oder dem Wechsel zu einer anderen Datei ist Ihre Nachricht jedoch unwiederbringlich gelöscht.

Unter `mailx` können Sie die Kommandos zum Löschen und Anzeigen miteinander verknüpfen. Die Eingabe lautet dann `dp`. Diese Eingabe ist gleichbedeutend mit: "Die gerade gelesene Nachricht löschen und die nächste anzeigen." Der folgende Bildschirm enthält einige Beispiele für das Kommando `d`:

```
? d *     [ Sämtliche Nachrichten löschen, die mir gehören. ]
? d :r    [ Alle gelesenen Nachrichten löschen. ]
? dp      [ Aktuelle Nachricht löschen und die nächste anzeigen. ]
? d 2-5   [ Nachrichten 2 bis 5 löschen. ]
```

Post abspeichern

Alle Nachrichten, die nicht ausdrücklich gelöscht worden sind, werden beim Verlassen von `mailx` abgespeichert. Alle gelesenen Nachrichten werden in eine Datei in Ihrem Home-Verzeichnis mit dem Namen `mbox` eingebracht. Nachrichten, die noch nicht gelesen worden sind, verbleiben in Ihrem Briefkasten (`/var/mail/`*benutzer_name*).

Das Kommando zum Sichern von Nachrichten gibt es in zwei Varianten: Als kleingeschriebenes und als großgeschriebenes `s`. Die großgeschriebene Variante hat folgende Syntax:

`S` [*nach_liste*]

Die Nachrichten, die über das Argument *nach_liste* angegeben worden sind, werden in eine Datei in Ihrem aktuellen Verzeichnis eingebracht. Der Dateiname entspricht dem Benutzernamen des Absenders der ersten Nachricht der Liste.

Die kleingeschriebene Variante hat folgende Syntax :

`s` [*nach_liste*]*dateiname*
oder
`s`

Die im Argument *nach_liste* enthaltenen Nachrichten werden in die Datei *dateiname* eingebracht. Wird keine *nach_liste* angegeben, wird die aktuelle Nachricht abgespeichert. Die Verwendung von Benutzernamen für die Dateinamen kann zu zweideutigen Bezugnahmen führen, die eine Fehlermeldung von `mailx` zur Folge haben können. Fehlt sowohl *nach_liste* als auch *dateiname*, wird die Post in die Datei `mbox` in ihrem Home-Verzeichnis eingebracht.

Beantworten von Post

Das Kommando zum Beantworten von Post gibt es in zwei Varianten: Als kleingeschriebenes und als großgeschriebenes `r`. Mit der großgeschriebenen Variante (`R`) wird Ihre Antwort nur an den Absender der Nachricht geschickt. Mit der kleingeschriebenen Variante (`r`) wird Ihre Antwort nicht nur an den Absender, sondern auch an alle anderen Empfänger geschickt.

Wenn Sie eine Nachricht beantworten, wird der Inhalt der Subject-Zeile (Titel der Nachricht) in die Subject-Zeile der Antwort übernommen. Der folgende Bildschirm zeigt dafür ein Beispiel:

```
$ mailx<CR>
mailx version 4.0  Type ? for help.
"/var/mail/sms":  3 messages 2 new 3 unread
  U 1 jaf         Wed May 9 07:55 7/137   test25
> N 2 todd        Wed May 9 08:59 9/377   UNITS requirements
  N 3 has         Wed May 9 11:08 29/1214  access to bailey

? R 2
To: todd
Subject: Re: UNITS requirements
```

Angenommen, die Nachricht über "UNITS requirements" wurde auch andere Personen geschickt; wenn dann für die Antwort die kleingeschriebene Variante `r` benutzt wird, sieht der Nachrichtenkopf folgendermaßen aus:

```
? r 2
To: todd eg has jcb bdr
Subject: Re:  UNITS requirements
```

Beenden von `mailx`

Zum Beenden von `mailx` gibt es zwei Standardmöglichkeiten: Die Kommandos `q` und `x`. Wird `mailx` über das Kommando `q` verlassen, erhalten Sie einen Überblick darüber, was Sie mit Ihren Nachrichten gemacht haben:

```
? q<CR>
Saved 1 message in /fsl/bdr/mbox
Held 1 message in /var/mail/bdr
$
```

Aus dieser Meldung können Sie ersehen, daß für Benutzer `bdr` mindestens zwei Nachrichten angekommen sind. Eine davon hat er gelesen, die andere hat er entweder nicht gelesen oder mit einem Kommando in die Datei `/var/mail/bdr` eingebracht. Wenn mehr als zwei Nachrichten vorhanden waren, sind die anderen entweder gelöscht oder in andere Dateien geschrieben worden. Über diese Nachrichten gibt `mailx` keine Meldungen aus.

Wenn Sie `mailx` mit dem Kommando `x` beenden, werden fast alle `mailx`-Aktionen ungeschehen gemacht. Die gelesene und gelöschte Post bleibt in Ihrem Briefkasten erhalten. Wenn Sie allerdings Nachrichten in anderen Dateien abgespeichert haben, ist diese Maßnahme bereits ausgeführt worden und kann nicht durch `x` rückgängig gemacht werden.

`mailx`-Kommandoübersicht

In den vorangegangenen Abschnitten wurden die wichtigsten `mailx`-Kommandos beschrieben (eine vollständige Übersicht über die Kommandos finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mailx`(1)). Wenn Sie sich im Kommandomodus von `mailx` befinden und Hilfe benötigen, können Sie diese durch die Eingabe von `?` oder `help` nach der Eingabeaufforderung `?` anfordern; auf Ihrem Bildschirm wird dann eine Liste der `mailx`-Kommandos und eine Erklärung ihrer Funktionen angezeigt.

Die Datei .mailrc

Die Datei .mailrc enthält die Kommandos, die beim Aufrufen von mailx ausgeführt werden sollen.

Möglicherweise ist auf Ihrem System eine systemweite Startdatei (/etc/mail/mailx.rc) vorhanden. In dieser Datei werden vom Systemverwalter die globalen (systemweit gültigen) Variablen eingestellt. Allerdings haben die Variablen, die Sie in Ihrer eigenen .mailrc-Datei festgelegt haben, gegenüber den entsprechenden Variablen der systemweiten Startdatei mailx.rc. die Priorität.

In die Datei .mailrc können die meisten mailx-Kommandos eingebracht werden. Die folgenden Kommandos sind jedoch NICHT zulässig:

! (oder) shell	Sub-Shell vorübergehend aufrufen.
Copy	Nachrichten in *nach_liste* in einer Datei abspeichern, deren Namen vom Absender abgeleitet wird.
edit	Aufrufen des Editors.
visual	Aufrufen von vi.
followup	Beantwortung einer Nachricht.
Followup	Beantwortung einer Nachricht mit Kopie an alle Absender aus *nach_liste*
mail	Wechsel in den Eingabemodus.
reply	Beantwortung einer Nachricht.
Reply	Beantwortung einer Nachricht mit Kopie an alle Absender aus *nach_liste.*

Sie können mit einem beliebigen Editor Ihre eigene .mailrc- Datei erstellen oder die eines anderen Benutzers kopieren. Bild 11-5 zeigt ein Beispiel für eine .mailrc-Datei.

Bild 11-5: Beispiel für eine `.mailrc`-Datei

```
if r
  cd $HOME/mail
endif
set allnet append asksub askcc
autoprint dot
set metoo quiet save showto header
hold keep keepsave
set outfolder
set folder='mail'
set record='outbox'
set crt=24
set EDITOR='/bin/ed'
set sign='Roberts'
set Sign='Jackson Roberts,
Supervisor'
set toplines=10
alias fred  fjs
alias bob  rcm
alias alice  ap
alias donna  dr
alias pat  pat
group robertsgrp  fred bob alice mark pat
group accounts  robertsgrp donna
```

Im Beispiel in Bild 11-5 wurden Kommandos verwendet, die wahrscheinlich sehr nützlich für Sie sind: Die Kommandos `set`, `alias` und `group`.

Mit dem Kommando `set` können Sie Werte für die Umgebungsvariablen festlegen. Die Kommandosyntax sieht folgendermaßen aus:

```
set
set name
set name=zeichenkette
set name=zahl
```

Wird das Kommando `set` ohne Argumente angegeben, wird eine Liste aller definierten Variablen und ihrer Werte angezeigt. Das Argument *name* steht für eine Umgebungsvariable. Nach dem Kommando `set` kann mehr als ein Argument *name* eingegeben werden. Manche Variablen nehmen einen alphanumerischen Wert (Zeichenkette) oder einen numerischen Wert an. Alphanumerische

Variablen werden in Hochkommata eingeschlossen.

Durch die Zuordnung eines Werts an eine Umgebungsvariable (z.B. HOME=*verz*) teilen Sie der Shell mit, wie sie diese Variable interpretieren soll. Allerdings ist das Ergebnis einer derartigen Variablenzuweisung auf Shell-Ebene nicht für andere UNIX- Programme gültig, die auf Umgebungsvariablen Bezug nehmen. Damit auch andere Programmen darauf zugreifen können, muß die Variable exportiert werden. Vielleicht erinnern Sie sich noch an die Einstellung der Variablen TERM (siehe Kapitel 7 oder Kapitel 9), die Sie mit dem folgenden `export`-Kommando exportiert haben:

```
$ TERM=5425
$ export TERM
```

Durch das Exportieren einer Shell-Variablen weisen Sie alle Programme, die auf diese Umgebungsvariable zugreifen, zum Importieren ihres Werts an. Einige der Variablen (wie EDITOR und VISUAL) sind keine speziellen `mailx`- Variablen, können jedoch als globale Umgebungsvariablen angegeben und aus Ihrer Ausführungsumgebung importiert werden. Wird in `.mailrc` ein Wert für eine importierte Variable festgesetzt, werden damit die importierten Werte überschrieben. Das Aufheben einer Variablen-Einstellung kann mit dem Kommando `unset` durchgeführt werden; dieses Kommando gilt jedoch nur für Variablen, die in `.mailrc` festgesetzt worden sind; auf die importierten Variablen hat es keinerlei Auswirkung.

In Ihrer `.mailrc`-Datei können eine Vielzahl von Umgebungsvariablen definiert werden, so daß ihre vollständige Beschreibung den Rahmen dieses Kapitels sprengen würde. Ausführliche Informationen über diese Variablen finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mailx`(1) .

Drei der in Bild 11-5 verwendeten Variablen, die die Verteilung der Nachrichten auf die Verzeichnisse und Dateien steuern, verdienen besondere Beachtung. Hierbei handelt es sich um folgende Variablen: `folder`, `record` und `outfolder`. Über diese drei voneinander abhängigen Variablen können Sie die Verzeichnisse und Dateien angegeben, in denen die Kopien der Nachrichten abgelegt werden sollen.

Die Wertzuweisung für die Variable `folder` hat folgendes Format:

```
set folder=verzeichnis
```

Damit wird das Verzeichnis angegeben, in das die Standard-Briefkastendateien gesetzt werden sollen. Wenn der angegebene Verzeichnisname nicht mit einem

Schrägstrich (/) beginnt, wird angenommen, daß das Verzeichnis sich in der Verzeichnishierarchie unterhalb von `$HOME` befindet. Wenn `folder` eine exportierte Shell-Variable ist, können auch (bei Kommandos, bei denen das Argument *dateiname* erforderlich ist) Dateinamen angegeben werden, deren Name mit einem (Schrägstrich (/) beginnt; der Dateiname wird dann so erweitert, daß die Datei in das bei `folder` angegebene Verzeichnis gesetzt wird.

Die Wertzuweisung für die Variable `record` hat folgendes Format:

```
set record=dateiname
```

Damit wird `mailx` angewiesen, eine Kopie sämtlicher abgeschickten Nachrichten in der angegebenen Datei abzuspeichern. Mit dem Nachrichtentext wird auch der Nachrichtenkopf abgespeichert. Diese Variable ist standardmäßig nicht gesetzt.

Die Variable `outfolder` bewirkt, daß die Datei, in der Sie die Kopien der abgeschickten Nachrichten abspeichern (festgelegt durch die Variable `record=`), in das Verzeichnis `folder` gesetzt wird. Um die Variable zu setzen, müssen Sie sie in einer `set`-Kommandozeile angeben. Der Standardwert ist `nooutfolder`.

Die Kommandos `alias` und `group` sind funktionsgleich. In Bild 11-5 wird das Kommando `alias` verwendet, um einem einzelnen Benutzernamen einen Alias-Namen zuzuweisen. Beim Kommando `group` können Sie mehrere Namen angeben, die mit einem einzigen Pseudonym aufgerufen werden können. Dies ermöglicht die problemlose Unterscheidung der Alias-Namen für Einzelpersonen von denen für Gruppen. Grundsätzlich können beide Kommandos jedoch für genau denselben Zweck verwendet werden. Beachten Sie auch, daß Alias-Namen ineinandergeschachtelt werden können.

In der `.mailrc`-Datei in Bild 11-5 steht der Alias-Name `robertsgrp` für fünf Benutzer. Vier dieser Benutzer wurden über zuvor definierte Alias-Namen angegeben und einer (`mark`) über einen Benutzernamen. Der fünfte Benutzer, `pat`, ist sowohl mit seinem Benutzernamen als auch mit einem Alias-Namen vertreten. Im nächsten Gruppierungs-Kommando werden der Gruppenbezeichnung `accounts` eine weitere Gruppenbezeichnung (`robertsgrp`) sowie ein Alias-Name (`donna`) zugeordnet. Die Gruppen-Bezeichnung `accounts` wird also in sechs Benutzernamen umgesetzt.

Die Datei `.mailrc` in Bild 11-5 enthält das Kommando `if-endif`. Die vollständige Syntax dieses Kommandos sieht folgendermaßen aus:

```
if s | r
  post_kommandos
else
  post_kommandos
endif
```

Das `s` (für send) und `r` (für receive) stehen für Senden und Empfangen; dadurch können Sie die Kommandos angeben, die beim Aufrufen von `mailx` im Eingabemodus (Senden) oder Kommandomodus (Empfangen) ausgeführt werden sollen. Im letzten Beispiel wurde dieses Kommando benutzt, um zum Lesen der Post in das Verzeichnis `$HOME/mail` zu wechseln. Damit hat der Benutzer festgelegt, daß die eingegangene Post in ein bestimmtes Unterverzeichnis gesetzt werden soll.

In diesem Abschnitt wurden die Umgebungsvariablen beschrieben, die in der Datei `.mailrc` am häufigsten benutzt werden. Zur Einstellung dieser Umgebungsvariablen für eine bestimmte Sitzung müssen Sie sich auf jeden Fall im Kommandomodus befinden. Eine vollständige Auflistung der Umgebungsvariablen, die unter `mailx` eingestellt werden können, finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mailx`(1).

12 Kommunikation mit fernen Systemen

Einführung 12-1

Übermittlung von Dateien 12-2
Das Kommando `uucp` 12-2
- Die Syntax 12-3
- Beispiel 12-4
- Ablauf des Übertragungsprozesses 12-6

Das Kommando `uuto` 12-9
- Syntax 12-9
- Beispiel 12-9

Das Kommando `uustat` 12-12
- Beispiel 12-12

Das Kommando `uupick` 12-14
- Beispiel 12-14

Das Netzwerk 12-18
An ein fernes Terminal anschließen: das Kommando `ct` 12-18
- Format der Kommandozeile 12-19
- Beispiel 12-19

Anrufen eines anderen UNIX-Systems: Das Kommando `cu` 12-21
- Syntax 12-22
- Beispiel 12-26

Kommandoausführung auf einem fernen System: Das Kommando `uux` 12-27
- Syntax 12-28
- Beispiel 12-28

Einführung

Unter UNIX stehen Ihnen eine Reihe von Kommandos zur Verfügung, die Ihnen die Kommunikation mit anderen UNIX-Benutzern ermöglichen. Diese Kommandos bieten verschiedene Möglichkeiten wie die Übermittlung von Nachrichten an andere Benutzer bzw. den Empfang von Nachrichten von anderen Benutzern (die an Ihrem oder einem anderen UNIX- System arbeiten), den Austausch von Dateien sowie die Kommunikation mit anderen UNIX-Rechnern, die an Ihr Netzwerk angeschlossen sind. Über ein Netzwerk kann der Benutzer am einen System Nachrichten und Dateien mit einem anderen Rechner austauschen und Kommandos auf fernen Rechnern ausführen.

Damit Sie diese Möglichkeiten nutzen können, lernen Sie in diesem Kapitel folgende Kommandos kennen:

```
ct
cu
uucp
uuname
uupick
uustat
uuto
uux
```

Übermittlung von Dateien

Die elektronische Kommunikation basiert auf der Möglichkeit, Dateien von einem Rechner an einen anderen zu senden. Das Senden der Dateien ist jedoch nur der erste Schritt des dreistufigen Gesamtprozesses:

Auftrag senden
: Das bzw. die Kommando(s), mit denen eine Datei (ein Auftrag) an ein fernes System oder an einen bestimmten Benutzer des fernen Systems übermittelt wird (`uucp`, `uuto`).

Auftrags-Status überprüfen
: Sie können den Fortgang der von Ihnen angeforderten Übertragungen kontrollieren und verfolgen sowie spezielle Informationen über diese Anforderungen abrufen (`mail`, `uustat`).

Auftrag empfangen
: Für die Behandlung der Datei nach ihrer Ankunft am fernen System gibt es verschiedene Möglichkeiten (`uupick`).

Das Kommando `uucp`

Ohne dabei den gesamten Prozeß aus dem Auge zu verlieren, wollen wir mit dem ersten Schritt beginnen -- dem Kommando zur Übertragung einer Datei an ein fernes System. Mit dem Kommando `uucp` (für UNIX-to-UNIX system copy) können Dateien von einem Rechner auf einen anderen kopiert werden. Hierbei handelt es sich nicht um ein interaktives Kommando; es führt seine Arbeit "stillschweigend" durch, so daß sein Ablauf vom Benutzer nicht unmittelbar verfolgt werden kann. Nach dem Aufrufen dieses Kommandos können Sie andere Prozesse durchführen.

Um mit dem Kommando `uucp` Dateien an einen fernen Rechner übermitteln zu können, müssen Sie von dem fernen Rechner lediglich seinen Namen und, nach Möglichkeit, den bzw. die Benutzernamen des Empfängers bzw. der Empfänger am fernen System kennen.

Die Syntax

Mit uucp können Sie:

- Eine Datei in eine Datei oder ein Verzeichnis übertragen.
- Mehrere Dateien in ein Verzeichnis übertragen.

Damit uucp Ihre Datei(en) übermitteln kann, müssen Sie sowohl von der *ausgangsdatei* als auch von der *zieldatei* den absoluten Pfadnamen kennen. Das bedeutet jedoch nicht, daß Sie bei jedem Aufruf von uucp den absoluten Pfadnamen beider Dateien angeben müssen. Für diesen Zweck gibt es Kürzel mit speziellen Formaten, die von uucp zu absoluten Pfadnamen erweitert werden.

Zur Angabe von *ausgangsdatei* und *zieldatei* müssen Sie zunächst die Position Ihrer *ausgangsdatei* in Relation zu Ihrer aktuellen Position im Dateisystem ermitteln. Wenn die *ausgangsdatei* in Ihrem aktuellen Verzeichnis enthalten ist, genügt es, nur ihren Namen (ohne Zugriffspfad) anzugeben; andernfalls müssen Sie ihren absoluten oder relativen Pfadnamen angeben.

Wie wird die *zieldatei* angegeben? Da *zieldatei* sich auf dem fernen System befindet, muß sie immer mit einem Pfadnamen angegeben werden, der mit dem Namen des fernen Systems beginnt. Danach bietet uucp Ihnen jedoch eine Auswahl an verschiedenen Formaten:

- *systemname*!*absoluter_pfadname*
- *systemname*!~/[*pfadname*]

absoluter_pfadname ist dabei ein expliziter Pfadname. ~/ wird in `/var/spool/uucppublic/` übersetzt, das öffentliche uucp-Verzeichnis auf dem fernen System. *pfadname* ist ein Unterverzeichnis, dessen Name normalerweise mit dem Benutzernamen des Empfängers identisch ist.

Hinweis: In den Releases bis 4.0 war das `/var/spool/uucppublic/` das öffentliche uucp-Verzeichnis.

In den bisherigen Abschnitten ging es lediglich um die Übermittlung einer Datei von Ihrem lokalen System an ein fernes System. Mit uucp ist auch die umgekehrte Richtung möglich, also die Übermittlung einer Datei von einem fernen zu Ihrem lokalen System. In beiden Fällen können Sie zur Angabe der *ausgangsdateien* oder *zieldateien* die oben angegebenen Formate benutzen. Die Entscheidung für eines dieser Formate ist nicht davon abhängig, ob Sie eine

ausgangsdatei oder eine *zieldatei* angeben, sondern an welcher Position innerhalb des Dateisystems Sie sich aktuell in Relation zu den betreffenden Dateien befinden.

Angenommen, Sie haben in einem System mit dem Namen `mickey` den Benutzernamen `kol`. Ihr Home-Verzeichnis sei `/home/kol`, und Sie möchten eine Datei mit dem Namen `chap1` (im Verzeichnis `text`, das sich in Ihrem Home-Verzeichnis befindet) an den Benutzernamen `wsm` auf einem System namens `minnie` übermitteln. Da Ihr Arbeitsverzeichnis `/home/kol/text` ist, können Sie die *ausgangsdatei* über ihren relativen Pfadnamen, also `chap1`, angeben. Die *zieldatei* kann dann folgendermaßen angegeben werden:

- Über ihren absoluten Pfadnamen:

  ```
  uucp chap1 minnie!/home/wsm/receive/chap1
  ```

- Im Format *~/pfadname*. Dieser Pfadname wird zum Unterverzeichnis des Empfängers im öffentlichen Verzeichnis des fernen Systems ergänzt.

  ```
  uucp chap1 minnie!~/wsm/chap1
  ```

  (Die Datei wird in diesem Fall nach `minnie!/var/spool/uucppublic/wsm/chap1` kopiert).

Hinweis: Dasselbe Ergebnis erzielen Sie, indem Sie `chap1` am Ende der letzten Kommandozeile weglassen.

Beispiel

Wenn Sie beispielsweise eine Datei mit dem Namen `minutes` an einen fernen Rechner mit dem Namen `eagle` schicken möchten, geben Sie die folgende Kommandozeile ein:

```
$ uucp -m -j minutes eagle!/home/gws/minutes<CR>
eagleN3f45
$
```

Damit wird die Datei `minutes` (die sich in Ihrem aktuellen Verzeichnis auf Ihrem lokalen Rechner befindet) an den fernen Rechner `eagle` übermittelt und in die Datei `minutes` gesetzt, die den Pfadnamen `/home/gws` hat. Nach Beendigung der Übertragung werden Sie, der Absender, über `mail` benachrichtigt.

Mit der Option `-m` können Sie sicherstellen, daß Sie als Absender von `mail` darüber benachrichtigt werden, ob die Übertragung erfolgreich ausgeführt worden ist. Die Option `-j` bewirkt, daß die Auftragsnummer (`eagleN3f45`) ausgegeben wird.

Selbst wenn Sie von `uucp` nicht unmittelbar nach dem Abschicken einer Datei eine Bestätigung erhalten, müssen Sie nicht unbedingt von einer Störung ausgehen. Nicht alle Systeme, auf denen die Netzwerk-Software installiert ist, sind mit der Hardware ausgerüstet, die zum Anrufen von anderen Systemen erforderlich ist. Dateien, die von diesen sogenannten passiven Systemen übertragen werden, müssen in regelmäßigen Abständen von den aktiven, mit der notwendigen Hardware ausgerüsteten Systemen abgeholt werden (weitere Informationen dazu finden Sie im Abschnitt "Ablauf des Übertragungsprozesses"). Daher kann es zu Verzögerungen kommen, wenn Sie Dateien von einem passiven System aus abschicken. Erkundigen Sie sich bei Ihrem Systemverwalter, ob Ihr System aktiv oder passiv ist.

Im letzten Beispiel ist zur Angabe der *zieldatei* ein absoluter Pfadname benutzt worden. Zur Angabe der *zieldatei* gibt es zwei weitere Möglichkeiten:

- Das Home-Verzeichnis von `gws` kann mit dem Tilde-Zeichen (~) wie folgt angegeben werden:

  ```
  eagle!~gws/minutes
  ```

  Diese Eingabe wird folgendermaßen interpretiert:

  ```
  eagle!/home/gws/minutes
  ```

- Ähnlich wird auf den `uucppublic`-Bereich durch den Pfadnamen mit vorangestellter Tilde (~) Bezug genommen:

  ```
  eagle!~/gws/minutes
  ```

  Diese Eingabe wird folgendermaßen interpretiert:

  ```
  /var/spool/uucppublic/gws/minutes
  ```

Ablauf des Übertragungsprozesses

Dieser Abschnitt enthält einen Überblick über die Vorgänge, die beim Aufrufen des Kommandos `uucp` automatisch ablaufen. Kenntnisse über diesen Prozeß werden Ihr Verständnis für die Grenzen und Anforderungen des Kommandos erweitern. Weitere Informationen finden Sie im *System Administrator's Guide* und im *Referenzhandbuch für Systemverwalter.*

Bei der Eingabe des Kommandos `uucp` erstellt das `uucp`-Programm für die angeforderte Übertragung eine Arbeitsdatei sowie gewöhnlich eine Daten-Datei (Letzteres können Sie über die Option `-c` ausschalten). Die Arbeitsdatei enthält Informationen, die für die Übertragung der Datei(en) benötigt werden. Die Daten-Datei ist eine Kopie der angegebenen Ausgangsdatei. Nachdem diese Dateien im Spool-Verzeichnis angelegt worden sind , wird der `uucico`-Dämon gestartet.

Der `uucico`-Dämon versucht, eine Verbindung mit dem fernen Rechner herzustellen, an den die Datei(en) übermittelt werden soll(en). Zunächst sucht er in der `Systems`-Datei nach den Informationen, die zur Herstellung einer Verbindung mit dem fernen Rechner notwendig sind. Auf diese Weise erfährt `uucico`, mit welchem Gerät die Verbindung hergestellt werden soll. `uucico` durchsucht dann die `Devices`-Datei nach den Geräten, die den in der `Systems`-Datei angegebenen Anforderungen entsprechen. Nachdem `uucico` ein verfügbares Gerät gefunden hat, versucht es, die Verbindung herzustellen und sich bei dem fernen Rechner anzumelden.

Sobald sich `uucico` auf dem fernen Rechner anmeldet, startet er den `uucico`-Dämon auf dem fernen Rechner. Nachdem sich die beiden `uucico`-Dämone über das Leitungsprotokoll für die Dateiübertragung geeinigt haben, überträgt der lokale `uucico`-Dämon die von Ihnen abgeschickte(n) Datei(en) an den fernen Rechner; der ferne `uucico`-Dämon setzt die Datei(en) in das/die angegebene(n) Verzeichnis(se) auf dem fernen Rechner. Nachdem Ihr lokaler Rechner die Übertragung der Datei(en) beendet hat, kann der ferne Rechner Dateien abschicken, die in einer Warteschlange auf ihre Übertragung an Ihren lokalen Rechner warten. Dem fernen Rechner kann die Berechtigung zur Übertragung dieser Dateien mit einem Eintrag in der `Permissions`- Datei verwehrt werden. In diesem Fall kann der ferne Rechner die Übertragungen erst durchführen, nachdem er eine Verbindung mit Ihrem lokalen Rechner hergestellt hat.

Ist der ferne Rechner oder das Gerät, das die Verbindung mit dem fernen Rechner herstellen soll, aktuell nicht verfügbar, so verbleibt die Anforderung im Spool-Verzeichnis. Der `cron`-Dämon ruft zweimal pro Stunde das `uudemon.hour`-Skript auf, das wiederum den `uusched`-Dämon startet. Der `uusched`-Dämon durchsucht das Spool-Verzeichnis nach den verbliebenen Arbeitsdateien, ermittelt eine willkürliche Reihenfolge, in der diese Anforderungen abgearbeitet werden, und startet dann den bereits bekannten Übertragungsprozeß `uucico`.

Der hier sehr allgemein beschriebene Übertragungsprozeß bezieht sich auf einen aktiven Rechner (mit installierter Netzwerk-Software und einer Hardware-Ausrüstung, die zum Anrufen von anderen Rechnern fähig ist). Ein aktiver Rechner kann so konfiguriert werden, daß er einen passiven Rechner in bestimmten Abständen aufruft (sog. Polling). Ein passiver Rechner kann aufgrund der installierten Netzwerk-Software die zu übertragenden Dateien in eine Warteschlange setzen. Er kann jedoch - mangels geeigneter Hardware - keine Verbindung mit dem fernen Rechner herstellen. Die `Poll`-Datei (`/etc/uucp/Poll`) enthält eine Liste mit den Rechnern, die auf diese Weise aufgerufen werden müssen.

Das folgende Bild faßt die Syntax und Funktion des Kommandos `uucp` zusammen.

Bild 12-1: `uucp`**-Kurzübersicht**

Kurzbeschreibung `uucp` – Dateien von einem Rechner zum anderen kopieren.		
Kommando	*Optionen*	*Argumente*
`uucp`	`-j`, `-m`, `-s` und andere*	*ausgangsdatei,zieldatei*
Funktion:	`uucp` führt die Aufgaben durch, die für das Kopieren einer Datei von einem Rechner auf einen anderen vorab notwendig sind. Es ruft `uucico` auf, den Dämon (Hintergrundprozeß) zur Übertragung der Datei. Der Benutzer muß zum Kopieren einer Datei nur das Kommando `uucp` eingeben.	
Hinweise:	Das Verzeichnis `/var/spool/uucppublic` ist standardmäßig das einzige Verzeichnis, in das Sie Dateien setzen können. Um Dateien in die Verzeichnisse eines anderen Benutzers setzen zu können, müssen Sie von ihm und dem Systemverwalter die Schreibberechtigung erhalten. Obwohl es mehrere Möglichkeiten gibt, um Pfadnamen als Argumente anzugeben, wird aus Gründen der Eindeutigkeit der absolute Pfadname empfohlen.	

* Eine vollständige Beschreibung sämtlicher verfügbarer Optionen und ihrer Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `uucp`(1).

Das Kommando `uuto`

Das Kommando `uuto` ist eine vereinfachte Schnittstelle zu `uucp`, die Ihnen die Übermittlung von Dateien an das öffentliche Verzeichnis (`/var/spool/uucppublic/`) eines fernen Systems erleichtert.

Syntax

Das Kommando `uuto` hat folgendes grundlegendes Format:

`uuto` *dateiname(n) system*`!`*empfänger*`<CR>`

Dabei ist *dateiname* der Name der zu übertragenden Datei, *system* das System des Empfängers und *empfänger* der Benutzername des Empfängers. Mit `uuto` können Sie Dateien auch über Zwischensysteme zum gewünschten Zielsystem schicken, d.h., *system!empfänger* kann auch im Format *system1!system2!...!empfänger* angegeben werden.

Bei der Übermittlung von Dateien an einen Benutzer Ihres lokalen Systems können Sie den Systemnamen weglassen und folgendes Format verwenden:

`uuto` *dateiname !empfänger*`<CR>`

Beispiel

Zur Verdeutlichung der Funktionsweise von `uuto` ein Beispiel:

Die Übermittlung einer Datei mit `uuto` wird Auftrag genannt. Wenn Sie das Kommando `uuto` eingeben, wird Ihr Auftrag nicht sofort abgeschickt, sondern zunächst in eine Warteschlange gesetzt; gleichzeitig wird der Datei eine Auftragsnummer zugeordnet. Sobald ihre Auftragsnummer an der Reihe ist, wird die Datei an das ferne System übertragen und dort in das öffentliche Verzeichnis gesetzt. Der Empfänger, der durch eine `mail`-Nachricht benachrichtigt wird, kann die Datei nun mit dem Kommando `uupick` (das in diesem Kapitel später besprochen wird) auf unterschiedliche Weise behandeln.

In den folgenden Beispielen wird mit den folgenden Namen gearbeitet:

`wombat`	Ihr Benutzername.
`sys1`	Der Name Ihres lokalen Systems.
`marie`	Der Benutzername des Empfängers.
`sys2`	Der Name des fernen Systems.
`money`	Die zu übertragende Datei.

Weiterhin wird davon ausgegangen, daß beide Systeme miteinander kommunizieren können. Um die Datei `money` an den Benutzer `marie` auf dem System `sys2` zu übermitteln, muß folgendes eingegeben werden:

```
$ uuto money sys2!marie<CR>
$
```

Die Eingabeaufforderung in der zweiten Zeile zeigt Ihnen, daß die Datei in die Warteschlange gesetzt worden ist. Der Auftrag ist nun Ihrer Kontrolle entzogen; Sie können jetzt nur noch auf die Bestätigung warten, daß der Auftrag sein Ziel erreicht hat.

Wie erfahren Sie als Absender, daß der Auftrag angekommen ist? Die einfachste Möglichkeit besteht darin, in der `uuto`-Kommandozeile die Option `-m` anzugeben:

```
$ uuto -m money sys2!marie<CR>
$
```

Mit dieser Option erhalten Sie als Absender eine `mail`-Nachricht, sobald der Auftrag das ferne System erreicht hat. Hierbei handelt es sich um die formelle Bestätigung, daß die Datei vollständig und fehlerfrei übertragen worden ist. Die Meldung sieht in etwa wie folgt aus:

```
$ mail<CR>
>From uucp Fri Feb  3 11:53 EST 1989 remote from sys1
REQUEST: sys1!wombat/money --> sys2!~/receive/marie/sys1/ (marie)
(SYSTEM sys2)  copy succeeded
?
```

Bild 12-2 enthält eine Übersicht über Syntax und Funktion des Kommandos `uuto`.

Bild 12-2: `uuto`-Kurzübersicht

Kurzbeschreibung `uuto` – Dateien an einen anderen Benutzer senden		
Kommando	*Optionen*	*Argumente*
`uuto`	`-m` und andere*	*datei system!empfänger*
Funktion:	`uuto` kopiert eine in der Kommandozeile angegebene Datei in das öffentliche Verzeichnis eines angegebenen Systems, und benachrichtigt den Empfänger (über eine `mail`-Nachricht an seinen Benutzernamen), daß die Datei angekommen ist.	
Hinweise:	Für die Datei(en), die Sie abschicken möchten, müssen Sie die Leseberechtigung haben. Für das Vaterverzeichnis der Datei müssen die anderen Systembenutzer über die Lese- und Ausführungsberechtigung verfügen. Mit der Option `-m` wird der Absender der Datei benachrichtigt, sobald die Datei ihr Ziel erreicht hat.	

* Eine vollständige Beschreibung sämtlicher verfügbarer Optionen und ihrer Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `uuto`(1).

Das Kommando `uustat`

Nach dem Abschicken der Datei können Sie jetzt mit dem nächsten Schritt fortfahren -- der Überprüfung des Auftragsstatus. Um zu erfahren, ob der Auftrag Ihr System verlassen hat, rufen Sie das Kommando `uustat` auf. Dieses Kommando verfolgt den Ablauf aller abgesetzten `uucp`- und `uuto`-Aufträge und gibt Informationen über ihren Status aus.

Beispiel

Diese Meldung sieht in etwa wie folgt aus:

```
$ uustat<CR>
sys1N2f01  02/03-16:06  S  sys2  wombat 10   money
$
```

Die Statuszeile in diesem Beispiel besteht aus folgenden Elementen:

- `sys1N2f01` ist die Nummer, die dem Auftrag von Ihrem Host zugeordnet worden ist.
- `02/03-16:06` ist das Datum und die Uhrzeit der Einreihung des Auftrags in die Warteschlange.
- `S` gibt an, daß in diesem Fall das Senden einer Datei angefordert worden ist (R (receive) würde für Empfangen stehen).
- `sys2` ist das Zielsystem, an das die Datei übertragen wird.

- `wombat` ist der Benutzername des Benutzers, der den Auftrag angefordert hat.
- `10` ist die Größe der zu übertragenden Datei in Byte.
- `money` ist die Datei, die übertragen werden soll.

Weitere Statusmeldungen und Optionen für das Kommando `uustat` finden Sie im *Referenzhandbuch für Benutzer.*

Mehr ist nicht notwendig, um Dateien zu übertragen und den Ablauf des Auftrags zu verfolgen. Eine Übersicht über Syntax und Funktion des Kommandos `uustat` ist in Bild 12-3 enthalten.

Bild 12-3: `uustat`**-Kurzübersicht**

Kurzbeschreibung `uustat` – Status eines `uucp`- oder `uuto`-Auftrags überprüfen.		
Kommando	*Optionen*	*Argumente*
`uustat`	`-k` und andere*	keine
Beschreibung:	`uustat` gibt Informationen über den Status aller von Ihnen angeforderten `uucp`- und `uuto`-Aufträge aus.	
Hinweis:	Mit der Option `-k` in Kombination mit einer Auftragsnummer können Sie den angegebenen Auftrag abbrechen.	

* Eine vollständige Beschreibung sämtlicher verfügbarer Optionen und ihrer Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `uustat`(1).

Das Kommando `uupick`

Nachdem es in den bisherigen Abschnitten dieses Kapitels um das Abschicken von Dateien und die Abfrage von Statusinformationen ging, wollen wir diesen Vorgang im folgenden vom Standpunkt des Empfängers aus betrachten. Wenn eine mit `uuto` abgeschickte Datei im öffentlichen Verzeichnis Ihres UNIX-Verzeichnisses ankommt, werden Sie formell über `mail` benachrichtigt.

Beispiel

Um bei unserem Beispiel zu bleiben, erhält die Benutzerin mit dem Benutzernamen `marie` per `mail` die folgende Meldung, nachdem die Datei `money` im öffentlichen Verzeichnis ihres Systems eingegangen ist:

```
$ mail<CR>
>From uucp Fri Feb  3 16:05 EST 1989 remote from sys2
/var/spool/uucppublic/receive/wombat/sys1/money from sys1!wombat arrived
$
```

Die Meldung enthält folgende Informationen:

- Die erste Zeile informiert den Empfänger darüber, wann die Datei an ihrem Ziel angekommen ist.
- Der erste Teil der zweiten Zeile (bis zum Wort "money") gibt den Pfadnamen des öffentlichen Verzeichnisses an, in dem die Datei `money` gespeichert worden ist.
- Der Rest der Zeile (nach dem Wort "from") gibt den Namen des fernen Systems, den Namen des fernen Absenders (Benutzers) und den Status der Dateiübertragung ("arrived") an.

Nachdem Sie die mail-Meldung z.B. gelöscht oder abgespeichert haben, können Sie die Datei mit dem Kommando uupick in einem Verzeichnis Ihrer Wahl abspeichern. Dazu geben Sie bei der Eingabeaufforderung folgendes ein:

```
uupick<CR>
```

Das Kommando durchsucht das öffentliche Verzeichnis nach Dateien, die für Sie angekommen sind und gibt gegebenenfalls die Dateinamen aus. Dann fordert es Sie mit der Eingabeaufforderung ? zur Eingabe von weiteren Anweisungen auf.

Wenn z.B. die Benutzerin mit dem Benutzernamen marie das Kommando uupick für die Datei money aufruft, erzeugt das Kommando die folgende Ausgabe:

```
$ uupick<CR>
from system sys1: file money ?
```

An dieser Stelle können Sie verschiedene Anweisungen eingeben. Die gängigsten Anweisungen und eine Beschreibung ihrer Funktion sind im folgenden aufgeführt:

Zunächst sollten Sie die Datei aus dem öffentlichen in Ihr aktuelles Verzeichnis versetzen. Hierzu geben Sie beim Fragezeichen (?) das Kommando m ein:

```
? m<CR>
$
```

Durch diese Kommandozeile wird die Datei in Ihr aktuelles Verzeichnis versetzt. Wenn Sie die Datei in ein anderes Verzeichnis versetzen wollen, müssen Sie den Namen dieses Verzeichnisses nach dem m eingeben:

? m *anderes_verzeichnis*<CR>

Wenn weitere Dateien im öffentlichen Verzeichnis vorhanden sind, wird nun die nächste Datei angezeigt, gefolgt von einem Fragezeichen (?); andernfalls wird uupick beendet, und das System gibt eine Eingabeaufforderung aus.

Wenn sich am Zustand dieser Datei im Augenblick nichts ändern soll, betätigen Sie beim Fragezeichen (?) die Taste RETURN:

```
? <CR>
```

Dadurch verbleibt die aktuelle Datei im öffentlichen Verzeichnis, bis das Kommando uupick erneut aufgerufen wird. Wenn keine weiteren Meldungen über eingegangene Dateien vorhanden sind, gibt das System seine

Eingabeaufforderung aus.

Wenn Sie bereits wissen, daß Sie die Datei nicht abspeichern wollen, können Sie sie mit der Eingabe von `d` nach dem Fragezeichen (?) löschen:

```
? d<CR>
```

Durch diese Anweisung wird die aktuelle Datei aus dem öffentlichen Verzeichnis gelöscht und die nächste Meldung angezeigt (falls noch eine Datei vorhanden ist). Wenn keine weiteren Meldungen über wartende Dateien mehr vorhanden sind, gibt das System eine Eingabeaufforderung aus.

Um das Kommando `uupick` zu beenden, muß nach dem Fragezeichen (?) ein `q` eingegeben werden:

```
? q<CR>
```

Alle Dateien, die weder in eine andere Datei versetzt noch gelöscht worden sind, verbleiben im öffentlichen Verzeichnis, bis Sie das nächste mal das Kommando `uupick` aufrufen.

Weitere Informationen hierzu finden Sie im *Referenzhandbuch für Benutzer.*

Sie wissen jetzt, wie man eine Datei an ein fernes System sendet, den Verlauf des Auftrags kontrolliert und eine Datei im öffentlichen Verzeichnis behandelt. Bild 12-4 faßt die Syntax und Funktion des Kommandos `uupick` zusammen.

Bild 12-4: `uupick`**-Kurzübersicht**

Kurzbeschreibung		
`uupick` – Dateien suchen, die mit `uuto` oder `uucp` übermittelt worden sind.		
Kommando	*Optionen*	*Argumente*
`uupick`	`-s`system	
Funktion:	`uupick` sucht im öffentlichen Verzeichnis Ihres Systems nach Dateien, die mit den Kommandos `uuto` oder `uucp` übermittelt worden sind. Sind Dateien vorhanden, zeigt das Kommando Informationen über diese Dateien an und fordert Sie zur Eingabe einer Anweisung auf. Wird das Kommando `uupick` mit der Option `-s`*system* eingegeben, so wird im öffentlichen Verzeichnis nur nach Dateien gesucht, die von *system* übermittelt worden sind.	
Hinweis:	Ein Fragezeichen (`?`) am Ende der Nachricht zeigt an, daß eine Anweisung erwartet wird. Eine vollständige Liste der Eingabemöglichkeiten finden Sie im *Referenzhandbuch für Benutzer*.	

Das Netzwerk

Über ein Netzwerk können Rechner und Terminals miteinander verbunden werden. Die Benutzer haben dann folgende Möglichkeiten:

- Sie können sich sowohl bei einem fernen als auch bei einem lokalen Rechner anmelden.
- Sie können an zwei Rechnern gleichzeitig angemeldet sein und an beiden arbeiten (ohne den einen abschalten zu müssen, wenn Sie sich am anderen anmelden).
- Sie können Daten zwischen Rechnern austauschen.

In diesem Abschnitt werden die Kommandos erklärt, mit denen Sie diese Funktionen durchführen können. Mit dem Kommando `ct` können Sie die Verbindung zwischen Ihrem Rechner und einem fernen Terminal herstellen, das mit einem Modem ausgestattet ist. Mit dem Kommando `cu` können Sie die Verbindung zwischen Ihrem und einem fernen Rechner herstellen. Das Kommando `uux` ermöglicht das Ausführen von Kommandos auf fernen Systemen, bei denen Sie nicht angemeldet sind.

Hinweis: Auf einigen Rechnern sind diese Kommandos nur dann verfügbar, wenn die Netzwerk-Software installiert ist. Ist dies bei Ihnen nicht der Fall, so erhalten Sie bei der Eingabe eines Netzwerk-Kommandos eine Meldung auf Ihrem Bildschirm, die in etwa wie folgt aussieht:

```
ct: not found
```

Erkundigen Sie sich bei Ihrem Systemverwalter, ob Netzwerkkommandos auf Ihrem UNIX-System zur Verfügung stehen.

An ein fernes Terminal anschließen: das Kommando `ct`

Mit dem Kommando `ct` können Sie eine Verbindung zwischen Ihrem Rechner und einem fernen, mit einem Modem ausgestatteten Terminal herstellen. Außerdem ermöglicht es dem Benutzer eines fernen Terminals sich am Rechner anzumelden. Um die Verbindung herzustellen, wählt das Kommando die Telephonnummer des Modems. Das Modem muß in der Lage sein, den Anruf entgegenzunehmen. Wenn `ct` feststellt, daß der Anruf entgegengenommen worden ist, gibt es eine Login-Aufforderung aus.

Dieses Kommando ist auch nützlich, wenn es vom entgegengesetzten Ende, d.h. am fernen Terminal, eingegeben wird. Wenn Sie an einem fernen Terminal arbeiten, das sehr weit von Ihrem Rechner entfernt ist und Sie sich die Kosten für ein "Ferngespräch" ersparen möchten, können Sie den Rechner über das Kommando `ct` zum Anrufen Ihres Terminals anweisen. Dazu rufen Sie den Rechner einfach an, melden sich an und geben das Kommando `ct` ein. Der Rechner unterbricht dann die aktuelle Verbindung und ruft Ihr (fernes) Terminal zurück.

Wenn das Kommando `ct` keine freie Wähleinheit (Dialer) finden kann, erhalten Sie die Meldung, daß alle Wähleinheiten besetzt sind. Sie werden gefragt, ob es warten soll, bis eine Wähleinheit frei wird. Wenn Sie mit "Ja" antworten, werden Sie gefragt, wie lange (in Minuten) es warten soll.

Format der Kommandozeile

Das Kommando `ct` hat folgendes Format:

`ct` [*optionen*] *tel_nr*<CR>

Das Argument *tel_nr* steht für die Telephonnummer des fernen Terminals.

Beispiel

Angenommen, Sie haben sich von einem lokalen Terminal aus an einem Rechner angemeldet und möchten zwischen einem fernen Terminal und Ihrem Rechner eine Verbindung herstellen; das Modem des fernen Terminals hat die Telephonnummer 555-3497. In diesem Fall geben Sie folgendes Kommando ein:

```
ct -h -w5 -s1200 9=5553497<CR>
```

Hinweis: Das Gleichheitszeichen (=) dient in einer 2-stufigen Verbindung zur Abgrenzung zwischen dem ersten und zweiten Teil der Nummer; es bedeutet, daß vor der Wahl der nächsten Ziffer der Wählton abgewartet werden muß.

`ct` ruft das Modem in diesem Fall mit einer Wähleinheit an, die mit einer Übertragungsgeschwindigkeit von 1200 Baud arbeitet. Steht die Wähleinheit nicht zur Verfügung, bewirkt die Option `-w5`, daß `ct` fünf Minuten lang auf eine Wähleinheit wartet, bevor es das Kommando beendet. Mit der Option `-h` weisen Sie das Kommando `ct` an, die Verbindung zwischen dem lokalen Terminal (dem Terminal, auf dem das Kommando eingegeben worden ist) und dem

Rechner nicht zu unterbrechen.

Angenommen, Sie möchten sich von zu Hause aus am Rechner anmelden. Um die Kosten für ein "Ferngespräch" zu vermeiden, können Sie mit dem Kommando `ct` erreichen, daß der Rechner Ihr Terminal anruft:

```
ct -s1200 9=5553497<CR>
```

Da die Option `-w` in diesem Fall nicht angegeben worden ist, informiert `ct` Sie gegebenenfalls über die folgende Meldung, daß kein Gerät verfügbar ist:

```
1 busy dialer at 1200 baud Wait for dialer?
```

Wenn Sie jetzt `n` (für no) eingeben, wird das Kommando `ct` beendet. Wenn Sie `y` (für yes) eingeben, fordert `ct` Sie zur Angabe der maximalen Wartezeit auf:

```
Time, in minutes?
```

Steht eine Wähleinheit zur Verfügung, so gibt `ct` die folgende Meldung aus:

```
Allocated dialer at 1200 baud
```

In jedem Fall werden Sie vom System gefragt, ob die Verbindung zwischen Ihrem fernen Terminal und dem Rechner abgebrochen werden soll:

```
Confirm hangup?
```

Wenn Sie jetzt `y` (für yes) eingeben, werden Sie abgemeldet; `ct` ruft Sie dann zurück, sobald eine Wähleinheit verfügbar ist. Wenn Sie `n` (für no) eingeben, wird das Kommando `ct` beendet. Sie bleiben am Rechner angemeldet; `ct` versucht dann nicht, Sie zurückzurufen.

Bild 12-5 enthält einen Überblick über die Syntax und Funktion des Kommandos `ct`.

Bild 12-5: `ct`-Kurzübersicht

Kurzbeschreibung `ct` – Rechner mit fernem Terminal verbinden.		
Kommando	*Optionen*	*Argumente*
`ct`	`-h, -w, -s` und andere*	*tel_nr*
Funktion:	`ct` stellt die Verbindung zwischen dem Rechner und einem fernen Terminal her und ermöglicht es einem Benutzer, sich von diesem Terminal aus anzumelden.	
Hinweis:	Das ferne Terminal muß mit einem Modem ausgestattet sein, das zur automatischen Entgegennahme von Anrufen (Autoanswering-Funktion) in der Lage ist.	

* Eine vollständige Beschreibung sämtlicher verfügbarer Optionen und ihrer Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `ct`(1).

Anrufen eines anderen UNIX-Systems: Das Kommando `cu`

Das Kommando `cu` stellt die Verbindung zwischen einem fernen Rechner und Ihrem Rechner her und ermöglicht Ihnen, auf beiden Rechnern gleichzeitig angemeldet zu sein. D. h., Sie können zwischen den beiden Rechnern umschalten, Dateien übertragen und auf beiden Rechnern Kommandos ausführen, ohne die Verbindung zwischendurch abzubrechen.

Der Ablauf des Kommandos cu hängt davon ab, ob Sie in der Kommandozeile eine Telephonnummer oder den Systemnamen des fernen Rechners angeben. Eine von Ihnen eingegebene Telephonnummer wird an das automatische Wählmodem weitergeleitet. Wenn Sie dagegen einen Systemnamen angeben, ermittelt cu die Telephonnummer anhand der Datei `Systems`. Wird zur Herstellung der Verbindung kein automatisches Wählmodem verwendet, kann in der Kommandozeile der Anschluß (port) angegeben werden, der der Direktleitung zum fernen Rechner zugeordnet ist.

Sobald die Verbindung hergestellt ist, fordert der ferne Rechner Sie auf, sich anzumelden. Wenn Sie nicht mehr am fernen Terminal arbeiten möchten, melden Sie sich ab und brechen die Verbindung durch die Eingabe von <~.> ab. Auf dem lokalen Rechner sind Sie danach immer noch angemeldet.

Hinweis: Das Kommando cu kann keine Fehlerüberprüfung und -berichtigung durchführen; daher kommen die Dateien möglicherweise unvollständig oder fehlerhaft an ihrem Ziel an. Einen eventuellen Datenverlust können Sie mit dem Kommando sum feststellen. Hierzu geben Sie, bevor Sie die *datei* auf Ihrem lokalen System abschicken, das Kommando sum mit dem Argument *datei* ein. Wiederholen Sie das Kommando auf dem fernen System, nachdem *datei* dort angekommen ist. Nach einer fehlerfreien Dateiübertragung müssen beide Ergebnisse übereinstimmen.

Syntax

Das Kommando cu hat folgendes Format:

cu [*optionen*] *tel_nr* | *systemname*<CR>

Die Kommandozeile besteht aus folgenden Komponenten:

tel_nr — Die Telephonnummer des fernen Rechners.

Mit einem Gleichheitszeichen (=) geben Sie in einer zweistufigen Verbindung an, daß vor dem Wählen der nächsten Ziffer der Wählton abgewartet werden muß. Ein Bindestrich (-) bedeutet, daß vier Sekunden lang gewartet werden soll.

systemname Ein Systemname, der in der `Systems`-Datei aufgeführt ist.

Das Kommando `cu` ermittelt die Telephonnummer und die Übertragungsgeschwindigkeit (Baud-Rate) anhand der `Systems`-Datei und sucht nach einer Wähleinheit. Die Optionen `-s`, `-n` und `-l` sollten nicht zusammen mit dem Argument *systemname* verwendet werden (die Liste der in der `Systems`-Datei aufgeführten Rechner können Sie mit dem Kommando `uuname` abrufen).

Nachdem die Verbindung zwischen Ihrem Terminal und dem Rechner hergestellt ist und Sie auf dem fernen Rechner angemeldet sind, wird die gesamte Standardeingabe (d.h. die über die Tastatur eingegebenen Zeichen) mit Ausnahme der Tilde-Kommandos (~) an den fernen Rechner geschickt. Bild 12-6 und Bild 12-7 zeigen die Kommandos, die Sie ausführen können, während Sie mit dem fernen Rechner über das Kommando `cu` verbunden sind.

Bild 12-6: Kommandofolgen, die bei `cu` verwendet werden

Kommandofolge	Funktion
`~.`	Verbindung abbrechen.
`~!`	Rückkehr zum lokalen Rechner ohne Abbruch der Verbindung. Um zum fernen Rechner zurückzuschalten, geben Sie `<^d>` (CONTROL-d) ein.
`~!`*kommando*	*kommando* auf dem lokalen Rechner ausführen.
`~$`*kommando*	*kommando* auf dem lokalen Rechner ausführen und die Ausgabe an das ferne System senden.
`~%cd` *pfad*	Verzeichnis auf dem lokalen Rechner wechseln. *pfad* ist der Pfad- oder Verzeichnisname.
`~%take` *von* [*nach*]	Eine Datei mit dem Namen *von* (auf dem fernen System) in eine Datei mit dem Namen *nach* (auf dem lokalen Rechner) kopieren. Fehlt *nach*, so wird für beide Angaben das Argument *von* verwendet.
`~%put` *vpm* [*nach*]	Eine Datei mit dem Namen *von* (auf dem lokalen Rechner) in eine Datei mit dem Namen *nach* (auf dem fernen Rechner) kopieren. Fehlt *nach*, so wird für beide Angaben das Argument *von* verwendet.
`~~...`	Die Zeile `~...` an den fernen Rechner übermitteln.
`~%break`	Ein BREAK an den fernen Rechner übertragen (kann auch mit `~%b` angegeben werden).

Bild 12-6: Kommandofolgen, die bei `cu` verwendet werden (Fortsetzung)

Kommandofolge	Funktion
`~%ifc`	Die Steuerung des Eingabeflusses wird ein- und ausgeschaltet. Ist diese Option aktiviert, so kann eine terminal-seitige Steuerung des Eingabeflusses stattfinden (kann auch mit `~%nostop` angegeben werden).
`~%ofc`	Die Steuerung des Ausgabeflusses wird ein- und ausgeschaltet. Ist diese Option aktiviert, so kann host-seitig eine Steuerung des Ausgabeflusses stattfinden (kann auch mit `~%noostop` angegeben werden).
`~%debug`	Die Fehlerprüf-Option `-d` (für debugging) wird aktiviert bzw. deaktiviert; diese Option kann auch im Format `~%d` angegeben werden.
`~t`	Die Werte der termio-(Terminal I/O)Strukturvariablen für Ihr Terminal werden angezeigt (nützlich zur Fehlersuche).
`~l`	Die Werte der termio-Strukturvariablen für die Übertragungsleitung zum fernen Rechner werden angezeigt (nützlich zur Fehlersuche).

Hinweis: `~%put` setzt auf dem fernen Rechner die Kommandos `stty` und `cat` voraus. Auch müssen auf beiden Rechnern zum Löschen von Zeichen und Zeilen aktuell dieselben (erase- und kill-) Zeichen benutzt werden.

`~%take` setzt auf dem fernen Rechner die Kommandos `echo` und `cat` voraus. Auch sollte `stty tabs` auf dem fernen Rechner aktiviert sein, wenn Tabulatorzeichen beim Kopieren nicht in Leerzeichen umgesetzt werden sollen.

Beispiel

Wenn Ihren Rechner beispielsweise mit einem fernen Rechner namens `eagle` verbinden möchten, der die Telephonnummer 555–7867 hat, geben Sie die folgende Kommandozeile ein:

```
cu -s2400 9=5557867<CR>
```

Die Option `-s2400` bewirkt, daß `cu` zum Anrufen von `eagle` eine Wähleinheit mit der Übertragungsgeschwindigkeit 2400 Baud verwendet. Ohne die Angabe von `-s` verwendet `cu` eine Wähleinheit mit einer Übertragungsgeschwindigkeit von 1200 Baud (Standard).

Sobald `eagle` den Anruf entgegennimmt, werden Sie von `cu` benachrichtigt, daß die Verbindung hergestellt worden ist. `eagle` gibt für Sie dann die Login-Aufforderung aus:

```
Connected
login:
```

Geben Sie jetzt Ihren Benutzernamen und Ihr Paßwort ein.

Mit dem Kommando `take` können Sie Dateien vom fernen zu Ihrem lokalen Rechner kopieren. Wenn Sie beispielsweise für Ihren lokalen Rechner eine Kopie der Datei `proposal` erstellen möchten, so erstellt das folgende Kommando von der Datei `proposal` in Ihrem aktuellen Verzeichnis eine Kopie auf dem fernen Rechner und setzt sie in das aktuelle Verzeichnis Ihres lokalen Rechners. Wird für die neue Datei kein Dateiname angegeben, erhält sie ebenfalls den Namen `proposal`.

```
~%take proposal<CR>
```

Das Kommando `put` bewirkt das Gegenteil: Es kopiert Dateien vom lokalen zum fernen Rechner. Wenn Sie die Datei `minutes` aus Ihrem aktuellen Verzeichnis auf dem lokalen Rechner auf den fernen Rechner kopieren wollen, müssen Sie folgendes eingeben:

```
~%put minutes minutes.9-18<CR>
```

In diesem Fall wurde für die neue Datei ein neuer Name eingegeben (`minutes.9-18`); daher erhält die Kopie der Datei `minutes` auf dem fernen Rechner den Namen `minutes.9-18`.

Bild 12-8 faßt die Syntax und Funktion des Kommandos `cu` zusammen.

Bild 12-7: `cu`**-Kurzübersicht**

Kurzbeschreibung `cu` – Rechner mit einem fernen Rechner verbinden.		
Kommando	*Optionen*	*Argumente*
`cu`	`-s` u.a.*	*tel_nr* (oder) *systemname*
Funktion:	`cu` stellt die Verbindung zwischen Ihrem Rechner und einem fernen Rechner her und ermöglicht Ihnen, auf beiden Systemen gleichzeitig angemeldet zu sein. Nach Ihrer Anmeldung können Sie zwischen beiden Rechnern umschalten und auf jeden Rechner Dateien übertragen, ohne die Verbindung zwischendurch abzubrechen.	

* Eine vollständige Beschreibung sämtlicher verfügbarer Optionen und ihrer Funktion finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `cu(1)`.

Kommandoausführung auf einem fernen System: Das Kommando `uux`

Mit dem Kommando `uux` (für UNIX-to-UNIX system command execution) können Sie UNIX-Kommandos auf einem fernen Rechner ausführen. Mit diesem Kommando können Sie Dateien auf verschiedenen Rechnern "abholen", ein Kommando auf einem bestimmten Rechner ausführen und die Standardausgabe zu einer Datei auf einem bestimmten Rechner umleiten. Möglicherweise gelten auf dem fernen System Ausführungsbeschränkungen für bestimmte Kommandos. In diesem Fall werden Sie über `mail` benachrichtigt, daß Ihre Anforderung auf dem fernen Rechner nicht ausgeführt werden kann.

Syntax

Das Kommando `uux` hat folgendes Format:

`uux` [*optionen*] *kommandofolge*<CR>

Die *Kommandofolge* besteht aus ein oder mehreren Argumenten. Sonderzeichen der Shell (wie "<>|^\fP'') müssen in Anführungszeichen (") eingeschlossen werden; hierzu setzen Sie entweder die gesamte *kommandofolge* in Anführungszeichen ("), oder das Sonderzeichen als separates Argument. Den Kommandos und Dateinamen in der *kommandofolge* kann ein *systemname!* vorangestellt werden. Alle Argumente, die keinen *systemnamen* enthalten, werden als Kommando-Argumente behandelt. Ein Dateiname kann entweder ein absoluter Pfadname oder der Name einer Datei im aktuellen Verzeichnis (des lokalen Rechners) sein.

Beispiel

Ist Ihr Rechner über eine Direktleitung mit einem größeren Host-Rechner verbunden, so können Sie über diesen Rechner mit dem Kommando `uux` Dateien ausdrucken, die auf Ihrem Rechner enthalten sind. Dafür geben Sie folgendes Kommando ein:

```
pr minutes | uux -p host!lp<CR>
```

Mit diesem Kommando wird die Datei `minutes` auf dem Bereichsdrucker des Rechners `host` ausgedruckt.

Weitere Informationen zu diesem Thema finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `uux`(1). Bild 12-9 faßt Syntax und Funktion des Kommandos `uux` zusammen.

Bild 12-8: `uux`**-Kurzübersicht**

Kurzbeschreibung `uux` – Kommandos auf fernen Rechnern ausführen.		
Kommando	*Optionen*	*Argumente*
`uux`	`-p` u.a. *	*kommandofolge*
Funktion:	Mit `uux` können UNIX-Kommandos auf fernen Rechnern ausgeführt werden. Sie können damit Dateien auf verschiedenen Rechnern "abholen", ein Kommando auf einem bestimmten Rechner ausführen und die Standardausgabe zu einer Datei auf einem bestimmten Rechner umleiten.	
Hinweis:	Standardmäßig kann über das Kommando `uux` nur das Kommando `mail` ausgeführt werden. Erkundigen Sie sich bei Ihrem Systemverwalter, welche anderen Kommandos über `uux` ausgeführt werden können.	

A Kurzübersicht über das Dateisystem

Das UNIX-Dateisystem A-1
Aufbau des Dateisystems A-1

Die Verzeichnisse des UNIX-Betriebssystems A-4

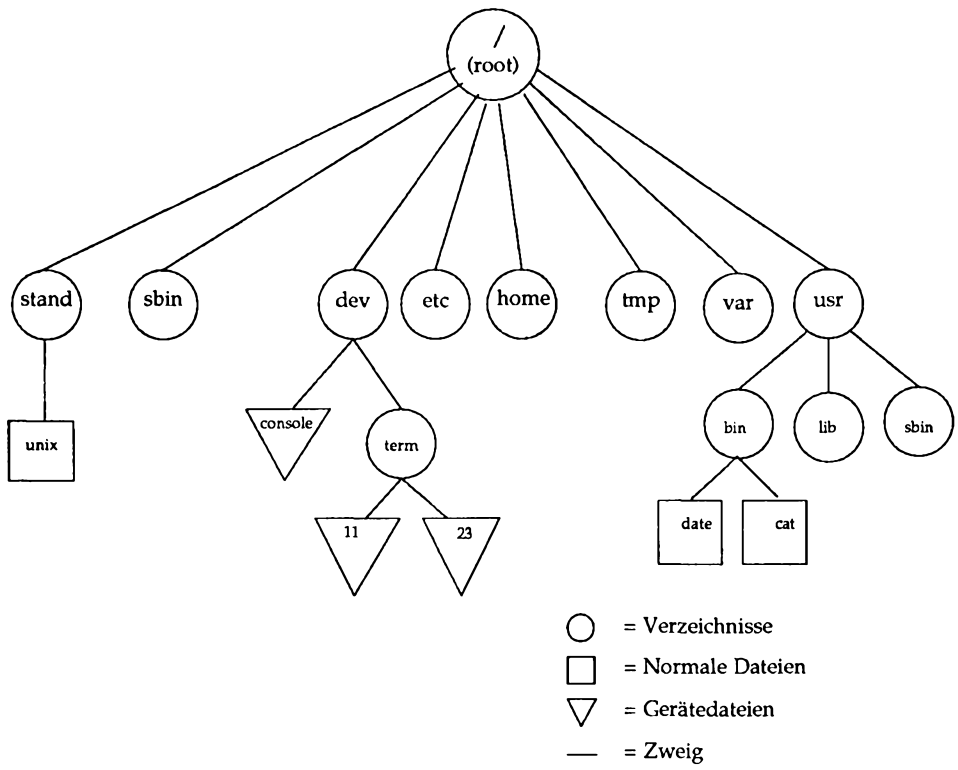
Das UNIX-Dateisystem

Dieser Anhang enthält eine Kurzübersicht über das in Kapitel 1 beschriebene Dateisystem und die wichtigsten im Root-Verzeichnis enthaltenen Systemverzeichnisse.

Aufbau des Dateisystems

Die Dateien des UNIX-Betriebssystems sind hierarchisch angeordnet. Die Verzeichnishierarchie wird oft auch als umgedrehter Baum beschrieben. An der Spitze der Hierarchie steht das Root-Verzeichnis, der Dreh- und Angelpunkt des gesamten Dateisystems. Es wird durch einen Schrägstrich (/) dargestellt. Alle anderen Verzeichnisse und Dateien stehen unter dem Root-Verzeichnis und verzweigen sich von dort (siehe Bild A-1).

Bild A-1: Aufbau des Dateisystems



Vom Root-Verzeichnis aus führt ein Pfad zum Home-Verzeichnis. In Ihrem Home-Verzeichnis können Sie eine eigene Verzeichnis- und Dateihierarchie erstellen, in der Sie Informationen organisieren und speichern können.

Andere Pfade führen vom Root-Verzeichnis zu Systemverzeichnissen, auf die auch andere Benutzer zugreifen können. Die in diesem Leitfaden beschriebenen Systemverzeichnisse sind in allen Implementierungen des UNIX System V Release 4 enthalten. Sie werden vom Betriebssystem mitgeliefert und gewartet.

Möglicherweise enthält Ihr UNIX-System zusätzlich zu diesen Standardverzeichnissen noch weitere Systemverzeichnisse. Über die folgende Kommandozeile erhalten Sie eine Übersicht über die Verzeichnisse und Dateien im Root-Verzeichnis Ihres UNIX-Systems:

```
ls -l / <CR>
```

Mit Hilfe der Pfadnamen können Sie sich innerhalb der Verzeichnishierarchie bewegen. Mit folgender Kommandozeile können Sie z.B. in das Verzeichnis `/usr/bin` (das die ausführbaren Dateien des UNIX-Systems enthält) wechseln:

```
cd /usr/bin <CR>
```

Über die folgenden Kommandozeilen können Sie den Inhalt eines Verzeichnisses abrufen:

```
ls <CR>         # Übersicht über die Dateien und Verzeichnisse
ls -l <CR>      # (long) Ausführliche Übersicht über die
                # Dateien und Verzeichnisse
```

Das folgende Beispiel zeigt, wie Sie den Inhalt eines Verzeichnisses, in dem Sie sich nicht befinden, abrufen können. Dazu verwenden Sie das Kommando `ls`:

```
ls /usr/bin <CR>        # kurze Liste
ls -l /usr/bin <CR>     # lange Liste
```

Im folgenden Abschnitt werden das Root-Verzeichnis und die darunter befindlichen Systemverzeichnisse beschrieben, die im Bild A-1 zu sehen sind.

Die Verzeichnisse des UNIX-Betriebssystems

`/`	Der Ausgangspunkt des Dateisystems (das sogenannte Root-Verzeichnis).
`/stand`	Enthält Programme und Daten-Dateien, die für beim Urladen verwendet werden.
`/sbin`	Enthält wichtige ausführbare Dateien, die beim Urladen und bei der manuellen Wiederherstellung des Systems verwendet werden.
`/dev`	Enthält Gerätedateien für folgende Peripheriegeräte: `console` Konsole `lp` Zeilendrucker `term/*` Benutzer-Terminal(s) `dsk/*` Platten
`/etc`	Enthält maschinenspezifische Konfigurationsdateien für die Systemverwaltung und Systemverwaltungs-Datenbanken.
`/home`	Die Root eines Teilbaums für Benutzerverzeichnisse.
`/tmp`	Enthält Temporäre Dateien, z.B. von Editoren angelegte Puffer.
`/var`	Enthält die Root eines Teilbaums für veränderliche Dateien, z.B. Protokolldateien.
`/usr`	Enthält weitere Verzeichnisse wie z.B. `lib` und `bin`.

`/usr/bin`	U.a. die folgenden ausführbaren Programme und Dienstprogramme,
	`cat`
	`date`
	`login`
	`grep`
	`mkdir`
	`who`

`/usr/lib`	Bibliotheken für Programme und Sprachen.

B UNIX-Kommandoübersicht

Grundlegende UNIX-Kommandos B-1

Grundlegende UNIX-Kommandos

`at` Kommandoausführung im Hintergrund zu einem bestimmten Zeitpunkt anfordern; der Zeitpunkt wird in der Kommandozeile angegeben.

Die Kommandozeile kann beispielsweise folgendermaßen aussehen:

```
at 8:45am Jun 09<CR>
kommando1<CR>
kommando2<CR>
<^d>
```

Wird `at` ohne Datum eingegeben, so wird das Kommando zum angegebenen Zeitpunkt (spätestens nach 24 Stunden) durchgeführt.

`banner` Meldung (die aus Wörtern mit bis zu 10 Zeichen besteht) im Großformat auf der Standard-Ausgabe anzeigen.

`batch` Kommandos in eine Warteschlange setzen und ausführen, sobald es die Systemauslastung zuläßt. Die Kommandozeile dieses Kommandos kann beispielsweise folgendermaßen aussehen:

```
batch<CR>
kommando1<CR>
kommando2<CR>
<^d>
```

Eines der bei `batch`(1) angegebenen Kommandos kann auch der Name einer Shell-Prozedur sein. In diese Shell-Prozedur können Sie Kommandos einbringen, die Sie häufig mit `batch`(1) aufrufen.

`cat` Der Inhalt einer bestimmten Datei wird auf Ihrem Terminal angezeigt. Um die Bildschirmausgabe auf einem ASCII- Terminal vorübergehend anzuhalten, geben Sie das Steuerzeichen <^s> ein; sobald Sie die Bildschirmausgabe wieder fortsetzen möchten, geben Sie <^q> ein. Zum endgültigen Abbruch der Bildschirmausgabe und zur Rückkehr zur Shell betätigen Sie auf einem ASCII- Terminal die Taste BREAK oder DELETE.

`cd`	Vom aktuellen Verzeichnis ins Home-Verzeichnis wechseln. Wenn Sie bei diesem Kommando einen Verzeichnisnamen angeben, werden Sie vom aktuellen Verzeichnis ins angegebene Verzeichnis versetzt. Um mit einem einzigen Kommando mehrere Verzeichnisebenen zu überspringen, geben Sie statt des Verzeichnisnamens einen Pfadnamen ein.
`cp`	Datei in eine neue Datei kopieren; die ursprüngliche Datei bleibt unverändert erhalten.
`cut`	Felder aus jeder Zeile einer Datei ausschneiden. Dieses Kommando gibt Ihnen beispielsweise die Möglichkeit, Spalten aus einer Tabelle zu extrahieren.
`date`	Datum und Uhrzeit anzeigen.
`diff`	Zwei Dateien vergleichen. Das Kommando `diff`(1) gibt die Zeilen aus, zwischen denen Unterschiede bestehen; außerdem informiert Sie es über die Änderungen, mit denen die zweite Datei inhaltsgleich mit der ersten Datei gemacht werden kann.
`echo`	Die Eingabe wird, einschließlich des Wagenrücklauf-Zeichens, auf der Standard-Eingabe (dem Terminal) angezeigt; das Kommando gibt abschließend die Eingabeaufforderung aus.
`ed`	Datei mit dem Zeileneditor editieren. Ist die angegebene Datei nicht vorhanden, so wird sie durch das Kommando `ed`(1) angelegt. Eine ausführliche Einführung in den Zeileneditor `ed`(1) finden Sie in Kapitel 6.
`grep`	Datei(en) nach einem bestimmten Muster durchsuchen und die Zeilen, in denen das Muster enthalten ist, ausgeben. Wenn Sie mehr als eine Datei angeben, gibt `grep`(1) diejenige Datei aus, die das Muster enthält.
`kill`	Hintergrund-Prozeß beenden; der betreffende Prozeß wird über seine Prozeßnummer (PID) angegeben. Die Prozeßnummer können Sie über das Kommando `ps`(1) ermitteln.

`lex`	Generierung von Programmen, die für einfache Textanalysen verwendet werden sollen; hierbei kann es sich z.B. um den ersten Schritt bei der Entwicklung eines Übersetzers (Compilers) handeln. Nähere Informationen hierzu finden Sie im *Leitfaden für Programmierer*.
`lp`	Inhalt einer bestimmten Datei über einen Zeilendrucker ausgeben, um eine Druckausgabe der Datei auf Papier zu erhalten.
`lpstat`	Anzeige des Status sämtlicher Anforderungen, die an den Zeilendrucker übermittelt worden sind. Für die Abfrage von bestimmten Informationen stehen spezielle Optionen zur Verfügung.
`ls`	Anzeige der Namen sämtlicher Dateien und Dateiverzeichnisse mit Ausnahme derjenigen, die mit einem Punkt (.) beginnen. Zum Abrufen von ausführlicheren Informationen über die Dateien innerhalb des Verzeichnisses stehen spezielle Optionen zur Vergügung (siehe auch Eintrag `ls`(1) im *Referenzhandbuch für Benutzer*.)
`mail`	Sämtliche Nachrichten, die über die elektronische Post an Ihr Terminal übermittelt werden sind, werden hintereinander angezeigt. Hinter jeder Nachricht steht eine Eingabeaufforderung in Form eines Fragezeichens (`?`); `mail`(1) fordert Sie damit zu einer bestimmten Aktion (z.B. Abspeichern, Weiterleiten oder Löschen der Nachricht) auf. Eine Übersicht über sämtliche verfügbaren Optionen erhalten Sie, indem Sie ein weiteres Fragezeichen (`?`) eingeben. Folgt auf `mail`(1) ein Benutzername, so schickt das Kommando eine Nachricht an den betreffenden Benutzer. Die Nachricht kann beliebig lang sein. Zum Abschluß der Nachricht geben Sie <^d> ein; die Nachricht wird dann an den Empfänger übermittelt. Zur Beendigung der Mail-Sitzung betätigen Sie die Taste BREAK.
`mailx`	`mailx`(1) ist eine komfortablere, erweiterte Version von `mail`(1).

`make`	Umfangreiche Programme oder Dokumente auf der Basis kleinerer Programme oder Dokumente verwalten und unterstützen (siehe auch Eintrag `make`(1) im *Referenzhandbuch für Benutzer.*
`mkdir`	Anlegen eines neuen Verzeichnisses. Das neue Verzeichnis wird als Unterverzeichnis des Verzeichnisses angelegt, in dem Sie das Kommando `mkdir` aufrufen. Wenn Sie im neuen Verzeichnis weitere Unterverzeichnisse oder Dateien anlegen möchten, sollten Sie mit dem Kommando `cd` in dieses Verzeichnis wechseln.
`mv`	Dateien an eine neue Position innerhalb des Dateisystems versetzen. Sie können eine Datei in eine Datei anderen Namens innerhalb desselben Verzeichnisses oder in ein anderes Verzeichnis versetzen. Wenn Sie die Datei in ein anderes Verzeichnis versetzen, können Sie sowohl den alten Dateinamen beibehalten als auch einen neuen wählen.
`nohup`	Die Ausführung eines Kommandos wird in den Hintergrund gestellt, so daß sie nach Ihrer Abmeldung vom System fortgesetzt werden kann. Die Fehlermeldungen und Ergebnisse werden in eine Datei namens `nohup.out` gesetzt.
`pg`	Der Inhalt einer bestimmten Datei wird seitenweise auf Ihrem Terminal ausgegeben. Nach jeder Seite legt das System eine Pause ein und wartet auf Ihre Anweisungen, bevor es fortfährt.
`pr`	Ausgabe einer teilweise formatierten Version einer bestimmten Datei auf Ihrem Temrinal. Das Kommando `pr`(1) zeigt zwar Seitenumbrüche an, enthält aber keines der Makros, wie es sie für Textformatierungssysteme gibt.
`ps`	Ausgabe von Status und Nummer aller aktuell laufenden Prozesse. Das Kommando `ps`(1) zeigt den Status der gerade aktiven Aufträge an, nicht aber denjenigen der Aufträge in der `at`(1)- oder `batch`(1)-Warteschlange.
`pwd`	Anzeige des absoluten Pfadnamens des aktuellen Arbeitsverzeichnisses.

`rm` Datei aus dem Dateisystem entfernen. Beim Kommando `rm`(1) sind Metazeichen zulässig; dabei ist jedoch Vorsicht geboten, da eine gelöschte Datei nicht wiederhergestellt werden kann.

`rmdir` Verzeichnis löschen. Das angegebene Verzeichnis darf nicht das aktuelle Verzeichnis sein. Außerdem kann nur ein Verzeichnis gelöscht werden, in dem keine Dateien mehr enthalten sind. Daher müssen Sie vor dem Aufrufen des Kommandos sämtliche Unterverzeichnisse und Dateien löschen, die gegebenenfalls noch im Verzeichnis enthalten sind (zum Löschen eines Verzeichnisses, das nicht leer ist, benutzen Sie die Option `-r` des Kommandos `rm`(1); siehe auch den entsprechenden Eintrag im *Referenzhandbuch für Benutzer*).

`sort` Sortieren einer Datei nach der Reihenfolge der ASCII-Tabelle und Anzeige des Ergebnisses auf dem Bildschirm. Die Sortierreihenfolge nach der ASCII- Tabelle sieht folgendermaßen aus:

1. Sonderzeichen
2. Zahlen vor Buchstaben
3. Großbuchstaben vor Kleinbuchstaben
4. Alphabetische Reihenfolge

Für das Sortieren einer Datei gibt es eine Reihe weiterer Optionen. Eine vollständige Übersicht über die `sort`(1)-Optionen finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `sort`(1).

`spell` Englische Rechtschreibprüfung. Die Wörter in einer bestimmten Datei werden mit einem (englischen) Wörterbuch verglichen. Sämtliche Wörter, die nicht im Wörterbuch enthalten sind oder (z.B. durch Suffixierung oder Präfigierung u.ä.) von Wörtern im Wörterbuch abgeleitet werden können, werden ausgegeben.

`stty` Anzeige der Konfigurationseinstellungen bestimmter Ein/Ausgabe- Optionen für Ihr Terminal. Wird das Kommando `stty`(1) mit den entsprechenden Optionen und Argumenten aufgerufen, so werden diese Ein-/Ausgabe-Optionen zusätzlich neu eingestellt (siehe auch Eintrag

	stty(1) im *Referenzhandbuch für Benutzer.*)
uname	Anzeige des Namens des UNIX-Systems, an dem Sie aktuell arbeiten.
uucp	Übermittlung einer bestimmten Datei an ein anderes UNIX-System (siehe auch Eintrag uucp(1) im *Referenzhandbuch für Benutzer*).
uuname	Auflistung der Namen der fernen UNIX-Systeme, die mit Ihrem UNIX-System Daten austauschen können.
uupick	Öffentliches Verzeichnis nach Dateien durchsuchen, die an Sie mit dem Kommando uuto(1) übermitteltworden sind. Ist eine Datei vorhanden, so zeigt uupick(1) ihren Namen sowie das System, von dem sie abgeschickt worden ist, an. Außerdem werden Sie (mit einem Fragezeichen (?) zur Durchführung einer Aktion aufgefordert.
uustat	Anzeige des Status des Kommandos uuto(1), mit dem Sie Dateien an andere Benutzer übermittelt haben.
uuto	Übermittlung einer bestimmten Datei an einen anderen Benutzer. Das Ziel wird im Format *system* ! *empfänger* angegeben. Das *system* muß auf der System-Liste des Kommandos uuname(1) enthalten sein.
vi	Editieren einer Datei mit dem bildschirmorientierten Editor vi(1). Ist keine Datei mit dem angegebenen Namen vorhanden, so wird sie von vi(1) angelegt (ausführliche Informationen über den Editor vi(1) finden Sie in Kapitel 7).
wc	Anzahl der Zeilen, Wörter und Zeichen in einer Datei zählen; Anzeige der Ergebnisse auf Ihrem Terminal.
who	Anzeige der Benutzernamen aller Benutzer, die aktuell an Ihrem UNIX-System angemeldet sind. Für jeden Benutzernamen wird die Adresse seines Terminals sowie der Zeitpunkt seiner Anmeldung aufgeführt.
yacc	Eingabe eines Programms mit einer Struktur versehen (weitere Informationen hierzu finden Sie im *Leitfaden für Programmierer*).

C FACE-Kurzübersicht

Einführung C-1

Kommandos und Funktionen für das Command Menu C-3
Stornieren eines Kommandos C-3
Hilfe über ein Kommando abrufen C-3
Inhalt des Command Menu abrufen C-3

Kommandos und Funktionen für Dateien und Kataloge C-4
Kopieren einer Datei bzw. eines Katalogs C-4
Anlegen von Dateien und Katalogen C-4
Löschen einer Datei bzw. eines Katalogs C-5
Inhalt einer Datei abrufen C-6
Absoluten Pfadnamen einer Datei bzw. eines Katalogs ausgeben C-7
Suchen einer bestimmten Datei bzw. eines bestimmten Katalogs C-7
Versetzen einer Datei C-8
Umordnen der Katalog-Übersicht C-9
Beschreibung einer Datei ändern C-10
Umbenennen einer Datei bzw. eines Katalogs C-11
Zugriffsberechtigungen für eine vorhandene Datei bzw. einen vorhandenen Katalog ändern C-11
Zugriffsberechtigungen für neue Dateien und Kataloge angeben C-12
Wiederherstellung einer gelöschten Datei bzw. eines gelöschten Katalogs C-13

Editieren einer Datei C-14
Druckausgabe einer Datei C-14

Kommandos und Funktionen für Ausschnitte C-16
Löschen eines Ausschnitts C-16
Löschen des FACE-Bildschirms C-16
Umschalten zu einem bestimmten Ausschnitt C-16
Informationen über den aktuellen Ausschnitt abrufen C-17
Inhalt eines Ausschnitts aktualisieren C-17
Ausschnitt versetzen oder vergrößern/verkleinern C-18

Das Menü Programs C-19
Eintragen eines benutzer-eigenen Programms C-19
Entfernen eines benutzer-eigenen Programms C-20
Aufrufen eines Programms C-21
Post abschicken C-21
Nachricht lesen C-23
Rechtschreibprüfung C-24

Verschiedene Funktionen C-25
Zugriff auf das FACE-Hilfesystem C-25
Beenden von FACE C-25
Wiederaufbau des Bildschirms C-25
Aufrufen einer Shell-Prozedur C-26
Aufrufen einer ausführbaren Datei C-26
Aufrufen des UNIX-Systems C-27
Verlassen des UNIX-Systems C-27
Vorübergehendes Verlassen einer Datei C-27
Rückkehr zu einer vorübergehend verlassenen Datei C-28

Einführung

Kapitel 5 enthielt eine Einführung in die Benutzung der FACE-Schnittstelle. Wenn Sie dieses Kapitel durchgearbeitet haben, kennen Sie die verschiedenen Möglichkeiten zum Ausführen eines Kommandos. Diese Möglichkeiten sind im folgenden nochmals als Gedächtnisstütze aufgeführt:

- Wählen Sie das Kommando im Command Menu. Sie werden gegebenenfalls zur Eingabe weiterer Informationen aufgefordert.
- Bewegen Sie den Cursor zu dem Kommando im Command Menu und betätigen Sie die Taste [CTRL-j]. Das Kommando erscheint in der Kommandozeile; nachdem Sie alle notwendigen Argumente eingegeben haben, betätigen Sie die Taste [ENTER]. Sie werden gegebenenfalls zur Eingabe weiterer Informationen aufgefordert.
- Betätigen Sie die Taste [CTRL-j], um auf die Kommandozeile zuzugreifen; dann geben Sie das Kommando mit Argumenten ein und betätigen die Taste [ENTER]. Sie werden gegebenenfalls zur Eingabe weiterer Informationen aufgefordert.
- Betätigen Sie die Funktionstaste (bzw. die entsprechende Ersatztaste), auf die das Kommando gelegt worden ist. Sie werden gegebenenfalls zur Eingabe weiterer erforderlicher Informationen aufgefordert.

Bei der Beschreibung eines Kommandoformats für die FACE- Kommandozeile wurden die folgenden Gestaltungsmittel benutzt:

- In `Computerschrift` (Schrift mit festem Zeichenabstand) wurden diejenigen Komponenten des Kommandos gedruckt, die exakt wie gezeigt eingegeben werden müssen.
- *Kursiv* wurden die Platzhalter des Kommandos gedruckt, d.h. diejenigen Komponenten, für die Sie einen Wert einsetzen müssen.
- In eckigen Klammern `[ ]` wurden optionale Optionen bzw. Argumente angegeben.
- Mehrere Punkte (`...`) zeigen, daß ein Argument bzw. eine Option mehr als einmal eingegeben werden kann.
- Die Eingabeaufforderung `-->` vor jedem Kommandobeispiel zeigt Ihnen, daß Sie das Kommando eingeben müssen, nachdem Sie auf die Kommandozeile über die Taste [CTRL-j] zugegriffen haben.

Zusammengehörige FACE-Kommandos werden in diesem Anhang nach Funktionsgruppen eingeteilt und beschrieben. Im einzelnen besteht dieser Anhang aus den folgenden Abschnitten:

- Kommandos und Funktionen für das Command Menu
- Kommandos und Funktionen für Dateien und Kataloge
- Kommandos und Funktionen für Ausschnitte
- Funktionen für Programme
- Verschiedene Funktionen

Für jedes Kommando und jede Funktion, das/die in diesen Abschnitten beschrieben wird, werden kurz die einzelnen Schritte aufgeführt, mit denen Sie es/sie über Funktionstasten, benannte Tasten bzw. FACE-Menüs durchführen können (die Ersatztasten werden für die Fälle aufgeführt, in denen die benutzte Tastatur nicht sämtliche benannten Tasten und Funktionstasten aufweist bzw. die angegebenen Tasten nicht funktionieren). Auf jede Kurzübersicht folgt ein Abschnitt mit dem Titel "Über die Kommandozeile", in dem die Durchführung derselben Funktion über die Kommandozeile beschrieben wird (falls sinnvoll).

Kommandos und Funktionen für das Command Menu

Stornieren eines Kommandos

1. Betätigen Sie die Taste [CANCEL] (bzw. [CTRL-f] [6]).

Hilfe über ein Kommando abrufen

1. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]).
2. Bewegen Sie den Cursor zu `command`.
3. Betätigen Sie die Taste [HELP] (bzw. [CTRL-f] [1]).
4. Wenn im rechten unteren Rand eine Verschiebeleiste erscheint, betätigen Sie die Taste [NEXTPAGE] (bzw. [CTRL-f] [3]).

Über die Kommandozeile

`-->help` *Kommando*

Inhalt des Command Menu abrufen

1. Betätigen Sie die Taste [CMD-MENU] (bzw. [CTRL-f] [7]).

Kommandos und Funktionen für Dateien und Kataloge

Kopieren einer Datei bzw. eines Katalogs

1. Bewegen Sie den Cursor zu der Datei bzw. dem Katalog, die/der kopiert werden soll.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]).
3. Betätigen Sie die Taste [COPY] (bzw. [CTRL-f] [2]).
4. Schalten Sie zum Ziel-Katalog um und betätigen Sie die Taste [ENTER].
5. Betätigen Sie die Taste [SELECT] (bzw. [CTRL-f] [8]).
6. Ist bereits eine Datei oder ein Katalog dieses Namens im Ziel-Katalog vorhanden, so müssen Sie für die Kopie einen neuen Namen eingeben und dann die Taste [ENTER] betätigen.

Über die Kommandozeile

-->copy *name* [[to]*ziel*[*name_neu*]]

Dabei ist *name* der Name der zu kopierenden Datei bzw. des zu kopierenden Katalogs, *ziel* der absolute Pfadname des Ziel-Katalogs. *name_neu* kann identisch sein mit *name*, vorausgesetzt, es ist noch keine Datei bzw. kein Katalog dieses Namens in *ziel* vorhanden. Fehlt die Angabe von *name_neu*, so wird der Kopie der Name *name* zugeordnet.

Anlegen von Dateien und Katalogen

1. Schalten Sie zu dem Katalog um, in dem Sie die neue Datei bzw. den neuen Katalog anlegen möchten.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]).
3. Betätigen Sie die Taste [CREATE] (bzw. [CTRL-f] [6]).
4. Geben Sie den Namen der neuen Datei bzw. des neuen Katalogs ein und betätigen Sie die Taste [ENTER].

5. Wenn Sie eine Datei anlegen, wählen Sie im Menü Choices die Option `Standard file:`
    1. Tragen Sie mit dem Editor, der beim Anlegen der Datei aufgerufen worden ist, Text ein.
    2. Speichern Sie die Datei ab und beenden Sie den Editor.
6. Beim Anlegen eines Katalogs wählen Sie im Menü Choices die Option `File folder.`
7. Die neue Datei bzw. der neue Katalog wird in dem Katalog aufgeführt, in dem Sie sie/ihn angelegt haben.

Über die Kommandozeile

`-->create` [*name_neu*] [[`in`] *ziel*]

Dabei ist *name_neu* der Name der neuen Datei bzw. des neuen Katalogs und *ziel* der Katalog, in dem sie/er angelegt wird. Fehlt die Angabe von *name_neu*, so werden Sie zur Eingabe eines Namens aufgefordert. Fehlt *ziel*, so wird *name_neu* im aktuellen Ausschnitt angelegt. Ist der aktuelle Ausschnitt kein Katalog, so wird *name_neu* im Katalog /home/login angelegt.

Löschen einer Datei bzw. eines Katalogs

1. Bewegen Sie den Cursor zu dem Namen der Datei bzw. des Katalogs, die/den Sie löschen möchten.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]).
3. Betätigen Sie die Taste [DELETE] (bzw. [CTRL-f] [4]).
4. Bei der Eingabeaufforderung in der Meldungszeile, die Ihnen signalisiert, daß FACE die angegebene Datei bzw. den angegebenen Katalog jetzt löscht, betätigen Sie die Taste [ENTER], um den Löschvorgang fortzusetzen. Sind in einem Katalog Dateien oder Kataloge enthalten, so stellt FACE eine entsprechende Warnung dar.

   Jetzt kommt es zu einer der folgenden Aktionen:

1. Die Datei bzw. der Katalog wird gelöscht (in den Katalog WASTEBASKET versetzt).
2. Wenn in WASTEBASKET bereits eine Datei bzw. ein Katalog gleichen Namens enthalten ist, werden Sie zur Eingabe eines neuen Namens für die Datei bzw. den Katalog in WASTEBASKET aufgefordert.
3. Beim Versuch, einen aktuell eröffneten Katalog zu löschen, bzw. einen Katalog, der eine aktuell eröffnete Datei bzw. einen aktuell eröffneten Katalog *enthält*, wird eine Warnung ausgegeben; Sie müssen die eröffnete Datei bzw. den eröffneten Katalog dann schließen.
4. Eine Datei bzw. ein Katalog, die/der bereits im Katalog WASTEBASKET enthalten ist, wird unwiderruflich gelöscht, wenn Sie für sie/ihn das Kommando `delete` ausführen.

Über die Kommandozeile

`-->delete` [*name*]

Fehlt *name*, so wird die Datei bzw. der Katalog gelöscht, bei der/dem der Cursor sich aktuell befindet.

Inhalt einer Datei abrufen

1. Bewegen Sie den Cursor zu dem Namen der Datei, deren Inhalt Sie abrufen möchten.
2. Wählen Sie im Command Menu die Option `display`.

Über die Kommandozeile

`-->display` [*dateiname*]

Dabei ist *dateiname* der Name einer Standard-Datei im aktuellen Katalog bzw. der absolute Pfadname einer Standard-Datei, die nicht im aktuellen Katalog enthalten ist. Fehlt *dateiname*, so wird standardmäßig die aktuelle Datei ausgegeben.

Absoluten Pfadnamen einer Datei bzw. eines Katalogs ausgeben

1. Bewegen Sie den Cursor zu dem Namen der Datei/des Katalogs, deren/dessen absoluten Pfadnamen Sie auf dem Bildschirm abrufen möchten.
2. Betätigen Sie die Taste CMD-MENU bzw. CTRL-f 7, um das Command Menu aufzurufen.
3. Wählen Sie die Option `show-path`.
4. Auf dem Bildschirm wird jetzt in einem Ausschnitt der absolute Pfadname der Datei bzw. des Katalogs angezeigt. Betätigen Sie die Taste CANCEL bzw. CTRL-f 7, um zum vorherigen Ausschnitt zurückzukehren.

Über die Kommandozeile

```
-->show-path
```

Suchen einer bestimmten Datei bzw. eines bestimmten Katalogs

1. Schalten Sie zu dem Katalog um, in dem der Suchvorgang *einsetzen* soll (mit dem Kommando `find` können Sie ausschließlich Kataloge unterhalb des aktuellen Katalogs durchsuchen).
2. Wählen Sie im Command Menu die Option `find`.
3. Auf dem Bildschirm wird eine Find-Schablone angezeigt, für deren vier Suchkriterien bereits Standardwerte eingetragen sind:

   Name: (Datei- oder Katalogname)
   Type: (Ein zulässiger Dateityp)
   Owner: (Benutzername des Eigentümers; als Eigentümer können Sie dieses Feld leer lassen)
   Age: (Anzahl der Tage)

1. Soll der Suchvorgang mit den Standardwerten durchgeführt werden, so betätigen Sie die Taste [ENTER]. In diesem Fall werden sämtliche Dateien und Kataloge unterhalb Ihres aktuellen Katalogs aufgelistet.
2. Wenn Sie die Suchkriterien näher eingrenzen möchten, so tragen Sie in ein oder mehreren Feldern einen neuen Wert ein, oder betätigen Sie die Taste [CHOICES] (bzw. [CTRL-f] [2]) Sie erhalten für das aktuelle Feld eine Übersicht über die zulässigen Einträge. Wählen Sie über die Cursor-Tasten für Schablonen und die Taste [ENTER] einen neuen Wert. Die eingetragenen Informationen müssen sich nur auf die gesuchte(n) Datei(en) beziehen.

4. Betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um `find` mit Ihren Suchkriterien aufzurufen.

Über die Kommandozeile

`-->find` [*name*] [[`in`] *katalog_name*]

Dabei ist *name* der Name der gesuchten Datei bzw. des gesuchten Katalogs und *katalog_name* der Katalog, in dem der Suchvorgang gestartet werden soll. Fehlt *katalog_name*, so beginnt der Suchvorgang im aktuellen Katalog (bzw. im Katalog /home/login, wenn Sie sich nicht in einem Katalog befinden).

Versetzen einer Datei

1. Bewegen Sie den Cursor zu dem Namen der Datei bzw. des Katalogs, die/der versetzt werden soll.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]).
3. Betätigen Sie die Taste [MOVE] (bzw. [CTRL-f] [3]).
4. Schalten Sie zum Ziel-Katalog um.
5. Betätigen Sie die Taste [SELECT] (bzw. [CTRL-f] [8]).

   Wenn im Ziel-Katalog bereits eine Datei bzw. ein Katalog gleichen Namens enthalten ist, so werden Sie zur Eingabe eines neuen Namens für die zu versetztende Datei bzw. den zu versetztenden Katalog

aufgefordert.

Über die Kommandozeile

`-->move` [*name*] [[`to`] *ziel* [*name_neu*]]

Dabei ist *name* der Name der zu versetzenden Datei bzw. des zu versetzenden Katalogs, und *ziel* der Katalog, in den Sie die Datei bzw. den Katalog versetzen möchten. Fehlt *name*, so wird die aktuelle Datei bzw. der aktuelle Katalog versetzt. Fehlt *ziel*, so werden Sie zur Eingabe eines Katalognamens aufgefordert. Fehlt *name_neu*, so erhält die Datei bzw. der Katalog den Namen *name*.

Umordnen der Katalog-Übersicht

1. Bewegen Sie den Cursor zu dem Namen des Katalogs, den Sie umordnen möchten.
2. Wählen Sie im Command Menu die Option `organize`.
3. Auf dem Bildschirm wird eine Organize-Schablone mit drei Feldern angezeigt. Die Standardwerte dieser drei Felder können Sie folgendermaßen ändern:

   Default Organization: Betätigen Sie die Taste CHOICES (bzw. CTRL-f 2), bis der gewünschte Wert im Feld erscheint. Enthält dieses Feld den Wert `no`, so hängt die Anordnung der Katalog-Übersicht von den nächsten beiden Feldern in dieser Schablone ab. Mit dem Wert `yes` wird die Katalogübersicht den Standardwerten in der Schablone Preferences entsprechend angezeigt.

   Folder Display Format: Betätigen Sie die Taste CHOICES (bzw. CTRL-f 2). Auf dem Bildschirm erscheint das Menü Choices Wählen Sie den gewünschten Wert.

   Folder Display Order: Betätigen Sie die Taste CHOICES (bzw. CTRL-f 2). Auf dem Bildschirm erscheint das Menü Choices. Wählen Sie den gewünschten Wert.

4. Betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um die Anordnung der Katalog-Übersicht zu ändern.

Über die Kommandozeile

`-->organize` [*ausschnitt_nr*]

Dabei kann *ausschnitt_nr* der absolute Pfadname des Ausschnitts bzw. die Ausschnitt-Kennung sein. Fehlt *ausschnitt_nr*, so wird vom aktiven Ausschnitt ausgegangen.

Beschreibung einer Datei ändern

1. Bewegen Sie den Cursor zu der Datei bzw. dem Katalog, deren/dessen Beschreibung Sie ändern möchten.
2. Wählen Sie im Command Menu die Option `redescribe`.
3. Geben Sie die neue Beschreibung bei der entsprechenden Eingabeaufforderung in der Kommandozeile ein und betätigen Sie die Taste [ENTER].

Über die Kommandozeile

`-- >redescribe` [*name* [*beschr_neu*]]

Dabei ist *name* der Name der Datei bzw. des Katalogs, deren/dessen Beschreibung Sie ändern möchten, und *beschr_neu* eine Beschreibung aus bis zu 23 Zeichen (einschließlich Leerzeichen). Werden keine Argumente eingegeben, so fordert `redescribe` Sie zur Eingabe einer Beschreibung für die aktuelle Datei bzw. den aktuellen Katalog auf. Fehlt *beschr_neu*, so werden Sie zur Eingabe einer neuen Beschreibung aufgefordert.

Umbenennen einer Datei bzw. eines Katalogs

1. Bewegen Sie den Cursor zu dem Namen der Datei bzw. des Katalogs, die/den Sie umbenennen möchten.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]) und anschließend die Taste [RENAME] (bzw. [CTRL-f] [5]).
3. Bei der Eingabeaufforderung in der Kommandozeile geben Sie den neuen Namen der Datei bzw. des Katalogs ein und betätigen dann die Taste [ENTER].

Über die Kommandozeile

`-- >rename` [*name_alt*] [[`to`] *name_neu*]

Wenn Sie überhaupt keine Argumente eingeben, so werden Sie zur Eingabe eines *name_neu* für die aktuelle Datei bzw. den aktuellen Katalog aufgefordert. Fehlt *name_neu*, so werden Sie zur Eingabe eines neuen Namens aufgefordert.

Zugriffsberechtigungen für eine vorhandene Datei bzw. einen vorhandenen Katalog ändern

1. Bewegen Sie den Cursor zum Namen der Datei bzw. des Katalogs, deren/dessen Zugriffsberechtigungen Sie ändern möchten.
2. Betätigen Sie die Taste [CHG-KEYS] (bzw. [CTRL-f] [8]) und anschließend die Taste [SECURITY] (bzw. [CTRL-f] [7]).
3. In der Security-Schablone können Sie jetzt die Zugriffsberechtigungen für sich selbst, die Mitglieder Ihrer Gruppe und die anderen Benutzer auf Ihrem Rechner ändern (vorausgesetzt, Sie sind der Eigentümer). Sind Sie nicht der Eigentümer, so können Sie sich die Schablone lediglich ansehen.

   Benutzen Sie die Cursorsteuerungs- und Editiertasten für Schablonen.
4. Nach der Änderung der Zugriffsberechtigungen gehen Sie folgendermaßen vor:
   1. Betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um die neuen Zugriffsberechtigungen für diese Datei bzw. diesen Katalog abzuspeichern.

ODER

2. Betätigen Sie die Taste CANCEL (bzw. CTRL-f 6), wenn Sie bei den Standard-Werten bleiben möchten.

Über die Kommandozeile

`-->security [name]`

Fehlt *name*, so wird automatisch die aktuelle Datei bzw. der aktuelle Katalog eingesetzt.

Zugriffsberechtigungen für neue Dateien und Kataloge angeben

1. Wählen Sie im Menü Office die Option `Preferences`.
2. Wählen Sie im Menü Preferences die Option `File Permissions`
3. In der Security-Schablone können Sie jetzt die Zugriffsberechtigungen für sich selbst, die Mitglieder Ihrer Gruppe und die übrigen Rechnerbenutzer angeben.

   Benutzen Sie die Cursorsteuerungs- und Editiertasten für Schablonen.
4. Nach der Änderung der Zugriffsberechtigungen für die neuen Dateien und Kataloge gehen Sie folgendermaßen vor:
   1. Betätigen Sie die Taste SAVE (bzw. CTRL-f 3) um die neuen Zugriffsberechtigungen für diese Datei bzw. diesen Katalog abzuspeichern.

      ODER
   2. Betätigen Sie die Taste CANCEL (bzw. CTRL-f 6) wenn Sie bei den Standard-Werten bleiben möchten.

Über die Kommandozeile

```
-->preferences
```

Hiermit können Sie auf das Menü Preferences ohne Umweg über das Menü Office of login zugreifen.

Wiederherstellung einer gelöschten Datei bzw. eines gelöschten Katalogs

1. Wählen Sie im Menü Office die Option `Wastebasket`.
2. Bewegen Sie den Cursor zu der Datei bzw. dem Katalog im Katalog WASTEBASKET, die/den Sie wiederherstellen möchten.
3. Wählen Sie im Command Menu die Option `undelete`, um die Datei bzw. den Katalog wieder in dem Katalog wiederherzustellen, in dem sie/er gelöscht worden war (wenn Sie den ursprünglichen Katalog jedoch umbenannt oder gelöscht haben, so können Sie die Datei bzw. den Katalog nicht mit `undelete` in diesem Katalog wiederherstellen).

Über die Kommandozeile

`-->undelete` [*name*]

Dabei ist *name* der Name einer Datei bzw. eines Katalogs, die/der aktuell im Katalog WASTEBASKET enthalten ist, bzw. (wenn Sie sich gerade in einem anderen Ausschnitt befinden) der absolute Dateiname der Datei bzw. des Katalogs im Katalog WASTEBASKET. Fehlt *name*, so wird die aktuelle Datei bzw. der aktuelle Katalog eingesetzt. Das Kommando `undelete` kann für Dateien bzw. Kataloge im Katalog WASTEBASKET durchgeführt werden, die dorthin mit dem Kommando `delete`, `move` oder `copy` gesetzt worden sind.

Editieren einer Datei

1. Bewegen Sie den Cursor zu der Datei, die Sie editieren möchten, und betätigen Sie die Taste ENTER.
2. Editieren Sie die Datei mit dem Editor, der für Sie automatisch aufgerufen wird (Standard: `vi`).
3. Speichern Sie Ihre Änderungen mit dem geeigneten Editor-Kommando ab und verlassen Sie die Datei.

Hinweis: Wenn Sie versehentlich eine Datei gewählt haben und den Editor `vi` nicht bedienen können, so geben Sie `:q!` ein, um die Datei unverändert zu verlassen und zum FACE-Office zurückzukehren. Lassen Sie sich dann von einem erfahrenen `vi`- Benutzer in die Bedienung dieses Editors einweisen.

Über die Kommandozeile

-->dateiname

Dabei ist *dateiname* der Name der Standard-Datei (falls Sie sich in demselben Katalog befinden) bzw. der absolute Pfadname der Datei (wenn Sie sich in einem anderen Katalog befinden). Zum Editieren der Datei *dateiname* wird standardmäßig der Editor `vi` aufgerufen bzw. der Editor, den Sie in der Schablone Preferences angegeben haben (siehe Abschnitt "Einstellung Ihrer Office-Parameter" in Kapitel 5 dieses Leitfadens).

Druckausgabe einer Datei

1. Bewegen Sie den Cursor zu der auszudruckenden Datei.
2. Wählen Sie im Command Menu die Option `print`.
3. Wählen Sie eines der drei Druck-Kommandos, die im Print-Menü angezeigt werden.

   Der Bildschirm wird jetzt gelöscht, und Sie werden in einer Meldung über die Auftragsnummer ihres Druckauftrags informiert.

4. Betätigen Sie die Taste ENTER, um zu Ihrem FACE-Office zurückzukehren.

Über die Kommandozeile

`-->print` [*dateiname*]

Dabei ist *dateiname* der Name der Datei, wenn sie sich in Ihrem aktuellen Katalog befindet, andernfalls der absolute Pfadname der Datei. Fehlt *dateiname*, so geht `print` von der aktuellen Datei aus.

Kommandos und Funktionen für Ausschnitte

Löschen eines Ausschnitts

1. Betätigen Sie die Taste CANCEL (bzw. CTRL-f 6), um den aktuellen Ausschnitt zu löschen.

Über die Kommandozeile

`-->cancel` [*ausschnitt_nr* ...]

Dabei ist *ausschnitt_nr* die Nummer des Ausschnitts, den Sie schließen möchten (links neben dem Ausschnitt-Titel). Beim Kommando `cancel` können Sie als Argumente bis zu 24 Ausschnitt-Nummern angeben. Fehlt *ausschnitt_nr*, so wird der aktuelle Ausschnitt eingesetzt.

Löschen des FACE-Bildschirms

1. Wählen Sie im Command Menu die Option `cleanup`.

   Hiermit schließen Sie sämtliche eröffneten Ausschnitte mit Ausnahme des Menüs AT&T FACE sowie sämtliche sonstigen Kataloge, die bei Ihrer Anmeldung automatisch eröffnet werden.

Über die Kommandozeile

```
-->cleanup
```

Umschalten zu einem bestimmten Ausschnitt

1. Wählen Sie im Command Menu die Option `goto`.
2. Bei der Eingabeaufforderung in der Kommandozeile geben Sie die Ausschnitt-Nummer bzw. den Pfadnamen des Ausschnitt-Titels eines beliebigen eröffneten Ausschnitts ein und betätigen dann die Taste ENTER.

Über die Kommandozeile

`-->goto` [*ausschnitt_nr*]

Dabei kann *ausschnitt_nr* entweder die Ausschnitt-Nummer oder der Pfadname des betreffenden Ausschnitts sein. Die *ausschnitt_nr* darf keine Leerzeichen enthalten. Fehlt *ausschnitt_nr*, so gibt `goto` eine entsprechende Eingabeaufforderung aus.

Informationen über den aktuellen Ausschnitt abrufen

1. Betätigen Sie die Taste HELP (bzw. CTRL-f 1).

Über die Kommandozeile

`-->help`

Hiermit erhalten Sie kontext-abhängige Hilfe über den aktuellen Ausschnitt bzw. über die Funktion, die Sie gerade durchführen.

Inhalt eines Ausschnitts aktualisieren

1. Wählen Sie in einem eröffneten Ausschnitt im Command Menu die Option `update`, wenn Sie einen Überblick über die Änderungen erhalten möchten, die Sie an seinem Inhalt vorgenommen haben.

Über die Kommandozeile

`-->update` [*ausschnitt_nr*]

Dabei ist *ausschnitt_nr* der absolute Pfadname bzw. die Nummer eines Ausschnitts. Fehlt *ausschnitt_nr*, so wird der aktuelle Ausschnitt eingesetzt.

Ausschnitt versetzen oder vergrößern/verkleinern

1. Schalten Sie zu dem betreffenden Ausschnitt um.
2. Wählen Sie im Command Menu die Option `frm-mgmt`.
3. Auf dem Bildschirm erscheint das Menü Frame Management.
    1. Wählen Sie die Option `move`, wenn Sie den Ausschnitt zwar an eine neue Position versetzen möchten, seine ursprüngliche Größe aber erhalten bleiben soll.
    2. Wählen Sie die Option `reshape`, wenn Sie die Größe des aktuellen Ausschnitts ändern und/oder ihn versetzen möchten. Beachten Sie, daß nur Text- und Menüausschnitte vergrößert bzw. verkleinert werden können.
4. Über die Pfeiltasten (bzw. ihre Ersatztasten) können Sie den Ausschnitt an die neue Position versetzen bzw. seine Größe ändern.

Über die Kommandozeile

`-->frm-mgmt` [*aktion*]

Dabei kann *aktion* das Kommando `list`, `move`, oder `reshape` sein. Fehlt *aktion*, so wird das Menü Frame Management aufgerufen. Die *aktion*en werden automatisch für den aktuellen Ausschnitt durchgeführt.

Bei der Wahl der Aktion `list` erhalten Sie eine Übersicht über sämtliche eröffneten Ausschnitte und vorübergehend verlassenen Dateien. Bei der Wahl der Aktion `move` und `reshape` können Sie einen Ausschnitt vorübergehend versetzen und seine Größe ändern. Die Position und die Größe des Ausschnitts wird wieder auf die FACE-Standardeinstellung zurückgesetzt, sobald Sie den Ausschnitt schließen oder sich abmelden. Ein Menü- oder Textausschnitt, für den die Aktion `reshape` durchgeführt worden ist, wird bei jedem Aktualisierungs-Kommando in seiner ursprünglichen Größe wiederhergestellt.

Das Menü Programs

Eintragen eines benutzer-eigenen Programms

1. Hiermit können Sie auf Ihrem Rechner eine ausführbare Datei (d.h. ein Programm) anlegen oder installieren.
2. Wählen Sie im Menü Office of login die Option `Programs Administration`.
3. Wählen Sie im Menü Programs Administration die Option `Add Programs`.
4. Auf dem Bildschirm erscheint die Schablone Add Programs. Bewegen Sie den Cursor zum Feld `Program Menu Name:`.
5. Geben Sie den Namen ein, unter dem das Programm im Menü Programs aufgeführt sein soll und betätigen Sie die Taste [ENTER].
6. Bewegen Sie den Cursor zum Feld `Name of Command:`.
7. Geben Sie den absoluten Pfadnamen der ausführbaren Datei ein.
8. Bewegen Sie den Cursor zum Feld `Working Directory:`.
9. Geben Sie den absoluten Pfadnamen des Verzeichnisses (des Katalogs) ein, in das (den) durch das Programm angelegte Dateien gesetzt werden sollen. Wenn die Dateien auf jeden Fall in den aktuellen Katalog gesetzt werden sollen, so geben Sie einen Punkt (.) ein.
10. Bewegen Sie den Cursor zum Feld `Prompt for Arguments:`.

    Wenn die Benutzer beim Aufrufen dieses Programms Argumente eingeben dürfen, so geben Sie `yes` ein, andernfalls `no`.
11. Speichern Sie die Werte über die Taste [SAVE] (bzw. [CTRL-f] [3]) ab.

    Das Menü Programs ist beim nächsten Aufrufen durch das neue Programm erweitert. Wenn es gerade eröffnet ist und Sie es sofort aktualisieren möchten, so schalten Sie in dieses Menü um und rufen das Kommando `update` auf.

Hinweis: Nachfolgend wird erklärt, wie Sie ein benutzer-eigenes Programm ändern können, das Sie in das Menü Programs eingetragen haben (indem Sie Felder in der Schablone Add Program geändert haben). Die Änderung einer durch das Programm aufgerufenen Shell-Prozedur wird dagegen nicht beschrieben. Um eine Datei zu ändern, die eine Shell- Prozedur enthält, rufen Sie die Datei einfach mit einem Editor auf und ändern die Prozedur

ab.

1. Wählen Sie im Menü Office of login die Option `Programs Administration`.
2. Wählen Sie im Menü Programs Administration die Option `Modify Programs`.
3. Wählen Sie im Menü Personal Programs das Programm aus, das Sie ändern möchten.
4. Auf dem Bildschirm erscheint die Schablone Modify Program. Ändern Sie die Felder über die Cursorsteuerungs- und Editier-Tasten für Schablonen ab.
5. Nach Abschluß der Änderungen betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]).

   Das Menü Programs enthält beim nächsten Aufrufen das neue Programm. Wenn das Menü bereits eröffnet ist und Sie es sofort aktualisieren möchten, so schalten Sie zu diesem Menü um und rufen das Kommando `update` auf.

Entfernen eines benutzer-eigenen Programms

Wählen Sie im Menü Office of login die Option `Programs Administration`

1. Wählen Sie im Menü Programs Administration die Option `Remove Programs`.
2. Wählen Sie das Programm, das Sie aus dem Menü Personal Programs entfernen möchten.

   Auf dem Bildschirm erscheint ein Ausschnitt, in dem Sie zur Eingabe einer Bestätigung aufgefordert werden.
3. Betätigen Sie die Taste [CONT] (bzw. [CTRL-f] [3]), wenn Sie das Programm entfernen möchten, andernfalls [CANCEL] (bzw. [CTRL-f] [6]) (das Programm wird dann nicht gelöscht).

Das Menü Programs enthält beim nächsten Aufrufen nicht mehr das gelöschte Programm. Wenn das Menü aktuell eröffnet ist und Sie es sofort aktualisieren möchten, so schalten Sie zu diesem Menü um und wählen das Kommando `update`.

Aufrufen eines Programms

1. Wählen Sie im Menü FACE die Option `Programs`.
2. Wählen Sie das Programm, das Sie im Menü Programs aufrufen möchten.

   HINWEIS: In diesem Leitfaden werden nur die Programme `Mail Services` und `Spell Checker` beschrieben (siehe Kapitel 5); die Bedienung der anderen Programme sollten Sie sich von Ihrem Systemverwalter oder einem erfahrenen Benutzer erklären lassen.

Über die Kommandozeile

```
-->programs
```

Hiermit wird das Menü Programs eröffnet, in dem Sie das gewünschte Programm auswählen und somit aufrufen können.

Post abschicken

1. Wählen Sie im Menü FACE die Option `Programs`.
2. Wählen Sie im Menü Programs die Option `Mail Services`.
3. Wählen Sie im Menü Mail Services die Option `send mail`.
   1. Geben Sie im Feld `To:` den Benutzernamen des Empfängers an Ihrem oder einem anderen Rechner ein (im letzteren Fall müssen Sie den Empfänger im Format *system*!*benutzername* eingeben). Bei mehreren Empfängern müssen Sie die einzelnen Benutzernamen durch ein Leerzeichen voneinander trennen.
   2. Sie können aber auch über die Taste [CHOICES] (bzw. [CTRL-f] [2]) ein Menü mit den Benutzernamen der Benutzer auf Ihrem Rechner abrufen, in dem Sie den betreffenden Benutzernamen auswählen können. Arbeiten an Ihrem Rechner

weniger als vier weitere Benutzer, so erscheinen die zur Wahl stehenden Benutzernamen im Feld selbst; Sie können den Cursor dann über die Taste [CHOICES] zu den verschiedenen Benutzernamen bewegen.

4. Nachdem Sie den Benutzernamen des Empfängers eingegeben bzw. ausgewählt haben, betätigen Sie die Taste [SAVE] (bzw. [CTRL-f] [3]), um diesen Wert im Feld `To:` abzuspeichern.
5. Der Bildschirm wird jetzt gelöscht, und das UNIX-Kommando `mailx` übernimmt die Kontrolle.

   Hinweis: Zur Stornierung des Kommandos geben Sie `~x` ein.

6. Tragen Sie den Titel Ihrer Nachricht in das Feld `Subject` ein, oder überspringen Sie es über die Taste [ENTER].
7. Geben Sie den Nachrichtentext ein.

   Informationen darüber, wie Sie eine bereits abgeschickte Nachricht editieren können, finden Sie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mailx`.
8. Betätigen Sie die Taste [ENTER], um in eine neue Zeile zu springen, dann die Taste [CTRL-d] um die Nachricht abzuschicken.
9. Über die Taste [ENTER] kehren Sie zu FACE zurück.

Über die Kommandozeile

`-->!mailx` *empfänger ...*

Dabei zeigt ein Ausrufezeichen (`!`), daß das darauffolgende Kommando ein UNIX-Kommando ist; *empfänger* ist der Benutzername des Empfängers dieser Nachricht. Beachten Sie, daß Sie mehr als einen Empfänger in der `mailx`-Kommandozeile angeben können.

Nachricht lesen

1. Wählen Sie im Menü Office die Option `Programs`.
2. Wählen Sie im Menü Programs die Option `Mail Services`.
3. Wählen Sie im Menü Mail Services die Option `New Mail`.

   Sind Nachrichten vorhanden, die Sie noch nicht gelesen haben, so wird der Bildschirm gelöscht.
4. Nachdem eine Übersicht über die Nachrichten-Köpfe ausgegeben worden ist, erscheint die Eingabeaufforderung `?`.

   Am Ende jeder angezeigten Nachricht steht die Eingabeaufforderung `?`. Ihnen stehen jetzt die folgenden Eingabemöglichkeiten zur Wahl:

   1. Betätigen Sie die Taste ENTER, wenn Sie die nächste Nachricht lesen möchten (bzw. die erste Nachricht, wenn Sie sich beim letzten Nachrichten-Kopf befinden).
   2. Geben Sie *n* ein, wobei *n* für die Nummer einer bestimmten Nachricht steht, die Sie lesen möchten.
   3. Geben Sie `s` ein, um die Nachricht in der Datei `mbox` in `$HOME` abzuspeichern.
   4. Geben Sie `s` *dateiname* ein, um die Nachricht in der Datei *dateiname* in `$HOME` abzuspeichern.
   5. Geben Sie `d` ein, wenn Sie die aktuelle Nachricht löschen möchten. Geben Sie `d` *n* ein, wenn Sie die Nachricht mit der Nummer *n* löschen möchten (dabei muß es sich nicht um die soeben gelesene Nachricht handeln). Geben Sie `d` *n-n* ein, wenn Sie bestimmte Nachrichten im Bereich *n* bis *n* löschen möchten.
   6. Geben Sie `quit` ein, wenn Sie das `Lesen der Nachrichten` beenden möchten.
5. Über die Taste ENTER kehren Sie zum Menü Mail Services zurück.

Ausführliche Informationen über das Kommando `mailx` finden Sie in Kapitel 11 dieses Leitfadens, "Die elektronische Post", sowie im *Referenzhandbuch für Benutzer* unter dem Eintrag `mailx`(1).

Über die Kommandozeile

```
-->!mailx
```

Dabei zeigen Sie mit dem Ausrufezeichen (`!`), daß das nachfolgende Kommando ein UNIX-Kommando ist; `mailx` ist das Kommando, das bei der Auswahl von `read mail` im Menü Mail Programs aufgerufen wird.

Rechtschreibprüfung

1. Wählen Sie im Menü Office of login die Option `Programs` .
2. Wählen Sie im Menü Programs die Option `Spell Checker` .
3. Auf dem Bildschirm erscheint eine Schablone mit dem Titel `Spell Checker`; Sie werden jetzt zur Eingabe des absoluten Pfadnamens der Datei aufgefordert, die Sie auf (englische) Rechtschreibfehler überprüfen möchten.

   Geben Sie den absoluten Pfadnamen der Datei ein und betätigen Sie die Taste [SAVE], um die Rechtschreibprüfung aufzurufen.
4. Auf dem Bildschirm erscheint ein Ausschnitt mit dem Titel `Spell Checker Output`, in dem Schreibfehler aufgeführt sind (wenn keine Schreibfehler gefunden wurden, wird die entsprechende Meldung angezeigt).

Verschiedene Funktionen

Zugriff auf das FACE-Hilfesystem

1. Betätigen Sie die Taste [HELP] (bzw. [CTRL-f] [1]), um Informationen über den jeweils aktuellen Ausschnitt abzurufen.
2. Wenn Sie die Taste [CONTENTS] (bzw. [CTRL-f] [8]) betätigen, wird eine Inhaltsübersicht (Menü Table Of Contents) mit weiteren Themen angezeigt, über die Sie Informationen abrufen können.

   Über die meisten Menüoptionen erhalten Sie allgemeine Informationen über FACE. Bei der Wahl von `Commands Overview` dagegen wird ein Menü mit FACE-Kommandos angezeigt; wenn Sie eines dieser Kommandos auswählen, erhalten Sie Informationen über seine Benutzung.

Beenden von FACE

1. Wählen Sie im Menü Office die Option `Exit FACE`, wenn Sie FACE beenden *und* sich vom Rechner abmelden möchten.

Über die Kommandozeile

```
-->exit
```

Wiederaufbau des Bildschirms

1. Bei einer gestörten Bildschirmanzeige wählen Sie im Command Menu die Option `refresh`, um den Terminal-Bildschirm neu aufzubauen.

   Anders als beim Kommando `cleanup` werden bei `refresh` keine Ausschnitte geschlossen.

```
-->refresh
```

Aufrufen einer Shell-Prozedur

1. Bewegen Sie den Cursor zu der Shell-Prozedur, die Sie aufrufen möchten.
2. Wählen Sie im Command Menu die Option `run`, um die Shell-Prozedur aufzurufen.

   HINWEIS: Wenn Sie die Shell-Prozedur nicht aufrufen können, so wählen Sie im Command Menu die Option `security`, um sicherzustellen, daß die Such-/Ausführungsberechtigung des Eigentümers für diese Datei auf `yes` ist.
3. Nach der Beendigung der Shell-Prozedur betätigen Sie die Taste (ENTER), um zu FACE zurückzukehren.

Über die Kommandozeile

`-->run` [*dateiname*]

Dabei ist *dateiname* der Name der Shell-Prozedur, die Sie aufrufen möchten. Fehlt *dateiname,* so benutzt `run` automatisch die aktuelle Datei.

Aufrufen einer ausführbaren Datei

1. Bewegen Sie den Cursor zu der ausführbaren Datei, die Sie aufrufen möchten, und betätigen Sie die Taste (ENTER).
2. Sobald die Durchführung der ausführbaren Datei beendet ist, betätigen Sie die Taste (ENTER), um zu FACE zurückzukehren.

Über die Kommandozeile

-->dateiname

Dabei ist *dateiname* der Name der ausführbaren Datei, die Sie aufrufen möchten.

Aufrufen des UNIX-Systems

1. Wählen Sie im Menü Office die Option `UNIX System`.

Über die Kommandozeile

```
-->unix-system
```

Verlassen des UNIX-Systems

1. Geben Sie bei der Eingabeaufforderung `UNIX:` das Kommando `exit` ein oder betätigen Sie die Taste CTRL-d.
2. Betätigen Sie die Taste ENTER, um zu FACE zurückzukehren.

Vorübergehendes Verlassen einer Datei

Hinweis: Eine Datei können Sie nur dann vorübergehend verlassen, wenn Ihr Editor eine Möglichkeit zur Ausführung von UNIX- Kommandos aufweist; zu diesen Editoren gehört z.B. `vi`.

Wenn Sie eine Datei vorübergehend verlassen können, hat dies für Sie den Vorteil, daß Sie andere Dateien abrufen und andere Arbeiten erledigen können, ohne die editierte Datei abzuspeichern und zu verlassen.

1. Geben Sie `:!facesuspend` ein und betätigen Sie die Taste ENTER.

   Sie kehren jetzt zur FACE-Schnittstelle zurück und können vor der Rückkehr zu der vorübergehend verlassenen Datei andere Aufgaben erledigen.

Sie können bis zu fünf Dateien gleichzeitig vorübergehend verlassen.

Rückkehr zu einer vorübergehend verlassenen Datei

1. Wählen Sie im Command Menu die Option `frm-mgmt`.
2. Wählen Sie im Menü Frame Management die Option `list` .

   Auf dem Bildschirm erscheint das Menü Open Frames mit einer Übersicht über sämtliche eröffneten Ausschnitte und sämtliche vorübergehend verlassenen Dateien.
3. Wählen Sie die Datei, zu der Sie zurückkehren möchten.

D Kurzübersicht über die ed-Kommandos

ed-Kurzübersicht D-1
Grundlegende Kommandos D-1
Kommandos zur Zeilenadressierung D-2
- Anzeige-Kommandos D-2
- Eingabe-Kommandos D-3
- Löschen von Text D-3
- Ersetzen von Text D-3
- Sonderzeichen zum Mustervergleich D-3
- Kommandos zum Versetzen von Text D-4

Weitere nützliche Kommandos und Informationen D-5

ed-Kurzübersicht

Ein ed-Kommando hat folgendes Format:

[adresse1,adresse2]kommando[parameter]...<CR>

adresse1 und *adresse2* geben dabei die Zeilenadressen an, *parameter* die Daten, die vom Kommando verarbeitet werden. Die Kommandos werden bei der Eingabe auf dem Terminal angezeigt. Ausführliche Informationen zur Benutzung der ed-Kommandos finden Sie in Kapitel 6, "Der Zeileneditor ed."

Im folgenden finden Sie Kurzbeschreibungen der ed-Kommandos. Die Kommandos sind nach ihrer Funktion in Gruppen eingeteilt.

Grundlegende Kommandos

ed *dateiname*	Der Zeileneditor ed wird mit einer bestimmten Datei aufgerufen.
a	Unterhalb der aktuellen Zeile wird Text eingefügt.
.	Der Eingabemodus wird verlassen und in den Kommandomodus zurückgeschaltet.
p	Die aktuelle Zeile wird ausgegeben.
d	Die aktuelle Zeile wird gelöscht.
<CR>	Die nächste Zeile im Puffer wird angezeigt.
–	Die vorhergehende Zeile im Puffer wird angezeigt.
w	Der Puffer-Inhalt wird in die Datei eingebracht, die dem Puffer aktuell zugeordnet ist.
q	Die Editor-Sitzung wird beendet. Sind Änderungen, die am Puffer vorgenommen worden sind, nicht zuvor in eine Datei eingebracht worden, wird eine Warnung in Form eines Fragezeichens (?) ausgegeben. Durch die erneute Eingabe von q wird die Sitzung beendet, ohne die Änderungen in eine Datei einzubringen.

Kommandos zur Zeilenadressierung

`1, 2, 3...`	Die Adressen von Zeilen im Puffer.
`.`	Steht für "Die Adresse der aktuellen Zeile im Puffer".
`.=`	Die Adresse der aktuellen Zeile wird angezeigt.
`$`	Steht für "Die letzte Zeile im Puffer".
`,`	Die erste bis letzte Zeile wird adressiert.
`;`	Der Bereich zwischen der aktuellen und der letzten Zeile wird adressiert.
`+`*x*	Eine relative Adresse, die durch Addition von *x* zur aktuellen Zeilennummer entsteht.
`–`*x*	Eine relative Adresse, die durch Subtraktion von *x* von der aktuellen Zeilennummer entsteht.
`/`*abc*	Der Puffer wird in Vorwärtsrichtung durchsucht; die erste Zeile unter der aktuellen Zeile, in der das Muster *abc* enthalten ist, wird adressiert.
`?`*abc*	Der Puffer wird in Rückwärtsrichtung durchsucht; die erste Zeile vor der aktuellen Zeile, in der das Muster *abc* enthalten ist, wird adressiert.
`g/`*abc*	Alle Zeilen im Puffer, in denen das Muster *abc* enthalten ist, werden adressiert.
`v/`*abc*	Alle Zeilen im Puffer, in denen das Muster *abc* nicht enthalten ist, werden adressiert.

Anzeige-Kommandos

`p`	Die angegebenen Zeilen im Puffer werden angezeigt.
`n`	Die angegebenen Zeilen werden mit Zeilenadressen sowie einem Tabulator-Zwischenraum angezeigt.

Eingabe-Kommandos

`a`	Unter der angegebenen Zeile wird Text in den Puffer eingefügt.
`i`	Vor der angegebenen Zeile wird Text in den Puffer eingefügt.
`c`	Die angegebenen Zeilen werden durch neuen Text ersetzt.
`.`	Bei der Eingabe in einer separaten Zeile wird der Eingabemodus verlassen und in den Kommandomodus zurückgeschaltet.

Löschen von Text

`d`	Eine oder mehrere Textzeilen werden gelöscht (Kommandomodus).
`u`	Das letzte Kommando wird rückgängig gemacht (Kommandomodus).

Ersetzen von Text

"*adresse1,adresse2*`s/`*text_alt*`/`*text_neu*`/`*kommando*"
text_neu wird im Zeilenbereich, der durch *adresse1,adresse2* angegeben wird, durch *text_alt* ersetzt (als Adresse sind Zahlen, Symbole oder Text zulässig). *kommando* kann sein: `g`, `l`, `n`, `p` oder `gp`.

Sonderzeichen zum Mustervergleich

`.`	Paßt zu einem beliebigen Einzelzeichen.
`*`	Paßt zu Null, einem oder mehr Auftreten des vorhergehenden Zeichens.
`[...]`	Paßt zu jedem Zeichen, das innerhalb der eckigen Klammern steht

`[^...]`	Paßt zu jedem Zeichen, das nicht innerhalb der eckigen Klammern steht.
`.*`	Paßt zu Null, einem oder mehr Auftreten eines beliebigen Zeichens.
`^`	Paßt zum Zeilenanfang.
`$`	Paßt zum Zeilenende.
`\`	Die Bedeutung des nachfolgenden Sonderzeichens wird aufgehoben.
`&`	Wird innerhalb des Ersetzungs-Musters durch die erste Zeichenkette ersetzt, die zum Suchmuster paßt.
`%`	Das letzte Ersetzungs-Muster wird erneut benutzt.

Kommandos zum Versetzen von Text

`m`	Die angegebenen Textzeilen werden gelöscht und nach einer bestimmten Zeile eingefügt; die Textzeilen werden also versetzt.
`t`	Die angegebenen Textzeilen werden kopiert; die Kopie der Zeilen wird nach einer bestimmten Zeile eingefügt.
`j`	Aufeinanderfolgende Zeilen werden verknüpft.
`w`	Der Puffer-Inhalt wird in eine Datei eingebracht (abgespeichert).
`r`	Aus einer anderen Datei wird Text eingelesen und in den Puffer eingefügt.
`W`	Am Ende einer vorhandenen Datei wird Text angefügt.

Weitere nützliche Kommandos und Informationen

`h`	Für die vorhergehende Fehleranzeige (`?`) wird eine kurze Erläuterung ausgegeben.
`H`	Der Hilfe-Modus wird eingeschaltet, in dem während der Editor-Sitzung für jede Fehleranzeige (`?`) eine kurze Erläuterung angezeigt wird.
`l`	Sämtliche nicht darstellbaren Zeichen, die im Text enthalten sind, werden angezeigt.
`f`	Der aktuelle Dateiname wird angezeigt.
`f` *datei_neu*	Der aktuell dem Puffer zugeordnete Dateiname wird in *datei_neu* geändert.
`!`*kommando*	Hiermit können Sie vorübergehend eine Shell aufrufen, um ein Shell-Kommando auszuführen.
`ed.hup`	Der Puffer-Inhalt wird in einer speziellen Datei namens `ed.hup` abgespeichert, falls die ed-Sitzung unterbrochen wird.

E Kurzübersicht über die vi-Kommandos

vi-Kurzübersicht E-1
Grundlegende Kommandos E-1
- Shell-Kommandos E-1
- Grundlegende vi-Kommandos E-2

Positionierungs-Kommandos E-2
- Zeichenweises Positionieren E-3
- Zeilenweises Positionieren E-3
- Wortweises Positionieren E-4
- Satzweises Positionieren E-4
- Absatzweises Positionieren E-4
- Bildschirmbezogenes Positionieren E-4

Dateibezogenes Positionieren E-5
- Verschieben der Bildschirmanzeige E-5
- In eine bestimmte Zeile springen E-5
- Suchen von Mustern E-6

Kommandos zum Einfügen von Text E-6
Kommandos zum Löschen von Text E-7
- Im Eingabemodus E-7
- Im Kommandomodus E-7

Kommandos zum Ändern von Text E-7
- Zeichen, Wörter und Textobjekte E-7
- Text ausschneiden und einfügen E-8

Sonstige Kommandos E-9
- Spezielle Kommandos E-9
- Zeileneditor-Kommandos E-9
- Kommandos zum Beenden von vi E-10

Spezielle vi-Optionen E-11

vi-Kurzübersicht

Dieser Anhang enthält Kurzbeschreibungen der Kommandos des Bildschirmeditors vi. Die Kommandos sind zu Funktionsgruppen zusammengefaßt.

Ein vi-Kommando hat folgendes allgemeines Format :

[x][kommando]text_objekt

Dabei steht *x* für eine Anzahl und *text_objekt* für den Text, der vom Kommando verarbeitet werden soll. Das Kommando erscheint bei der Eingabe auf dem Bildschirm. Eine Einführung in die vi-Kommandos finden Sie in Kapitel 7, "Der Bildschirmeditor vi."

Grundlegende Kommandos

Shell-Kommandos

TERM=*code*	Die Umgebungsvariable TERM wird mit dem Codenamen für Ihr Terminal belegt.
export TERM	Der Wert der Umgebungsvariablen TERM (der Codename des Terminals) wird jedem UNIX-Programm zur Verfügung gestellt, dessen Funktionsweise vom benutzten Terminal abhängig ist.
tput init	Das Terminal wird so initialisiert, daß verschiedene UNIX-Programme darauf ablauffähig sind.

Hinweis: Bevor Sie mit vi arbeiten können, müssen Sie zunächst die drei Schritte durchführen, die oben beschrieben wurden: Einstellung der Umgebungsvariablen TERM, Exportieren des Werts von TERM und Aufrufen des Kommandos tput init.

vi *datei_name*	Der bildschirmorientierte Editor vi wird mit einer zu editierenden Datei aufgerufen.

Grundlegende vi-Kommandos

<a>	Umschalten des Eingabemodus und Anfügen von Text hinter der aktuellen Cursor-Position.
<ESC>	(Escape) Beenden des Eingabemodus und Rückkehr in den Kommandomodus.
<h>	Cursor um ein Zeichen nach links bewegen.
<j>	Cursor innerhalb der aktuellen Spalte in die nächste Zeile.
<k>	Cursor innerhalb der aktuellen Spalte in die vorhergehende Zeile bewegen.
<l>	Cursor um ein Zeichen nach rechts bewegen.
<x>	Aktuelles Zeichen löschen.
<CR>	(Carriage return) Cursor zum Anfang der nächsten Zeile bewegen ("Wagenrücklauf").
<ZZ>	Diejenigen Änderungen, die seit dem letzten Abspeichern vorgenommen worden sind, werden in die aktuelle Datei eingebracht, und vi wird beendet.
:w	Die Änderungen am Puffer werden in die aktuelle Datei eingebracht.
:q	vi wird beendet, wenn die am Puffer-Inhalt vorgenommenen Änderungen zuvor abgespeichert worden sind.

Positionierungs-Kommandos

Zeichenweises Positionieren

<h>	Cursor um ein Zeichen nach links bewegen.
<BACKSPACE>	Backspace; Cursor um ein Zeichen nach links bewegen.
<l>	Cursor um ein Zeichen nach rechts bewegen.
<LEERTASTE>	Cursor um ein Zeichen nach rechts bewegen.
<f*x*>	Cursor nach rechts zum Zeichen *x* bewegen.
<F*x*>	Cursor nach links zum Zeichen *x* bewegen.
<t*x*>	Cursor nach links zu dem Zeichen unmittelbar vor dem Zeichen *x* bewegen.
<T*x*>	Cursor nach rechts zu dem Zeichen unmittelbar hinter dem Zeichen *x* bewegen.
<;>	Der Suchvorgang nach dem Zeichen, das bei einem der Kommandos <f>, <F>, <t> oder <T> angegeben worden ist, wird fortgesetzt. Aufgrund des Strichpunkts (;) wird das angegebene Zeichen zwischengespeichert, so daß es zur Fortsetzung des Suchvorgangs in der aktuellen Zeile nicht erneut eingegeben werden muß.
<,>	Die Suche nach dem Zeichen, das bei einem der Kommandos <f>, <F>, <t> oder <T> angegeben worden ist, wird fortgesetzt. Aufgrund des Kommas (,) wird das angegebene Zeichen zwischengespeichert, so daß es zur Fortsetzung des Suchvorgangs in Rückwärtsrichtung in der aktuellen Zeile nicht erneut eingegeben werden muß.

Zeilenweises Positionieren

<j>	Cursor innerhalb der aktuellen Spalte um eine Zeile nach unten bewegen.

<k>	Cursor innerhalb der aktuellen Spalte um eine Zeile nach oben.
<+>	Cursor zum Anfang der nächsten Zeile bewegen.
<CR>	(Carriage return) Cursor zum Anfang der nächsten Zeile bewegen.
<–>	Cursor zum Anfang der vorhergehenden Zeile bewegen.

Wortweises Positionieren

<w>	Cursor nach rechts zum ersten Zeichen des nächsten Worts bewegen.
<b>	Cursor nach links zum ersten Zeichen des vorhergehenden Worts bewegen.
<e>	Cursor zum Ende des aktuellen Worts.

Satzweises Positionieren

<(>	Cursor zum Anfang des Satzes bewegen.
<)>	Cursor zum Anfang des nächsten Satzes bewegen.

Absatzweises Positionieren

<{>	Cursor zum Anfang des Absatzes bewegen.
<}>	Cursor zum Anfang des nächsten Absatzes bewegen.

Bildschirmbezogenes Positionieren

<H>	(Home) Cursor zur ersten Zeile auf dem Bildschirm bewegen.

<M>	Cursor in die mittlere Zeile auf dem Bildschirm bewegen.
<L>	(Last) Cursor zur letzten Zeile auf dem Bildschirm bewegen.

Dateibezogenes Positionieren

Verschieben der Bildschirmanzeige

<^f>	Die Bildschirmanzeige wird um ein volles Fenster in Vorwärtsrichtung verschoben, um den Text unterhalb des aktuellen Fensters zur Darstellung zu bringen.
<^d>	Die Bildschirmanzeige wird um ein halbes Fenster in Vorwärtsrichtung verschoben, um die Textzeilen unterhalb des aktuellen Fensters zur Darstellung zu bringen.
<^b>	Die Bildschirmanzeige wird um ein volles Fenster in Rückwärtsrichtung verschoben, um den Text oberhalb des aktuellen Fensters zur Darstellung zu bringen.
<^u>	Die Bildschirmanzeige wird um ein halbes Fenster in Rückwärtsrichtung verschoben, um die Textzeilen oberhalb des aktuellen Fensters zur Darstellung zu bringen.

In eine bestimmte Zeile springen

<G>	Cursor zum Anfang der letzten Zeile im Puffer.
<*n*G>	Cursor zum Anfang der *n*-ten Zeile der Datei (*n* = Zeilennummer) bewegen.

Suchen von Mustern

/muster	Der Puffer wird in Vorwärtsrichtung nach der ersten Stelle durchsucht, an der das Textmuster vorkommt; der Cursor wird zum ersten Zeichen des Musters bewegt.
?*muster*	Der Puffer wird in Rückwärtsrichtung nach der ersten Stelle durchsucht, an der das Textmuster vorkommt; der Cursor wird zum ersten Zeichen des Musters bewegt.
<n>	Das letzte Suchen-Kommando wird wiederholt.
<N>	Das letzte Suchen-Kommando wird in der entgegengesetzten Richtung wiederholt.

Kommandos zum Einfügen von Text

<a>	Aufrufen des Eingabemodus und Anfügen von Text hinter der aktuellen Cursor-Position.
<i>	Aufrufen des Eingabemodus und Einfügen von Text vor der aktuellen Cursor-Position.
<o>	Aufrufen des Eingabemodus, indem unmittelbar unter der aktuellen Zeile eine Leerzeile eingefügt wird.
<O>	Aufrufen des Eingabemodus, indem unmittelbar über der aktuellen Zeile eine Leerzeile eingefügt wird.
<ESC>	(Escape) Umschalten vom Eingabemodus (der mit einem der oben aufgeführten Kommandos aufgerufen worden ist) in den Kommandomodus.

Kommandos zum Löschen von Text

Im Eingabemodus

<BACKSPACE>	Das aktuelle Zeichen wird gelöscht.
<^w>	Das aktuelle, in Leerzeichen eingeschlossene Wort wird gelöscht.
<@>	Die aktuelle Textzeile wird gelöscht.

Im Kommandomodus

<x>	Das aktuelle Zeichen wird gelöscht.
<dw>	Die Zeichen zwischen der aktuellen Cursor-Position und dem nächsten Leer- bzw. Satzzeichen werden gelöscht (maximal ein Wort).
<dd>	Die aktuelle Zeile wird gelöscht.
<*n*d*x*>	*n* Textobjekte vom Typ *x* werden gelöscht; *x* kann ein Wort, eine Zeile, ein Satz oder ein Absatz sein.
<D>	Die Zeichen zwischen der aktuellen Cursorpositoin und dem Zeilenende werden gelöscht.

Kommandos zum Ändern von Text

Zeichen, Wörter und Textobjekte

<r>	Das aktuelle Zeichen wird ersetzt.
<s>	Das aktuelle Zeichen wird gelöscht; die bis zur Betätigung der Taste <ESC> eingegebenen Zeichen werden in den Text angefügt.
<S>	Alle Zeichen in der aktuellen Zeile werden ersetzt.
<~>	Groß- werden in Kleinbuchstaben umgewandelt und umgekehrt.

<cw>	Das aktuelle Wort bzw. die restlichen Zeichen im aktuellen Wort (bis zum nächsten Leer- oder Satzzeichen) werden durch neuen Text überschrieben.
<cc>	Alle Zeichen in der aktuellen Zeile werden überschrieben.
<*n*c*x*>	*n* Textobjekte vom Typ *x* werden überschrieben; *x* kann ein Wort, eine Zeile, ein Satz oder ein Absatz sein.
<C>	Die Zeichen zwischen der aktuellen Cursor-Position und dem Zeilenende werden überschrieben.

Text ausschneiden und einfügen

<p>	Der Inhalt des temporären Puffers (d.h. das Ergebnis des letzten Löschen- oder Speichern-Kommandos) wird hinter der aktuellen Cursor-Position oder unter der aktuellen Zeile eingefügt.
<yy>	Die angegebene Textzeile extrahiert und in einen temporären Puffer gesichert.
<*n*y*x*>	Von *n* Textobjekten vom Typ *x* wird eine Kopie erstellt und in einen temporären Puffer gesichert.
<"*l*y*x*>	Von *n* Textobjekten vom Typ *x* wird eine Kopie erstellt und in das Register *l* gesichert; *x* kann ein Wort, eine Zeile, ein Satz oder ein Absatz sein.
<"*x*p>	Der Inhalt von Register *x* wird hinter der aktuellen Curosrposition bzw. unter der aktuellen Zeile eingefügt.

Sonstige Kommandos

Spezielle Kommandos

<^g>	Anzeige der Zeilennummer, in der der Cursor sich aktuell befindet, sowie des Änderungs-Status der Datei.
<.>	Das zuletzt aufgerufene Kommando wird erneut durchgeführt.
<u>	Die Auswirkungen des zuletzt aufgerufenen Kommandos werden rückgängig gemacht.
<U>	Die aktuelle Zeile wird in den Zustand zurückgesetzt, in dem sie sich vor der letzten Änderung befunden hat.
<J>	Die Zeile unmittelbar unter der aktuellen Zeile wird mit der aktuellen Zeile verknüpft.
<^l>	Die aktuelle Bildschirmanzeige wird gelöscht und neu aufgebaut.

Zeileneditor-Kommandos

:	`vi` wird darüber informiert, daß die nächsten Kommandos Zeileneditor-Kommandos sind.
:sh	Vorübergehendes Aufrufen einer Shell, um Shell-Kommandos durchzuführen; `vi` wird nicht beendet.
<^d>	Vorübergehendes Aufrufen einer Shell, um Shell-Kommandos durchzuführen; danach wird `vi` wieder zum Editieren des aktuellen Fensters aufgerufen.
:*n*	Der Cursor springt zur *n*-ten Zeile im Puffer.
:*x,z*w *datei_name*	Der Inhalt der Zeilen mit den Nummern im Bereich *x* bis *z* werden in eine neue Datei namens *datei_name* eingebracht.

`:$`	Cursor zum Anfang der letzten Zeile im Puffer bewegen.
`:.,$d`	Sämtliche Zeilen zwischen der aktuellen Zeile (einschließlich) und dem Ende des Puffers werden gelöscht.
`:r` *datei_name*	Der Inhalt der Datei *datei_name* wird unter der aktuellen Zeile in den Puffer eingefügt.
`:s/`*text_alt*`/`*text_neu*`/`	*text_alt* wird beim ersten Auftreten in der aktuellen Zeile durch *text_neu* ersetzt.
`:s/`*text_alt*`/`*text_neu*`/g`	*text_alt* wird bei jedem Auftreten in der aktuellen Zeile durch *text_neu* ersetzt.
`:g/`*text_alt*`/s//`*text_neu*`/g`	*text_alt* wird bei jedem Auftreten im Puffer durch *text_neu* ersetzt.

Kommandos zum Beenden von `vi`

`<ZZ>`	Der Puffer-Inhalt wird in einer Datei abgespeichert (falls nicht bereits geschehen), und `vi` wird beendet.
`:wq`	Der Puffer-Inhalt wird in der Datei abgespeichert, und `vi` wird beendet.
`:w` *datei_name* `:q`	Der Puffer-Inhalt wird in der neuen Datei *datei_name* abgespeichert, und `vi` wird beendet.
`:w!` *datei_name* `:q`	Die vorhandene Datei *datei_name* wird durch den Puffer-Inhalt überschrieben, und `vi` wird beendet.
`:q!`	`vi` wird beendet, unabhängig davon, ob die Änderungen, die seit dem letzten Komando `:w` am Puffer-Inhalt vorgenommen worden sind, zuvor abgesichert worden oder nicht.

`:q`	`vi` wird beendet, wenn die Änderungen am Puffer-Inhalt zuvor in eine Datei eingebracht worden sind.

Spezielle `vi`-Optionen

`vi` *datei1 datei2 datei3*	In den `vi`-Puffer werden drei zu editierende Dateien (*datei1*, *datei2* und *datei3*) geschrieben.
`:w` `:n`	Sind in ein und derselben `vi`-Kommandozeile zwei oder mehr Dateien zum Editieren aufgerufen worden, so wird der Puffer-Inhalt in die aktuell editierte Datei eingebracht und die nächste Datei im Puffer geladen (vor `:n` sollten Sie auf jeden Fall `:w` durchführen).
`vi -r` *datei1*	Die Änderungen an *datei1*, die aufgrund eines Systemabsturzes verlorengegangen sind, werden wiederhergestellt.
`view` *datei1*	*datei1* wird schreibgeschützt mit `vi` aufgerufen. Die Änderungen, die am Puffer-Inhalt vorgenommen werden, können nicht in die Datei eingebracht werden.

F Die Kommandosprache der Shell - Kurzübersicht

Die Kommandosprache der Shell - Kurzübersicht F-1

Die Terminologie der Shell-Kommandosprache F-1

- Sonderzeichen der Shell F-1
- Umleitung der Ein- und Ausgabe F-2
- Ausführen und Beenden von Prozessen F-2
- Dateizugriff für die Shell ermöglichen F-3
- Variablen F-3
- Umgebungsvariablen F-4

Shell-Programmstrukturen F-5

- Here-Dokument F-5
- For-Schleife F-5
- While-Schleife F-6
- If...Then F-6
- If...Then...Else F-7
- Case-Konstruktion F-7
- break- und continue-Anweisungen F-8

Die Kommandosprache der Shell - Kurzübersicht

Dieser Anhang enthält eine Übersicht über die Shell-Kommandosprache sowie die Programstrukturen, die in Kapitel 9, "Die Shell", beschrieben worden sind. Im ersten Abschnitt geht es um Metazeichen, Sonderzeichen, die Umleitung der Ein-/Ausgabe, Variablen und Prozesse. Diese Punkte sind nach Themen zusammengefaßt und in der Reihenfolge aufgeführt, in der sie in Kapitel 9 beschrieben worden sind. Der zweite Abschnitt enthält die Syntax der Shell-Programmstrukturen.

Die Terminologie der Shell-Kommandosprache

Sonderzeichen der Shell

`* ? [ ]`	Metazeichen; hierbei handelt es sich um Kurzformate zur Bezugnahme auf Dateinamen in Suchmustern.
`&`	Kommandoausführung im Hintergrund.
`;`	Die in ein und derselben Kommandozeile enthaltenen Kommandos werden nacheinander abgearbeitet; die Kommandos sind durch Semikolon (;) voneinander getrennt.
`\`	Die spezielle Bedeutung des unmittelbar darauffolgenden Sonderzeichens wird aufgehoben.
`'...'`	Durch das Einschließen in Hochkommata (') wird die spezielle Bedeutung sämtlicher Zeichen mit Ausnahme des Hochkommas selbst aufgehoben.
`"..."`	Durch das Einschließen in Anführungszeichen ("") wird die spezielle Bedeutung sämtlicher Zeichen mit Ausnahme des Dollar-Zeichens ($), des Hochkommas sowie des Anführungszeichens selbst aufgehoben.

Umleitung der Ein- und Ausgabe

<	Der Inhalt einer Datei wird zu einem Kommando umgeleitet.
>	Die Ausgabe eines Kommandos wird zu einer neuen Datei umgeleitet; ist die angegebene Datei bereits vorhanden, so wird ihr ursprünglicher Inhalt durch die Ausgabe überschrieben.
>>	Die Ausgabe eines Kommandos wird so umgeleitet, daß sie am Ende einer Datei angefügt wird.
|	Die Ausgabe eines Kommandos wird so umgeleitet, daß sie zur Eingabe des darauffolgenden Kommandos wird.
`` `kommando` ``	Die Ausgabe des eingeschlossenen Kommandos wird für `` `kommando` `` eingesetzt.

Ausführen und Beenden von Prozessen

`batch`	Die darauffolgenden Kommandos sollen in eine Warteschlange gesetzt und erst dann ausgeführt werden, wenn es die Systemauslastung zuläßt. Mit <^d> wird das Kommando `batch` beendet.
`at`	Die nachfolgenden Kommandos sollen zu einem bestimmten Zeitpunkt ausgeführt werden. Mit <^d> wird das Kommando beendet.
`at -l`	Informationen über die Aufträge werden ausgegeben, die aktuell in der `at`- oder `batch`-Warteschlange enthalten sind.
`at -r`	Der mit `at` oder `batch` abgesetzte Auftrag wird aus der Warteschlange entfernt.
`ps`	Der Status der Shell-Prozesse wird ausgegeben.
`kill` *PID*	Der Shell-Prozeß mit der angegebenen Prozeßnummer (PID) wird abgebrochen.
`nohup` *kommando_liste* `&`	Die Ausführung der Hintergrund-Prozesse wird auch nach der Abmeldung des Benutzers fortgesetzt.

Dateizugriff für die Shell ermöglichen

`chmod u+x` *dateiname*
: Der Benutzer erhält die Berechtigung zur Ausführung der Datei (vor allem bei Shell-Programmdateien hilfreich).

`mv` *dateiname* `$HOME/bin/`*dateiname*
: Ihre Datei wird in das Verzeichnis `bin` in Ihrem Home-Verzeichnis versetzt. Im Verzeichnis `bin` sind ausführbare Shell-Programme enthalten, auf die Sie zugreifen können. Stellen Sie sicher, daß dieses Verzeichnis im Zugriffspfad der Umgebungsvariablen `PATH` in Ihrer `.profile`-Datei aufgeführt ist. Ist dies der Fall, so sucht die Shell im Verzeichnis `$HOME/bin` nach der Datei, wenn sie von Ihnen aufgerufen wird; andernfalls kann die Shell Ihre Datei nicht ausfindig machen und ausführen.

`dateiname`
: Der Name einer Datei, die ein Shell-Programm enthält, wird zu einem Kommandonamen gemacht, den Sie zum Aufrufen dieses Shell-Programms eingeben können.

Variablen

`Stellungsparameter`
: Eine Variable in Form einer Zahl, die in einem Shell-Programm zur Bezugnahme auf Werte benutzt wird, die durch die Shell automatisch aufgrund der Argumente in der Aufrufzeile des Shell-Programms zugeordnet werden.

`echo`
: Ein Kommando, mit dem Sie den Wert einer Variablen auf Ihrem Terminal abrufen können.

`$#`
: Ein spezieller Parameter, der die Anzahl der Argumente angibt, mit denen das Shell-Programm aufgerufen worden ist.

`$*`
: Ein spezieller Parameter, der die Werte sämtlicher Argumente enthält, mit denen das Shell-Programm aufgerufen worden ist.

`Schlüsselwortparameter`
: Eine Variable, der vom Benutzer ein Name oder ein Wert zugeordnet werden kann.

Umgebungsvariablen

`HOME`
: Ihr Home-Verzeichnis; die Standard-Variable für das Kommando `cd`.

`PATH`
: Der Zugriffspfad, den Ihre Login-Shell bei der Suche nach einem Kommando durchsuchen muß.

`MAIL`
: Der Name der Datei, der als Ihr elektronischer Briefkasten benutzt wird.

`PS1, PS2`
: Die primäre bzw. sekundäre Eingabeaufforderung.

`TERM`
: Der Terminal-Typ.

`LOGNAME`
: Der Benutzername des Benutzers.

`IFS`
: Die internen Feldbegrenzer (im Normalfall das Leerzeichen, Tabulatorzeichen sowie das Wagenrücklauf-Zeichen).

`TERMINFO`
: Die Unterprogramme `curses` und `terminfo` sollen einen bestimmten Teilbaum des Dateisystems durchsuchen, bevor sie im Standard-Verzeichnis nach dem Typ Ihres Terminals suchen.

`TZ`
: Einstellen und Speichern der lokalen Zeitzone.

Shell-Programmstrukturen

Here-Dokument

```
kommando <<!
eingabezeilen
!
```

For-Schleife

```
for variable<CR>
        in dieser werteliste<CR>
do folgende kommandos<CR>
        kommando 1<CR>
        kommando 2<CR>
            .<CR>
            .<CR>
        letztes kommando<CR>
done<CR>
```

While-Schleife

```
while kommando_liste<CR>
do<CR>
    kommando1<CR>
    kommando2<CR>
        .<CR>
        .<CR>
    letztes kommando<CR>
done<CR>
```

If...Then

```
if diese kommando_liste Wert Null ergibt<CR>
then kommando1<CR>
    kommando2<CR>
        .<CR>
        .<CR>
    letztes kommando<CR>
fi<CR>
```

If...Then...Else

```
if kommando_liste<CR>
   then kommando_liste<CR>
   else kommando_liste<CR>
fi<CR>
```

Case-Konstruktion

```
case wort<CR>
in<CR>
    muster1)<CR>
        kommando_zeile 1<CR>
            .<CR>
            .<CR>
        letzte kommando_zeile<CR>
    ;;<CR>
    muster2)<CR>
        kommando_zeile 1<CR>
            .<CR>
            .<CR>
        letzte kommando_zeile<CR>
    ;;<CR>
    muster3)<CR>
        kommando_zeile 1<CR>
            .<CR>
            .<CR>
        letzte kommando_zeile<CR>
    ;;<CR>
esac<CR>
```

break- und continue-Anweisungen

Bei einer `break`- oder `continue`-Anweisung beendet das Programm eine beliebige Schleife und fährt bei dem Kommando fort, das auf das Schleifen-Ende folgt.

G Einstellung des Terminals

Einstellung der Variablen TERM G-1
Zulässige Terminal-Namen G-2

Beispiel G-4

Arbeiten mit Shell-Fenstern G-6
Eröffnen von Fenstern G-6
- Zeichnen von Fenstern mit der Maus G-7
- Zeichnen von Fenstern ohne Maus G-7

Arbeiten mit Shell-Fenstern G-10

Einstellung der Variablen TERM

AT&T unterstützt eine Vielzahl von Terminals, auf denen UNIX lauffähig ist. Da die Ausführung einiger Kommandos abhängig vom jeweiligen Terminal ist, muß der Terminal-Typ dem System bei jeder Anmeldung mitgeteilt werden. Das System ermittelt die Eigenschaften Ihres Terminals anhand der Variablen TERM; dieser Variablen ist der Name des Terminals zugeordnet. Nachdem Sie dieser Variablen den Namen Ihres Terminals zugeordnet haben, kann das System die Ausführung aller Programme an Ihr Terminal anpassen.

Die Parameter, aus denen das UNIX-System den Typ des benutzten UNIX-Systems erkennen kann, werden unter dem Begriff Terminal-Konfiguration zusammengefaßt. Zur Einstellung der Terminal-Konfiguration geben Sie die Kommandozeilen ein, die auf dem folgenden Bildschirm gezeigt werden; *terminal_name* steht dabei für den Namen Ihres Terminals:

```
$ TERM=terminal_name
$ export TERM
$ tput init
```

Diese Zeilen müssen in der angegebenen Reihenfolge eingegeben werden; andernfalls haben Sie nicht die gewünschte Wirkung. Beachten Sie außerdem, daß diese Prozedur bei jeder Anmeldung durchgeführt werden muß. Aus diesem Grund bringen die meisten Benutzer diese Zeilen in eine Datei namens `.profile` ein; diese Datei wird bei jeder Anmeldung automatisch abgearbeitet. Ausführliche Informationen zur Datei `.profile` finden Sie in Kapitel 7.

Mit den ersten beiden Zeilen teilen Sie der UNIX-Shell den Typ Ihres Terminals mit. Durch die Kommandozeile `tput init` funktioniert Ihr Terminal so, wie UNIX es von einem Terminal dieses Typs erwartet. So werden durch diese Kommandozeile beispielsweise der linke Rand und die Behandlung der Tabulatorzeichen eingestellt - vorausgesetzt, diese Möglichkeiten sind für Ihr Terminal vorhanden.

Das Kommando `tput` ermittelt anhand des Eintrags, der in seiner Datenbank Ihrem Terminal zugeordnet ist, die Eigenschaften Ihres Terminals und gibt diese Informationen an die Shell weiter. Da den verschiedenen Eigenschaften der verschiedenen Terminal-Typen unterschiedliche Werte zugeordnet sind, müssen Sie die Kommandozeile `tput init` jedesmal nach einer Änderung der Variablen TERM aufrufen.

Für jeden Terminal-Typ sind in einer Datenbank eine Reihe von Eigenschaften angegeben. Diese Datenbank ist, je nach System, entweder im Verzeichnis `/usr/share/lib/terminfo` oder im Verzeichnis `/usr/lib/.COREterm` enthalten.

Hinweis: Zumindest eines dieser Verzeichnisse ist auf jedem System vorhanden, in manchen Fällen auch beide. Erkundigen Sie sich bei Ihrem Systemverwalter, ob es in Ihrem System das Verzeichnis `terminfo` und/oder `.COREterm` gibt.

In den folgenden Abschnitten wird beschrieben, wie Sie die Zulässigkeit eines *terminal_name*s überprüfen können. Weitere Informationen über die Eigenschaften, die in der `terminfo`-Datenbank beschrieben werden, finden Sie im *Referenzhandbuch für Programmierer* unter dem Eintrag `terminfo(4)`.

Zulässige Terminal-Namen

Das UNIX-System ist auf einer Vielzahl von Terminal-Typen lauffähig. Bevor Sie der Umgebungsvariablen `TERM` einen Terminal-Namen zuordnen, müssen Sie sicherstellen, daß Ihr Terminal zu den unterstützten Terminals gehört.

Außerdem müssen Sie sicherstellen, daß Sie der Umgebungsvariablen `TERM` einen zulässigen Terminal-Namen zugeordnet haben. Im Normalfall gibt es mindestens zwei zulässige Namen: Den Namen des Herstellers sowie die Modell-Nummer. Allerdings gibt es für die Darstellung dieser Namen mehrere Möglichkeiten (durch unterschiedliche Groß- und Kleinschreibung, die Benutzung von Abkürzungen usw.). Vor der Zuordnung eines Terminal-Namens an die Umgebungsvariable `TERM` muß auf jeden Fall sichergestellt sein, daß er vom System akzeptiert wird.

Mit dem Kommando `tput` können Sie schnell und einfach sicherstellen, daß Ihr Terminal vom System unterstützt wird. Geben Sie folgendes ein:

```
tput -Tterminal_name longname<CR>
```

Wird Ihr Terminal vom UNIX-System unterstützt, so gibt es den vollständigen Terminal-Namen aus; andernfalls wird eine Fehlermeldung angezeigt.

Die zulässigen Terminal-Namen, die der Umgebungsvariablen TERM zugeordnet werden können, sind in einem der folgenden zwei Verzeichnisse enthalten: /usr/share/lib/terminfo oder /usr/lib/.COREterm. Jedes dieser Verzeichnisse enthält eine Reihe von Dateien mit einstelligem Namen. In den einzelnen Dateien wiederum ist eine Liste von Terminal-Namen enthalten, die alle mit dem Namen der jeweiligen Datei beginnen (beim Namen kann es sich entweder um einen Buchstaben wie den Großbuchstaben A für AT&T oder eine Ziffer wie 5 für 5425) handeln). Suchen Sie die Datei heraus, deren Name mit dem ersten Zeichen des Terminal-Namens übereinstimmt. Rufen Sie dann den Inhalt der Datei auf dem Bildschirm ab und suchen Sie nach Ihrem Terminal.

Sie können sich aber auch bei Ihrem Systemverwalter nach einer Liste der unterstützten Terminals sowie den zulässigen Namen, die Sie der Umgebungsvariablen TERM zuordnen können, erkundigen.

Beispiel

Angenommen, Sie arbeiten an einem AT&T-Terminal vom Typ Teletype 5425; Ihr Benutzername sei `jim`, und Sie befinden sich aktuell in Ihrem Home-Verzeichnis. Überprüfen Sie zunächst mit Hilfe des Kommandos `tput`, ob Ihr Terminal von Ihrem UNIX-System unterstützt wird. Ermitteln Sie dann einen zulässigen Namen anhand des Verzeichnisses `/usr/share/lib/terminfo/A`. Der folgende Bildschirm zeigt die Kommandos, die Sie dafür eingeben müssen:

```
$ tput -T5425 longname
AT&T 4425/5425
$ cd /usr/share/lib/terminfo/A
$ ls
ATT4410
ATT4415
ATT4418
ATT4424
ATT4424-2
ATT4425
ATT4426
ATT513
ATT5410
ATT5418
ATT5420
ATT5420-2
ATT5425
ATT5620
ATT610BCT
ATTPT505
$
```

Jetzt können Sie den Namen des Terminals (`ATT5425`) der Umgebungsvariablen `TERM` zuordnen. Danach müssen Sie `TERM` in jedem Fall exportieren und `tput init` aufrufen.

```
$ TERM=ATT5425
$ export TERM
$ tput init
$
```

Das UNIX-System kennt jetzt den Typ Ihres Terminals und führt die Kommandos entsprechend durch.

Arbeiten mit Shell-Fenstern

Der Bereich des Terminal-Bildschirms, in dem Sie arbeiten und Dateinamen abrufen, hat eine ähnliche Funktion wie das Fenster in einem Haus: Bei beiden handelt es sich um Einrichtungen, die den Rahmen einer Einheit bilden und Einblick in diese Einheit gewähren. Aus diesem Grund wird der Arbeitsbereich eines Terminal-Bildschirms auch als Fenster bezeichnet. Bisher sind wir davon ausgegangen, daß es auf Ihrem Terminal-Bildschirm nur ein einziges Fenster (den ganzen Bildschirm) gibt. Allerdings kann der Bildschirm auf einer Reihe von Terminals —nicht auf dem Terminal 5425— in mehrere Fenster unterteilt werden. Jedem Fenster auf einem so unterteilten Bildschirm ist eine separate Shell zugeordnet; in jedem Fenster lassen sich fast dieselben Funktionen durchführen wie auf einem separaten Terminal. Um Ihnen die Nutzung dieser Einrichtung zu ermöglichen, gibt es unter UNIX eine Gruppe von Software-Werkzeugen, die als Basic Windowing Utilities bezeichnet werden.

Die Durchführung mehrerer Aufgaben gleichzeitig auf ein und demselben Bildschirm z.B. mit Hilfe des Hintergrund-Modus und dem Kommando `at` ist bereits an einer früheren Stelle dieses Leitfadens beschrieben worden. Mit mehreren Fenstern haben Sie zusätzlich die Möglichkeit, mit mehr als einem Prozeß gleichzeitig im Dialog zu arbeiten. So können Sie den Ablauf mehrerer Prozesse gleichzeitig verfolgen oder mehrere Dateien gleichzeitig ansehen. Auf einem Terminal mit Fensterdarstellung, auf dem die Basic Windowing Utilities installiert sind, stehen Ihnen die in diesem Abschnitt beschriebenen Verfahren zur effizienten Nutzung Ihres Terminals zur Verfügung.

Eröffnen von Fenstern

Um ein Fenster zu eröffnen, müssen Sie es auf Ihrem Bildschirm zeichnen und die zugehörige Shell einrichten. Die Shell ist der Kommando-Interpreter und ermöglicht Ihnen das Arbeiten mit UNIX im Dialogbetrieb. Wenn Sie einem Fenster keine Shell zuordnen, ist das eröffnete Fenster lediglich eine Zeichnung auf Ihrem Bildschirm.

Das Kommando `layers` ermöglicht Ihnen das Zeichnen eines Fensters auf einem beliebigen Terminal mit Fensterdarstellung. Wenn Sie das Kommando ohne Argumente aufrufen, müssen Sie zum Zeichnen des Fensters die Maus benutzen. Wenn Sie in der Kommandozeile von `layers` dagegen Argumente eingeben, können Sie das Zeichnen des Fensters programmgesteuert durchführen, so daß Sie keine Maus benötigen; Ihr Fenster wird vom Kommando `layers` dann automatisch gezeichnet.

Zeichnen von Fenstern mit der Maus

Am einfachsten geht das Zeichnen von Fenstern mit der Maus. Rufen Sie zunächst das Kommando `layers` auf:

```
layers<CR>
```

Drücken Sie jetzt eine der Tasten Ihrer Maus; auf dem Bildschirm wird ein Menü mit Optionen zur Manipulation von Shell-Fenstern angezeigt. Wählen Sie die Menüoption zum Zeichnen von Fenstern (z.B. `New`) und zeichnen Sie das Fenster mit Hilfe der Maus (die genauen Anweisungen schlagen Sie bitte in der Terminal-Dokumentation nach).

Um zwei oder mehr Fenster zu eröffnen, rufen Sie erneut das Menü auf, wählen die entsprechende Option und zeichnen das/die Fenster mit der Maus (das Kommando `layers` kann nicht noch einmal aufgerufen werden). Auf dem Bildschirm werden Ihre Fenster gezeichnet; Sie können jetzt Kommandos am Terminal eingeben.

Zeichnen von Fenstern ohne Maus

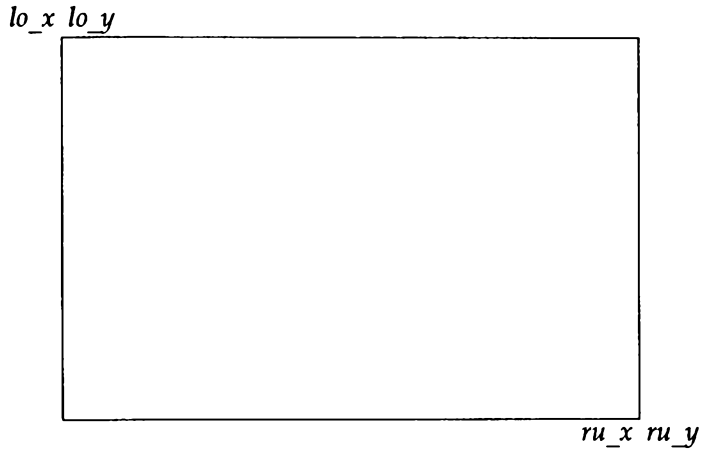
Wenn Sie das Zeichnen der Fenster programmgesteuert durchführen möchten, so müssen Sie zunächst eine Datei anlegen, in die Sie die Nummer sowie die Abmessungen des gewünschten Fensters einbringen. Dann rufen Sie das Kommando `layers` mit dem Namen der Datei als Argument sowie mit der Option `-f` auf (diese Option weist das Kommando an, Ihre Parameterdatei zu lesen). Die Kommandozeile hat folgendes allgemeines Format:

```
layers -f datei<CR>
```

Die Parameter-Datei muß für jedes zu eröffnende Fenster eine Zeile in folgendem Format enthalten:

lo_x lo_y ru_x ru_y kommando_liste

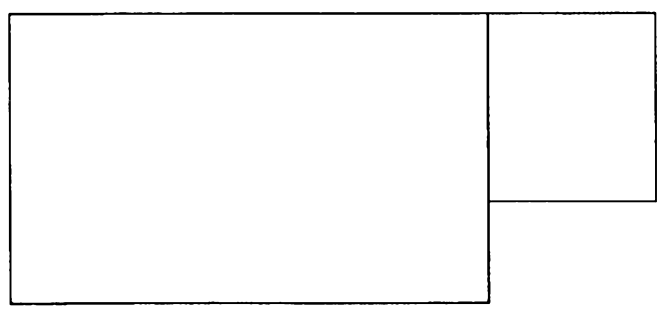
Die ersten vier Felder der Zeile stehen für die Koordinaten der Fenster-Ecken. Die Einträge *lo_x* und *lo_y* definieren die x- und y-Koordinaten der oberen linken Ecke des Fensters, die Einträge *ru_x* und *ru_y* die x- und y-Koordinaten seiner unteren rechten Ecke.



Um beispielsweise ein großes und ein kleines rechteckiges Fenster zu eröffnen, geben Sie die folgenden Zeilen in die Parameterdatei ein:

```
0       0       650     300
650     0       792     175
```

Die so angegebenen Fenster sehen folgendermaßen aus:



Im fünften Feld jeder Zeile ist eine *kommando-liste* enthalten. Hier müssen Sie ein Kommando eintragen, mit dem dem Fenster eine Shell zugeordnet wird. Außerdem können Sie dem Fenster einen bestimmten Terminal-Typ oder einen bestimmten Editor zuordnen.

Damit Sie Ihrem Terminal eine Shell zuordnen können, müssen Sie zunächst die Shell-Variable einstellen; hierzu rufen Sie das Kommando `exec` (für execute) auf. In der Kommandozeile dieses Kommandos geben Sie ein Argument ein, das für die gewünschte Shell steht. Wenn Sie mit der Shell arbeiten möchten, die auch normalerweise auf Ihrem Terminal aktiv ist, so geben Sie folgendes ein:

```
exec $SHELL
```

Zum Aufrufen der Standard-UNIX-Shell geben Sie folgendes ein:

```
exec /usr/bin/sh
```

Vielleicht möchten Sie in Ihrem Fenster mit Funktionen arbeiten, die auf Ihrem aktuellen Terminal- Typ nicht verfügbar sind. Geben Sie den gewünschten Terminal-Typ ein und weisen Sie ihn der Umgebungsvariablen `TERM` zu. Wenn Sie diese Zuweisung in die *kommando_liste* eingeben, müssen Sie sie vor das Kommando `exec` setzen. Die drei Anforderungen (Terminal-Typ, Einstellung der Umgebungsvariablen `TERM` sowie das Kommando `exec`) sind durch Semikolon (;) voneinander, zu trennen; jedes Semikolon muß in Leerzeichen eingeschlossen sein. Angenommen, Ihr Fenster soll die Funktionen eines HP 2621-Terminals unterstützen, wobei mit der Shell gearbeitet werden soll, die auch normalerweise auf Ihrem Terminal aktiv ist. In diesem Fall sieht das Feld *kommando_liste* in Ihrer Parameter-Datei folgendermaßen aus:

```
hp2621 ; TERM=hp2621 ; exec ; $SHELL
```

Die Parameter-Datei muß also für jedes Fenster, das Sie eröffnen möchten, eine separate Zeile enthalten; jede Zeile muß fünf Felder enthalten: Vier Koordinaten-Felder für die obere linke und untere rechte Ecke des Fensters sowie eine Kommandozeile zur Angabe einer Shell für das Fenster. Außerdem kann dem Fenster in der Kommandozeile ein bestimmter Editor oder ein bestimmter Terminal- Typ zugeordnet werden. Die folgende Beispiel-Parameterdatei enthält die oben beschriebenen Einzelbeispiele:

```
8       0       650     300     exec $SHELL
675     0       800     175     exec /usr/bin/sh
0       200     800     900     vi ; exec $SHELL
0       800     792     1024    hp2621 ; TERM=hp2621 ; exec $SHELL
```

Wenn Ihre Parameterdatei vollständig ist, rufen Sie das Kommando `layers` wie folgt auf:

```
layers -f parameter_datei<CR>
```

Die von Ihnen dargestellten Fenster werden auf dem Bildschirm gezeichnet; die Shell, die Sie dem jeweiligen Fenster zugeordnet haben, wird aktiviert und ist nun für Ihre Kommandos empfangsbereit.

Arbeiten mit Shell-Fenstern

Sobald die Fenster auf Ihrem Bildschirm dargestellt werden, müssen Sie den Umgang damit lernen, d.h., wie Sie zu einem anderen Fenster wechseln, wie Sie mit jedem Fenster wie mit einem Terminal arbeiten, und wie Sie die Fenster wieder entfernen können. Diese Funktionen können Sie über die verschiedenen Tasten der Maus durchführen (nähere Informationen hierzu finden Sie in der Terminal-Dokumentation).

Für Programmierer, die selbst Programme zum Eröffnen und Benutzen von Fenstern erstellen möchten, gibt es die Funktions-Bibliothek `libwindows` (siehe auch den Eintrag (`libwindows`(3X) im *Referenzhandbuch für Programmierer*.)

GL Fachwörter

Fachwörter

Abmelden

Die Prozedur, mit der Sie das UNIX-System verlassen.

Absoluter Pfadname

Ein Pfadname, der beim Root-Verzeichnis des UNIX-Dateisystems beginnt und zu einer bestimmten Datei oder einem bestimmten Verzeichnis führt. Jede Datei und jedes Verzeichnis im UNIX-System hat einen eindeutigen absoluten Pfadnamen, der auch als "vollständiger Pfadname" bezeichnet wird. Siehe Pfadname, vgl. Relativer Pfadname.

Adresse

Im allgemeinen eine Zahl zur Angabe eines Speicherplatzes. Unter UNIX ist die Adresse Teil eines Editor-Kommandos, in dem eine Zeilennummer oder ein Zeilenbereich angegeben wird.

Aktuelles Verzeichnis

Das Verzeichnis, in dem Sie sich gerade befinden. In Ihrem aktuellen Verzeichnis können Sie auf sämtliche Dateien und Unterverzeichnisse direkt zugreifen. Die Kurzform für die Angabe des aktuellen Verzeichnisses ist der Punkt (.).

Alphanumerische Variable

Eine Folge von Zeichen, die als Wert einer Shell-Variablen benutzt werden kann. Siehe
Variable.

Anfügen-Modus

Ein Editor-Modus, in dem die von Ihnen eingegebenen Zeichen als Text in den Puffer des Texteditors eingebracht werden. In diesen Modus geben Sie hinter der aktuellen Position im Puffer Text ein (fügen Text an. Siehe Eingabemodus, vgl. Kommandomodus und Einfügen-Modus.

Angleichung

Die Umwandlung eines Datenobjekts, so daß es auf bestimmte Weise verarbeitet werden kann.

Anmeldung

Die Prozedur, die Ihnen den Zugriff auf das UNIX-System ermöglicht.

Arbeitsverzeichnis

Siehe Aktuelles Verzeichnis.

Argument

Das Element einer Kommandozeile, durch das die vom Kommando zu verarbeitenden Daten angegeben werden. Das Argument folgt auf den Kommandonamen und kann aus Zahlen, Buchstaben oder Zeichenketten bestehen. So ist in der Kommandozeile `lp -m myfile`, `lp` das Kommando und `myfile` das Argument. Siehe Option.

ASCII

American Standard Code for Information Interchange (Amerikanischer Standard-Code für den Informationsaustausch). Der Standard-Code für die Übertragung von Daten; UNIX arbeitet mit dem ASCII-Code. Der ASCII-Zeichensatz besteht aus 128 Zeichen, die als Folgen von Nullen und Einsen dargestellt werden; der ASCII-Zeichensatz enthält Buchstaben, Zahlen sowie Standard-Sonderzeichen wie #, $, % und &.

Assoziatives Feld

Eine Gruppe von Daten, deren einzelne Elemente über eine Zeichenkette indiziert (angesprochen) werden können, und nicht, wie in den meisten Programmiersprachen üblich, durch eine Ganzzahl. Man sagt, die Elemente sind "assoziiert" mit dem Paar *feld_name: zeichenkette*, wobei *zeichenkette* der Index ist.

ASCII

American Standard Code for Information Interchange (Amerikanischer Standard-Code für den Informationsaustausch). Der Standard-Code für die Übertragung von Daten; UNIX arbeitet mit dem ASCII-Code. Der ASCII-Zeichensatz besteht aus 128 Zeichen, die als Folgen von Nullen und Einsen dargestellt werden; der ASCII-Zeichensatz enthält Buchstaben, Zahlen sowie Standard-Sonderzeichen wie #, $, % und &.

AT&T 3B Rechner

Rechner von AT&T Technologies, Inc.

Ausführbare Datei

Eine Datei, die vom Rechner ohne vorherige Umsetzung verarbeitet bzw. ausgeführt werden kann. Bei der Eingabe des entsprechenden Dateinamens werden die Kommandos ausgeführt, die in der Datei enthalten sind. Siehe Shell-Prozedur.

Ausführung

Ein Vorgang, bei dem auf dem Rechner ein Programm oder Kommando aufgerufen und die angeforderten Funktionen durchgeführt werden.

Ausgabe

Informationen, die vom Rechner bei der Verarbeitung ermittelt worden sind und an Sie über einen Drucker, ein Terminal oder ein ähnliches Gerät übermittelt werden.

Baudrate

Ein Maß für die Geschwindigkeit, mit der die Datenübertragung zwischen einem Rechner und einem Peripheriegerät (z.B. einem Terminal) oder zwischen zwei Geräten stattfindet. Gängige Baudraten sind 300, 1200, 4800 und 9600. Um die ungefähre Anzahl der Zeichen zu ermitteln, die pro Sekunde übertragen werden, müssen Sie einfach die Baudrate durch 10 dividieren.

Begrenzer

Ein Zeichen zur logischen Abgrenzung der Wörter bzw. Argumente in einer Kommandozeile. Unter UNIX sind das Leer- und das Tabulatorzeichen die gängigsten Begrenzer.

Benutzer

Eine Person, die mit einem Rechner oder einem Betriebssystem arbeitet.

Benutzer-definiert

Variablen oder Parameter, die vom Benutzer festgelegt werden.

Benutzer-definierte Variable

Eine benannte Variable, der vom Benutzer ein Wert zugeordnet worden ist. Siehe Variable.

Benutzername

Eine Folge von Zeichen zur Bezeichnung eines Benutzers. Jedem Benutzer ist ein anderer Benutzername zugeordnet.

Betriebssystem

Die System-Software auf einem Rechner, die sämtliche Rechner-Aktivitäten steuert. Ein Beispiel für ein Betriebssystem ist das UNIX-System.

Bildschirmeditor

Ein Editier-Programm, bei dem der Text bearbeitet wird, der sich an der aktuellen Position des Cursors auf dem Bildschirm befindet. Damit Text an einer bestimmten Stelle eingegeben, geändert oder gelöscht werden kann, muß zunächst der Cursor zu dieser Stelle bewegt werden. Die Änderungen wirken sich unmittelbar auf die Bildschirmanzeige aus. Siehe Texteditor, vgl. Zeileneditor.

Cursor

Ein Zeichen, das auf dem Terminal-Bildschirm angezeigt wird und zur Markierung der Position dient, an der Sie ein Zeichen eingeben oder löschen können. Im Normalfall handelt es sich hierbei um ein Rechteck oder einen blinkenden Unterstrich.

Datei

Eine Zusammenfassung von Informationen in Form eines Zeichenstroms. In einer Datei können Daten, Programme und sonstiger Text enthalten sein. Unter UNIX wird auf eine Datei über ihren Namen zugegriffen. Siehe Normale Datei, Permanente Datei, und Ausführbare Datei.

Dateiname

Eine Folge von Zeichen zur Bezeichnung einer Datei (unter UNIX darf in einem Dateinamen kein Schrägstrich (/) enthalten sein).

Dateisystem

Eine Ansammlung von Dateien sowie die Struktur, durch die sie zusammenhängen. Unter UNIX hat das Dateisystem eine hierarchische Struktur (weitere Informationen hierzu finden Sie in Anhang A, "Kurzübersicht über das Dateisystem.")

Datensichtgerät

Ein Terminal, bei dem die Informationen auf einem Bildschirm ausgegeben werden, der dem eines Fernsehgeräts sehr ähnlich ist. Auf einem Datensichtgerät können die Informationen wesentlich schneller dargestellt werden als auf einem Drucker-Terminal.

Dienstprogramm

Ein Programm, mit denen häufig anfallende Aufgaben durchgeführt und den Programmierer bzw. System-Benutzer bei der Rationalisierung von häufig anfallenden Aufgaben unterstützt.

Drucker

Ein Ausgabegerät, das die vom Rechner empfangenen Daten auf Papier ausgibt.

Ein/Ausgabe

Der Weg, über den die Informationseingabe am Rechner und die Informationsausgabe des Systems erfolgt. Das Eingabegerät ist die Terminal-Tastatur, das Ausgabegerät der Terminal-Bildschirm.

Einfügen-Modus

Ein Editor-Modus, in dem die von Ihnen eingegebenen Zeichen als Text in den Puffer des Texteditors eingebracht werden. In diesem Modus können Sie den Text vor der aktuellen Position im Puffer einfügen. Siehe Eingabemodus, vgl. Anfügen-Modus und Kommandomodus.

Eingabemodus

Ein Editor-Modus, in dem die von Ihnen eingegebenen Zeichen als Text in den Puffer des Texteditors eingebracht werden. Wenn Sie ein Kommando durchführen möchten, müssen Sie den Eingabemodus verlassen. Siehe Kommandomodus, vgl. Anfügen-Modus and Einfügen-Modus.

Eingabeaufforderung

Das Zeichen bzw. die Zeichenkette, das/die die Shell auf dem Terminal anzeigt, um Ihnen ihre Empfangsbereitschaft zu signalisieren. Die Standard-Eingabeaufforderung ist unter UNIX das Dollar-Symbol (`$`).

Elektronische Post

Die Einrichtung eines Betriebssystems, mit der die Benutzer des Rechners Nachrichten über den Rechner austauschen können. Unter UNIX gibt es zu diesem Zweck die Kommandos `mail` und `mailx`, bei denen die Adressierung über die Benutzernamen erfolgt.

Fehlermeldung

Eine Meldung, die auf Ihrem Terminal ausgegeben wird und Sie über einen Fehler informiert, der bei der Durchführung eines Kommandos oder Programms aufgetreten ist. Auf eine Fehlermeldung müssen Sie normalerweise nicht direkt reagieren.

Fernes System

Ein System, an dem Sie aktuell nicht arbeiten.

Filter

Ein Kommando, das die Standard-Eingabe einliest, sie auf eine bestimmte Art verarbeitet und das Ergebnis dann als Standard-Ausgabe ausgibt.

Gerätedatei

Eine (als Gerätetreiber bezeichnete) Datei, die als Schnittstelle zu einem Ein/Ausgabegerät wie einem Benutzer-Terminal, einem Plattenlaufwerk oder einem Zeilendrucker benutzt wird.

Global

Ein Begriff, mit dem die vollständige bzw. ganze Datei angegeben wird. Während normale Editor-Kommandos lediglich das erste Auftreten eines Musters in der Datei bearbeiten, können globale Kommandos das Muster bei jedem Auftreten in der Datei verarbeiten.

Hardware

Die physischen Komponenten eines Rechners und der zugehörigen Peripheriegeräte.

Hintergrund

Eine Methode zur Ausführung eines Programms, bei der während des Programmlaufs kein Dialog zwischen Ihnen und dem Rechner stattfindet. Die Shell gibt während des Programmlaufs ihre Eingabeaufforderung aus, so daß Sie am Terminal weitere Kommandos aufrufen können. Vgl. Vordergrund.

Home-Verzeichnis

Das Verzeichnis, das nach Ihrer Anmeldung beim UNIX-System automatisch für Sie aufgerufen wird.

Interaktiv

Die Eigenschaft eines Betriebssystems (z.B. des UNIX-Systems), bei dem Sie auf Ihre Eingaben sofort eine Reaktion erhalten. Zwischen Ihnen und dem Rechner findet also ein direkter Dialog statt; daher wird auch häufig das Synonym dialogfähig benutzt.

Kommando

Der Name einer Datei, die ein Programm enthält; dieses Programm kann vom Rechner bei der entsprechenden Anforderung ausgeführt werden. Übersetzte Programme und Shell-Programme sind spezielle Kommandovarianten.

Kommandodatei

Siehe Ausführbare Datei.

Kommandointerpreter

Ein Programm, das als direkte Schnittstelle zwischen Ihnen und dem Rechner fungiert. Unter UNIX liest ein spezielles Programm, die sogenannte Shell, die Kommandos ein und setzt sie in eine Sprache um, die vom Rechner verstanden wird.

Kommandomodus

Der Modus eines Texteditors, in dem die von Ihnen eingegebenen Zeichen als Editierkommandos interpretiert werden. Dies ermöglicht Ihnen das Anspringen von verschiedenen Stellen im Puffer, das Löschen von Text sowie das Versetzen von Textzeilen. Siehe Eingabemodus; vgl. Anfügen-Modus und Einfügen-Modus.

Kommandozeile

Eine Zeile, die ein oder mehr Kommandos enthält und mit dem Wagenrücklauf-Zeichen (<CR>) abgeschlossen wird. Die Zeile kann außerdem Optionen und Argumente für die Kommandos enthalten. Die Kommandozeile geben Sie bei einer Shell ein, um den Rechner zur Durchführung von ein oder mehreren Funktionen anzuweisen.

Konkatenation

(Aneinanderreihung). Die Verknüpfung von zwei Zeichenketten zu einer einzigen, in der die Zeichen der zweiten Zeichenkette unmittelbar auf die Zeichen der ersten folgen.

Kontextbezogener Suchvorgang

Eine Methode zur Suche eines bestimmten Zeichenmusters (einer sogenannten Zeichenkette) in einer Editor-Sitzung. Die Editier-Kommandos, mit denen der kontextbezogene Suchvorgang gestartet wird, durchsuchen den Puffer nach einer Stelle, an der die im Kommando angegebene Zeichenkette vorkommt. Siehe Zeichenkette.

Metazeichen

Eine bestimmte Art von Sonderzeichen, die für die Shell eine spezielle Bedeutung haben. Als Metazeichen werden `*`, `?` sowie die Zeichenkette `[ ]` benutzt. Metazeichen werden in Mustern benutzt, um mehrere Dateien gleichzeitig anzusprechen.

Modem

Ein Gerät, mit dem die Verbindung zwischen einem Terminal und einem Rechner über eine Telephonleitung hergestellt werden kann. Ein Modem wandelt digitale Signale in analoge Impulse um und umgekehrt. Dadurch wird einem Terminal und einem Rechner der Datenaustausch über das öffentliche Telephonnetz ermöglicht.

Modus

Im allgemeinen eine Bezeichnung für eine bestimmte Betriebsart (z.B. den Anfügen-Modus eines Editors).

Multitasking

Die Fähigkeit eines Betriebssystems, mehrere Programme gleichzeitig auszuführen. Ein derartiges Betriebssystem wird auch als Mehrprozeß-Betriebssystem bezeichnet.

Multiuser

Die Fähigkeit eines Betriebssystems, mehrere Benutzer gleichzeitig zu bedienen. Man spricht hier auch von Mehrbenutzer-Umgebung.

Normale Datei

Eine Datei, die Text oder sonstige Daten enthält; sie ist nicht ausführbar. Siehe Ausführbare Datei.

Option

Eine spezielle Anweisung, über die der Ablauf eines Kommandos verändert werden kann. Eine Option ist ein Argument, das auf den Kommandonamen folgt und im Normalfall den übrigen Argumenten in der Kommandozeile vorangestellt ist. Einer Option wird üblicherweise ein Minus-Zeichen (–) vorangestellt, um sie gegenüber den anderen Argumenten abzugrenzen. Bei einigen UNIX-Kommandos sind zwei oder mehr Optionen zulässig. So sind beim Kommando `ls -l -a verzeichnis`, `-l` und `-a` Optionen, mit denen der Ablauf des Kommandos `ls` geändert werden kann. Siehe Argument.

Parameter

Ein bestimmter Variablen-Typ innerhalb von Shell-Programmen, der zum Zugriff auf Werte benutzt wird, die zu Argumenten in der Kommandozeile oder zur Ausführungs-Umgebung des Programms gehören. Siehe Stellungsparameter.

Parität

Eine Methode, mit der auf einem Rechner überprüft wird, ob die empfangenen Daten den abgeschickten entsprechen.

Paßwort

Ein nur Ihnen bekanntes Code-Wort, mit dem Sie sich bei UNIX anmelden. Der Rechner überprüft anhand des Paßworts Ihre Zugriffsberechtigung für das System.

Peripheriegerät

Geräte, die durch den Host angesteuert und in der Regel für die Ein-/Ausgabe und Datenspeicherung benutzt werden. Zu den Peripheriegeräten zählen z.B. das Terminal, der Drucker sowie die Plattenlaufwerke.

Permanente Datei

Die Daten, die permanent im Dateisystem abgespeichert sind. Die Änderung einer permanenten Datei kann mit Hilfe eines Texteditors erfolgen; die Änderungen werden während einer Editor-Sitzung in einem temporären Arbeitsbereich, dem sogenannten Puffer vorgenommen. Damit die Änderungen am Puffer erhalten bleiben, müssen Sie nach Abschluß der Änderungsarbeiten in die permanente Datei eingebracht werden. Siehe Puffer.

Pfadname

Eine Folge von Verzeichnis-Namen, die durch einen Schrägstrich (/) voneinander getrennt sind; ein Pfadname endet mit dem Namen einer Datei oder eines Verzeichnisses. Der Pfadname gibt die Verbindung zwischen einem bestimmten Verzeichnis und der angegebenen Datei an.

Pipe

Eine Methode zur Umleitung der Ausgabe eines Kommandos, so daß sie von einem anderen Kommando als Eingabe benutzt werden kann. Das Pipe-Symbol ist der senkrechte Strich (|). So übergibt das Shell-Kommando `who | wc -l` die Ausgabe des Kommandos `who` an das Kommando `wc`; hiermit können Sie die Gesamtzahl der aktuell angemeldeten Benutzer ermitteln.

Pipeline

Eine Aneinanderreihung von Filtern, die durch das Pipe-Symbol (|) voneinander getrennt sind. Die Ausgabe jedes Filters wird zur Eingabe des nächsten Filters in der Kommandozeile. Der letzte Filter innerhalb der Pipeline gibt sein Ergebnis auf seiner Standard-Ausgabe aus; auch die Umleitung der Ausgabe in eine Datei ist möglich. Siehe Filter.

Platte

Ein Magnetspeicher, der aus einer Reihe von runden, der Schallplatte ähnlichen Scheiben besteht. Eine Platte kann große Datenmengen aufnehmen und ermöglicht den schnellen Zugriff auf alle Daten.

Programm

Die Anweisungen, mit denen der Rechner zur Durchführung einer bestimmten Aufgabe aufgefordert wird. Programme können vom Benutzer ausgeführt werden.

Prozeß

Im allgemeinen ein Programm, das gerade ausgeführt wird. Unter UNIX werden damit außerdem der aktuelle Stand einer Rechnerumgebung wie z.B. der Speicher-Inhalt, Register-Werte, der Name des aktuellen Verzeichnisses, der Status von Dateien, die bei der Anmeldung protokollierten Informationen und andere Daten bezeichnet.

Puffer

Ein temporärer Speicherbereich des Rechners, in dem die Texteditoren ihre Änderungen an der Kopie einer vorhandenen Datei durchführen. Während einer Editor-Sitzung wird der Inhalt der betreffenden Datei in den Puffer eingelesen; die Änderungen finden dann nur am Puffer-Inhalt statt. Damit die Änderungen permanent in die Datei eingebracht werden, müssen Sie den Puffer-Inhalt wieder in die Datei zurückschreiben, d.h. ihn abspeichern. Siehe Permanente Datei.

Quellcode

Die nicht übersetzte Version eines Programms, das in einer Programmiersprache wie C oder Pascal erstellt worden ist. Bevor der Rechner das Programm durchführen kann, muß der Quellcode durch ein Programm, den sogenannten Compiler, in die Maschinensprache übersetzt werden.

Relativer Pfadname

Der Zugriffspfad für eine Datei oder ein Verzeichnis, der von der Position des aktuellen Verzeichnisses innerhalb des Dateisystems abhängig ist. Siehe Pfadname, vgl. Absoluter Pfadname.

Root

Das Verzeichnis, dem sämtliche Dateien und Verzeichnisse innerhalb des Dateisystems untergeordnet sind; es wird über den Schrägstrich (/) angegeben.

Sekundäre Eingabeaufforderung

Ein Zeichen oder eine Zeichenkette, das/die von der Shell auf Ihrem Terminal immer dann angezeigt wird, wenn die bei der primären Eingabeaufforderung getätigte Eingabe unvollständig ist. Unter UNIX wird als sekundäre Eingabeaufforderung standardmäßig das Größer-als-Zeichen (>) benutzt.

Shell

Ein UNIX-Programm, das den Dialog zwischen Ihnen und dem Rechner steuert. Die Shell wird auch als Kommandosprachen-Interpreter bezeichnet, da sie Ihre Kommandos in eine Sprache übersetzt, die vom Rechner verstanden wird. Die Shell liest die Kommandos ein und bewirkt die Ausführung des entsprechenden Programms. Unter den Einträgen `sh`(1) und `ksh`(1) im *Referenzhandbuch für Benutzer* werden zwei der verfügbaren Shells beschrieben.

Shell-Prozedur

Eine ausführbare Datei, bei der es sich nicht um ein übersetztes Programm handelt. Eine Shell-Prozedur ruft die Shell auf, damit sie die in einer Datei enthaltenen Dateien einliest und ausführt. Dies gibt Ihnen die Möglichkeit, eine Kommandofolge, die Sie öfter benötigen, in einer Datei abzuspeichern. Statt "Shell-Prozedur" werden auch häufig die Begriffe "Shell-Programm" oder "Kommandodatei" benutzt. Siehe Ausführbare Datei.

Software

Anweisungen und Programme, die den Rechner zur Durchführung von bestimmten Funktionen auffordern. Das Gegenstück zur Hardware.

Sohn-Verzeichnis

Siehe Unterverzeichnis.

Sonderzeichen

Ein Zeichen, das in einem Programm eine spezielle Bedeutung hat. Die Sonderzeichen der Shell werden für gängige Shell-Funktionen wie die Umleitung einer Datei, die Pipe, die Ausführung im Hintergrund sowie die Dateinamenerweiterung benutzt. Zu den Sonderzeichen der Shell gehören z.B. <, >, |, ;, &, *, ?, [und]. In Editoren wie ed und vi gibt es ebenfalls Sonderzeichen.

Standard

Ein automatisch zugewiesener Wert bzw. eine automatisch vorausgesetzte Bedingung, der/die in jedem Fall gilt, es sei denn, Sie geben explizit eine andere Einstellung an. So hat die Zeichenkette für die Eingabeaufforderung der Shell den Standard-Wert $, es sei denn, Sie legen eine andere Eingabeaufforderung an.

Standard-Ausgabe

Eine eröffnete Datei, die im Normalfall direkt mit einem primären Ausgabegerät wie z.B. einem Terminal- Drucker oder -Bildschirm verbunden ist. Die Standard-Ausgabe des Rechners wird normalerweise zuerst in diese Datei gesetzt und dann zum Ausgabegerät weitergeleitet. Um als Standard-Ausgabe eine Datei zu verwenden, die nicht dem Drucker bzw. Bildschirm zugeordnet ist, geben Sie ein Argument im Format > *datei* ein. Die Ausgabe wird dann zur angegebenen Datei umgeleitet.

Standard-Eingabe

Eine eröffnete Datei, die im Normalfall direkt mit der Tastatur verbunden ist. Die Standard-Eingabe für ein Kommando wird normalerweise von der Tastatur zu dieser Datei geleitet und dann zur Shell. Wenn die Standard-Eingabe nicht der Tastatur zugeord[illegible] sein soll, können Sie sie so umleiten, daß sie aus einer anderen Datei g[illegible] wird; hierzu benutzen Sie ein Argument im Format < *datei*. Die Eingal[illegible]es Kommandos wird dann der angegebenen Datei entnommen.

Standard-Fehlerausgabe

Eine eröffnete Datei, die im Normalfall direkt einem primären Ausgabegerät wie z.B. einem Terminal-Drucker oder -Bildschirm zugeordnet ist. Sämtliche Fehlermeldungen u.ä. werden normalerweise zuerst zu dieser Datei und dann an das Ausgabegerät weitergeleitet. Wenn die Fehlermeldungen nicht auf dem Drucker oder dem Bildschirm ausgegeben werden sollen, können Sie die Standard-Fehlerausgabe in eine andere Datei umleiten; hierzu benutzen Sie ein Argument im Format `2>` *datei*. Die Fehlerausgabe wird dann zu der angegebenen Datei umgeleitet.

Stellungsparameter

Numerische Variablen innerhalb einer Shell-Prozedur, über die auf die Zeichenketten zugegriffen wird, die in der Kommandozeile zum Aufrufen der Shell-Prozedur enthalten sind. Der Name der Shell-Prozedur hat den Stellungsparameter `$0`. Siehe Variable und Shell-Prozedur.

Steuerzeichen

Ein nicht-darstellbares Zeichen, das eingegeben wird, indem man die CONTROL-Taste gedrückt hält und gleichzeitig ein Zeichen eingibt. Die Steuerzeichen werden häufig für bestimmte Aufgaben benutzt. Wenn Sie beispielsweise mit dem Kommando `cat` eine lange Datei auf dem Bildschirm abrufen, können Sie die Bildschirmausgabe mit CONTROL-s (`^s`) anhalten, so daß Sie die Anzeige in Ruhe durchlesen können; durch die Eingabe von CONTROL-q (`^q`) können Sie die Bildschirmausgabe dann wieder fortsetzen.

Suchmuster

Siehe Zeichenkette.

Systemverwalter

Die Person, die für den ordnungsgemäßen Betrieb des Rechners verantwortlich ist, auf dem Ihr UNIX-System installiert ist.

Tastaturpuffer

Der Puffer, in den das UNIX-System Ihre Eingabe einliest, während es Ihnen als Antwort auf eine vorhergehende Eingabe Informationen sendet. Unter UNIX wird die Eingabe ordnungsgemäß von der Ausgabe und den Prozessen getrennt.

Terminal

Ein Ein/Ausgabegerät, das an einen Rechner angeschlossen ist und im Normalfall aus einer Tastatur sowie einem Datensichtgerät oder einem Drucker besteht. An einem Terminal können Sie Anweisungen für den Rechner eingeben, der seine Antworten dann wiederum auf dem Terminal ausgibt.

Texteditor

Ein Programm zum Erstellen, Ändern und Löschen von Text über den Rechner. Bei den meisten Texteditoren gibt es zwei Modi: Den Eingabemodus, in dem Text eingegeben werden kann, und den Kommandomodus, in dem Text versetzt oder abgeändert werden kann. UNIX stellt beispielsweise die Editoren `ed` und `vi` zur Verfügung. Siehe Zeileneditor rnd Bildschirmeditor.

Textformatierungssystem

Ein Programm, das eine Textdatei für die Druckausgabe vorbereitet. Die Verwendung eines Textformatierungssystems setzt voraus, daß Ihre Datei spezielle Kommandos zur Strukturierung der Druckausgabe enthält. Diese speziellen Kommandos legen die Randeinstellungen fest, den Beginn neuer Absätze, das Format von Listen und Tabellen, die Position von Abbildungen usw. Unter UNIX gibt es die beiden Textformatierungssysteme `nroff` und `troff`.

Teilnehmersystem (Timesharing)

Ein Betriebssystem, das mehreren Benutzern das gleichzeitige Arbeiten am Rechner ermöglicht. Der Rechner führt den Dialog mit den einzelnen Benutzern in einem so schnellen Wechsel durch, daß bei den Benutzern der Eindruck entsteht, sie würden alleine am Rechner arbeiten. Ein derartiges System wird auch häufig als Timesharing-System bezeichnet.

tty

Ursprünglich die Abkürzung für ein Terminal der Firma Teletype®. Inzwischen wird diese Abkürzung als Bezeichner für jedes beliebige Benutzer-Terminal verwendet.

Umgebung

Die Gesamtheit der Parameter, die beim Arbeiten unter UNIX gültig sind. In der Umgebung ist z.B. Ihre Anmelde-Prozedur festgelegt, außerdem die spezielle Schnittstelle zwischen Ihnen und dem UNIX-System bzw. dem Rechner. So enthält Ihre Shell-Umgebung beispielsweise die Zeichenkette für die

Eingabeaufforderung der Shell, die Korrektur- und Zeichenlöschzeichen, sowie außerdem Kommandos zur Übermittlung der Ausgabe von Ihrem Terminal an den Rechner.

UNIX-System

Ein universelles, dialogfähiges Mehrplatz-/Teilnehmersystem, das bei den Bell Laboratories von AT&T entwickelt worden ist. Das UNIX-System ermöglicht die Nutzung der begrenzten Rechner-Ressourcen durch mehrere Benutzer gleichzeitig und stellt eine effiziente Schnittstelle zwischen dem Benutzer und dem Rechnersystem zur Verfügung.

Unterverzeichnis

Ein Verzeichnis, zwischen dem und dem unmittelbar übergeordneten Verzeichnis innerhalb des Dateisystems ein Verweis vorhanden ist; häufig ist auch der Begriff Sohn-Verzeichnis anzutreffen.

Variable

Ein Symbol, dessen Wert sich ändern kann. Bei der Shell ist eine Variable ein Symbol, das für eine bestimmte Zeichenkette steht. Die Variablen können sowohl im Dialog mit der Shell als auch innerhalb einer Shell-Prozedur benutzt werden. In einer Shell-Prozedur wird zwischen zwei Variablen-Typen (Stellungsparameter und Schlüsselwortparameter) unterschieden (nähere Informationen zu den Schlüsselwortparametern finden Sie im Kapitel 9, "Die Shell").

Vaterverzeichnis

Das Verzeichnis, das einem Unterverzeichnis oder einer Datei innerhalb des Dateisystems unmittelbar übergeordnet ist. Als Kurzform für das Vater-Verzeichnis werden häufig zwei Punkte (..) benutzt.

Verborgenes Zeichen

Ein Zeichen im ASCII- Zeichensatz, das nicht dargestellt werden kann. Hierzu gehören beispielsweise der Rückschritt (Backspace), das Escape-Zeichen sowie <^d>.

Verzeichnis

Ein Datei-Typ, mit dem andere Dateien oder Verzeichnisse nach bestimmten Gesichtspunkten zusammengefaßt werden. In einem Verzeichnis können Text- oder sonstige Daten nicht direkt eingetragen werden (weitere Informationen hierzu finden Sie in Anhang A, "Kurzübersicht über das Dateisystem").

Vollduplex

Eine Methode zur Datenübertragung, bei der ein Rechnersystem gleichzeitig Daten übermitteln und empfangen kann. Terminals und Modems sind normalerweise auf Halbduplex- Betrieb eingestellt; das UNIX-System dagegen arbeitet im Vollduplex-Betrieb.

Vordergrund

Die normale Methode der Kommandoausführung. Wenn Sie ein Kommando im Vordergrund ausführen, wartet die Shell auf die Beendigung des einen Kommandos, bevor sie Sie zur Eingabe eines weiteren Kommandos auffordert. Wenn Sie also eine Anweisung an den Rechner eingeben, muß der Rechner zuerst "antworten", bevor Sie die nächste Eingabe vornehmen können. Siehe auch Hintergrund.

Werkzeug

Ein Programm-Paket.

Zeichenkette

Eine Folge bzw. ein Muster von Zeichen (wie z.B. ein oder mehrere Wörter), die Sonderzeichen enthalten können. Die Sonderzeichen werden in einer Editor-Sitzung während eines kontextbezogenen Suchvorgangs interpretiert, bei dem der Editor-Puffer nach der Zeichenkette durchsucht wird, die zu dem Muster paßt.

Zeichenlösch-Zeichen

(erase). Das Zeichen, über das Sie das zuletzt eingegebene Zeichen löschen. Unter UNIX wird hierzu standardmäßig das Zeichen # benutzt; häufig wird es auch auf die Taste BACKSPACE gelegt.

Zeileneditor

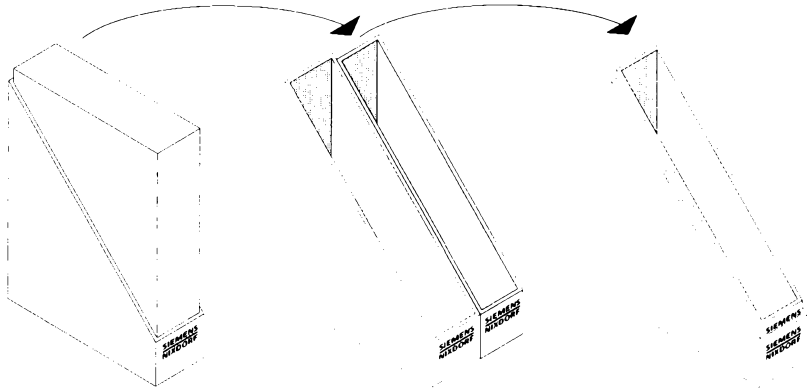
Ein Editier-Programm, bei denen der Inhalt einer Datei zeilenweise verarbeitet wird. Die als nächstes zu bearbeitende(n) Zeile(n) werden in den entsprechenden Kommandozeilen über Zeilenadressen angegeben. Die Änderungen wirken sich sofort auf die Bildschirmanzeige aus. Siehe Texteditor, vgl. Bildschirmorientierter Editor.

Zugriffsberechtigungen

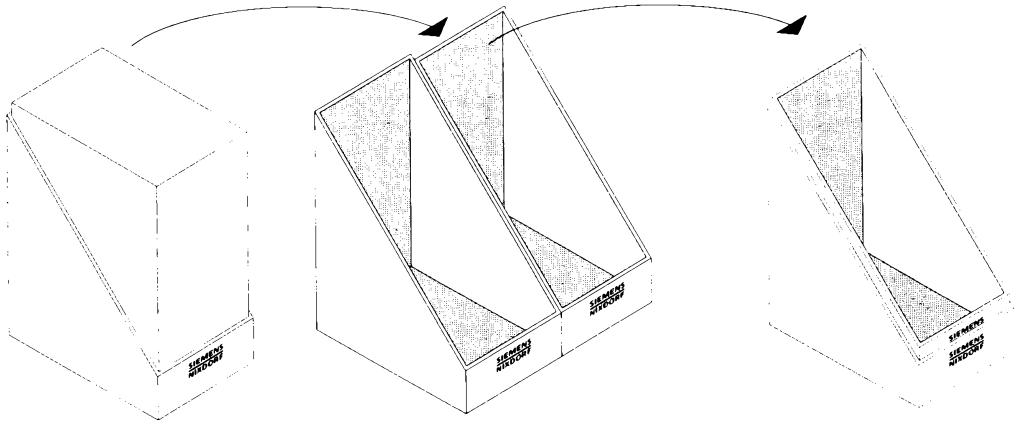
Einstellungen, die Verzeichnissen und Dateien zugeordnet sind und einem Benutzer die Möglichkeit zum Lesen, Ändern und/oder Ausführen der Verzeichnisse bzw. Dateien erteilen bzw. verweigern. Die Zugriffsberechtigungen für Ihre Verzeichnisse und Dateien können Sie über das Kommando `chmod` festlegen.

Sammelboxen

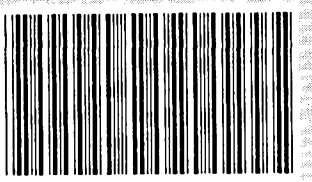
Für Handbücher des vorliegenden Formates bieten wir zweiteilige Sammelboxen in zweierlei Größen an. Der Bestellvorgang entspricht dem für Handbücher.



Breite: ca. 5 cm
Bestellnummer: U3775-J-Z18-1



Breite: ca. 10 cm
Bestellnummer: U3776-J-Z18-1



926219

Herausgegeben von/Published by
Siemens Nixdorf Informationssysteme AG
Postfach 2160, W-4790 Paderborn
Postfach 830951, W-8000 München 83

Bestell-Nr./Order No. ***U6404-J-Z95-1***
Printed in the Federal Republic of Germany
14250 AG 12902. (17820)