Deutsch

BS2000/OSD

POSIX

Kommandos

Benutzerhandbuch

Stand der Beschreibung:
BS2000/OSD V7.0/V8.0/V9.0

Ausgabe November 2012

Kritik… Anregungen… Korrekturen…

Die Redaktion ist interessiert an Ihren Kommentaren zu diesem Handbuch. Ihre Rückmeldungen helfen uns, die Dokumentation zu optimieren und auf Ihre Wünsche und Bedürfnisse abzustimmen.

Sie können uns Ihre Kommentare per E-Mail an manuals@ts.fujitsu.com senden.

Zertifizierte Dokumentation nach DIN EN ISO 9001:2008

Um eine gleichbleibend hohe Qualität und Anwenderfreundlichkeit zu gewährleisten, wurde diese Dokumentation nach den Vorgaben eines Qualitätsmanagementsystems erstellt, welches die Forderungen der DIN EN ISO 9001:2008 erfüllt.

cognitas. Gesellschaft für Technik-Dokumentation mbH
www.cognitas.de

Copyright und Handelsmarken

Inhalt

1 Einleitung

Unter POSIX (**P**ortable **O**pen **S**ystem **I**nterface for UNI**X**) versteht man eine Reihe von Standards auf UNIX-Basis. Diese Standards gewährleisten die Kompatibilität und Interoperabilität von Anwendungen in einem heterogenen Netz. Ein heterogenes Netz besteht aus Rechnern von verschiedenen Herstellern sowie aus System- und Anwendersoftware von verschiedenen Softwareanbietern.

Der POSIX-Standard wurde vom Institute of Electrical and Electronics Engineers (IEEE) 1989 als nationaler amerikanischer Standard definiert. Anschließend wurde er vom X/OPEN-Konsortium erweitert und 1990 als internationaler Standard verabschiedet (X/OPEN Portability Guide IV).

Der X/OPEN Portability Guide IV, kurz auch XPG4-Standard genannt, umfasst 7 Bände, die u.a. Schnittstellendefinitionen zu Basisbetriebssystemen, Programmiersprachen, Datenverwaltung und Vernetzung enthalten. Das Betriebssystem BS2000/OSD unterstützt ab V2.0 die XPG4-Standards, die in den ersten beiden Bänden enthalten sind:

- Band 1: System Interfaces and Headers (ca. 350 Programmschnittstellen)
- Band 2: Commands and Utilities (ca. 200 Benutzerschnittstellen)

Zur Unterstützung dieser Schnittstellen wurde die POSIX-Funktionalität im BS2000/OSD integriert. POSIX bezeichnet also sowohl den Standard vom IEEE als auch die BS2000/OSD-Funktionalität „POSIX“. Mit POSIX wurden die Voraussetzungen für eine erfolgreiche Zertifizierung nach dem XPG4-Standard geschaffen, die in zwei Stufen erfolgte: Ende 2008 erhielt BS2000/OSD von „The Open Group“ (vormals X/OPEN) das „XPG4 Base Branding“ (XPG4) und Mitte 1997 das Branding nach „XPG4 UNIX profile“ (auch XPG4.2 oder UNIX95 genannt). Außerdem wurde BS2000/OSD mit seinem POSIX-Subsystem 1999 von „The Open Group“ als Internet Server zertifiziert.

Der Kern der POSIX-Funktionalität ist als privilegiertes BS2000-Subsystem realisiert. Dem Benutzer stehen die Bibliotheksfunktionen des XPG4-Standards über eine C-Bibliothek und eine definierte Menge von Kommandos über eine Shell (POSIX-Shell) zur Verfügung. Die C-Bibliothek ist Bestandteil des Produkts CRTE (Common Runtime Environment).

Mit POSIX lassen sich Anwendungsprogramme leicht portieren - unabhängig vom ausführenden Betriebssystem. Deshalb können XPG4-konforme Programme nach einer Neuübersetzung auch im BS2000/OSD ablaufen.

Die POSIX-Programmschnittstellen werden parallel zu den BS2000-Programmschnittstellen angeboten. Die gemischte Nutzung von BS2000- und POSIX-Programmschnittstellen in einem Programm ist mit Einschränkungen möglich.

1.1 Konzept der POSIX-Dokumentation

Für das Kennenlernen von POSIX und das Arbeiten mit dem Subsystem POSIX und der POSIX-Shell im BS2000/OSD steht Ihnen folgende Dokumentation zur Verfügung:

- Eine Einführung in das Arbeiten mit dem Subsystem POSIX erhalten Sie im POSIX-Handbuch „Grundlagen für Anwender und Systemverwalter“ [1]. Darüber hinaus werden die Verwaltungsaufgaben beschrieben, die im Zusammenhang mit dem Subsystem POSIX anfallen. Außerdem erfahren Sie, mit welchen BS2000-Softwareprodukten Sie das Subsystem POSIX nutzen können.
- Eine vollständige Beschreibung der POSIX-Kommandos, mit denen Sie in der POSIX-Shell arbeiten können, enthält das vorliegende Handbuch.
- Alle notwendigen Informationen zum Einsatz von bs2fs-Dateisystemen finden Sie im POSIX-Handbuch „BS2000-Dateisystem bs2fs“ [2].

POSIX-Dokumentation im BS2000/OSD-Umfeld

Im BS2000/OSD werden Softwareprodukte funktionell erweitert, so dass Sie auch mit diesen Produkten die POSIX-Funktionalität nutzen können.

Eine Reihe von Dienstprogrammen ermöglichen den Zugriff auf das POSIX-Dateisystem. So können Sie z.B. mit dem EDT Dateien des POSIX-Dateisystems bearbeiten.

Durch die Erweiterung des CRTE gemäß dem XPG4-Standard können Sie mit den C-Bibliotheksfunktionen unabhängig vom ausführenden Betriebssystem portable C-Programme schreiben.

Als Grundlage für den Zugriff auf die POSIX-Funktionalität aus anderen Softwareprodukten wird das POSIX-Handbuch „Grundlagen für Anwender und Systemverwalter“ [1] vorausgesetzt.

1.2 Zielsetzung und Zielgruppen des Handbuchs

Die Kommandos von POSIX stellen eine umfangreiche Sammlung von vielseitig verwendbaren Programmen und Prozeduren zur Steuerung des Arbeitsablaufes dar.
Das vorliegende Handbuch enthält die Beschreibung der POSIX-Kommandos. Es ist ein grundlegendes Nachschlagewerk für alle Benutzer von POSIX, die bereits Grundkenntnisse im Umgang mit POSIX haben. Die Grundkenntnisse sollten dem Wissensstand entsprechen, der durch das POSIX-Handbuch „Grundlagen für Anwender und Systemverwalter“ [1] vermittelt wird. Weiterführende Literatur ist im Literaturverzeichnis aufgeführt.

1.3 Konzept des Handbuchs

Das Kapitel „Arbeiten mit der POSIX-Shell“ auf Seite 23 gibt grundlegende Informationen über den Umgang mit der POSIX-Shell. Zudem enthält es eine Übersicht der Kommandos nach Funktionen und ein Beispiel für den Zugang und das Arbeiten mit der POSIX-Shell.

Im Kapitel „Internationale Umgebung - NLS“ auf Seite 91 erfahren Sie etwas über das Native Language System.

Die Kommandos sind in alphabetischer Reihenfolge im Kapitel „Kommandos“ auf Seite 97 beschrieben. Als Suchhilfe erscheint im äußeren Kolumnentitel jeder Seite der Name des jeweils beschriebenen Kommandos und bei sehr langen Beschreibungen zusätzlich ein Schlagwort.

Im Kapitel „Tabellen und Verzeichnisse“ auf Seite 935 finden Sie einen Überblick über die XPG4-Konformität der POSIX-Kommandos sowie Sonderzeichen und reguläre Ausdrücke der POSIX-Shell. Die Tabellen über die Zeichensätze ASCII und EBCDIC schließen dieses Kapitel ab.

Die Stichwörter am Ende des Buches dienen zum schnelleren Auffinden von Problem- und Begriffserläuterungen. Das gedruckte Handbuch ist aus technischen Gründen auf zwei Bände aufgeteilt.

Readme-Datei

Funktionelle Änderungen der aktuellen Produktversion und Nachträge zu diesem Handbuch entnehmen Sie bitte ggf. der produktspezifischen Readme-Datei.

Readme-Dateien stehen Ihnen online bei dem jeweiligen Produkt zusätzlich zu den Produkthandbüchern unter *http://manuals.ts.fujitsu.com* zur Verfügung. Alternativ finden Sie Readme-Dateien auch auf der Softbook-DVD.

Informationen unter BS2000/OSD

Wenn für eine Produktversion eine Readme-Datei existiert, finden Sie im BS2000-System die folgende Datei:

`SYSRME.<product>.<version>.<lang>`

Diese Datei enthält eine kurze Information zur Readme-Datei in deutscher oder englischer Sprache (<lang>=D/E). Die Information können Sie am Bildschirm mit dem Kommando `/SHOW-FILE` oder mit einem Editor ansehen.
Das Kommando `/SHOW-INSTALLATION-PATH INSTALLATION-UNIT=<product>` zeigt, unter welcher Benutzerkennung die Dateien des Produkts abgelegt sind.

Ergänzende Produkt-Informationen

Aktuelle Informationen, Versions-, Hardware-Abhängigkeiten und Hinweise für Installation und Einsatz einer Produktversion enthält die zugehörige Freigabemitteilung. Solche Freigabemitteilungen finden Sie online unter *http://manuals.ts.fujitsu.com*.

1.4 Änderungen gegenüber dem Vorgänger-Handbuch

Die Ausgabe dieses Handbuches enthält gegenüber der letzten Ausgabe (Bestellnummer: U22794-J-Z125-6) folgende Änderungen:

Das Kapitel „Kommandos“ wurde um folgende Kommandos erweitert:

edtu	BS2000-Dateibearbeiter EDT aufrufen
fuser	Dateinutzer anzeigen
last	Zuletzt angemeldete Benutzer anzeigen
logrotate	Wechsel der Protokolldateien des syslog-Dämonen
ping	Senden von Echo-Request-Paketen an Netzwerkkomponenten
su	Benutzerkennung wechseln

Die Beschreibung der folgenden Kommandos wurde geändert:

man	Online-Dokumentation nutzen
posdbl	Globalen Programm-Cache einrichten und verwalten
tail	Den letzten Teil einer Datei ausgeben

Der Abschnitt „Aufträge“ auf Seite 66 wurde um die Beschreibung der Tastenkombination @@z ergänzt.

1.5 Darstellungsmittel

Die Beschreibung eines jeden Kommandos hält sich, soweit möglich, an ein festes Raster:

- Kopfzeile/Äußerer Kolumnentitel
- Kopfzeile/Innerer Kolumnentitel (optional)
- Hauptüberschrift
- Beschreibung des Kommandos
- Syntax
- Syntaxbeschreibung
- Endestatus (optional)
- Fehlermeldung(en) (optional)
- Datei(en) (optional)
- Umgebungsvariable(n) (optional)
- Internationale Umgebung (optional)
- Beispiel(e) (optional)
- Siehe auch (optional)

Die hier aufgeführten Bestandteile werden im Folgenden erläutert.

Äußerer Kolumnentitel

Der äußere Kolumnentitel enthält den Kommandonamen. Bei umfangreichen Kommandos dient der äußere Kolumnentitel außerdem als Orientierungshilfe. Dem Kommandonamen folgt dann ein Stichwort zum Seiteninhalt.

Innerer Kolumnentitel (optional)

Der innere Kolumnentitel wird bei einer zum Verständnis nötigen Klassifizierung eines Kommandos verwendet, z.B. *eingebautes sh-Kommando*.

Hauptüberschrift

Die Hauptüberschrift enthält:
– den Namen des Kommandos
– eine Kurzbeschreibung des Kommandos
– die englische Bezeichnung des Kommandos

– den Zusatz *(BS2000)*, wenn das Kommando speziell für das Zusammenspiel mit dem BS2000 ergänzend zum XPG4-Standard angeboten wird.

Beschreibung des Kommandos

Hier ist beschrieben:
– die Arbeitsweise des Kommandos
– die unterschiedlichen Aufgaben der verschiedenen Kommando-Formate, falls mehrere Formate vorhanden sind
– in welcher Umgebung das Kommando zu verwenden ist (z.B. Einträge in Dateien, Zugriffsrechte, ...)
– Hintergrundinformationen
– was vor oder nach dem Kommandoaufruf zu beachten ist.

Bei Kommandos, die komplexe Programme aufrufen (z.B. *awk, sh*), ist hier nur der Programmaufruf beschrieben. Weitergehende Informationen, z.B. die Arbeitsweise von *awk*, finden Sie im Anschluss an die Syntax.

Syntax

Syntax **cmd**[␣**-a**][␣**-b**][␣**-c**][␣**-d**arg1][␣**-f**␣arg2]␣datei␣...

Die Bedeutung der Metasyntax können Sie der folgenden Beschreibung entnehmen.

In der Syntax:

halbfette Zeichen
: Konstanten: Diese Zeichen sind direkt so einzugeben, wie sie gedruckt sind.

normale Zeichen
: Variablen: Diese Zeichen sind Stellvertreter für andere Zeichen, die Sie auswählen und eingeben.

[ ]
: Optional: Alles, was zwischen den eckigen Klammern steht, können, aber müssen Sie nicht eingeben. Welche Auswirkungen das hat, entnehmen Sie der Beschreibung der Optionen und Argumente. Die eckigen Klammern selbst dürfen Sie nicht eingeben, es sei denn, es ist ausdrücklich angegeben.

|
: Ein senkrechter Strich kennzeichnet Alternativen, von denen nur eine ausgewählt werden darf.

␣
: Ein Leerzeichen, das Sie eingeben müssen.

...
: Der vorhergehende Ausdruck kann wiederholt werden. Falls zwischen den Wiederholungen Leerzeichen eingegeben werden müssen, die nicht im Ausdruck enthalten sind, steht vor ... ein ␣ (Leerzeichen).

Beispiel

```
cmd[␣-a][␣-b][␣-c][␣-darg1][␣-f␣arg2]␣datei␣...
```

Sie müssen eingeben:

– *cmd*

– für *datei* eine oder mehrere Dateien, die jeweils durch ein Leerzeichen getrennt werden.

Sie können zusätzlich angeben:

– eine oder mehrere der Optionen *-a, -b, -c*; diese Optionen können Sie einzeln angeben:
-a -b -c
oder zusammengefasst angeben:
-abc

– die Option *-d*, dabei muss *arg1* durch ein Argument ersetzt werden

– die Option *-f*, dabei muss *arg2* durch ein Argument ersetzt werden.

Im Fließtext:

Im Fließtext wird nicht zwischen Konstanten und Variablen unterschieden; alle Elemente der Syntax sowie sonstige Dateinamen, Pfadnamen und Kommandos sind dort in *kursiver* Schrift wiedergegeben.

Eingabe

In Anwendungsbeispielen sind Eingaben in das System `dicktengleich halbfett` dargestellt. Alle Eingabezeilen werden bei Zeichenterminals mit der Taste ↵ abgeschlossen, bei Blockterminals mit den Tasten EM DUE. Deshalb werden die Tasten am Ende der Eingabezeilen nicht angegeben.

Manche Eingaben sind terminalabhängig, d.h. sie unterscheiden sich bei Block- und Zeichenterminals (siehe dazu auch Abschnitt „Unterstützung von Terminals“ auf Seite 29).

Ausgabe

Ausgaben des Betriebssystems werden, außer im Fließtext, `dicktengleich` dargestellt. Im Fließtext erscheinen die Ausgaben des Systems *kursiv*.

Syntaxbeschreibung

option
(Näheres zu Optionen siehe Abschnitt „Optionen angeben“ auf Seite 36)

argument
Beschreibung der übrigen Argumente, die Sie beim Aufruf an ein Kommando übergeben können, z.B. Eingabedateien, Ausgabedateien, Parameter, Variablen, Feldtrenner usw.

Endestatus (optional)

Der Endestatus ist der Wert, den ein Kommando nach seiner Ausführung an den aufrufenden Prozess zurückliefert. Der Endestatus gibt Auskunft darüber, wie das Kommando abgelaufen ist. Der Endestatus ist ein Zahlenwert und wird in der Variablen ? abgelegt. Sie können den Endestatus mit dem Befehl *echo $?* abfragen.

Der Abschnitt Endestatus ist nur angegeben, wenn er von folgendem Regelfall abweicht:

0 nach korrekter Durchführung des Kommandos

> 0 bei Fehler

Fehlermeldung(en) (optional)

Hier werden wichtige Fehlermeldungen angegeben und erläutert sowie Hinweise zur Fehlervermeidung und -behebung gegeben.

Fehlermeldungen werden standardmäßig auf die Standard-Fehlerausgabe ausgegeben. Normalerweise ist der Bildschirm die Standard-Fehlerausgabe.

Datei(en) (optional)

Hier werden solche Dateien angegeben, auf die das betreffende Kommando zugreift oder die dieses anlegt.

Umgebungsvariable(n) (optional)

Einige Kommandos greifen auf Umgebungsvariablen zu. Diese werden hier aufgeführt.

Internationale Umgebung (optional)

Hier sind die Auswirkungen des NLS auf das Kommando beschrieben (siehe Kapitel „Internationale Umgebung - NLS“ auf Seite 91).

Beispiel(e) (optional)

Beispiele sollen veranschaulichen:

– die Hauptfunktion des Kommandos

– den Einsatz der wesentlichen Optionen

– sinnvolle komplexe Kombinationen von Optionen und Argumenten

Siehe auch (optional)

In diesem Abschnitt finden Sie Verweise auf andere Kommandos, die eine ähnliche Funktionsweise haben oder mit dem betreffenden Kommando zusammenarbeiten. Außerdem wird auf weitere Literatur zu diesem Kommando verwiesen.
Systemaufrufe und Bibliotheksfunktionen für den C-Software-Entwickler sind durch ein Paar runder Klammern gekennzeichnet, z.B. *chdir()*.

Hinweise und Warnungen

BS2000 Abschnitte, die auf Besonderheiten im Zusammenspiel mit dem BS2000 hinweisen, sind auf diese Weise gekennzeichnet.

Dieses Symbol kennzeichnet wichtige Hinweise, die Sie unbedingt beachten sollten.

Dieses Symbol steht vor Warnungen, die Sie im Interesse der System- und Betriebssicherheit unbedingt beachten müssen.

Verweise

Die folgenden Beispiele zeigen, in welcher Form auf andere Textabschnitte, Handbücher, Kommandos, Funktionen, Systemaufrufe und Dateiformate verwiesen wird.

siehe Abschnitt „Kommando-Übersicht“ auf Seite 70	Verweis auf den Abschnitt „Kommando-Übersicht“ im vorliegenden Handbuch
siehe „Grundlagen für Anwender und Systemverwalter“ [1]	Verweis auf ein Handbuch, das im Literaturverzeichnis unter der Nummer [1] aufgeführt ist
siehe auch *awk*	Verweis auf das Kommando *awk* im vorliegenden Handbuch
siehe auch *chdir()* [4]	Verweis auf die C-Funktion, den Systemaufruf oder das Dateiformat *chdir()* in dem Handbuch, das im Literaturverzeichnis unter der Nummer [4] aufgeführt ist

2 Arbeiten mit der POSIX-Shell

Die POSIX-Shell ist die Schnittstelle, die Sie über das C-Laufzeitsystem/Bibliotheken mit dem POSIX-Subsystem verbindet.

Bild 1 zeigt die Struktur von POSIX im BS2000 und die Einbettung der POSIX-Shell.

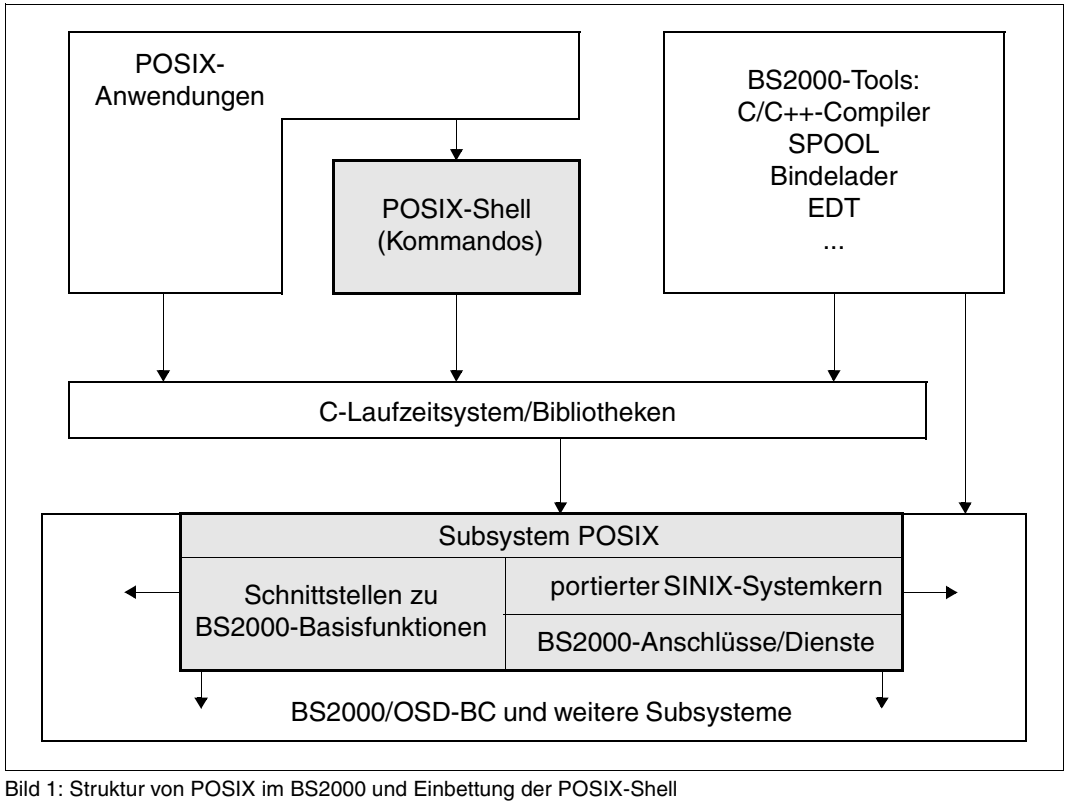


Bild 1: Struktur von POSIX im BS2000 und Einbettung der POSIX-Shell

Die POSIX-Shell ist eine Kommandoschnittstelle, die Sie zusätzlich zur Kommandoschnittstelle des BS2000 verwenden können (siehe Bild 2 auf Seite 24).

Nach erfolgreichem Zugang zur POSIX-Shell stehen Ihnen alle Kommandos der POSIX-Shell zur Verfügung. Nach dem Verlassen der POSIX-Shell können Sie wieder BS2000-Kommandos eingeben.

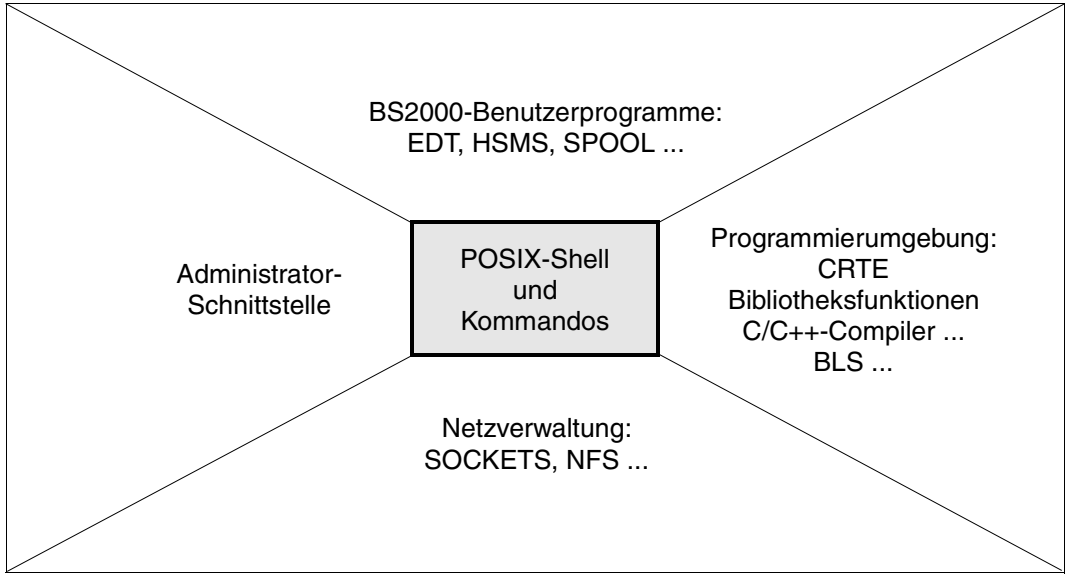


Bild 2: Die Kommandoebene POSIX-Shell

Die POSIX-Shell liest Kommandos von einer Datensichtstation oder aus einer Datei des POSIX-Dateisystems, interpretiert sie nach bestimmten Regeln und sorgt für die Ausführung. Eine Datei, die Kommandos für die POSIX-Shell enthält, heißt Shell-Prozedur (Shell-Skript).

Die Bedienung und Leistung der POSIX-Shell hängen davon ab, ob die Datensichtstation, an der der Benutzer arbeitet, ein Block- oder ein Zeichenterminal ist.

Die POSIX-Shell bietet Ihnen eine umfangreiche Kommandosprache, die sich wie eine Programmiersprache anwenden lässt. Sie können mit den vorhandenen Kommandos eigene Programme erstellen und ohne vorheriges Übersetzen ausführen.

2.1 Zugang zur POSIX-Shell

Zur POSIX-Shell gibt es folgende Zugangsmöglichkeiten:

- über ein BS2000-Terminal (Blockterminal)
- von einem UNIX-System (Zeichenterminal)
- über eine Emulation

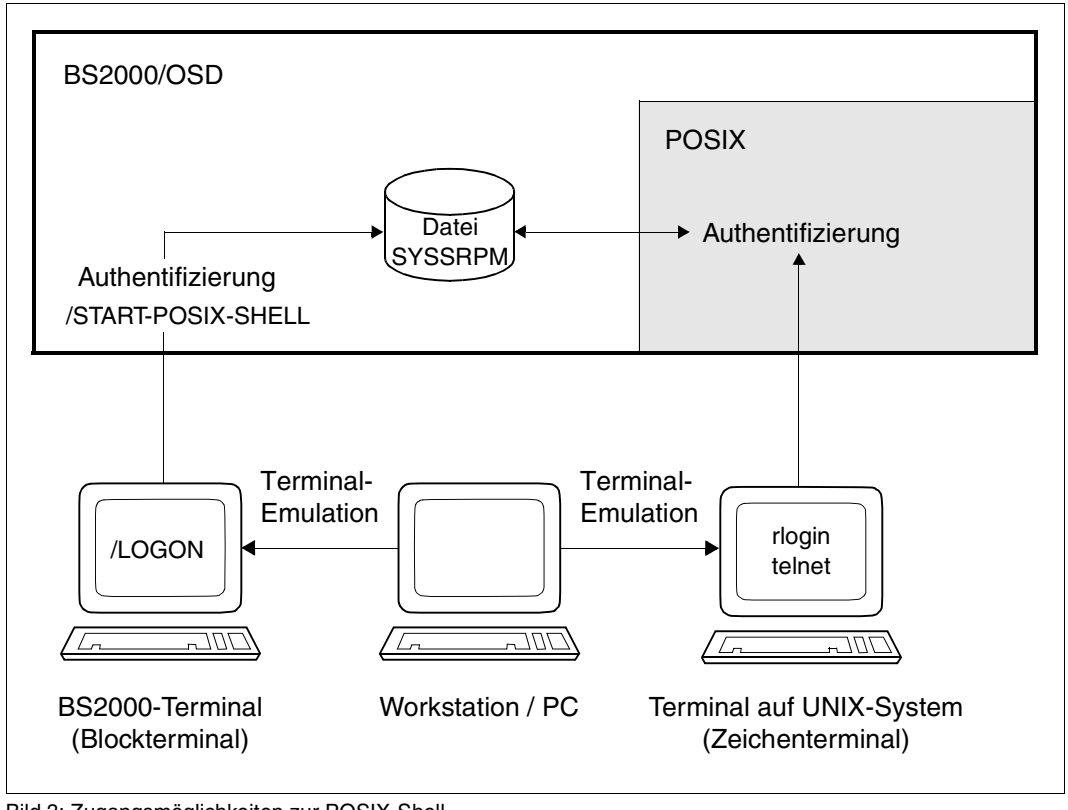


Bild 3: Zugangsmöglichkeiten zur POSIX-Shell

Zugang über ein BS2000-Terminal

Jeder BS2000-Benutzer kann nach dem erfolgreichen BS2000-LOGON mit dem BS2000-Kommando /START-POSIX-SHELL die POSIX-Shell anfordern:

START-POSIX-SHELL
VERSION = **<u>*STD</u>** / <product-version without-man-corr>

VERSION = <u>*STD</u> / <product-version without-man-corr>
Versionsnummer des aufzurufenden Programms (hier der POSIX-Shell).
Voreingestellt ist *STD, d.h. es wird die aktuell verfügbare Version aufgerufen.

Kommando-Returncodes

(SC2)	SC1	Maincode	Bedeutung / garantierte Meldungen
	0	CMD0001	Ohne Fehler
255	1	CCM0999	Die POSIX-Shell konnte nicht gestartet werden.

Wirkung des Kommandos START-POSIX-SHELL

Mit diesem Kommando wird die POSIX-Umgebung aufgebaut und das Programm aufgerufen, das in der SYSSRPM-Datei für den betreffenden Benutzer eingetragen ist (siehe Kommando SHOW-POSIX-USER-ATTRIBUTES).

Hat der Benutzer die POSIX-Shell als Standardprogramm in seinen Benutzerdaten eingetragen, kann er nach /START-POSIX-SHELL interaktiv mit der POSIX-Shell arbeiten und es stehen ihm alle Kommandos und Funktionen der POSIX-Shell zur Verfügung. Für Interaktionen zwischen dem BS2000 und dem POSIX-Subsystem gibt es in der POSIX-Shell eigene Kommandos.

Beenden der POSIX-Shell

Eine Rückkehr ins BS2000 ist nur möglich, indem der Benutzer mit dem Kommando *exit* die POSIX-Shell beendet.

Zugang von einem Zeichenterminal

Der Benutzer kann sich von einem Terminals eines UNIX-Systems mit dem Kommando *rlogin* an einem BS2000-Rechner anmelden, wenn er für den BS2000-Rechner eine Zugriffsberechtigung besitzt. Dazu benötigt er am BS2000-Rechner eine Benutzerken-

nung, für die der POSIX-*rlogin*-Zugang erlaubt ist, mit dem zugehörigen Kennwort und einer Abrechnungsnummer, die auch für die Abrechnung des *rlogin*-Zugangs verwendbar ist. Nach dem Anmelden kann er POSIX wie im lokalen Modus benutzen.

Um sich am BS2000-Rechner anzuschließen, muss der Benutzer folgendes Kommando in der POSIX-Shell eingeben:

```
rlogin [-l <benutzerkennung>] <host>
```

Wenn der Benutzer keine Benutzerkennung eingibt, wird die Kennung verwendet, unter der er am fernen Rechner angemeldet ist. Beim *rlogin* wird für die gewünschte Benutzerkennung das Kennwort erfragt. Das Kennwort wird über die BS2000-Komponente SRPM (**S**ystem **R**esources and **P**rivileges **M**anagement) verifiziert: Die Angaben für die BS2000-Benutzerkennung und das Kennwort werden gegen die Zugangskontroll-Attribute des Home-Pubsets geprüft. Wenn Übereinstimmung besteht, erhält der Benutzer Zugang zum POSIX-Subsystem. Ist das Produkt SECOS im Einsatz, kann die Zugangskontrolle über die LOGON-Protection noch weiter verfeinert werden.

Beim Zugang über *rlogin* kann der Benutzer nur eingeschränkt auf BS2000-Kommandos zugreifen (z.B. wegen der fehlenden SYSFILE-Umgebung).

Zugang über telnet

Mit dem *telnet*-Dämon *telnetd* wird ein direkter Zugang zum BS2000 über das *telnet*-Protokoll sowohl vom UNIX-System aus als auch vom PC direkt über die *telnet*-Anwendung realisiert, die sich gegenüber POSIX wie ein Zeichenterminal verhält. Die Zugangskontrolle erfolgt in derselben Weise wie bei *rlogin*, d.h. über BS2000-Zugangsmechanismen. Der Zugang ohne Angabe eines Kennwortes, wie es zwischen UNIX-Systemen möglich ist (durch einen Eintrag in der *.rhosts*-Datei), wird nicht unterstützt.

Zugang über eine Terminal-Emulation

Die dritte Zugangsmöglichkeit besteht über eine Terminal-Emulation. Dazu meldet sich der Benutzer an einer Workstation oder an einem PC an. Anschließend startet er eine Terminal-Emulation, wobei entweder ein Terminal eines UNIX-Systems oder ein BS2000-Terminal emuliert wird:

BS2000-Terminal-Emulation

Beim Zugang über eine BS2000-Terminal-Emulation wie z.B. EM9750 oder MT9750 steht dem Benutzer ein Terminal in der Art eines Blockterminals zur Verfügung. BS2000-Kommandos und /START-POSIX-SHELL können wie über ein BS2000-Terminal eingegeben werden (siehe Seite 26).

Terminal-Emulation für UNIX-Systeme

Terminal-Emulationen stehen für Workstations mit UNIX-System und grafischen, OSF/Motif-basierten Oberflächen und für Windows-PCs zur Verfügung (z.B. EM97801, SINIX-TE). Dabei wird ein Zeichenterminals eines UNIX-Systems emuliert und der Benutzer kann Kommandos wie z.B. *rlogin* eingeben (siehe Seite 26).

2.2 Besonderheiten für das Arbeiten mit der POSIX-Shell

Voreinstellungen in der Benutzerumgebung

Nach erfolgreichem Zugang zum POSIX-Subsystem wird die POSIX-Shell gestartet. Bevor die POSIX-Shell ihr Bereitzeichen ausgibt, werden folgende Voreinstellungen in der Benutzerumgebung getroffen:

- Die POSIX-Shell initialisiert die Standard-Shell-Variablen. Sie weist den folgenden Variablen ihre Standard-Werte zu: *HOME*, *LANG*, *LOGNAME*, *MAIL*, *PATH*, *SHELL*, *TTY*, *TERM*, *TZ* und *USER*.
  Falls eine Variable durch die SDF-P-Variable *SYSPOSIX* bereits definiert ist, wird dieser Wert genommen. Es dürfen jedoch nicht die Shell-Variablen *USER*, *TERM*, *TYP*, *LOGNAME*, *HZ*, *HOME* und *MAIL* vom Anwender gesetzt werden.
- Die Datei */etc/profile* wird ausgeführt.
- Die Datei *$HOME/.profile* wird ausgeführt, falls Sie diese Datei angelegt haben.

Unterstützung von Terminals

POSIX unterstützt neben den im BS2000 verwendeten Blockterminals auch die an UNIX-Systemen verwendeten Zeichenterminals. Diese Terminals sind an UNIX-Mehrplatzsystemen angeschlossen und werden von POSIX über Netze bedient. Beim Zugriff auf POSIX über eine Workstation wird ein Zeichenterminal emuliert.

Block- und Zeichenterminals haben eine unterschiedliche Funktionsweise:

- Bei Zeichenterminals wird jedes eingegebene Zeichen sofort an das UNIX-System übertragen und von dort als Antwort auf die Eingabe an den Bildschirm übergeben und abgebildet. Kontrollfunktionen, wie Cursorbewegung, Groß-/Kleinschreibung oder Pufferung der Übertragung werden von dem Rechner durchgeführt, an dem das Terminal angeschlossen ist. Bildschirmorientierte Anwendungen sind nur an Zeichenterminals zulässig.
- Blockterminals dagegen übergeben den gesamten am Bildschirm eingegebenen Text als Datenblock an den Rechner. Kontrollfunktionen werden im Gerät selbst durchgeführt.

Blockterminals sind direkt mit BS2000 und POSIX verbunden. Von diesem Terminaltyp aus können BS2000- und POSIX-Kommandos eingegeben werden.

Aus BS2000 Sicherheitsgründen kann über Blockterminals nur auf ein Terminal der eigenen Prozessgruppe geschrieben werden. So würde z.B. ein `cp <datei> /dev/term/xxx` für ein Blockterminal einer anderen LOGON-Task mit `permission denied` abgelehnt.

Manche Eingaben sind terminalabhängig, d.h. sie unterscheiden sich bei Block- und Zeichenterminals:

Blockterminal	Zeichenterminal
@@d	END
@@c	DEL
@@/	CTRL \
EM DUE	↵
@@z	CTRL Z
-	CTRL S / CTRL Q ...

Sonderfunktionen (P-Tasten, Ctrl-Taste)

Sie können die P-Tasten P3, P4 und P5 durch den Aufruf des Kommandos *bs2pkey* folgendermaßen belegen:

P3 mit @@c ([CTRL] C)
P4 mit @@d ([CTRL] D)
P5 mit @@z ([CTRL] Z)

Das Programm wird entweder in der POSIX-Shell (ohne Optionen) aufgerufen oder kann in die Datei */etc/profile* aufgenommen werden. Dann wird das Programm bei jedem Aufruf der Shell aktiviert.

Austausch von Dateien

POSIX-Dateien enthalten keine Datensätze, sondern sind byte-orientiert. BS2000-Dateien dagegen enthalten satzorientierte und/oder PAM-Block-orientierte Daten.

POSIX behandelt Dateien standardmäßig im EBCDIC-Format, UNIX-Systeme sowie MS-DOS und Windows behandeln Dateien dagegen im ASCII-Format. Im POSIX-Dateisystem abgelegte ASCII-Dateien können in der POSIX-Shell nur bearbeitet werden, wenn sie vorher konvertiert wurden.

Damit POSIX- und BS2000-Dateien ausgetauscht werden können, stehen Konvertierungsroutinen von POSIX nach BS2000 und umgekehrt zur Verfügung. Es wird vom Zeichensatz EBCDIC.DF.03 in den ASCII-ISO-7-Bit-Code konvertiert und umgekehrt (siehe auch Kapitel „Tabellen und Verzeichnisse“ auf Seite 935). Die Konvertierung ist nur für Textdateien sinnvoll.

Automatische Konvertierung

Mit der Umgebungsvariable *IO_CONVERSION* wird gesteuert, ob Dateien beim Zugriff mit POSIX-Kommandos (z.B. *awk*, *cat*, *grep*...) auf montierte ASCII-Dateisysteme automatisch konvertiert werden. Standardmäßig ist die Umgebungsvariable *IO_CONVERSION* mit dem Wert „NO“ belegt, d.h. es erfolgt keine automatische Konvertierung. Die automatische Konvertierung wird mit folgendem Kommando eingeschaltet:

```
export IO_CONVERSION=YES
```

Falls die automatische Konvertierung für einen POSIX-Benutzer bereits beim Starten der POSIX-Shell voreingestellt sein soll, muss dieses *export*-Kommando in die Datei *.profile* im HOME-Verzeichnis dieses Benutzers eingetragen werden.

Bei Verwendung der folgenden Tools darf die automatische Konvertierung nicht eingeschaltet sein, da diese Tools selbst konvertieren:
dd, *iconv*, *bs2cp* mit Schalter *-k*.

Behandlung von Archiven/Bibliotheken:
ar konvertiert nicht automatisch, da *ar*-Bibliotheken oft binäre Daten enthalten.

pax und *tar* konvertieren automatisch. Ein *pax*- oder *tar*-Archiv darf jedoch nicht mit *cp* kopiert werden, wenn die automatische Konvertierung eingeschaltet ist.

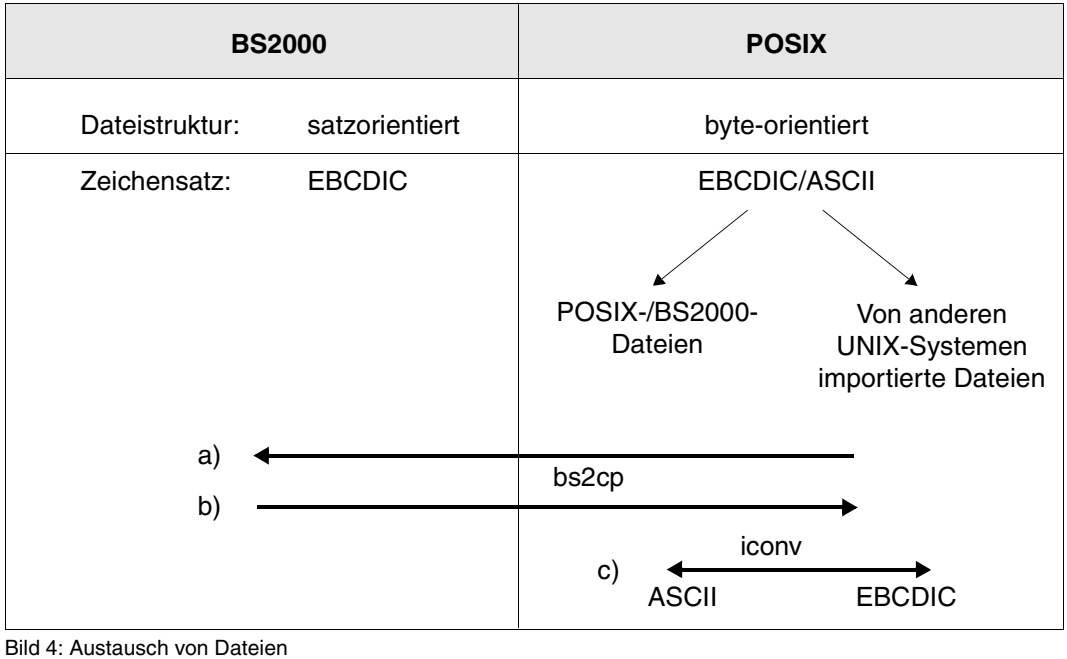


Bild 4: Austausch von Dateien

a) Dateien von POSIX nach BS2000 übertragen (aus Sicht der POSIX-Shell):

   Mit dem POSIX-Kommando *bs2cp* übertragen Sie Dateien von POSIX nach BS2000. Sie müssen nicht die Option *-k* angeben, wenn in beiden Dateisystemen die Dateien im EBCDIC-Zeichensatz vorliegen.

   Zusätzlich können Sie für die BS2000-Datei noch Dateiattribute bestimmen. Dazu müssen Sie vor dem „Kopier"kommando *bs2cp* mit dem POSIX-Kommando *bs2file* die BS2000-Dateiattribute festlegen. *bs2file* wird auf das BS2000-Kommando SET-FILE-ATTRIBUTES abgebildet.

b) Dateien von BS2000 nach POSIX übertragen (aus Sicht der POSIX-Shell):

   Mit dem POSIX-Kommando *bs2cp* übertragen Sie Dateien von BS2000 nach POSIX. Sie müssen nicht die Option *-k* angeben, wenn in beiden Dateisystemen die Dateien im EBCDIC-Zeichensatz vorliegen.

Abhängig von der Art der BS2000-Datei (SAM, ISAM oder PAM) ist folgendes zu berücksichtigen:

- PAM-Dateien werden immer binär abgelegt.
- ISAM-Dateien werden generell als Textdateien im POSIX-Dateisystem abgelegt.
- Bei SAM-Dateien und LMS-Elementen (Typ S, J, X ,D) können Sie wählen, ob die Datei/das Element als Textdatei, als Binärdatei oder als binäre Textdatei im POSIX-Dateisystem hinterlegt wird. Dazu müssen Sie vor dem Kopierkommando *bs2cp* noch die Bearbeitungsart der Datei mit dem POSIX-Kommando *ftyp* festlegen.

c) Zur Konvertierung innerhalb eines POSIX-Dateisystems dient das POSIX-Kommando *iconv*. Es werden die Datei-Inhalte konvertiert.

2.3 Kommandos von der POSIX-Shell aus eingeben

Nach erfolgreichem Zugang zum POSIX-Subsystem wird die POSIX-Shell gestartet.

Wenn Sie die POSIX-Shell interaktiv benutzen, gibt die POSIX-Shell den Wert der Umgebungsvariablen *PS1* als Bereitzeichen aus, bevor sie ein Kommando einliest. Im Standardfall ist dies das Dollarzeichen ($) bzw. (#) für den privilegierten Benutzer und ein anschließendes Leerzeichen (␣).

Die Kommandoeingabe hat folgendes Format:

```
kommando[␣optionen][␣parameter]␣ ...↵
```

Bei *kommando* müssen Sie den Namen eines POSIX-Kommandos oder einer Shell-Prozedur angeben, die ausgeführt werden soll. Mit *optionen* geben Sie Steueranweisungen zur Kommandoausführung. Bei *parameter* können Sie ein Aufrufargument eingeben, das die POSIX-Shell an *kommando* übergibt. Abhängig vom Kommando können Sie auch mehrere Aufrufargumente angeben.

Den Kommandonamen und die Aufrufargumente müssen Sie bei Zeichenterminals durch Tabulator- oder Leerzeichen voneinander trennen. Das letzte Aufrufargument und damit die Eingabe des Kommandos schließen Sie bei Zeichenterminals durch ↵ und bei Blockterminals durch EM DUE ab.

Das Starten von reinen BS2000-Programmen aus der POSIX-Shell wird nicht unterstützt.

Wenn die Bildschirmzeile für eine Eingabe zu kurz ist, haben Sie zwei Möglichkeiten:

- Sie schreiben am Zeilenende einfach weiter, ohne die Taste ↵ zu drücken. Nachdem Sie das Kommando vollständig eingegeben haben, schließen Sie es mit der Taste ↵ ab.
- Sie setzen die Zeile mit \ ↵ fort. Das Zeichen Gegenschrägstrich (\) entwertet die Kommandoabschlussfunktion der Taste ↵. Anschließend können Sie die Kommandoeingabe fortsetzen. Nach dem Drücken von ↵ (ohne \) wird das Kommando ausgeführt.

Jedes POSIX-Kommando gibt an die POSIX-Shell, in der es aufgerufen wurde, einen Wert zurück, nämlich seinen Endestatus. Dieser Wert ist bei einem fehlerfreien Ablauf 0, bei einem fehlerhaften Ablauf ungleich 0.

Wenn ein Kommando Informationen auf den Bildschirm ausgibt und die Ausgabe größer als eine Bildschirmseite ist, können Sie bei Zeichenterminals die Ausgabe durch Drücken der Tasten CTRL s anhalten und anschließend mit CTRL q fortsetzen. Bei Blockterminals wird diese Funktion nicht unterstützt.

2.3.1 Kommandos miteinander verknüpfen

Sie können mehrere Kommandos aneinanderhängen:

- Mit dem Zeichen | können Sie zwei Kommandos zu einer **Pipe** verbinden.
- Mit den Zeichen ; oder & oder && oder || können Sie mehrere Kommandos oder Pipes zu einer **Kommandofolge** verbinden.

Pipe

Eine Pipe ist eine Folge von zwei oder mehr Kommandos, die jeweils durch das Zeichen | verbunden sind. Eine Pipe zwischen zwei Kommandos lenkt die Standard-Ausgabe des ersten Kommandos auf die Standard-Eingabe des zweiten Kommandos um (siehe Abschnitt „Eingabe und Ausgabe eines Kommandos umlenken“ auf Seite 38). Deshalb müssen die so verbundenen Kommandos folgende Bedingungen erfüllen:

- Das erste Kommando einer Pipe muss auf die Standard-Ausgabe schreiben.
- Das nächste Kommando einer Pipe muss von der Standard-Eingabe lesen.
- Wenn die Pipe aus mehr als zwei Kommandos besteht, müssen alle Kommandos nach dem ersten und vor dem letzten Kommando der Pipe von der Standard-Eingabe lesen und auf die Standard-Ausgabe schreiben.

Die POSIX-Shell startet alle Kommandos der Pipe gleichzeitig als eigenständige parallele Prozesse. In einem Ein-Prozessor-System kann der Prozessor aber pro Zeiteinheit immer nur einen dieser Prozesse bearbeiten. Daher wechseln sich die einzelnen Prozesse am Prozessor ab.

Nur der Prozess, der zum ersten Kommando der Pipe gehört, erhält seine Eingabe *nicht* von einem anderen Prozess.
Nur der Prozess, der zum letzten Kommando der Pipe gehört, leitet seine Ausgabe *nicht* an einen anderen Prozess weiter.

Die POSIX-Shell wartet so lange, bis das letzte Kommando der Pipe beendet ist. Erst dann bearbeitet sie weitere Eingaben.

Kommandofolgen

Eine Kommandofolge ist die Aneinanderreihung einer oder mehrerer Pipes, die durch eines der Symbole

; & && ||

getrennt und optional durch

; & |&

abgeschlossen werden. Die Symbole ; & und |& haben gleichen Vorrang, der niedriger ist als der von && und ||, die beide ebenfalls gleichen Vorrang haben. Der Strichpunkt ; nach einer Pipe steht für sequentielle Ausführung der Pipe, während das kommerzielle Und & asynchrone Ausführung bedeutet, d.h. die Shell wartet nicht auf die Beendigung der Pipe. Das Symbol |& nach einer Kommandofolge veranlasst die asynchrone Ausführung der Pipe oder des Kommandos mit einer bidirektionalen Pipe zur Vater-Shell.

Im folgenden steht der Begriff Kommando für ein Kommando oder eine Pipe. Die Zeichen haben folgende Wirkung:

Zeichen	Wirkung
;	Die POSIX-Shell bearbeitet das nächste Kommando erst, wenn das vorausgehende beendet ist. Sie gibt aber kein Bereitzeichen aus.
&	Die POSIX-Shell startet das vorausgehende Kommando im Hintergrund und bearbeitet sofort das nachfolgende Kommando. Die POSIX-Shell lenkt die Standard-Eingabe aller Hintergrund-Kommandos auf eine Datei mit den Eigenschaften von */dev/null* um. Sie können die Standard-Eingabe eines Hintergrund-Kommandos aber auch auf eine andere Datei umlenken.
&&	Die POSIX-Shell führt das nachfolgende Kommando nur aus, wenn das vorausgehende Kommando als Endestatus den Wert 0 zurückgegeben hat.
||	Die POSIX-Shell führt das nachfolgende Kommando nur aus, wenn das vorausgehende Kommando als Endestatus einen Wert ungleich 0 zurückgegeben hat.
|&	Die POSIX-Shell startet das vorausgehende Kommando im Hintergrund mit einer bidirektionalen Pipe zur Vater-Shell.

2.3.2 Weitere Eingaben nach einem Kommandoaufruf

Einige Kommandos warten nach dem Abschluss mit ↵ auf weitere Eingaben von der Tastatur. Sie können dies daran erkennen, dass die Schreibmarke auf den Beginn der nächsten Zeile gesetzt wird, wobei entweder kein Bereitzeichen erscheint oder ein für das Kommando spezifisches Bereitzeichen ausgegeben wird. Die Zeilen, die Sie nun eingeben, müssen sie jeweils mit der Taste ↵ abschließen. Beenden können Sie die Eingabe in der Regel mit der Taste END (bzw. mit @@d bei Blockterminals) oder durch eine spezielle Anweisung des Kommandos.

Wenn Sie ein Kommando mit ↵ abschließen und diese Eingabe noch keine vollständige Kommandozeile ist oder ↵ durch einen Gegenschrägstrich (\) entwertet wurde, gibt die POSIX-Shell den Wert der Shell-Variablen PS2 als Bereitzeichen aus. Im Standardfall ist dies das Größer-Zeichen (>) und ein anschließendes Leerzeichen (␣).

2.3.3 Optionen angeben

Beim Aufruf von Kommandos können Sie keine, eine oder mehrere Optionen angeben. Optionen modifizieren die Funktionsweise eines Kommandos. Sie bestehen i.a. aus einem Buchstaben und sind so in der Syntax angegeben, wie Sie diese eingeben müssen.

Für die Angabe mehrerer Optionen gelten i.a. die folgenden Regeln:

- Optionen können Sie in beliebiger Reihenfolge angeben.
- Optionen ohne Argumente können Sie auf zwei Arten eingeben:
  - einzeln, z.B.: *cmd␣-a␣-b␣-c*
  - zusammengefasst, z.B.: *cmd␣-abc*
- Optionen, die ein Argument benötigen, z.B. *-dargument* oder *-f␣argument*, können Sie nicht mit Optionen ohne Argument zusammenfassen. Sie müssen diese einzeln, jeweils durch ein Leerzeichen (␣) getrennt, eingeben:

  cmd␣-abc␣-dargument␣-f␣argument
- Optionen bzw. zusammengefasste Optionen beginnen mit einem Bindestrich (-): *-abc*.
- Das Ende der Optionen kann mit zwei Bindestrichen (--) gekennzeichnet werden. Die Angabe -- ist nötig, wenn das erste Argument mit - beginnt.

Die meisten Kommandos geben eine Usage-Meldung aus, wenn sie mit einer falschen Option aufgerufen werden. Sie können der Usage-Meldung entnehmen, mit welchen Optionen und Operanden ein Kommando aufgerufen werden darf.

Beispiel Ausgabe einer Usage-Meldung bei Angabe einer falschen Option

```
$ ls␣-y
ls: Illegal option -- y

ls: Usage: ls -1RadCxmnlogrtucpFbqisfLe [files]
```

2.3.4 Dateinamen angeben

Für Dateinamen gelten folgende Konventionen:

- Erlaubt sind alle Zeichen mit Ausnahme von / und \0 (Nullbyte als Abschluss von Zeichenreihen).
- Am Anfang eines Dateinamens sollten Sie die Zeichen Plus (+), Minus (-) und Punkt (.) vermeiden. Der Punkt ist für spezielle Dateien reserviert, z.B. für die Datei *.profile*.

  Beispiel zur Benutzung eines Dateinamens, der mit Minus (-) beginnt:

  touch -- -file1 (vor Angabe des Dateinamens müssen die (leeren) Optionen beendet werden)
- Zur Verwendung der Zeichen *, ?, [...] siehe Abschnitt „Dateinamen-Erzeugung“ auf Seite 57.
- Verschiedene Sonderzeichen sollten Sie nicht verwenden (siehe Kapitel „Tabellen und Verzeichnisse“ auf Seite 935).
- Wenn der Dateiname Leerzeichen oder Tabulatorzeichen enthält, müssen Sie ihn in Anführungsstriche einschließen.
- POSIX unterscheidet im Gegensatz zum BS2000 zwischen Groß- und Kleinbuchstaben im Dateinamen.

Sie haben zwei Möglichkeiten, um die Namen von Dateien oder Dateiverzeichnissen in der POSIX-Shell anzugeben:

- Sie können einen relativen Pfadnamen angeben. Relative Pfadnamen gehen immer vom aktuellen Dateiverzeichnis aus.
- Sie können einen absoluten Pfadnamen angeben. Absolute Pfadnamen beginnen mit einem Schrägstrich (/) und geben den Pfadnamen in Bezug zum Root-Verzeichnis an.
- Ein Dateiname darf maximal 1024 Zeichen lang sein. Dabei sind die Namen der Dateiverzeichnisse, der eigentliche Dateiname sowie die begrenzenden Schrägstriche (Slashes) bereits mitgerechnet.

2.3.5 Eingebaute POSIX-Kommandos

Eingebaute POSIX-Kommandos sind Kommandos, die die POSIX-Shell selbst interpretiert und verarbeitet, für deren Bearbeitung sie also keinen neuen Prozess erzeugt. Sie unterscheiden sich von allen anderen „externen“ Kommandos in folgenden Punkten:

- Sie sind als Unterprogramme Bestandteil der Binärdatei */bin/sh*. Deshalb können Sie ihre Namen nicht ändern.
  Sie können die Zugriffsrechte nur für die Binärdatei */bin/sh* ändern, nicht aber für ein einzelnes eingebautes POSIX-Kommando.
- Die POSIX-Shell führt die eingebauten POSIX-Kommandos bevorzugt aus. Wenn es zu einem eingebauten POSIX-Kommando ein gleichnamiges externes Kommando gibt, führt die POSIX-Shell bei Angabe dieses Namens immer das eingebaute POSIX-Kommando aus. Das externe Kommando wird ausgeführt, wenn der Aufrufname einen Schrägstrich (/) enthält, also z.B. beim Aufruf mit dem entsprechenden absoluten Pfadnamen.
- Die eingebauten POSIX-Kommandos sind schneller, weil sie die POSIX-Shell selbst ausführt.

2.3.6 Eingabe und Ausgabe eines Kommandos umlenken

Vor der Ausführung eines Kommandos können Sie dessen Ein- und Ausgabe umlenken. Dazu benutzen Sie eine spezielle Notation, die von der POSIX-Shell interpretiert wird. Die folgenden Angaben können bei einem einfachen Kommando an beliebiger Stelle oder vor oder nach einem Kommando stehen. Diese Angaben werden nicht an das aufzurufende Kommando übergeben, sondern von der POSIX-Shell interpretiert. Parameter- und Kommando-Ersetzung werden durchgeführt, bevor *datei* oder *dateikennzahl* eingesetzt werden. Die Dateinamen-Erzeugung wird nur dann durchgeführt, wenn das Muster zu genau einem Dateinamen führt. Blankersetzung wird nicht durchgeführt.

<datei
: lenkt die Standard-Eingabe (Dateikennzahl 0) des Kommandos auf *datei* um; das Kommando liest seine Eingabe aus *datei*.

>datei
: lenkt die Standard-Ausgabe (Dateikennzahl 1) des Kommandos auf *datei* um; das Kommando schreibt seine Ausgabe in *datei*. Existiert die Datei noch nicht, wird sie neu angelegt. Existiert *datei* als einfache Datei bereits und ist die Option *noclobber* (siehe *Kommandos, set -o*) gesetzt, dann führt dies zu einem Fehler. Ist *noclobber* nicht gesetzt, dann wird der bisherige Inhalt der Datei gelöscht.

>|datei
: Entspricht **>**datei, mit dem Unterschied, dass die Einstellung der Option *noclobber* ignoriert wird.

>>datei
lenkt die Standard-Ausgabe (Dateikennzahl 1) des Kommandos auf *datei* um, das Kommando schreibt seine Ausgabe in *datei*. Wenn *datei* bereits existiert, wird die Ausgabe an den bisherigen Inhalt angehängt. Sonst wird die Datei neu angelegt.

<>datei
datei wird zum Lesen und Schreiben als Standard-Eingabe geöffnet.

<<[-]zeichenkette
leitet ein Here-Dokument ein. Ein Here-Dokument wird vor allem in Shell-Prozeduren eingesetzt, um Kommandos, die von der Standard-Eingabe lesen können, mit Werten zu versorgen.
An *zeichenkette* wird weder Parameter- und Kommando-Ersetzung noch Dateinamen-Erzeugung durchgeführt. Die Eingabe für die POSIX-Shell wird bis ausschließlich zu einer Zeile, die nur *zeichenkette* enthält, oder bis zum Dateiende gelesen.

Ist eines der Zeichen von *zeichenkette* entwertet, dann sind für die POSIX-Shell alle Zeichen des Here-Dokuments entwertet.

i *zeichenkette* in der schließenden Zeile darf nicht entwertet sein.

Ist *zeichenkette* nicht entwertet, dann

- wird Parameter- und Kommando-Ersetzung durchgeführt
- wird die Zeichenfolge *Neue-Zeile-Zeichen* ignoriert
- müssen durch Gegenschrägstrich (\) der Gegenschrägstrich (\), das Dollarzeichen (*$*), das Gegenhochkomma (`) und der erste Buchstabe von *zeichenkette* entwertet werden, wenn Bedarf dafür im Text besteht.

Durch die Angabe von <<- werden alle führenden Tabulatorzeichen von den Zeilen des Here-Dokuments und vor *zeichenkette* gelöscht.

<&dateikennzahl
>&dateikennzahl
Bei der ersten Form wird die Standard-Eingabe durch Duplizieren von *dateikennzahl* umgelenkt. Die Standard-Eingabe liest aus der Datei, die an die Dateikennzahl angeschlossen ist. Die zweite Form gilt analog für die Standard-Ausgabe.

<&-
>&-
Durch die erste Form wird für das Kommando die Standard-Eingabe geschlossen, das Kommando erhält nur Datei-Ende als Eingabe. Die zweite Form gilt für die Standard-Ausgabe, das Kommando gibt nichts aus.

<&p
>&p
Die Eingabe vom Koprozess bei einer bidirektionalen Pipe wird auf die Standard-Eingabe umgelenkt. Analog wird die Ausgabe auf die Standard-Ausgabe umgelenkt.

Wird einer der obigen Anweisungen eine Nummer vorangestellt, dann wird die Dateikennzahl mit dieser Nummer (anstelle der 0 bzw. *stdin* und 1 bzw. *stdout*) angesprochen.

Beispiel Das folgende Beispiel öffnet Dateikennzahl 2 (*stderr*) zum Schreiben als ein Duplikat von Dateikennzahl 1 (*stdout*):

```
... 2>&1
```

Reihenfolge der Umlenkungen und Hintergrundkommandos

Die Reihenfolge der Umlenkungen ist signifikant. Die POSIX-Shell bewertet jede Umlenkung bezogen auf die Verbindung (Dateikennzahl, Datei) zum Zeitpunkt der Bewertung. D.h.

```
... 1>datei 2>&1
```

verbindet zuerst die Standard-Ausgabe (Dateikennzahl 1) mit *datei* und verbindet dann die Standard-Fehlerausgabe (Dateikennzahl 2) mit der Datei, die mit Dateikennzahl 1 (also mit *datei*) verbunden ist. *datei* enthält nach der Ausführung die Standard-Ausgabe und die Fehlermeldungen des Kommandos.
Wäre die Reihenfolge umgekehrt, dann wäre Dateikennzahl 2 mit der Datensichtstation (falls Dateikennzahl 1 damit verbunden war) und Dateikennzahl 1 mit *datei* verbunden. Das heißt also: *datei* enthält dann nur noch die Standard-Ausgabe, aber nicht die Fehlermeldungen.

Wird ein Kommando mit *&* im Hintergrund gestartet, ohne aktive Auftragssteuerung, dann wird standardmäßig die Standard-Eingabe mit einer Datei verbunden, die die gleichen Eigenschaften wie */dev/null* besitzt. Bei aktiver Auftragssteuerung enthält die Umgebung für die Ausführung des Kommandos die Dateikennzahlen der ausführenden POSIX-Shell, wie sie von den Anweisungen für Ein- und Ausgabe geändert wurden.

Sie können aus einem Skript Kommandos oder Pipes im Hintergrund starten. Diese können dann mit Ihrem Programm kommunizieren. Um einen solchen Koprozess zu starten, stellen Sie dem Kommando den Operator |& nach. Verwenden Sie bidirektionale Pipes nur in Skripts, jedoch nicht auf der Kommandozeile.

Kommandofolgen lassen sich nur einmal als bidirektionale Pipes aufrufen. Wenn Sie den ursprünglichen Prozess abgebrochen haben (z.B. mit *kill -9 PID*) und zu einem späteren Zeitpunkt nochmals versuchen, ein Kommando in eine bidirektionale Pipe zu schreiben, wird zwar eine Subshell aufgerufen, aber der Prozess gestoppt. Sie erhalten die Fehlermeldung:

```
sh: bad file unit number.
```

2.3.7 Shell-Prozeduren und Prozesse

Shell-Prozeduren sind Kommandos, die zu einem Programm zusammengefasst sind. Um eine Shell-Prozedur zu erstellen, müssen Sie die gewünschte Kommandofolge in eine Datei schreiben. Um eine Shell-Prozedur auszuführen, rufen Sie sie mit *sh <dateiname>* auf. Besitzt die Datei das Ausführrecht (*chmod +x <dateiname>*), können Sie diese Kommandofolge direkt, also ohne *sh*, ausführen.

Dieser Abschnitt beschreibt die Auswirkungen der POSIX-Prozess-Struktur, wenn Sie mit der POSIX-Shell arbeiten, d.h. wenn Sie Kommandos eingeben oder mit Shell-Prozeduren arbeiten.

Alle Eingaben, die Sie nach /START-POSIX-SHELL machen, nimmt die POSIX-Shell in Empfang und führt entsprechende Aktionen aus.

Die wichtigsten Kommands sind als eingebaute Kommandos („builtin“) realisiert. Für einige Kommandos wird ein eigener Prozess erzeugt. Der Prozess, in dem die POSIX-Shell abläuft, wird damit selbst zu einem Vaterprozess.

Grundsätzlich gilt: Jeder Prozess kann einen neuen Prozess erzeugen. Solange ein Prozess keinen anderen Prozess erzeugt, ist er der Sohnprozess eines Vaterprozesses. Sobald ein Sohnprozess einen anderen Prozess erzeugt, wird er selbst vom Sohn- zum Vaterprozess.
Er behält aber dennoch dem übergeordneten Prozess gegenüber seinen Status als Sohnprozess, d.h. ein Prozess kann sowohl Vater- als auch Sohnprozess sein.

Die Vater-Sohn-Prozesshierarchie ist von großer Bedeutung, wenn Sie in Shell-Prozeduren Umgebungsvariablen weiterreichen möchten. Jeder Benutzer kann für seine POSIX-Shell (=Vaterprozess) Umgebungsvariablen definieren, mit denen die POSIX-Shell arbeiten soll. Diese Umgebungsvariablen sind aber erst einmal nur der POSIX-Shell bekannt. Wenn in einem Sohnprozess mit Umgebungsvariablen aus dem Vaterprozess (=POSIX-Shell) gearbeitet werden soll, müssen Sie diese an den Sohnprozess exportieren. Dazu steht das Kommando *export* zur Verfügung.

2.3.8 Kommandos durch die POSIX-Shell verarbeiten

Mit Ausnahme der eingebauten POSIX-Kommandos ist jedem Kommando eine Datei im POSIX-Dateisystem zugeordnet. Die Namen der meisten Kommandodateien sind im Dateiverzeichnis */usr/bin* eingetragen. Die Shell-Variable *PATH* enthält eine Liste von Dateiverzeichnissen, in denen die Kommandos gesucht werden. Bei der Ausführung eines Kommandos liest die POSIX-Shell das Kommando, sucht in den in *PATH* enthaltenen Dateiverzeichnissen nach dem Dateinamen, der mit dem Kommandonamen identisch ist, und führt das Kommando aus.

Für die Ausführung der nicht eingebauten Kommandos erzeugt die POSIX-Shell einen neuen Prozess und wartet dann, bis dieser fertig ist. Anschließend gibt die POSIX-Shell den Wert der Shell-Variablen *PS1* als Bereitzeichen aus (standardmäßig $ für nichtprivilegierte Benutzer oder # für priviligierte Benutzer); sie kann jetzt ein neues Kommando verarbeiten. Wenn Sie z.B. das Kommando *date* eingeben, bekommen Sie umgehend das aktuelle Datum ausgegeben. Anschließend gibt die POSIX-Shell das Bereitzeichen aus und ist damit bereit, ein neues Kommando zu verarbeiten.

Bei einem Kommando wie *date* ist die Zeitspanne sehr klein, die zwischen der Eingabe des Kommandos und der erneuten Empfangsbereitschaft der POSIX-Shell vergeht. Diese Zeitspanne kann aber bei Eingabe eines anderen Kommandos wesentlich größer sein, z.B. beim Übersetzen einer Datei. Denn die POSIX-Shell meldet sich erst zurück, nachdem sie ein Kommando ausgeführt hat. Um längere Wartezeiten an der Datensichtstation zu vermeiden, gibt es das Sonderzeichen &. Wenn Sie eine Kommandoeingabe mit diesem Zeichen beenden, meldet sich die POSIX-Shell sofort mit dem Bereitzeichen am Bildschirm zurück, ohne auf die Ausführung des eingegebenen Kommandos zu warten. Kommandos, die mit & abgeschlossen werden, werden als Hintergrundprozess gestartet, um den Sie sich nicht weiter kümmern müssen.

2.3.9 Zusammengesetzte Anweisungen

Syntax **for**␣bezeichner␣[␣**in**␣wort...];**do**␣kommandofolge;**done**

oder

Syntax **for**␣bezeichner␣[␣**in**␣wortliste]
do␣kommandofolge
done

Mit der *for*-Anweisung können Sie *kommandofolge* mehrmals wiederholen. Dabei wird *bezeichner* auf jedes Wort der *wortliste* gesetzt und die Schleife einmal durchlaufen. Ist *in wortliste* nicht angegeben, dann wird *kommandofolge* für jeden gesetzten Stellungsparameter (*"$@"*) einmal durchlaufen. Die Ausführung endet, wenn *wortliste* abgearbeitet ist.

Syntax **select**␣bezeichner␣[␣**in**␣wortliste];**do**␣kommandofolge;**done**

oder

Syntax **select**␣bezeichner␣[␣**in**␣wortliste]
do␣kommandofolge
done

Mit der *select*-Anweisung können Sie *kommandofolge* mehrmals durch Eingabe gesteuert wiederholen. *select* schreibt *wortliste* auf die Standard-Ausgabe. Dabei wird jedem Wort eine Nummer vorangestellt. Danach wird das Bereitzeichen *PS3* ausgegeben und eine Zeile von der Standard-Eingabe eingelesen. Der Inhalt der gelesenen Zeile wird der Variablen *REPLY* als Wert zugewiesen. Enthält diese Zeile die Nummer eines der angezeigten Wörter, dann wird an *bezeichner* das entsprechende Wort als Wert zugewiesen. Bei leerer Zeile wird die Ausgabe von *wortliste* wiederholt und *bezeichner* der Name der Shell-Prozedur zugewiesen. Wurde *in* wortliste weggelassen, dann werden dafür, wie bei der *for*-Schleife, die Stellungsparameter eingesetzt. Die *select*-Schleife wird solange durchlaufen, bis sie durch das eingebaute Kommando *break* oder durch Eingabe von Dateiende abgebrochen wird.

Beispiel Mit dieser POSIX-Shell-Prozedur können Sie Informationen über jede Datei des aktuellen Verzeichnisses einzeln abfragen:

```
select datei in `ls`
  do
    if [ -z "$datei" ]
      then        echo "Zahl auswaehlen"
      else        ls -lsid $datei
    fi
  done
```

Syntax

case wort **in**
[[(]muster[|muster]...**)**kommandofolge**;;**]...
esac

Die *case*-Anweisung gestattet Ihnen, mustergesteuert auf eine Kommadofolge zu verzweigen. Die erste *kommandoliste*, deren *muster* auf *wort* passt, wird ausgeführt. Die Muster werden in derselben Art angegeben, wie sie für die Erzeugung von Dateinamen verwendet werden (siehe Abschnitt „Dateinamen-Erzeugung“ auf Seite 57).

Die optionale öffnende Klammer vor *muster* müssen Sie angeben, wenn Sie *case*-Anweisungen innerhalb der Kommandosubstitution mit *$(...)* verwenden. Damit erreichen Sie paarweise Klammerung innerhalb von *$(...)*.

Syntax

if kommandofolge1
then Kommandofolge2
[**elif** kommandofolge3
then kommandofolge4]...
[**else** kommandofolge5]
fi

Mit der *if*-Anweisung können Sie entsprechend einer Bedingung verschiedene Kommandofolgen ausführen lassen. *kommandofolge1* folgend auf *if* wird als Bedingung ausgeführt. Ist diese Bedingung wahr (Endestatus = 0), dann wird *kommandofolge2* (*then* folgend) ausgeführt. Ist diese Bedingung falsch (Endestatus ≠ 0), wird die Bedingung *kommandofolge3* nach *elif* ausgeführt. Ist der Endestatus dieser Bedingung 0, wird *kommandofolge4* hinter dem nächsten *then* ausgeführt. Im falsch-Fall wird *kommandofolge5* nach *else* ausgeführt. Wurde keine der Kommandofolgen nach *then* oder *else* ausgeführt, dann bekommt die *if*-Anweisung den Endestatus 0 zugewiesen.

Syntax

while kommandofolge1
do kommandofolge2
done
until kommandofolge1
do kommandofolge2
done

Durch die *while*- und *until*-Anweisungen erhalten Sie Schleifen mit Abbruchbedingung. Die *while*-Anweisung führt die Bedingung *kommandofolge1* aus. Ist die Bedingung wahr (Endestatus des letzten ausgeführten einfachen Kommandos = 0), wird der Schleifenkörper *kommandofolge2* ausgeführt und die Bedingung erneut geprüft. Ist die Bedingung falsch (Endestatus ≠ 0), wird die Schleife beendet. Wurde kein Kommando aus *kommandofolge2* (nach *do*) ausgeführt, dann wird der *while*-Anweisung der Endestatus 0 zugewiesen.

Die *until*-Anweisung können Sie anstelle der *while*-Anweisung bei negierter Bedingung verwenden. *until* prüft nach, ob die Bedingung falsch ist und bricht, sobald das Ergebnis wahr wird, die Schleife ab.

Syntax **continue**␣zahl

continue fährt mit der nächsten Iteration der umschließenden *for*-, *while*-, *until*- oder *select*-Schleife fort. Ist *zahl* angegeben, wird in der *zahl*ten umschließenden Schleife mit der nächsten Iteration fortgefahren. Ist *zahl* gleich 0, wird hinter der äußersten Schleife fortgefahren. *continue 1* entspricht *continue*.

Syntax **break**␣zahl

break bricht die umschließende *for*-, *while*-, *until*- oder *select*-Schleife ab. Ist *zahl* angegeben, wird in der *zahl*ten umschließenden Schleife mit der Ausführung fortgefahren. Ist *zahl* gleich 0, wird die äußerste Schleife verlassen. *break 1* entspricht *break*.

Syntax **return**␣zahl

Wird *return* in einer Funktion aufgerufen, kehrt dieses Kommando zur aufrufenden Prozedur zurück. Der *return*-Status wird durch *zahl* oder das letzte ausgeführte Kommando bestimmt. Wird *return* außerhalb von Funktionen oder in einer mit Punkt ausgeführten Prozedur aufgerufen, dann entspricht dies dem Aufruf von *exit*.

Syntax **(**kommandofolge**)**

kommandofolge wird in einer eigenen Umgebung ausgeführt.

Werden zur Schachtelung zwei aufeinanderfolgende öffnende runde Klammern (benötigt, dann müssen Sie beide durch ein Leerzeichen trennen. Es könnte sonst zu einer Verwechslung mit einem arithmetischen Ausdruck kommen (siehe *Kommando-Ersetzung*).

Syntax **{**␣kommandofolge**;}**

kommandofolge wird in der aktuellen POSIX-Shell einfach ausgeführt.

Auf die öffnende Klammer { muss ein Leerzeichen folgen!
Im Unterschied zu den runden Klammern (und) sind die geschweiften Klammern { und } reservierte Wörter. Damit sie als solche erkannt werden, müssen { und } am Zeilenanfang oder nach einem Strichpunkt ; stehen.

Syntax **[[**␣bedingung␣**]]**

Der bedingte Ausdruck **bedingung** wird bewertet, und der Endestatus bei wahr wird auf 0 und bei falsch auf 1 gesetzt.

Siehe *Bedingter Ausdruck* für eine Beschreibung von *bedingung*.

Syntax **function**␣bezeichner␣**{**␣kommandofolge**;}**
bezeichner␣**()**␣**{**␣kommandofolge**;}**

Definition der Funktion mit dem Namen *bezeichner*. Dieser Name wird wie bei einem Kommando zum Aufruf verwendet. Der Funktionskörper besteht aus der *kommandofolge*, die in geschweifte Klammern eingeschlossen ist.

Auf die öffnende Klammer { muss ein Leerzeichen folgen!

Die geschweiften Klammern {} können weggelassen werden, wenn

– es sich um eine einzelne *for*-, *case*-, *if*-, *while*-, *until*- oder *select*-Anweisung handelt

  Beispiel

```
function␣bezeichner␣case␣wort␣in␣...
bezeichner␣()␣for␣bezeichner...
```

– die Kommandofolge aus einer Subshell gestartet wird, d.h. () statt {}.

  Beispiel

```
function␣bezeichner␣(kommandofolge)
bezeichner␣()␣(kommandofolge)
```

Syntax **time**␣pipe

Die Kommandos von *pipe* werden ausgeführt und als Statistik die verstrichene Zeit, die *user*- und die *system*-Zeit auf der Standard-Fehlerausgabe berichtet. Die folgenden reservierten Wörter werden nur als erstes Wort eines Kommandos erkannt und wenn sie nicht entwertet sind.

```
if      then    elif    else    fi      case    esac    for
select  while   until   do      done    {       }       function
time    [[      ]]
```

2.3.10 Kommentare

Beginnt ein Wort mit dem Nummernzeichen #, dann werden die restlichen Zeichen der Zeile einschließlich # ignoriert.

2.3.11 Alias-Variablen

Wenn Sie für das erste Wort eines Kommandos eine Alias-Variable definiert haben, dann wird dieses durch den Text der Alias-Variablen ersetzt. Der Name einer Alias-Variablen kann aus einer beliebigen Anzahl von Zeichen bestehen, zu denen nicht die Metazeichen, die Entwertungszeichen, die Zeichen für die Dateinamen-, die Parameter- und Kommando-Ersetzung und das Gleichheitszeichen = gehören. Die Ersetzungs-Zeichenkette kann jede korrekte POSIX-Shell-Prozedur, inklusive der vorhergehenden Metazeichen-Aufzählung, enthalten. Das erste Wort eines jeden Kommandos im ersetzten Text wird wiederum auf weitere Alias-Variablen getestet. Wenn das letzte Zeichen des Alias-Werts ein Leerzeichen ist, dann wird das auf die Alias-Ersetzung folgende Wort ebenfalls auf Alias-Ersetzung untersucht.

Mit Alias-Variablen können Sie eingebaute Shell-Kommandos neu definieren, Sie können sie aber nicht zur Neudefinition der reservierten Wörter (siehe oben) verwenden. Mit Hilfe des *alias*-Kommandos können Alias-Variablen definiert, exportiert und zum Auflisten auf die Standard-Ausgabe geschrieben werden, während sie mit dem *unalias*-Kommando wieder gelöscht werden können. Exportierte Alias-Variablen bleiben wirksam für Prozeduren, die per Name aufgerufen werden, müssen aber neu für explizite weitere Aufrufe der POSIX-Shell initialisiert werden.

Aliasing wird beim Lesen von Prozeduren durchgeführt, und nicht wenn diese ausgeführt werden. Dementsprechend muss eine Alias-Definition vor deren erster Verwendung beim Lesen eines Kommandos mit der Alias-Variablen erfolgen.

Alias-Variablen werden häufig als Abkürzung für volle Pfadnamen verwendet. Eine Option des eingebauten *alias*-Kommandos sorgt dafür, dass der Wert der Alias-Variablen automatisch auf den vollen Pfadnamen des zugehörenden Kommandos gesetzt wird. Diese Alias-Variablen nennt man „mit Pfad versehene Alias-Variablen“ (tracked alias). Der Wert einer solchen Variablen wird beim ersten Aufsuchen des zugehörenden Kommandos definiert und jedesmal zurückgesetzt, wenn die *PATH*-Variable neu gesetzt wird. Diese Alias-Variablen behalten den Zustand „mit Pfad versehen“, so dass der nächste folgende Zugriff den Wert neu definieren wird. Einige mit Pfad versehene Alias-Variablen sind mit in die POSIX-Shell übersetzt und compiliert. Die Option *-h* des eingebauten Kommandos *set* macht jeden aufgerufenen Kommandonamen zu einer mit Pfad versehenen Alias-Variablen.

Die folgenden exportierten Alias-Variablen sind in die POSIX-Shell compiliert und können neu definiert oder mit *unalias* gelöscht werden:

```
autoload='typeset -fu'
false='let 0'
functions='typeset -f'
hash='alias -t'
history='fc -l'
integer='typeset -i'
nohup='nohup'
r='fc -e -'
true=':'
type='whence -v'
```

Beispiel Durch die folgenden Alias-Variablen können Sie *l*, *ll* und *lf* nachbilden:

```
alias l='/bin/ls -m'
alias ll='/bin/ls -l'
alias lf='/bin/ls -CF'
```

2.3.12 Tilde-Ersetzung

Nach der Alias-Ersetzung wird jedes Wort getestet, ob es mit einer nicht entwerteten Tilde ~ beginnt. Wenn dies der Fall ist, dann wird das Wort bis zum nächsten Schrägstrich / auf die Übereinstimmung mit einem Login-Namen untersucht. Ist dies gegeben, dann wird die Tilde und der Login-Name durch das Login-Verzeichnis des gefundenen Benutzers ersetzt. Dies wird Tilde-Ersetzung genannt. Wird keine Übereinstimmung gefunden, dann bleibt der Originaltext unverändert. Eine alleinstehende Tilde oder eine Tilde vor einem Schrägstrich wird durch *$HOME* ersetzt. Eine Tilde gefolgt von einem Plus- oder Minuszeichen wird durch *$PWD* (+) oder *$OLDPWD* (-) ersetzt.

Tilde-Ersetzung wird auch versucht, wenn der Wert einer Variablenwertzuweisung mit einer Tilde beginnt.

Beispiel Mit dem Kommando

```
cat ~/.profile
```

können Sie von beliebiger Stelle die Datei *.profile* in Ihrem Login-Verzeichnis auflisten.

2.3.13 Kommando-Ersetzung

Durch die Standard-Ausgabe eines Kommandos können Sie auf zwei Arten einen Teil eines Wortes oder ein ganzes Wort ersetzen. Bei der ersten (neuen) Art wird das Kommando in Dollarzeichen, runde Klammer auf, runde Klammer zu *$(...)* eingeschlossen. Bei der zweiten (archaischen) Art wird das Kommando in Gegenhochkommata `...` eingeschlossen. Hier wird die Zeichenkette zwischen den Hochkommata auf Entwertungen oder Anführungszeichen untersucht, bevor das Kommando ausgeführt wird (siehe Abschnitt „Entwerten von Metazeichen (quoting)“ auf Seite 58). Bei beiden Arten werden Neue-Zeile-Zeichen der Kommandoausgabe gelöscht.

Die Ersetzung *$(cat datei)* kann durch die äquivalente, aber schnellere Version *$(< datei)* durchgeführt werden. Die Ersetzung wird von eingebauten Shell-Kommandos, die keine Umlenkung der Ein- oder Ausgabe durchführen, ohne die Erzeugung eines neuen Prozesses durchgeführt.

Beispiel Wenn Sie in einem Verzeichnis alle Dateien mit der Endung .c, in denen die Zeichenkette *include* enthalten ist, editieren wollen, dann hilft Ihnen folgendes:

```
for name in $( grep -l include *.c )
  do
      edt $name
  done
```

*grep -l include *.c* durchsucht alle Dateien **.c* nach der Zeichenkette *include*.
Mit der Option *-l* veranlassen Sie, dass alle Dateien ausgegeben werden, in denen der Suchstring gefunden wurde.

Ein arithmetischer Ausdruck, in Dollarzeichen, doppelte runde Klammer auf und doppelte runde Klammer zu *$((...))* eingeschlossen, wird durch den Wert des arithmetischen Ausdrucks ersetzt.

Beispiel Sie wollen den vorletzten Parameter einer Shell-Prozedur ausgeben:

```
eval print \$$(( $#-1 ))
```

2.3.14 POSIX-Shell-Variablen und Parameter-Ersetzung

Ein Parameter wird durch einen Bezeichner, eine oder mehrere Ziffern oder eines der folgenden Zeichen dargestellt:

* @ # ? - $!

Eine Variable (ein Parameter gekennzeichnet durch einen Bezeichner) hat einen Wert und keine oder mehrere Attribute. Variablen können Sie Werte und Attribute durch das eingebaute Shell-Kommando *typeset* zuweisen. Die verwendbaren Attribute werden bei *typeset* beschrieben. Exportierte Parameter geben ihren Wert und die Attribute an die Umgebung weiter.

Die POSIX-Shell stellt eindimensionale Felder zur Verfügung. Ein Element des Feldes wird über einen Index angesprochen. Ein Index wird durch eckige Klammer auf [, gefolgt von einem arithmetischen Ausdruck und eckige Klammer zu], beschrieben (zu den arithmetischen Ausdrücken siehe Abschnitt „Arithmetische Berechnungen“ auf Seite 59).

Einem Feld können Sie mit dem eingebauten Shell-Kommando *set* Werte zuweisen:
set -A bezeichner werte...
Die Werte aller Indizes müssen im zulässigen Wertebereich liegen. Felder müssen Sie nicht deklarieren, jeder Zugriff auf ein Feldelement mit zulässigem Index ist erlaubt. Wenn erforderlich, wird das Feld angelegt. Greifen Sie auf das Feld ohne Index zu, dann greifen Sie auf das nullte Element zu.

bezeichner**=**wert[␣bezeichner**=**wert]...
: Einer Variablen können Sie in dieser Form einen Wert zuweisen. Ist für eine Variable das Integer-Attribut gesetzt, dann kann bei arithmetischen Berechnungen ihr Wert eingesetzt werden. An Stellungsparameter (gekennzeichnet durch eine Zahl) können Sie Werte mit Hilfe des eingebauten Shell-Kommandos *set* zuweisen. Der Parameter *$0* wird beim Aufruf der Shell auf das nullte Argument gesetzt.

${parameter**}**
: Das Dollar-Zeichen dient zur Einführung von ersetzbaren Parametern. Die POSIX-Shell liest alle Zeichen von *${* bis *}* als Teil desselben Wortes, auch wenn darin Klammern oder Metazeichen enthalten sind. Falls existent, wird der Wert des Parameters eingesetzt. Die geschweiften Klammern benötigen Sie auch, wenn hinter *parameter* Buchstaben, Ziffern oder Unterstrich folgen, die nicht Teil des Namens des Parameters sind, oder wenn eine Variable indiziert wird. Besteht *parameter* aus einer oder mehreren Ziffern, dann ist es ein Stellungsparameter. Einen Stellungsparameter mit mehreren Ziffern müssen Sie in geschweifte Klammern einschließen.

  Wird *parameter* durch die Zeichen Stern * oder Klammeraffe @ beschrieben, dann werden alle Stellungsparameter ab *$1* eingesetzt. Als Trenner wird dabei das erste Zeichen der Variablen *IFS* verwendet (siehe unten). Ein Feldbezeichner mit Index Stern * oder Klammeraffe @ wird durch die Werte aller Elemente ersetzt. Auch hier wird der gleiche Trenner verwendet.

${#parameter**}**
Wird *parameter* durch die Zeichen Stern * oder Klammeraffe @ gegeben, dann wird die Anzahl der Stellungsparameter eingesetzt. Sonst wird die Länge des Wertes von *parameter* eingesetzt.

${#bezeichner[*****]**}**
Die Anzahl der Elemente des Feldes *bezeichner* wird eingesetzt.

${parameter**:-**wort**}**
Wenn *parameter* gesetzt und nicht die leere Zeichenkette ist, dann wird der Wert, sonst *wort* eingesetzt.

${parameter**:=**wort**}**
Ist *parameter* nicht gesetzt oder der Wert gleich der leeren Zeichenkette, dann wird *parameter* auf *wort* gesetzt. Anschließend wird der Wert eingesetzt.
Stellungsparametern können Sie mit dieser Methode keine Werte zuweisen.

${parameter**:?**wort**}**
Ist *parameter* gesetzt und der Wert nicht gleich der leeren Zeichenkette, dann wird der Wert eingesetzt; sonst wird *wort* ausgegeben und die Shell beendet. Fehlt *wort*, dann wird eine Standard-Fehlermeldung ausgegeben.

${parameter**:+**wort**}**
Wenn *parameter* gesetzt und nicht die leere Zeichenkette ist, dann wird *wort* eingesetzt, sonst die leere Zeichenkette.

${**parameter**#**muster**}** oder **${parameter**##**muster**}**
Sollte das POSIX-Shell-Muster *muster* auf den Anfang des Wertes von *parameter* passen, dann besteht der Ersetzungstext aus dem Wert von *parameter*, aus dem der auf *muster* passende Teil gelöscht wurde. Sonst wird der Wert von *parameter* eingesetzt. Bei der ersten Form der Angabe wird das kürzeste passende Muster, in der zweiten Form das längste Auftreten des Musters gelöscht.

${**parameter**%**muster**}** oder **${parameter**%%**muster**}**
Passt das POSIX-Shell-Muster *muster* auf das Ende des Wertes von *parameter*, dann wird der Ersetzungstext aus dem Wert von *parameter* ohne den gelöschten auf *muster* passenden Teil gebildet. Bei der ersten Form der Angabe wird das kürzeste, bei der zweiten Form das längste passende Muster gelöscht.

Bei den letzten 8 Ersetzungen wird *wort* erst dann bewertet, wenn es als Ersetzungs-Zeichenkette verwendet wird.

Beispiel

pwd wird erst dann ausgeführt, wenn *dvz* nicht gesetzt oder gleich der leeren Zeichenkette ist.

```
echo ${dvz:-$(pwd)}
```

Wird der Doppelpunkt : bei den obigen Ausdrücken weggelassen, dann kontrolliert die POSIX-Shell nur, ob *parameter* gesetzt ist oder nicht. Die Überprüfung auf die leere Zeichenkette entfällt.

Die folgenden Parameter werden automatisch von der POSIX-Shell gesetzt:

Die Anzahl (dezimal) der positionellen Parameter.

- Alle Optionen, die beim Aufruf der POSIX-Shell oder durch das eingebaute Kommando *set* gesetzt wurden.

? Der Endestatus des zuletzt ausgeführten Kommandos.

$ Die Prozess-Nummer der aktuellen POSIX-Shell.

_ Am Anfang wird der Wert von Unterstrich _ auf den absoluten Pfadnamen der POSIX-Shell oder der ausgeführten POSIX-Shell-Prozedur, wie er an die Umgebung übergeben wird, gesetzt. Später wird _ immer das letzte Argument des vorhergehenden Kommandos als Wert zugewiesen. Dieser Parameter wird nicht für asynchrone Hintergrund-Kommandos gesetzt. Bei der Suche nach Post wird der Parameter _ für die Speicherung des Namens der passenden *MAIL*-Datei verwendet.

Beispiel

Das Kommando *tail* greift über *$_* auf die letzte Datei des vorhergehenden *cat*-Kommandos zu.

```
cat /usr/tmp/mydir/xyz* > alles
tail $_
```

! Die Prozess-Nummer des zuletzt als Hintergrundprozess gestarteten Kommandos.

ERRNO
Der Wert dieses Parameters wird vom letzten fehlerhaften Systemaufruf gesetzt. Der Wert ist systemabhängig und dient der Fehlersuche.

LINENO
Die Zeilennummer der aktuellen Zeile in der Prozedur oder in der Funktion, die gerade ausgeführt wird.

OLDPWD
Das letzte aktuelle Dateiverzeichnis, gesetzt durch das *cd*-Kommando.

OPTARG
Der Wert des letzten Options-Arguments, das von dem eingebauten Kommando *getopt* bearbeitet wurde.

OPTIND
Der Index des letzten Options-Arguments, das von dem eingebauten Kommando *getopt* bearbeitet wurde.

PPID
Die Prozess-Nummer des Vaterprozesses der aktuellen POSIX-Shell.

PWD
Das aktuelle Dateiverzeichnis, gesetzt durch das *cd*-Kommando.

RANDOM
Jedesmal, wenn auf diese Variable zugegriffen wird, wird eine Zufallszahl aus dem Wertebereich von 0 bis 32767 berechnet. Die Folge der Zufallszahlen kann durch eine Wertzuweisung an die Variable *RANDOM* initialisiert werden.

REPLY
Diese Variable wird bei fehlenden Argumenten von der *select*-Anweisung oder vom eingebauten Kommando *read* gesetzt.

SECONDS
Beim Zugriff auf diese Variable enthält ihr Wert die Zeit in Sekunden, die seit dem Aufruf der POSIX-Shell vergangen ist. Wird der Variablen ein Wert zugewiesen, dann wird ihr Wert beim Zugriff auf den Zuweisungswert plus die vergangene Zeit seit der Zuweisung gesetzt.

Die folgenden Variablen werden von der POSIX-Shell benutzt:

CDPATH
Der Suchpfad für das *cd*-Kommando.

COLUMNS
Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Definition der Breite des Editierfensters beim Editiermodus der POSIX-Shell und für die Ausgabe der *select*-Liste verwendet. Diese Variable ist sinnvoll, wenn der Zugang zur POSIX-Shell über *rlogin* erfolgt.

EDITOR
Endet der Wert dieser Variablen mit *vi* und ist die Variable *VISUAL* nicht gesetzt, dann wird die entsprechende Option (siehe *set*) gesetzt. Diese Variable ist sinnvoll, wenn der Zugang zur POSIX-Shell über *rlogin* erfolgt.

ENV
Wenn diese Variable gesetzt ist, dann enthält ihr Wert den Pfadnamen der Prozedur, die bei Aufruf der POSIX-Shell ausgeführt wird. Diese Prozedur wird meist für Funktions- und Alias-Definitionen benutzt. Auf den Wert der Variablen wird Parameter-Ersetzung zur Dateinamen-Erzeugung durchgeführt.

FCEDIT
Der Name des Standard-Editors für das eingebaute Kommando *fc*. Momentan ist für diese Variable nur der Wert *edt* zulässig.

FPATH
: Der Suchpfad für Funktionsdefinitionen. Dieser Pfad wird verwendet, wenn auf eine Funktion mit Attribut *-u* zugegriffen wird und wenn kein Kommando gefunden wurde. Wird eine ausführbare Datei gefunden, dann wird sie gelesen und in der momentanen Umgebung ausgeführt.

HISTFILE
: Ist diese Variable bei Aufruf der POSIX-Shell gesetzt, dann wird der Wert als Pfadname für die Datei zur Speicherung der Kommando-history verwendet (siehe Abschnitt „Kommando-Wiederaufruf“ auf Seite 69).

HISTSIZE
: Wenn diese Variable bei Aufruf der POSIX-Shell gesetzt ist, dann behält die Shell den Text eingegebener Kommandos in Erinnerung. Sie können mindestens auf die, als Wert angegebene, Anzahl von früher eingegebenen, und der POSIX-Shell zugänglichen, Kommandos zurückgreifen. Der Standard-Wert ist 128.

HOME
: Das Standard-Argument (Login-Dateiverzeichnis) für das Kommando *cd*.

IFS
: (Internal field separators) Interner Feldtrenner der POSIX-Shell, der zur Trennung von Wörtern dient, die aus Kommando- oder Parameter-Ersatz entstehen. Der Feldtrenner wird auch durch das eingebaute Kommando *read* verwendet. Normalerweise ist der Wert auf Leer-, Tabulator- und Neue-Zeile-Zeichen gesetzt. Das erste Zeichen der *IFS*-Variable wird zur Trennung der Argumente bei der Ersetzung von *"$*"* verwendet (siehe Abschnitt „Entwerten von Metazeichen (quoting)“ auf Seite 58).

IO_CONVERSION
: Ist diese Variable auf „YES“ gesetzt, und wird mit POSIX-Kommandos (z.B. awk, cat, grep...) auf Dateien zugegriffen, die in einem (gemounteten) ASCII-Dateisystem liegen, dann wird automatisch konvertiert.

LINES
: Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Berechnung der Spaltenzahl für die Ausgabe von *select*-Listen verwendet. *select*-Listen werden vertikal ausgegeben, bis ungefähr zwei drittel der Zeilenzahl *LINES* gefüllt sind.

MAIL
: Enthält der Wert dieser Variablen den Namen einer *mail*-Datei und ist die Variable *MAILPATH* nicht gesetzt, dann informiert Sie die POSIX-Shell über das Eintreffen von Post in dieser Datei.

MAILCHECK
: Der Wert dieser Variablen gibt an, nach welchem Zeitintervall in Sekunden die POSIX-Shell jeweils nach Änderungen der Modifikationszeit der, durch die Variablen *MAIL* oder *MAILPATH*, ausgewiesenen Dateien sehen soll. Der Standard-Wert für *MAILCHECK* ist 600. Wenn die angegebene Zeit verstrichen ist, dann prüft die POSIX-Shell vor der Ausgabe des nächsten Bereitzeichens nach.

MAILPATH
: Eine Liste von Dateinamen, die durch Doppelpunkt *:* voneinander getrennt ist. Bei gesetzter Variable informiert Sie die POSIX-Shell über jede Änderung an den Dateien der Liste, die innerhalb der letzten *MAILCHECK* Sekunden erfolgt sind. Jeder Dateiname kann in der Liste von einem Fragezeichen *?* und einem Mitteilungstext, der ausgegeben werden soll, gefolgt werden. Diese Mitteilung wird der Parametererersetzung mit der Variablen *$_* unterzogen. *$_* enthält zu diesem Zeitpunkt den Namen der Datei, die sich geändert hat. Die Standard-Mitteilung ist: „*you have mail in $_*“.

PATH
: Der Suchpfad für Kommandos (siehe *Ausführung*). Sie können den Wert dieser Variablen nicht verändern, wenn Sie eine eingeschränkte POSIX-Shell benutzen.

PS1
: Der Wert dieser Variablen wird für den Parameter-Ersatz expandiert, um das Bereitzeichen der POSIX-Shell zu definieren. Der Standard-Wert ist „*$*␣“ bzw. für privilegierte Benutzer „#␣“. Das Ausrufezeichen *!* im Bereitzeichen wird durch die Kommando-Nummer ersetzt (siehe Abschnitt „Kommando-Wiederaufruf“ auf Seite 69).

PS2
: Das zweite Bereitzeichen, das die POSIX-Shell ausgibt, wenn sie noch weitere Eingaben nach einem Neue-Zeile-Zeichen erwartet. Der Standard-Wert ist „>␣“.

PS3
: Das Bereitzeichen für die *select*-Anweisung, das zur Abfrage der Nummer innerhalb der *select*-Schleife verwendet wird. Der Standard-Wert ist „*#?*␣“.

PS4
: Der Wert dieser Variablen wird vor jeder Zeile einer Ausführungs-Verfolgung (execution trace) ausgegeben. Der Wert dieser Variablen wird für den Parameter-Ersatz expandiert. Der Standard-Wert ist „+␣“.

SHELL
: Der Pfadname der POSIX-Shell wird in der Umgebung gehalten. Beim Aufruf wird die POSIX-Shell zu einer eingeschränkten Shell, wenn auf den Dateinamensteil des Pfadnamens (siehe *basename*) das Muster **r*sh** passt.

TMOUT
: Ist der Wert dieser Variablen positiv, dann beendet sich die POSIX-Shell selbständig, wenn nach Ausgabe des Bereitzeichens (*PS1*) nicht innerhalb der angegebenen Zeitspanne (in Sekunden) ein Kommando eingegeben wird.
(Vorsicht: Die POSIX-Shell kann mit einem Maximalwert für *TMOUT* compiliert worden sein, der nicht überschritten werden kann.)

VISUAL
: Endet der Wert dieser Variablen mit *vi*, dann wird die entsprechende Option (siehe *Eingebaute Kommandos*) gesetzt. Diese Variable ist sinnvoll, wenn der Zugang zur POSIX-Shell über *rlogin* erfolgt.

Die POSIX-Shell weist den folgenden Variablen Standard-Werte zu:
PATH, PS1, PS2, PS3, PS4, MAILCHECK, TMOUT und *IFS.*

Die Variablen *HOME, MAIL* und *SHELL* werden durch das Kommando /START-POSIX-SHELL gesetzt.

2.3.15 Blank-Ersetzung

Nach Parameter- und Kommando-Ersetzung wird das Ergebnis nach Feldtrennzeichen durchsucht und an den Fundstellen in selbständige Argumente unterteilt. Die Feldtrenner werden durch den Wert der Variable *IFS* definiert. Explizit vorhandene leere Zeichenketten (z.B. *""* oder *''*) bleiben dabei erhalten. Implizit vorhandene leere Zeichenketten, wie sie z.B. von Parametern ohne Wert herrühren, werden entfernt.

2.3.16 Dateinamen-Erzeugung

Nach den verschiedenen Ersetzungen wird jedes Wort auf das Auftreten von Stern *, Fragezeichen ? oder öffnende eckige Klammer [hin untersucht. Dies geschieht aber nur dann, wenn die Option *-f* (siehe *set*) nicht gesetzt wurde. Wird eines dieser Zeichen in einem Wort gefunden, dann wird dieses Wort als Muster betrachtet. Das Wort wird dann durch lexikographisch sortierte Dateinamen ersetzt, die auf das Muster passen. Wurde für das Muster kein Dateiname gefunden, dann wird das Wort unverändert gelassen.

Wenn Sie Muster für die Erzeugung von Dateinamen verwenden, müssen Sie den Zeichen Punkt . und Schrägstrich / besondere Aufmerksamkeit widmen: Punkt am Anfang eines Dateinamens oder unmittelbar nach /, sowie / selber müssen explizit passen. Bei anderen Ersetzungen gilt diese Sonderbehandlung der beiden Zeichen nicht.

***** wird durch jede Zeichenkette, auch die leere, ersetzt.

? wird durch ein beliebiges Zeichen ersetzt. (Zur Behandlung von / und . siehe oben.)

[...**]** wird durch genau ein Zeichen ersetzt, das in der Zeichenmenge innerhalb der eckigen Klammern enthalten ist.

Ein Zeichenpaar, getrennt durch den Bindestrich -, steht für alle Zeichen, die lexikographisch zwischen diesem Paar (inklusiv) liegen. Der Bindestrich kann als erstes oder letztes Zeichen in die Menge aufgenommen werden.
Ein Ausrufezeichen ! nach der öffnenden eckigen Klammer [negiert die Zeichenmenge, d.h. es werden alle nicht enthaltenen Zeichen angesprochen.

Eine *musterliste* ist eine Liste aus einem oder mehreren Mustern, die voneinander durch den senkrechten Strich | getrennt werden. Zusammengesetzte Muster können aus einem oder mehreren der folgenden Konstrukte geformt werden:

?(musterliste**)**
steht für kein- oder einmaliges Auftreten eines der angegebenen Muster.

***(**musterliste**)**
steht für kein- oder mehrmaliges Auftreten eines der angegebenen Muster.

+(musterliste**)**
steht für mindestens einmaliges Auftreten eines der angegebenen Muster.

@(musterliste**)**
Es muss genau eines der angegebenen Muster passen.

!(musterliste**)**
steht für alle anderen Muster außer den angegebenen Mustern.

2.3.17 Entwerten von Metazeichen (quoting)

Ein Metazeichen (metacharacter) ist eines der folgenden Zeichen:
; & ( ) | < > Leerzeichen Tabulatorzeichen Neue-Zeile-Zeichen

Ein *Blank* ist ein Tabulator oder ein Leerzeichen. Ein Bezeichner besteht aus einer Folge von Buchstaben, Ziffern und dem Unterstrich _, die mit einem Buchstaben oder dem Unterstrich beginnt. Bezeichner werden als Namen für Funktionen und Variable benutzt. Ein Wort ist eine Folge von Zeichen, die durch ein oder mehrere nicht entwertete Metazeichen getrennt wird. Jedes der Metazeichen hat eine spezielle Bedeutung für die POSIX-Shell und dient auch als Trenner von Wörtern, falls es nicht entwertet wurde.

Entwerter	Bedeutung
\	Ein Metazeichen kann durch Voranstellen des Gegenschrägstriches \ entwertet werden, damit es allein für sich steht. Das Paar *Neue-Zeile-Zeichen* wird von der POSIX-Shell ignoriert oder gelöscht.
'...'	Alle Zeichen, die in Hochkommata '...' eingeschlossen sind, sind entwertet. Ein einzelnes Hochkomma kann jedoch nicht darin vorkommen.
"..."	Zeichenketten in Anführungszeichen "..." eingeschlossen unterliegen der Parameter- und Kommando-Ersetzung. Durch Gegenschrägstrich können Sie hier Gegenschrägstrich \, Gegenhochkomma `, Anführungszeichen " und Dollarzeichen $ entwerten. Die Bedeutung der Angaben $* und $@ ist gleich, wenn sie nicht in Anführungszeichen eingeschlossen sind oder als Dateiname oder Wert für die Variablenwertzuweisung verwendet werden. Ihre Bedeutung ist unterschiedlich, wenn beide für sich in Anführungszeichen eingeschlossen als Kommandoargument verwendet werden. "$*" entspricht dann "$1␣$2␣...", wenn ␣ das erste Zeichen im Wert der Variable *IFS* ist, und "$@" steht dann für "$1"␣"$2"␣..., d.h. die einzelnen Aufrufargumente bleiben erhalten.
`...`	Bei in Gegenhochkommata `...` eingeschlossenen Zeichenketten können Sie mit dem Gegenschrägstrich \ den Gegenschrägstrich \, das Gegenhochkomma ` und das Dollarzeichen $ entwerten. Sollte das Ganze noch in Anführungszeichen "...`...`..." eingeschlossen sein, dann können Sie mit dem Gegenschrägstrich \ auch noch das Anführungszeichen " entwerten.

Die spezielle Bedeutung der reservierten Wörter oder Alias-Variablen kann durch Entwertung jedes einzelnen Zeichens annulliert werden. Es genügt aber auch ein einzelner „\“ vor dem Namen, um das gesamte Wort zu entwerten (z.B. „\while“).

Das Erkennen der Namen von Funktionen und von eingebauten Kommandos kann nicht auf diese Weise unterdrückt werden.

2.3.18 Arithmetische Berechnungen

Durch das eingebaute Kommando *let* steht Ganzzahl-Arithmetik zur Verfügung. Die Berechnungen werden auf der Basis von *Long*-Arithmetik durchgeführt. Konstanten werden in der Form [*basis#*]*n* dargestellt. Dabei ist *basis* eine ganze Zahl zwischen 2 und 36 und gibt die Basis an; *n* ist eine Zahl zu dieser Basis. Fehlt *basis#*, dann wird im Zehnersystem gerechnet.

Der arithmetische Ausdruck ist stark an die Programmiersprache C angelehnt. Er benutzt dieselbe Syntax, gleiche Vorrangregeln und Assoziativität. Alle unerlässlichen Operatoren außer ++, --, *?, :* und , sind vorhanden. Auf den Wert von Variablen kann über deren Namen zugegriffen werden, Sie müssen kein Dollarzeichen verwenden. Wenn der Wert einer Variablen eingesetzt wird, dann wird ihr Wert als arithmetischer Ausdruck berechnet.

Durch die Option *-i* des eingebauten Kommandos *typeset* kann als Attribut für die interne Darstellung des Wertes einer Variablen die Ganzzahldarstellung gewählt werden. Bei jeder Wertzuweisung einer Variablen mit dem *-i*-Attribut wird eine arithmetische Berechnung durchgeführt. Wird keine Basis für die Berechnungen angegeben, dann wird die Basis der ersten Wertzuweisung an die Variable verwendet. Diese Basis wird auch bei der Durchführung von Parameter-Ersetzung verwendet.

Da einige der arithmetischen Operatoren für die POSIX-Shell entwertet werden müssen, wurde eine alternative Form zum eingebauten Kommando *let* eingeführt. Bei jedem Kommando, das mit doppelter runder Klammer auf *((* beginnt, werden alle Zeichen bis zum schließenden runden Klammernpaar *))* als entwertet genommen.
((a=a+b)) entspricht *let "a=a+b"*.

2.3.19 Bedingte Ausdrücke

Einen bedingten Ausdruck verwenden Sie zum Testen der Eigenschaften von Dateien, für algebraische Vergleiche und zum Vergleich von Zeichenketten. In der POSIX-Shell werden die bedingten Ausdrücke innerhalb der Anweisung *[[...]]* angegeben. Die Ersetzung von Blanks und die Erzeugung von Dateinamen werden nicht auf die Wörter des bedingten Ausdrucks angewandt. Jeder bedingte Ausdruck kann aus einem oder mehreren der folgenden unären oder binären Ausdrücke gebildet werden:

Ist in den folgenden Ausdrücken *datei* von der Form /dev/fd/*n* (fd - file descriptor, *n* ist eine ganze Zahl), dann wird der Test auf die geöffnete Datei mit Dateikennzahl *n* durchgeführt.

-a␣datei
(a - access) wahr, wenn *datei* existiert.

-b␣datei
(b - block device) wahr, wenn *datei* existiert und ein blockorientiertes Gerät ist.

-c␣datei
(c - character device) wahr, wenn *datei* existiert und ein zeichenorientiertes Gerät ist.

-d␣datei
(d - directory) wahr, wenn *datei* existiert und ein Dateiverzeichnis ist.

-f␣datei
(f - file) wahr, wenn *datei* existiert und eine einfache Datei ist.

-g␣datei
(g - group ID) wahr, wenn *datei* existiert und das set-user-ID-Bit für die Gruppe gesetzt ist.

-k␣datei
(k - stic**k**y) wahr, wenn *datei* existiert und das sticky- oder t-Bit gesetzt ist.

-o␣option
(o - option) wahr, wenn die angegebene Option *option* aktiv ist, wobei *option* mit dem vollen Optionsnamen angegeben sein muss, z.B. *errexit*. (*option* kann mit *set* gesetzt werden).

-p␣datei
(p - pipe) wahr, wenn *datei* existiert und eine benannte Pipe (FIFO) oder eine Pipe ist.

-r␣datei
(r - read) wahr, wenn *datei* existiert und der aktuelle Prozess das Leserecht hat.

-s␣datei
(s - size) wahr, wenn *datei* existiert und nicht leer ist.

-t␣dateikennzahl
(t - terminal) wahr, wenn *dateikennzahl* geöffnet und einer Datensichtstation zugeordnet ist.

-u␣datei
(u - user ID) wahr, wenn *datei* existiert und das set-user-ID-Bit für den Eigentümer gesetzt ist.

-w␣datei
(w - write) wahr, wenn *datei* existiert und der aktuelle Prozess das Schreibrecht hat.

-x␣datei
(x - execute) wahr, wenn *datei* existiert und der aktuelle Prozess das Ausführrecht hat. Existiert *datei* und ist sie ein Dateiverzeichnis, dann muss für wahr der aktuelle Prozess das Recht zum Durchlaufen haben.

-G␣datei
(G - group) wahr, wenn *datei* existiert und die Gruppe der Datei der effektiven Gruppennummer des aktuellen Prozesses entspricht.

-L␣datei
(L - symbolic link) wahr, wenn *datei* existiert und ein symbolischer Link ist.

-O␣datei
(O - owner) wahr, wenn *datei* existiert und der Eigentümer der Datei der effektiven Benutzernummer des aktuellen Prozesses entspricht.

-S␣datei
(S - socket) wahr, wenn *datei* existiert und ein Socket ist.

datei1␣**-nt**␣datei2
(nt - newer than) wahr, wenn *datei1* existiert und neuer als *datei2* ist.

datei1␣**-ot**␣datei2
(ot - older than) wahr, wenn *datei1* existiert und älter als *datei2* ist.

datei1␣**-ef**␣datei2
(ef - equal file) wahr, wenn *datei1* und *datei2* existieren und beide ein Verweis auf dieselbe Datei sind.

Eigenschaften und Vergleiche von Zeichenketten

-n␣zeichenkette
(n - non zero) wahr, wenn die *zeichenkette* existiert und nicht die leere Zeichenkette ist, also eine Länge größer 0 hat.

-z␣zeichenkette
(z -zero) wahr, wenn die angegebene *zeichenkette* die leere Zeichenkette ist, also die Länge 0 hat.

zeichenkette␣**=**␣muster
wahr, wenn *zeichenkette1* auf *muster* passt.

zeichenkette␣**!=**␣muster
wahr, wenn *zeichenkette1* auf *muster* passt.

zeichenkette1␣**<**␣zeichenkette2
wahr, wenn *zeichenkette1* alphabetisch (EBCDIC-Ordnung) vor *zeichenkette2* liegt.

zeichenkette1␣**>**␣zeichenkette2
wahr, wenn *zeichenkette1* alphabetisch (EBCDIC-Ordnung) nach *zeichenkette2* liegt.

Algebraischer Vergleich ganzer Zahlen

zahl1␣**-eq**␣zahl2
(eq - equal) wahr, wenn *zahl1* gleich *zahl2* ist.

zahl1␣**-ne**␣zahl2
(ne - not equal) wahr, wenn *zahl1* ungleich *zahl2* ist.

zahl1␣**-lt**␣zahl2
(lt - less than) wahr, wenn *zahl1* kleiner als *zahl2* ist.

zahl1␣**-gt**␣zahl2
(gt - greater than) wahr, wenn *zahl1* größer als *zahl2* ist.

zahl1␣**-le**␣zahl2
(le - less than or equal) wahr, wenn *zahl1* kleiner oder gleich *zahl2* ist.

zahl1␣**-ge**␣zahl2
(ge - greater than or equal) wahr, wenn *zahl1* größer oder gleich *zahl2* ist.

Bedingungen verknüpfen oder negieren

Mehrere Bedingungen können Sie miteinander zu einem Ausdruck verknüpfen. Die folgenden Konstrukte sind nach Priorität geordnet; Klammerung hat die höchste, das logische ODER die niedrigste Priorität:

(␣bedingung␣**)**
bedingung steht hier für (eine oder) mehrere Bedingungen, die beliebig miteinander verknüpft sind. Der Ausdruck ist wahr, wenn *bedingung* wahr ist.

!␣bedingung
Negation: wahr, wenn *bedingung* falsch ist.

bedingung1␣**&&**␣bedingung2
Logisches UND: wahr, wenn *bedingung1* und *bedingung2* wahr sind.

bedingung1␣**||**␣bedingung2
Logisches ODER: wahr, wenn entweder *bedingung1* oder *bedingung2* wahr ist.

2.3.20 Umgebung

Die Umgebung eines Prozesses enthält eine Liste aus Paaren *name=wert*, die an ein ausgeführtes Programm in gleicher Weise wie eine normale Argumentliste übergeben wird. Die Namen *name* müssen Bezeichner im Sinne der POSIX-Shell sein, die Werte *wert* Zeichenketten (auch die leere).

Die POSIX-Shell und die Umgebung beeinflussen sich gegenseitig. Beim Aufruf durchsucht die POSIX-Shell die Umgebung und erzeugt eine Variable für jeden gefundenen Namen, weist ihr den entsprechenden Wert zu und markiert sie als exportiert. Ausgeführte Kommandos erben diese Umgebung. Modifiziert der Benutzer die Werte dieser Variablen oder fügt neue Variablen mit Hilfe der eingebauten Kommandos *export* oder *typeset -x* dazu, dann werden diese Teil der Umgebung. Die Umgebung, die von jedem ausgeführten Kommando gesehen wird, besteht aus den Paaren *name=wert*, die ursprünglich von der POSIX-Shell geerbt wurden, deren Werte von der aktuellen Shell modifiziert worden sein können, plus den Erweiterungen, die mit *export* oder *typeset -x* markiert wurden.

Die Umgebung eines einfachen Kommandos oder einer Funktion kann durch Voranstellen von Variablenwertzuweisungen erweitert werden. Eine Variablenwertzuweisung wird als Wort behandelt und hat die Form *bezeichner=wert*.

Die folgenden Zeilen sind für *kommando* äquivalent:

```
TERM=450 kommando argumente
( export TERM ; TERM=450 ; kommando argumente )
```

Wurde die Option *-k* beim Aufruf der POSIX-Shell oder durch das eingebaute Kommando *set* gesetzt, dann werden alle Variablenwertzuweisungen in die Umgebung exportiert, auch wenn sie nach dem Kommandonamen kommen.

2.3.21 Funktionen

Das reservierte Wort *function* (siehe *Zusammengesetzte Kommandos*) wird zur Definition von POSIX-Shell-Funktionen benutzt. Funktionen werden von der Shell eingelesen und intern gespeichert. Alias-Namen werden aufgelöst, wenn die Funktion gelesen wird. Funktionen werden wie Kommandos ausgeführt, Argumente werden als Stellungsparameter übergeben (siehe *Ausführung*).

Funktionen werden im aktuellen Prozess ausgeführt und haben Zugriff auf alle geöffneten Dateien und das aktuelle Verzeichnis.

Signale (traps), die vom Aufrufer abgefangen werden, werden innerhalb der Funktion auf ihre Standard-Aktion zurückgesetzt. Ein Signal, das von der Funktion weder abgefangen noch ignoriert wird, führt zum Abbruch der Funktion. Dieses Signal wird an den Aufrufer der Funktion weitergegeben. Eine *EXIT-trap*-Behandlung, die innerhalb einer Funktion gesetzt wurde, wird nach Beendigung der Funktion in der Umgebung des Aufrufers ausgeführt. Normalerweise werden die Variablen von Aufrufer und Funktion gemeinsam benutzt.

Das eingebaute Kommando *typeset* kann jedoch dazu benutzt werden, innerhalb einer Funktion lokale Variablen zu definieren, deren Gültigkeitsbereich dann die Funktion und alle aufgerufenen Funktionen umfasst.

Das eingebaute Kommando *return* wird zur Rückkehr von Funktionen verwendet. Fehler, die innerhalb einer Funktion auftreten, geben die Kontrolle an den Aufrufer zurück.

Die Bezeichner oder Namen von Funktionen können Sie mit Hilfe des eingebauten Kommandos *typeset* und einer der Optionen *-f* oder *+f* auflisten. Den Text der Funktionen können Sie mit der Option *-f* auflisten. Eine Funktion können Sie mit dem eingebauten Kommando *unset* und der Option *-f* löschen.

Normalerweise kann man auf die Funktionen nicht zugreifen, während die POSIX-Shell eine Prozedur ausführt.

Durch die Option *-xf* des eingebauten Kommandos *typeset* können Funktionen als exportierbar markiert werden. Diese Funktionen können Sie dann in Prozeduren verwenden, die ohne einen weiteren Aufruf der POSIX-Shell ausgeführt werden. Funktionen, die über mehrere Aufrufe der POSIX-Shell bekannt sein sollen, müssen Sie in der *ENV*-Datei mit *typeset -xf* definieren.

Beispiel Die folgende Funktion *lh* gibt Ihnen die obersten zwei Schichten der Dateiverzeichnis-Hierarchie aus, an der Sie stehen oder die Sie als Argument angegeben haben. Einfache Dateien als Argument werden ignoriert.

```
function lh
{
  for i in ${*:-.}
  do
     if [[ -d $i ]]
     then
        print $i:
        cd $i
        ls -CF $( ls )
        cd - >/dev/null
     fi
  done
}
```

Die *for*-Schleife arbeitet die Argumente einzeln ab. Haben Sie kein Argument angegeben, wird das aktuelle Verzeichnis (.) für *$** gesetzt.
Liegt ein Verzeichnis vor, dann wird dessen Name und Inhalt angezeigt: *cd $i* wechselt zu dem anzuzeigenden Verzeichnis. *$(ls)* wird durch den Inhalt des Verzeichnisses *$i* ersetzt. *ls -CF verzeichnis_inhalt* gibt Ihnen die Dateinamen, gefolgt von den Inhalten der Verzeichnisse, mit Markierungen für Zugriffsrechte und Verzeichnisse wieder. Das folgende *cd*-Kommando wechselt in das Ausgangsverzeichnis (vor dem ersten *cd $i*).

2.3.22 Aufträge

Ist die *monitor*-Option des eingebauten Kommandos *set* eingeschaltet (*set -m*), dann verknüpft eine interaktive POSIX-Shell mit jeder Pipe einen Auftrag. Die Shell hält eine Tabelle der aktuellen Aufträge, die Sie mit dem eingebauten Kommando *jobs* auf die Standard-Ausgabe schreiben können. Jedem Auftrag wird ein kleine ganze Zahl zugewiesen. Wird ein Auftrag im Hintergrund (*&*) gestartet, dann gibt die Shell eine Zeile der folgenden Form aus:

```
[1] 1234
```

Diese Zeile besagt, dass der Auftrag im Hintergrund mit der Auftragsnummer 1 gestartet wurde und einen (Top-Level) Prozess mit der Prozessnummer 1234 hat.

Möchten Sie andere Prozesse starten, während ein Prozess ausgeführt wird, dann brauchen Sie nur die Tastenkombination CTRL Z (bzw. @@z an Blockterminals) zu drücken. Diese bewirkt, dass ein STOP-Signal an den laufenden Prozess gesendet wird. Die POSIX-Shell wird dann anzeigen, dass der Auftrag gestoppt (*stopped*) wurde, und ein Bereitzeichen ausgeben. Sie können dann den Zustand dieses Auftrages manipulieren: Den Auftrag mit Hilfe des eingebauten Kommandos *bg* (background) im Hintergrund weiterlaufen lassen, ihn im gestoppten Zustand lassen und andere Kommandos ausführen oder wieder in den Vordergrund durch das eingebaute Kommando *fg* (foreground) holen.
CTRL Z (bzw. @@z) wirkt nicht auf eingebaute (builtin)Kommandos, sondern nur auf Kommandos, die in einem geforkten Prozess ausgeführt werden.

Sie haben mehrere Möglichkeiten, auf die Aufträge zuzugreifen. Sie können über die Prozessnummer eines Prozesses auf den Auftrag oder mit einem der folgenden Ausdrücke zugreifen:

%nummer
: der Auftrag mit der gegebenen Auftrags-*nummer*.

%zeichenkette
: jeder Auftrag, dessen Kommando-Zeile mit *zeichenkette* beginnt.

%?zeichenkette
: jeder Auftrag, dessen Kommando-Zeile *zeichenkette* enthält.

%%
: der aktuelle Auftrag.

%+
: synonym für *%%*

%-
: der letzte Auftrag.

Die POSIX-Shell registriert jeden Zustandswechsel eines Auftrags. Normalerweise informiert Sie die POSIX-Shell sofort, wenn ein Auftrag gestoppt wurde und nicht weiter ausgeführt werden kann. Damit Ihre Arbeit nicht gestört wird, wird diese Information vor der Ausgabe eines Bereitzeichens ausgegeben.

Wurde die *monitor*-Option eingeschaltet, dann löst jeder beendete Hintergrundauftrag jeden für *CHLD* gesetzten *trap* aus.

Wenn Sie versuchen, die POSIX-Shell zu verlassen, während Sie noch gestoppte Aufträge im Hintergrund haben, dann werden Sie gewarnt:

Sie können dann das Kommando *jobs* benutzen, um sich eine Übersicht der augenblicklichen Situation zu verschaffen. Wenn Sie dies getan haben oder unmittelbar nach dem ersten Versuch ein zweites Mal die POSIX-Shell verlassen möchten, werden Sie kein zweites Mal gewarnt und die gestoppten Aufträge werden beendet.

2.3.23 Signale

Die Signale INT und QUIT werden für ein aufgerufenes Hintergrund-Kommando (*&*) ignoriert, wenn die Option *monitor* der Auftragssteuerung nicht aktiv ist. Sonst haben die Signale die Werte, die von der POSIX-Shell von ihrem Vaterprozess geerbt wurden. (Lesen Sie dazu auch beim eingebauten Kommando *trap* nach.)

2.3.24 Ausführung

Jedesmal, wenn ein Kommando ausgeführt wird, werden die bereits beschriebenen Ersetzungen in dieser Reihenfolge durchgeführt:

- Entwertung
- Parameter-Ersetzung
- Tilde-Ersetzung
- Alias-Ersetzung
- Dateinamen-Erzeugung
- Ein- und Ausgabeumlenkung
- Kommando-Ersetzung

Entspricht der Kommandoname dem Namen eines eingebauten Kommandos, dann wird dieses in der aktuellen POSIX-Shell ausgeführt. Als nächstes wird geprüft, ob der Kommandoname dem Namen einer benutzerdefinierten Funktion entspricht. Ist dies der Fall, dann werden die Stellungsparameter gesichert und auf die Werte der Argumente des Funktionsaufrufs gesetzt. Wenn die Funktion beendet ist oder das eingebaute Kommando *return* ausgeführt wurde, werden die Stellungsparameter wiederhergestellt und jeder in der Funktion für *EXIT* gesetzte *trap* ausgeführt. Der Exit-Wert ist der Wert des letzten Kommandos in der Funktion. Eine Funktion wird auch in der aktuellen POSIX-Shell ausgeführt. Ist ein Kommandoname nicht unter den eingebauten Kommandos oder den benutzerdefinierten Funktionen zu finden, dann wird ein Prozess erzeugt und versucht, das Kommando durch den Systemaufruf *exec* ausführen zu lassen.

Die Variable *PATH* definiert den Suchpfad für die Dateiverzeichnisse mit den Kommandos. Die einzelnen Verzeichnisse werden durch Doppelpunkt *:* getrennt. Der Standard-Suchpfad ist */usr/bin:*. Damit sind, in dieser Reihenfolge, das Verzeichnis */usr/bin* und das aktuelle Verzeichnis gemeint. Das aktuelle Verzeichnis kann durch einen Doppelpunkt am Anfang oder Ende oder durch zwei (oder mehr) aufeinanderfolgende Doppelpunkte angegeben werden. Der Suchpfad wird nicht benutzt, wenn der Kommandoname einen Schrägstrich */* enthält. Ohne */* im Kommandonamen wird jedes Verzeichnis im Suchpfad nach einer ausführbaren Datei durchsucht. Existiert für die Datei das Ausführrecht und ist sie eine einfache Datei ohne das *a.out*-Format, dann wird angenommen, dass sie eine Shell-Prozedur enthält. In diesem Fall wird eine Subshell zum Lesen der Datei aufgerufen. Alle nicht exportierten Alias-Variablen, Funktionen und Umgebungsvariablen werden nicht in die Subshell kopiert. Ein geklammertes Kommando wird in einer Subshell ohne Löschung der nicht exportierten Teile ausgeführt.

2.3.25 Kommando-Wiederaufruf

Der Text der letzten *HISTSIZE*-Kommandos, die an einer Datensichtstation eingegeben wurden, wird in einer *History*-Datei gespeichert. Der Standard-Wert für *HISTSIZE* ist 128. Wenn die Variable *HISTFILE* nicht gesetzt ist oder die durch den Wert angegebene Datei nicht schreibbar ist, wird die Datei *$HOME/.sh_history* zur Speicherung benutzt.

Eine POSIX-Shell kann auf alle Kommandos von interaktiven POSIX-Shells zugreifen, welche die gleiche *History*-Datei benutzen. Das eingebaute Kommando *fc* können Sie zum Auslisten oder Editieren eines Bereiches dieser Datei benutzt. Der Dateibereich, der angesprochen wird, kann durch eine Zahl, den ersten Buchstaben des Kommandos oder eine Zeichenkette aus dem Kommando angesprochen werden. Der Bereich kann aus einem oder mehreren Kommandos bestehen.

Wenn Sie keinen Editor durch ein Argument beim Aufruf von *fc* definieren, dann wird der Wert der Variablen *FCEDIT* verwendet. Ist *FCEDIT* nicht definiert, dann wird */usr/bin/ed* benutzt. Die editierten Kommandos werden ausgegeben und nach Verlassen des Editors wieder ausgeführt.

Der Editorname Bindestrich - (wenn FCEDIT=-) wird zum Überspringen der Editierphase und zur direkten Ausführung benutzt. In diesem Fall kann eine Variable der Form *alt=neu* zur Änderung des Kommandos vor der Ausführung benutzt werden. Ist beispielsweise *r* die Aliasvariable für *fc -e -*, und Sie geben *r unsinn=sinn c* ein, dann wird das letzte Kommando, das mit dem Zeichen *c* begann, ausgeführt. Zuvor wird aber das erste Auftreten von *unsinn* im Kommando durch *sinn* ersetzt.

2.4 Kommando-Übersicht

Nachfolgende Übersichten zeigen

- eine Übersicht über alle POSIX-Kommandos der POSIX-Shell
- eine Übersicht nach Funktionen

Es gibt auch Kommandos, die von der POSIX-Shell aus eingegeben werden können, die aber nicht in diesem Handbuch beschrieben sind, wie beispielsweise die Kommandos *cc* oder *c89* zum Aufruf eines Compilers (siehe dazu das Handbuch „C/C++ (BS2000/OSD)“ [5]).

2.4.1 Übersicht über alle Kommandos der POSIX-Shell

Die POSIX-Shell setzt sich zusammen aus der Basis-Shell (POSIX-BC) und der erweiterten Shell (POSIX-SH). Sie beinhaltet die in folgender Tabelle aufgeführten POSIX-Kommandos.

Die Einträge in der Spalte *Typ* beschreiben den Typ des Kommandos:

bin eigener Modul

blt Builtin in Shell (= eingebautes POSIX-Kommando, siehe Seite 38).

Zusätzlich zu diesen Kommandos gibt es noch weitere eingebaute Kommandos (wie *for*, *while*, *if*, *break* usw.), die im Abschnitt „Zusammengesetzte Anweisungen“ auf Seite 43 beschrieben sind.

scr Skript

Die Spalte *LFS* beschreibt, ob die Kommandos große POSIX-Dateien bearbeiten können:

A (large file **a**ware): arbeitet korrekt mit großen Dateien

S (large file **s**afe): erkennt große Dateien, weist die Bearbeitung jedoch definiert zurück

Name	Ort	Typ	Auslieferung	Beschreibung	LFS
alias	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	Alias-Namen definieren oder anzeigen	
ar	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Bibliotheken verwalten	S
asa	/usr/bin	bin	SINLIB.POSIX-SH	Steuerzeichen für die Positionierung umsetzen	
at	/usr/bin	bin	SINLIB.POSIX-SH	Kommandos zu einem späteren Zeitpunkt ausführen	
awk	/usr/bin	bin	SINLIB.POSIX-SH	Programmierbare Bearbeitung von Textdateien	A
basename	/usr/bin	scr	SINLIB.POSIX-BC.SHELL	Dateinamen vom Pfad trennen	
batch	/usr/bin	scr	SINLIB.POSIX-BC.SHELL	Kommandos zu einer späteren Zeit ausführen	
bc	/usr/bin	bin	SINLIB.POSIX-SH	Arithmetische Sprache	
bg	-	blt	SINLIB.POSIX-BC.SHELL	Jobs in den Hintergrund schicken	
bs2cmd	-	blt	SINLIB.POSIX-BC.SHELL	BS2000-Kommando ausführen	
bs2cp	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	BS2000-Dateien kopieren (BS2000)	A
bs2do	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	BS2000-Prozeduren aus POSIX aufrufen	
bs2file	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Dateiattribute für BS2000-Dateien festlegen (BS2000)	
bs2lp	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Dateien ausdrucken (BS2000)	
bs2pkey	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	P-Tasten belegen (BS2000)	
cal	/usr/bin	bin	SINLIB.POSIX-SH	Kalender ausgeben	
cancel	/usr/bin	bin	SINLIB.POSIX-SH	Druckaufträge löschen	
cat	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Dateien aneinanderfügen und ausgeben	A
cd	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	Aktuelles Dateiverzeichnis wechseln	A
chgrp	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Gruppennummer einer Datei ändern	A
chmod	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Zugriffsrechte ändern	A
chown	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Eigentümer einer Datei ändern	A
cksum	/usr/bin	bin	SINLIB.POSIX-SH	Prüfsummen und Größen von Dateien schreiben	A
cmp	/usr/bin	bin	SINLIB.POSIX-SH	Dateien zeichenweise vergleichen	A
comm	/usr/bin	bin	SINLIB.POSIX-SH	Gleiche Zeilen in zwei sortierten Dateien suchen	S
command	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	einfaches Kommando ausführen	
compress	/usr/bin	bin	SINLIB.POSIX-SH	Dateien komprimieren	A
cp	/sbin	blt+bin	SINLIB.POSIX-BC.SHELL	Dateien kopieren	A
cp	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Dateien kopieren	A
cpio	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Dateien und Dateiverzeichnisse ein- und auslagern	A
crontab	/usr/bin	bin	SINLIB.POSIX-SH	Kommandos regelmäßig zu bestimmten Zeitpunkten ausführen	
csplit	/usr/bin	bin	SINLIB.POSIX-SH	Datei nach bestimmten Kriterien unterteilen	S
cut	/usr/bin	bin	SINLIB.POSIX-SH	Bytes, Zeichen oder Felder aus den Zeilen einer Datei ausschneiden	S
date	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Datum und Uhrzeit ausgeben	

Name	Ort	Typ	Auslieferung	Beschreibung	LFS
dd	/sbin	bin	SINLIB.POSIX-SH	Dateien kopieren und konvertieren	S
debug	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Testen von POSIX-Programmen	
df	/sbin	bin	SINLIB.POSIX-BC.SHELL	Anzahl der freien und belegten Plattenblöcke ausgeben	A
diff	/usr/bin	bin	SINLIB.POSIX-SH	Dateien zeilenweise vergleichen	A
dirname	/usr/bin	scr	SINLIB.POSIX-BC.SHELL	Pfad-Präfix vom Dateinamen trennen	
du	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Belegten Speicherplatz ausgeben	A
dumpfs	/sbin	bin	SINLIB.POSIX-BC.ROOT	interne Dateisystem-Information ausgeben	
echo	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Aufruf-Argumente ausgeben	
ed	/sbin	bin	SINLIB.POSIX-SH	Zeilenorientierter Editor im Dialogbetrieb	
edt	-	blt	SINLIB.POSIX-BC.SHELL	BS2000-Dateibearbeiter EDT aufrufen	S
edtu	-	blt	SINLIB.POSIX-BC.SHELL	BS2000-Dateibearbeiter EDT aufrufen	S
egrep	/usr/bin	bin	SINLIB.POSIX-SH	Muster suchen	S
env	/usr/bin	bin	SINLIB.POSIX-SH	Umgebung bei Ausführung von Kommandos ändern	
eval	-	blt	SINLIB.POSIX-BC.SHELL	Aufrufargumente bearbeiten und als Kommando ausführen	
ex	/usr/bin	bin	SINLIB.POSIX-SH	Zeilenorientierter Editor	
exec	-	blt	SINLIB.POSIX-BC.SHELL	Die aktuelle Shell überlagern	
exit	-	blt	SINLIB.POSIX-BC.SHELL	Shell-Prozedur beenden	
expand	/usr/bin	bin	SINLIB.POSIX-SH	Tabulatorzeichen in Leerzeichen umwandeln	S
export	-	blt	SINLIB.POSIX-BC.SHELL	Shell-Variablen exportieren	
expr	/sbin	blt+bin	SINLIB.POSIX-BC.SHELL	Ausdrücke auswerten	
expr	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Ausdrücke auswerten	
false	/usr/bin	alias+scr	SINLIB.POSIX-BC.SHELL	Endestatus ungleich 0 zurückgeben	
fc	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	Zugriff auf die History-Datei	
fg	-	blt	SINLIB.POSIX-BC.SHELL	Jobs in den Vordergrund bringen	
fgrep	/usr/bin	bin	SINLIB.POSIX-SH	Zeichenketten suchen	S
file	/usr/bin	bin	SINLIB.POSIX-SH	Art einer Datei bestimmen	A
find	/usr/bin	bin	SINLIB.POSIX-SH	Dateiverzeichnisse durchsuchen	A
fold	/usr/bin	bin	SINLIB.POSIX-SH	Lange Zeilen zerlegen	S
fsck	/sbin	bin	SINLIB.POSIX-BC.ROOT	Konsistenzprüfung des Dateisystems und Korrektur im Benutzer-Dialog	
fsexpand	/sbin	bin	SINLIB.POSIX-BC.ROOT	Existierende Dateisysteme vergrößern	A
ftyp	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Bearbeitungsart für Dateien festlegen (BS2000)	
fuser	/usr/sbin	bin	SINLIB.POSIX-BC.ROOT	Dateinutzer anzeigen	
gencat	/usr/bin	bin	SINLIB.POSIX-SH	Binär codierten Meldungskatalog erzeugen	
genso	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Shared Object erzeugen	

Name	Ort	Typ	Auslieferung	Beschreibung	LFS
getconf	/usr/bin	bin	SINLIB.POSIX-SH	Konfigurationswerte abrufen	A
getopts	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	Argumente einer Prozedur nach Optionen durchsuchen	
grep	/sbin	bin	SINLIB.POSIX-BC.SHELL	Muster suchen	A
hash	/usr/bin	alias+scr	SINLIB.POSIX-BC.SHELL	Hash-Tabelle der Shell bearbeiten	
hd	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Dateiinhalt hexadezimal ausgeben	A
head	/usr/bin	bin	SINLIB.POSIX-SH	Anfang einer Datei ausgeben	A
iconv	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Code konvertieren	A
id	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Benutzer-Identifikation ausgeben	
inetd	/usr/sbin	bin	SINLIB.POSIX-BC.ROOT	Dämon für Internet-Dienste	
info	/sbin	bin	SINLIB.POSIX-BC.ROOT	Online-Diagnosetool	
ipcrm	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Einrichtungen zur Interprozess-Kommunikation löschen	
ipcs	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Zustand von Interprozess-Kommunikationseinrichtungen anzeigen	
jobs	-	blt	SINLIB.POSIX-BC.SHELL	Auftragsinformationen ausgeben	
join	/usr/bin	bin	SINLIB.POSIX-SH	Zwei Dateien nach Vergleichsfeldern verbinden	A
kill	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	Signale an Prozesse senden	
last	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Zuletzt angemeldete Benutzer anzeigen	
let	-	blt	SINLIB.POSIX-BC.SHELL	Arithmetische Berechnungen	
lex	/usr/bin	bin	SINLIB.POSIX-SH	Scanner erstellen	
ln	/sbin	blt+bin	SINLIB.POSIX-BC.SHELL	Verweis auf eine Datei eintragen	A
locale	/usr/bin	bin	SINLIB.POSIX-SH	Informationen über die internationale Umgebung abrufen	
localedef	/usr/bin	bin	SINLIB.POSIX-SH	Internationale Umgebung definieren	
logger	/usr/bin	bin	SINLIB.POSIX-SH	Meldungen protokollieren	
logname	/usr/bin	bin	SINLIB.POSIX-SH	Login-Kennung abfragen	
logrotate	/usr/sbin	scr	SINLIB.POSIX-BC.ROOT	Wechsel der Protokolldateien des syslog-Dämonen	S
lp	/usr/bin	bin	SINLIB.POSIX-SH	Dateien ausdrucken	
lpstat	/usr/bin	bin	SINLIB.POSIX-SH	Informationen über Druckaufträge ausgeben	
ls	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Informationen über Dateiverzeichnisse und Dateien ausgeben	A
mailx	/usr/bin	bin	SINLIB.POSIX-SH	Nachrichten interaktiv bearbeiten	
make	/usr/bin	bin	SINLIB.POSIX-SH	Gruppen von Dateien verwalten	
man	/usr/bin	scr	SINLIB.POSIX-SH	Online-Dokumentation nutzen	
mesg	/usr/bin	bin	SINLIB.POSIX-SH	Nachrichtenempfang verbieten oder erlauben	
mkdir	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Dateiverzeichnis erzeugen	

Name	Ort	Typ	Auslieferung	Beschreibung	LFS
mkfifo	/usr/bin	bin	SINLIB.POSIX-SH	FIFO erstellen	A
mkfs	/sbin	bin	SINLIB.POSIX-BC.ROOT	Dateisystem erstellen	
mknod	/sbin	bin	SINLIB.POSIX-BC.ROOT	Gerätedatei anlegen	A
more	/usr/bin	bin	SINLIB.POSIX-SH	Bildschirmausgabe steuern	A
mount	/sbin	bin	SINLIB.POSIX-BC.ROOT	Dateisysteme und ferne Ressourcen einhängen	
mountall	/sbin	scr	SINLIB.POSIX-BC.ROOT	Mehrere Dateisysteme einhängen	
mv	/sbin	blt+bin	SINLIB.POSIX-BC.SHELL	Dateien versetzen oder umbenennen	A
newgrp	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Gruppenzugehörigkeit ändern	
nice	/usr/bin	bin	SINLIB.POSIX-SH	Priorität von Kommandos ändern	
nl	/usr/bin	bin	SINLIB.POSIX-SH	Textzeilen nummerieren	S
nm	/usr/bin	bin	SINLIB.POSIX-SH	Symboltabelle einer Objektdatei ausgeben	
nohup	/usr/bin	bin	SINLIB.POSIX-SH	Kommando ausführen und dabei Signale ignorieren	
od	/usr/bin	bin	SINLIB.POSIX-SH	Inhalt einer Datei oktal ausgeben	S
paste	/usr/bin	bin	SINLIB.POSIX-SH	Zeilen zusammenfügen	S
patch	/usr/bin	bin	SINLIB.POSIX-SH	Differenzliste anwenden	
pathchk	/usr/bin	bin	SINLIB.POSIX-SH	Pfadnamen überprüfen	
pax	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Bearbeitung portierbarer Archive	A
pdbl	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Privaten POSIX-Lader verwalten	
ping	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Senden von Echo-Request-Paketen an Netzwerkkomponenten	
pkginfo	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Informationen über Software-Pakete anzeigen	
posdbl	/usr/sbin	bin	SINLIB.POSIX-BC.ROOT	Verwalten des POSIX-Laders	
pr	/usr/bin	bin	SINLIB.POSIX-SH	Dateien formatieren und auf Standard-Ausgabe ausgeben	
print	-	blt	SINLIB.POSIX-BC.SHELL	Ausgabemechanismus ähnlich echo	
printf	/usr/bin	blt+bin	SINLIB.POSIX-SH	Formatierte Ausgabe	
ps	/sbin	bin	SINLIB.POSIX-BC.SHELL	Prozessdaten abfragen	
pwd	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Pfadnamen des aktuellen Dateiverzeichnisses ausgeben	
rcp	/usr/bin	bin	SINLIB.POSIX-BC.INET	Datei von oder zu einem fernen Rechner kopieren	A
read	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	Argumente von der Standard-Eingabe lesen, Shell-Variablen zuweisen	
readonly	-	blt	SINLIB.POSIX-BC.SHELL	Shell-Variablen schützen	
renice	/usr/bin	bin	SINLIB.POSIX-SH	Priorität laufender Prozesse ändern	
rm	/sbin	blt+bin	SINLIB.POSIX-BC.SHELL	Dateien löschen	
rmdir	/sbin	blt+bin	SINLIB.POSIX-BC.SHELL	Dateiverzeichnisse löschen	A
rmpart	/sbin	bin	SINLIB.POSIX-BC.ROOT	Partition entfernen	

Name	Ort	Typ	Auslieferung	Beschreibung	LFS
rsh	/usr/bin	bin	SINLIB.POSIX-BC.INET	Shell-Kommando am fernen Rechner ausführen	
sed	/usr/bin	bin	SINLIB.POSIX-SH	Editor im Prozedurbetrieb	
set	-	blt	SINLIB.POSIX-BC.SHELL	Parameter oder Optionen setzen, Variablen ausgeben	
sh	/sbin	bin	SINLIB.POSIX-BC.SHELL	Kommandointerpreter und Programmiersprache der POSIX-Shell	A
shift	-	blt	SINLIB.POSIX-BC.SHELL	Werte der Stellungsparameter nach links verschieben	
show_pubset_ export	/sbin	scr	SINLIB.POSIX-BC.ROOT	vom EXPORT-PUBSET betroffene Dateisysteme anzeigen	
sleep	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Prozesse zeitweise stillegen	
sort	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Dateien sortieren und/oder mischen	
split	/usr/bin	bin	SINLIB.POSIX-SH	Datei auf mehrere Dateien verteilen	A
start_bs2fsd	/sbin	scr	SINLIB.POSIX-BC.ROOT	Kopierdämönen starten	
strings	/usr/bin	bin	SINLIB.POSIX-SH	Druckbare Zeichenketten in Objekt- oder Binärdateien suchen	S
stty	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Eigenschaften einer Datensichtstation ausgeben oder ändern	
su	/sbin	bin	SINLIB.POSIX-BC.ROOT	Benutzerkennung wechseln	
sum	/usr/bin	bin	SINLIB.POSIX-SH	Prüfsumme einer Datei berechnen	A
sync	/sbin	bin	SINLIB.POSIX-BC.ROOT	Systempuffer zurückschreiben	
tabs	/usr/bin	bin	SINLIB.POSIX-SH	Tabulatorstops setzen	
tail	/usr/bin	bin	SINLIB.POSIX-SH	Den letzten Teil einer Datei ausgeben	A
talk	/usr/bin	bin	SINLIB.POSIX-SH	Dialog mit anderem Benutzer führen	
tar	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Archivieren von Dateien	A
tee	/usr/bin	bin	SINLIB.POSIX-SH	Pipes zusammenfügen und Eingabe kopieren	
test	/usr/bin	blt-scr	SINLIB.POSIX-BC.SHELL	Bedingungen prüfen	
time	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Laufzeit eines Kommandos messen	
times	-	blt	SINLIB.POSIX-BC.SHELL	Gesamtlaufzeit der bisher gestarteten Prozesse ausgeben	
touch	/usr/bin	blt+bin	SINLIB.POSIX-BC.SHELL	Änderungs- und Zugriffszeiten aktualisieren	A
tput	/usr/bin	bin	SINLIB.POSIX-SH	Datensichtstation initialisieren oder Datenbank terminfo abfragen	
tr	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Zeichen ersetzen oder löschen	A
trap	-	blt	SINLIB.POSIX-BC.SHELL	Signalbehandlung ändern	
true	/usr/bin	alias+scr	SINLIB.POSIX-BC.SHELL	Endestatus 0 zurückgeben	
tsort	/usr/bin	bin	SINLIB.POSIX-SH	Topologisch sortieren	
tty	/usr/bin	bin	SINLIB.POSIX-SH	Pfadnamen der aktuellen Datensichtstation ausgeben	
type	/usr/bin	alias+scr	SINLIB.POSIX-BC.SHELL	Typ eines Kommandos abfragen	

Name	Ort	Typ	Auslieferung	Beschreibung	LFS
typeset	-	blt	SINLIB.POSIX-BC.SHELL	Attribute für Shell-Variablen setzen	
ulimit	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	Datei-Größe für das Schreiben begrenzen oder Grenzwert abfragen	A
umask	/usr/bin	blt-scr	SINLIB.POSIX-BC.SHELL	Standard-Vergabe der Zugriffsrechte ausgeben oder ändern	
umount	/sbin	bin	SINLIB.POSIX-BC.ROOT	Dateisysteme und ferne Ressourcen aushängen	
umountall	/sbin	scr	SINLIB.POSIX-BC.ROOT	Aushängen mehrerer Dateisysteme	
unalias	/usr/bin	blt-scr	SINLIB.POSIX-BC.SHELL	Variablen aus der alias-Tabelle löschen	
uname	/sbin	bin	SINLIB.POSIX-BC.SHELL	Basisdaten über das aktuelle Betriebssystem ausgeben	
uncompress	/usr/bin	bin	SINLIB.POSIX-SH	Komprimierte Dateien dekomprimieren	A
unexpand	/usr/bin	bin	SINLIB.POSIX-SH	Leerzeichen in Tabulatorzeichen umwandeln	S
uniq	/usr/bin	bin	SINLIB.POSIX-BC.SHELL	Mehrfache Zeilen suchen	
uudecode	/usr/bin	bin	SINLIB.POSIX-SH	Datei nach der Übertragung per mailx decodieren	
uuencode	/usr/bin	bin	SINLIB.POSIX-SH	Datei für die Übertragung per mailx codieren	
uuname	/usr/bin	bin	SINLIB.POSIX-SH	Namen des Systems auflisten	
usp	/usr/bin	bin	SINLIB.POSIX-BC.ROOT	Dynamisches Setzen von POSIX-Steuerparametern	
vi	/usr/bin	bin	SINLIB.POSIX-SH	Bildschirmorientierter Editor	
wait	/usr/bin	blt+scr	SINLIB.POSIX-BC.SHELL	Auf die Beendigung von Hintergrundprozessen warten	
wc	/usr/bin	bin	SINLIB.POSIX-SH	Wörter, Zeichen und Zeilen zählen	A
whence	-	blt	SINLIB.POSIX-BC.SHELL	Kommando-Lokalisation	
who	/sbin	bin	SINLIB.POSIX-SH	Aktive Benutzerkennungen anzeigen	
write	/usr/bin	bin	SINLIB.POSIX-SH	Nachricht an einen Benutzer senden	
xargs	/usr/bin	bin	SINLIB.POSIX-SH	Argumentliste(n) aufbauen und Kommando ausführen	
yacc	/usr/bin	bin	SINLIB.POSIX-SH	Parser erstellen	
zcat	/usr/bin	bin	SINLIB.POSIX-SH	Komprimierte Dateien ausgeben	A

2.4.2 Kommando-Übersicht nach Funktionen

Nachfolgende Übersicht zeigt die funktionelle Einteilung der POSIX-Kommandos. Nachdem einige Kommandos mehreren Funktionen zugeordnet werden können, treten auch Doppelnennungen auf. Eine ausführliche Beschreibung der Kommandos in alphabetischer Reihenfolge finden Sie im Kapitel „Kommandos“ auf Seite 97.

Die Kommandos werden folgenden Hauptfunktionen zugeordnet:

- Kommando-Interpreter
- Benutzerumgebung abfragen bzw. ändern
- Dateien und Texte verwalten und bearbeiten
  - ausgeben
  - bearbeiten
  - sichern und archivieren
  - komprimiert speichern bzw. in Ursprungszustand zurückversetzen
  - Dateieigenschaften abfragen und ändern
- Dateisystem verändern und verwalten
  - Dateisystem verändern
  - Dateisystem verwalten
- Drucken und Druckerverwaltung
- Editoren
- Hilfskommandos für Shell-Prozeduren
- Zeichen einlesen, umwandeln und ausgeben
- Benutzereigenschaften abfragen und ändern
- Benutzer verwalten
- Kommunikation mit anderen Benutzern
- Kalenderfunktionen und Termine
- Rechenfunktionen
- Compilerkommandos
- Programmtest
- Auftragsverwaltung
- Informationen über Prozesse
- Prozesse steuern
- Interprozess-Kommunikation

- Datensichtstation
- Speicherplatzbelegung überprüfen
- Informationen über Systemdaten
- On-Line-Dokumentation
- Systempuffer leeren
- BS2000-Prozeduren aufrufen
- POSIX-Programm-Cache verwalten
- Netzkommandos
- NLS-Kommandos (Native Language System)

Kommando-Interpreter

sh	POSIX-Shell

Benutzerumgebung abfragen bzw. ändern

cd	Aktuelles Dateiverzeichnis wechseln
env	Umgebung bei Ausführung von Kommandos ändern
fuser	Dateinutzer anzeigen
id	Benutzer- und Gruppennummer und zugehörige Kennung ausgeben
last	Zuletzt angemeldete Benutzer anzeigen
logname	Login-Kennung abfragen
ls	Informationen über Dateiverzeichnisse und Dateien ausgeben
pwd	Pfadnamen des aktuellen Dateiverzeichnisses ausgeben
su	Benutzerkennung wechseln
tty	Pfadnamen der aktuellen Datensichtstation ausgeben
who	Aktive Benutzerkennungen anzeigen

Dateien und Texte verwalten und bearbeiten

- ausgeben

cat	Dateien aneinanderfügen und ausgeben
hd	Datei-Inhalt hexadezimal ausgeben
head	Anfang einer Datei ausgeben
more	Bildschirmausgabe steuern
od	Datei-Inhalt oktal ausgeben
pr	Dateien für Ausgabe aufbereiten
strings	Druckbare Zeichenketten in Objekt- oder Binärdateien suchen
tail	Letzten Teil einer Datei ausgeben
zcat	Komprimierte Dateien ausgeben

- bearbeiten

awk	Programmierbare Bearbeitung von Textdateien
cksum	Prüfsummen und Größen von Dateien schreiben
cmp	Dateien zeichenweise vergleichen
comm	Gleiche Zeilen in zwei sortierten Dateien suchen
csplit	Datei nach bestimmten Kriterien unterteilen
cut	Felder oder Spalten aus den Zeilen einer Datei herausschneiden
diff	Dateien zeilenweise vergleichen
edt, edtu	Datei mit EDT (BS2000) bearbeiten
egrep	Muster suchen
fgrep	Zeichenketten suchen
find	Dateiverzeichnisse durchsuchen
fold	Lange Zeilen zerlegen
grep	Muster suchen
join	Zwei Dateien nach Vergleichsfeldern verbinden
nl	Textzeilen nummerieren
paste	Zeilen zusammenfügen
patch	Diff-Liste anwenden

sort	Dateien sortieren und/oder mischen
split	Datei auf mehrere Dateien verteilen
sum	Prüfsumme einer Datei berechnen
tr	Zeichen ersetzen oder löschen
tsort	Topologisch sortieren
uniq	Mehrfache Zeilen suchen
wc	Wörter, Zeichen und Zeilen zählen

- sichern und archivieren

ar	Bibliotheken verwalten
cpio	Dateien und Dateiverzeichnisse ein- und auslagern
dd	Dateien kopieren und konvertieren
iconv	Code konvertieren
nm	Symboltabelle einer Objektdatei ausgeben
pax	Bearbeiten portierbarer Archive
tar	Archivieren von Dateien

- komprimiert speichern bzw. in Ursprungszustand zurückversetzen

compress	Dateien komprimieren
uncompress	Komprimierte Dateien expandieren
zcat	Komprimierte Dateien ausgeben

- Dateieigenschaften abfragen und ändern

chgrp	Gruppennummer einer Datei ändern
chmod	Zugriffsrechte ändern
chown	Eigentümer einer Datei ändern
file	Art einer Datei bestimmen
ls	Informationen über Dateiverzeichnisse und Dateien ausgeben
touch	Änderungs- und Zugriffszeiten aktualisieren
umask	Standardvergabe der Zugriffsrechte ändern

Dateisystem verändern und verwalten

- Dateisystem verändern

bs2cp	BS2000-Dateien kopieren
bs2file	BS2000-Dateiattribute festlegen
cp	Dateien kopieren
csplit	Datei nach bestimmten Kriterien unterteilen
find	Dateiverzeichnisse durchsuchen
fsexpand	Existierende Dateiverzeichnisse vergrößern
ftyp	Bearbeitungsart für BS2000-Dateien festlegen
ln	Verweis auf eine Datei eintragen
make	Gruppen von Dateien verwalten
mkdir	Dateiverzeichnis erzeugen
mv	Dateien versetzen oder umbenennen
rm	Dateien löschen
rmdir	Dateiverzeichnisse löschen
split	Datei auf mehrere Dateien verteilen

- Dateisystem verwalten

dumpfs	interne Dateisystem-Information ausgeben
fsck	Konsistenzprüfung des Dateisystems und Korrektur im Dialog mit dem Benutzer
mkfifo	FIFO erstellen
mknod	Gerätedatei anlegen
mount	Dateisystem einhängen
mountall	mehrere Dateisysteme einhängen
pathchk	Pfadnamen überprüfen
show-pubset_export	vom EXPORT-PUBSET betroffene Dateisysteme anzeigen
start_bs2fsd	Kopierdämönen starten
umount	Dateisystem aushängen
umountall	mehrere Dateisysteme aushängen

Drucken und Druckerverwaltung

asa	Steuerzeichen für die Positionierung umsetzen
bs2lp	Dateien ausdrucken
cancel	Druckaufträge löschen
lp	Dateien ausdrucken
lpstat	Informationen über Druckaufträge ausgeben
pr	Dateien für Ausgabe aufbereiten

Editoren

ed	Zeilenorientierter Editor im Dialogbetrieb
edt, edtu	Datei mit EDT (BS2000) bearbeiten
ex	Zeilenorientierter Editor
sed	Editor im Prozedurbetrieb
vi	Bildschirmorientierter Editor

Hilfskommandos für Shell-Prozeduren

basename	Dateinamen vom Pfad trennen
dirname	Pfad-Präfix vom Dateinamen trennen
expr	Ausdrücke auswerten
false	Endestatus ungleich 0 zurückgeben
getopts	Argumente einer Prozedur nach Optionen durchsuchen
pathchk	Pfadnamen überprüfen
sleep	Prozesse zeitweise stillegen
test	Bedingungen prüfen
true	Endestatus 0 zurückgeben
xargs	Argumentliste aufbauen und Kommando ausführen
[␣...␣]	Bedingungen prüfen (wie *test*)

Zeichen einlesen, umwandeln und ausgeben

echo	Aufruf-Argumente ausgeben
hd	Datei-Inhalt hexadezimal ausgeben
od	Datei-Inhalt oktal ausgeben
print	Ausgabemechanismus ähnlich echo
printf	Formatierte Ausgabe
tee	Pipes zusammenfügen und Eingabe kopieren

Benutzereigenschaften abfragen und ändern

id	Benutzer- und Gruppennummer und zugehörige Kennung ausgeben
logname	Login-Kennung abfragen
mesg	Nachrichtenempfang verbieten oder erlauben
newgrp	Gruppenzugehörigkeit ändern

Benutzer verwalten

fuser	Dateinutzer anzeigen
last	Zuletzt angemeldete Benutzer anzeigen
who	Aktive Benutzerkennungen anzeigen

Kommunikation mit anderen Benutzern

mailx	Nachrichten interaktiv bearbeiten
mesg	Nachrichtenempfang verbieten oder erlauben
talk	Dialog mit anderem Benutzer führen
write	Nachricht an einen Benutzer senden

Kalenderfunktionen und Termine

at	Kommandos zu einem späteren Zeitpunkt ausführen
batch	Kommandos zu einer späteren Zeit ausführen
cal	Kalender ausgeben
crontab	Kommandos regelmäßig zu bestimmten Zeitpunkten ausführen
date	Datum und Uhrzeit ausgeben

Rechenfunktionen

bc	Arithmetische Sprache
expr	Ausdrücke auswerten
let	Integer-Arithmetik

Compilerkommandos

lex	Scanner erstellen
yacc	Parser erstellen

Programmtest

debug	Testen von POSIX-Programmen

Auftragsverwaltung

bg	Jobs im Hintergrund bearbeiten
fg	Jobs in den Vordergrund bringen
jobs	Auftragsinformation ausgeben

Informationen über Prozesse

logger	Meldungen protokollieren
ps	Prozessdaten abfragen
time	Laufzeit eines Kommandos messen
times	Gesamt-Laufzeit der bisher gestarteten Prozesse ausgeben

Prozesse steuern

at / batch	Kommandos zu einer späteren Zeit ausführen
kill	Signale an Prozesse senden
nice	Priorität von Kommandos ändern
nohup	Kommando ausführen und dabei Signale ignorieren
renice	Priorität laufender Prozesse ändern
sleep	Prozesse zeitweise stillegen
wait	Auf die Beendigung von Hintergrund-Prozessen warten

Interprozess-Kommunikation

ipcrm	Einrichtungen zur Interprozess-Kommunikation löschen
ipcs	Zustand von Interprozess-Kommunikationseinrichtungen anzeigen

Datensichtstation

bs2pkey	P-Tasten belegen
expand	Tabulatorzeichen in Leerzeichen umwandeln
stty	Eigenschaften einer Datensichtstation ausgeben oder ändern
tabs	Tabulatorstops setzen
tput	Datensichtstation initialisieren oder Datenbasis *terminfo* abfragen
tty	Pfadnamen der aktuellen Datensichtstation ausgeben
unexpand	Leerzeichen in Tabulatorzeichen umwandlen

Speicherplatzbelegung überprüfen

df	Dateisystem auf freien Platz prüfen
du	Belegten Speicherplatz ausgeben

Informationen über Systemdaten

info	Online-Diagnosetool
pkginfo	Informationen über Software-Pakete anzeigen
ps	Prozessdaten abfragen
uname	Basisdaten über das aktuelle Betriebssystem ausgeben
who	Aktive Benutzerkennungen anzeigen

Online-Dokumentation

man	Online-Dokumentation nutzen

Systempuffer leeren

sync	Systempuffer zurückschreiben

BS2000-Prozeduren aufrufen

bs2do	BS2000-Prozeduren aus POSIX aufrufen

POSIX-Programm-Cache verwalten

pdbl	Privaten POSIX-Lader verwalten
posdbl	Verwalten des POSIX-Laders

Netzkommandos

ping	Senden von Echo-Request-Paketen an Netzwerkkomponenten
rcp	Dateien von oder zu einem fernen Rechner kopieren
rsh	Shell-Kommando am fernen Rechner ausführen
uudecode	Datei nach der Übertragung per *mailx* decodieren
uuencode	Datei für die Übertragung per *mailx* codieren
uuname	Namen von UUCP-Systemen auflisten

NLS-Kommandos (NATIVE LANGUAGE SYSTEM)

gencat	Binär codierten Meldungskatalog erzeugen
locale	Informationen über die internationale Umgebung abrufen
localedef	Internationale Umgebung definieren

Subsystem-Administration

usp	Dynamisches Setzen von POSIX-Steuerparametern

2.5 Beispiel

In diesem Abschnitt finden Sie ein Beispiel für das Arbeiten mit der POSIX-Shell.
Sie melden sich an das BS2000 an, lassen sich das Inhaltsverzeichnis Ihrer Benutzerkennung ausgeben und starten dann die POSIX-Shell.
In der POSIX-Shell erstellen Sie zuerst eine *.profile*-Datei, in der Sie zur Arbeitsvereinfachung Aliasvariablen und zur besseren Orientierung ein neues Bereitzeichen definieren, das den jeweils aktuellen Pfad ausgibt. Nach der Ausführung der *.profile*-Datei sind die dort getroffenen Definitionen wirksam.
Anschließend übertragen Sie eine Datei des BS2000-Dateisystems in das POSIX-Dateisystem und bearbeiten sie dort.

```
/set-logon-parameters user-id=user1,account=...  ————————————————————  (1)
/show-file-attributes  ——————————————————————————————————————————————  (2)
%     114 :1OSN:$USER1.ANHANG.V2
%       3 :1OSN:$USER1.AVASQUER
%      78 :1OSN:$USER1.BIB.EXAMPLES.SDF
%       6 :1OSN:$USER1.DO.MSGCHECK
%    5007 :1OSN:$USER1.FS.USER1
%       3 :1OSN:$USER1.MSG.PROT
%       3 :1OSN:$USER1.OUTPUT
%       3 :1OSN:$USER1.PROG.C
%       3 :1OSN:$USER1.SYS.SDF.LOGON.USERPROC
/start-posix-shell  ——————————————————————————————————————————————————  (3)
POSIX Basisshell 09.0A43 created Feb 20 2012
POSIX Shell 08.0A43 created Aug 02 2011
Copyright (C) Fujitsu Technology Solutions 2009
              All Rights reserved
Last login: Thu Jul 19 11:50:17 on term/001  ————————————————————————  (4)
$ edt .profile  ——————————————————————————————————————————————————————  (5)
```

(1) Melden Sie sich in gewohnter Weise an das BS2000 an.

(2) Lassen Sie sich mit dem BS2000-Kommando SHOW-FILE-ATTRIBUTES das Inhaltsverzeichnis Ihrer Benutzerkennung ausgeben.

(3) Rufen Sie die POSIX-Shell mit dem BS2000-Kommando START-POSIX-SHELL auf.

(4) Sie sind als POSIX-Shell-Benutzer akzeptiert.

(5) Erzeugen Sie die Datei *.profile* mit dem POSIX-Editor edt.
Da die Datei noch nicht vorhanden ist, legt *edt* eine neue Datei an (siehe Seite 89).

```
   1.00 alias ll='ls -l'
   2.00 alias la='ls -al'
   3.00 PS1='$PWD> '
   4.00
   5.00
   6.00
   7.00
   8.00
   9.00
  10.00
  11.00
  12.00
  13.00
  14.00
  15.00
  16.00
  17.00
  18.00
  19.00
  20.00
  21.00
  22.00
            POSIX editor ready for file .profile: new file
return                                                    0000.00:001(0)
LTG     EM:1                                             TAST
```

```
$ . .profile ──────────────────────────────────────────────────────────  (6)
/home/user1> la ───────────────────────────────────────────────────────  (7)
total 84
drwxr-xr-x   5 USER1    USROTHER    2048 Dec 22 14:03 .
drwxr-xr-x  63 SYSROOT  POSSYS      2048 Dec 22 06:35 ..
-rw-r--r--   1 USER1    USROTHER      48 Dec 22 14:02 .profile
-rw-------   1 USER1    USROTHER    2576 Dec 22 14:06 .sh_history
drwxr-xr-x   2 USER1    USROTHER    2048 Dec 15 17:18 c-source
drwxr-xr-x   2 USER1    USROTHER    8192 Dec  5 13:47 lost+found
-rw-r--r--   1 USER1    USROTHER      94 Dec 21 14:02 letter1
drwxr-xr-x   2 USER1    USROTHER    2048 Dec 19 15:05 test
  ...
/home/user1> cd c-source ──────────────────────────────────────────────  (8)
```

(6) Nach der Erstellung der *.profile*-Datei mit dem *edt* und Verlassen des Editors mit dem Kommando *return* soll die *.profile*-Datei in der aktuellen Shell ausgewertet werden. Dazu geben Sie . *.profile* ein.

(7) Die POSIX-Shell meldet sich mit dem neu definierten Bereitzeichen, das den aktuellen Pfad */home/user1* ausgibt. Sie lassen sich das Inhaltsverzeichnis mit allen Dateien über das mit dem Aliasnamen *la* definierte Kommando anzeigen.

(8) Wechseln Sie in das Unterverzeichnis *c-source*, in dem Sie beispielsweise Ihre C-Programme speichern.

```
/home/user1/c-source> bs2cp bs2:prog.c prog.c ———————————————————— (9)
/home/user1/c-source> la
total 60
drwxr-xr-x  2  USER1      USER1GRP     2048   Jul 6 .
drwxr-xr-x  2  USER1      USER1GRP     2048   Jul 6 ..
-rw-r--r--  1  USER1      USER1GRP     2048   Jul 6 prog.c
/home/user1/c-source> cat prog.c ———————————————————————————— (10)
#include <stdio.h>
main()
{
  printf("hello world\n");
  return(0);
}
/home/user1/c-source> cc -o prog prog.c ————————————————————— (11)
/home/user1/c-source> prog ———————————————————————————————— (12)
hello world
/home/user1/c-source> exit ———————————————————————————————— (13)
.... ————————————————————————————————————————————————————— (14)
/exit-job ————————————————————————————————————————————————— (15)
```

(9) Kopieren Sie die im BS2000-Dateisystem liegende Datei *prog.c* in das POSIX-Dateisystem. Die Datei wird in das aktuelle Verzeichnis *c-source* geschrieben.

(10) Lassen Sie sich den Inhalt der Datei *prog.c* mit *cat* ausgeben.

(11) Übersetzen Sie die Datei *prog.c* mit dem C-Compiler. Das Ergebnis des Übersetzungslaufs soll in die Datei *prog* geschrieben werden.

(12) Lassen Sie das Programm *prog* ablaufen. Es gibt die Zeichenfolge „hello world“ auf dem Bildschirm aus.

(13) Beenden Sie mit dem Kommando *exit* die POSIX-Shell.

(14) Eingabe von weiteren BS2000-Kommandos, falls gewünscht.

(15) Melden Sie sich am BS2000 ab.

3 Internationale Umgebung - NLS

3.1 Definition des NLS

Die meisten UNIX-System basierten bisher auf dem ASCII-Zeichensatz und auf dem amerikanischen Englisch als Sprache, in der der Benutzer mit dem Rechner kommunizieren konnte. Im kommerziellen Bereich mussten aber auch bisher schon interaktive Programme in der Sprache des jeweiligen Benutzers angeboten werden (mit Sprache sind in diesem Sinn auch regional übliche Konventionen wie z.B. Währungsformat gemeint). So wurden sogenannte Ländervarianten von Programmen für die einzelnen nationale Märkte produziert.

Die zunehmende internationale Verbreitung von UNIX-Systemen erforderte jedoch eine Erweiterung, die den unterschiedlichen Schriften, Sprachen und den länderspezifischen Konventionen der unterschiedlichen Benutzer auf flexiblere Weise gerecht wird.

Mit dem NLS (Native Language System) hat X/Open eine Schnittstelle definiert, die es ermöglicht,

– Anwendungsprogramme zu entwickeln, die in verschiedenen Landessprachen mit dem Benutzer kommunizieren und den dazugehörigen länderspezifischen Konventionen entsprechen. Solche Programme machen keine Annahmen über die Sprache des Benutzers und halten die Datenspezifikationen getrennt von der Programmlogik. Sie werden als *internationalisierte Programme* bezeichnet.

– die Ablaufumgebung eines internationalisierten Anwendungsprogramms zur Laufzeit mit der richtigen Landessprache und den entsprechenden länderspezifischen Konventionen zu versorgen. Dieses Verfahren nennt man *Lokalisierung*. Daher wird die Menge der länder- bzw. sprachspezifischen Konventionen als *Locale* bezeichnet.

Das NLS umfasst dafür insbesondere:

– Einen Bindungsmechanismus, der es dem Benutzer erlaubt, die Landessprache, die Verarbeitung nach regionalen Konventionen und den zugrundeliegenden Zeichensatz zur Laufzeit eines Anwendungsprogramms nach individuellem Bedarf einzustellen. Dies erfolgt mit Hilfe von Umgebungsvariablen.

- Meldungskataloge, die es erlauben, Meldungen eines Anwendungsprogramms getrennt von der Programmlogik zu halten, in verschiedene Landessprachen zu übersetzen und zur Laufzeit an die Anwendung zu binden.
- Internationalisierte C-Bibliotheksfunktionen, die keine Annahmen über Landessprache, Land und Zeichensatz machen und deshalb geeignet sind, universelle Zeichenklassifizierung, Umwandlung von Groß-/Kleinbuchstaben und Zahlenformaten durchzuführen sowie Zeichenketten zu sortieren.
- Einen Satz von C-Bibliotheksfunktionen, die es erlauben, zur Laufzeit eines Anwendungsprogramms die persönliche Sprachumgebung einzustellen, zu verändern und wieder abzustellen.

Alle POSIX-Kommandos, die in diesem Handbuch beschrieben sind, sind 8-bit-transparent. Dies schließt alle auf der Kommandozeile angegebenen Argumente und alle Daten und Zeichen, die von diesen Kommandos verarbeitet werden, ein.

Es bestehen jedoch weiterhin Einschränkungen in der Portabilität von 8-bit-Daten zwischen POSIX und anderen Systemen:

- Der Austausch von Daten zwischen Systemen über elektronische Mail-Verbindungen kann durch die eingesetzten Mail- oder Netzprotokolle auf 7-bit-Daten begrenzt sein.
- 8-bit-Daten und 8-bit-Dateinamen sind möglicherweise nur auf Systeme portierbar, die den Anforderungen der X/Open-Systemschnittstelle genügen.

Detaillierte Informationen zu NLS finden Sie im „Leitfaden für Programmierer: Internationalisierung - Lokalisierung“ [14].

3.2 Einstellen der internationalisierten Umgebung

3.2.1 Die persönliche internationalisierte Umgebung

POSIX unterstützt Bindungsmechanismen, die es erlauben, die benötigte Sprache, nationale Konventionen und den Zeichensatz sowohl für das gesamte System als auch individuell für einzelne Benutzer oder Benutzergruppen zu setzen.

Eine bestimmte Arbeitsumgebung wird gesetzt, indem der Name der gewünschten Locale einem Satz von Umgebungsvariablen zugewiesen wird, die für diesen Zweck reserviert wurden. Bei jedem Aufruf eines internationalisierten Programms werden diese Umgebungsvariablen gelesen und die Informationen der zugewiesenen Locale werden zu der Ablaufumgebung des Programms gebunden.

Die relevanten Umgebungsvariablen sind:

Variable	**Einfluss auf:**
LANG	Gesamte internationale Umgebung
LC_ALL	Gesamte internationale Umgebung
LC_CTYPE	Zeichenklassen und Umwandlung von Klein- in Großbuchstaben und umgekehrt
LC_COLLATE	Sortierreihenfolge
LC_TIME	Datums- und Zeitangaben
LC_MONETARY	Währungszeichen und Währungsformat
LC_NUMERIC	Dezimalpunktdarstellung, Exponentzeichen und Tausendertrennzeichen
LC_MESSAGES	Meldungstexte und Antworten auf ja/nein-Fragen

Die Prioritäten der einzelnen Variablen finden Sie im Abschnitt „Prioritäten der Umgebungsvariablen“ auf Seite 95.

Diese Variablen definieren Sie in einem der folgenden Formate:

Format 1: Variable=locale-name

Format 2: Variable=sprache[_territorium][.zeichensatz]

locale-name
sprache[_territorium][.zeichensatz]

Name eines Dateiverzeichnisses unter */usr/lib/locale*. Die Länge dieser Zeichenkette darf *{NL_LANGMAX}* nicht überschreiten.

Darüber hinaus können Sie für die Variablen *LC_CTYPE*, *LC_COLLATE*, *LC_TIME*, *LC_MONETARY*, *LC_NUMERIC* und *LC_MESSAGES* durch die zusätzliche Angabe von *@parameter* weitere Alternativen nutzen, beispielsweise für eine bestimmte Art der Sortierreihenfolge:

LC_Variable=sprache_territorium.zeichensatz@parameter

Beispiel

Bei der Angabe der Sortierreihenfolge durch *LC_COLLATE* kann in der deutschen Sprache durch die Angabe von *@parameter* noch zwischen der Standard-Sortierreihenfolge (wie im Duden) und der Sortierreihenfolge des Telefonbuchs gewählt werden. Die verschiedenen Sortierregeln unterscheiden sich durch die Reihenfolge von Groß- und Kleinbuchstaben, Umlauten und Sonderzeichen. Zur Auswahl der Sortierreihenfolge des Telefonbuchs geben Sie an:

```
LC_COLLATE=De_DE.88591@TE
```

Ist eine der Variablen *LC_CTYPE, LC_COLLATE, LC_TIME, LC_MONETARY, LC_NUMERIC, LC_MESSAGES* nicht definiert oder ist ihr die leere Zeichenkette zugewiesen, dann gilt für diese Variable als Standardwert der Wert von *LANG*.

Hat eine der Variablen *LC_CTYPE, LC_COLLATE, LC_TIME, LC_MONETARY, LC_NUMERIC, LC_MESSAGES* einen ungültigen Wert oder ist die Variable *LANG* nicht definiert bzw. leer, dann verhält sich das System, als wäre es nicht internationalisiert, d.h. es wird dann nach der Reihenfolge der Zeichen im ASCII-Zeichensatz sortiert; es gelten das amerikanische Datumsformat, englische Wochentags- und Monatsnamen etc. Dies ist die Standardeinstellung.

3.2.2 Prioritäten der Umgebungsvariablen

Die Umgebungsvariable *LC_ALL* stellt ebenso wie die Umgebungsvariable *LANG* einen allgemeinen Bindungsmechanismus für die gesamte Locale dar und verwendet die gleiche Syntax. Im Unterschied zu *LANG* besitzt die Umgebungsvariable *LC_ALL* jedoch höchste Priorität und hat den Vorrang vor allen anderen Umgebungsvariablen im Bereich der Internationalisierung. Das Setzen von *LC_ALL* ist dann ausreichend, wenn eine vorgegebene Locale den gesamten Bedürfnissen der Benutzer an die internationale Umgebung genügt. Die Umgebungsvariable *LANG* dagegen besitzt die niedrigste Priorität. Durch setzen weiterer internationaler Umgebungsvariablen kann die Arbeitsumgebung einzelner Benutzer individuell an die jeweiligen Bedürfnisse angepasst werden.

Priorität	**Umgebungsvariable**
Hoch	LC_ALL
Mittel	LC_CTYPE LC_COLLATE LC_TIME LC_MONETARY LC_NUMERIC LC_MESSAGES
Niedrig	LANG

Weder *LANG* noch *LC_ALL* decken beispielsweise den Fall ab, dass ein Benutzer in einer Sprache mit dem System kommunizieren und gleichzeitig in einer anderen Sprache Textdateien sortieren will. In solch einem Fall definiert der Benutzer jene Umgebungsvariablen, die die Veränderung einzelner Aspekte (Kategorien) der internationalen Umgebung ermöglichen.

Möchte also z.B. ein Benutzer mit dem System in Deutsch kommunizieren, daneben aber englische Textdateien sortieren, so muss er die Umgebungsvariablen *LANG* und *LC_COLLATE* wie folgt setzen:

```
LANG=De
LC_COLLATE=C
```

Durch das Setzen weiterer Variablen kann sich der Benutzer eine multilinguale Arbeitsumgebung gestalten.

Beachten Sie, dass die Umgebungsvariable *LC_ALL* in diesem Fall nicht eingesetzt werden kann, da sie Vorrang vor allen internationalen Umgebungsvariablen hat. Die Angabe von *LC_COLLATE* wird in Verbindung mit *LC_ALL* nicht ausgewertet und hätte keine Auswirkungen.

3.2.3 Ausgelieferte Lokalitäten

POSIX stellt verschiedene vordefinierte Lokalitäten zur Verfügung:

- C
- De
- De.EDF04F
- De_DE.EDF04
- En_US.EDF04
- De.EDF04F@euro
- De_DE.EDF04@EU
- POSIX

Die „POSIX“- bzw. die „C“-Lokalität entspricht den im XPG4-Standard festgelegten Eigenschaften.

Die „De“-Lokalität ist deutschsprachig, die Sortierung erfolgt nach der Standardsortierung (wie im Duden).

3.2.4 Einschränkungen

Kommandos, die nicht im XPG4-Standard enthalten sind (siehe auch Seite 937), werden nicht internationalisiert.

4 Kommandos

alias Alias-Namen definieren oder anzeigen (define or display aliases)

Mit dem in die POSIX-Shell *sh* eingebauten Kommando *alias* können Sie eingebaute Shell-Kommandos neu definieren, Sie können sie aber nicht zur Neudefinition der reservierten Wörter verwenden. Mit Hilfe des *alias*-Kommandos können Alias-Variablen definiert, exportiert und zum Auflisten auf die Standard-Ausgabe geschrieben werden, während sie mit dem *unalias*-Kommando wieder gelöscht werden können. Exportierte Alias-Variablen bleiben wirksam für Prozeduren, die per Name aufgerufen werden, müssen aber neu für explizite weitere Aufrufe der POSIX-Shell initialisiert werden.
Zur Verwendung von Alias-Variablen siehe Abschnitt „Alias-Variablen“ auf Seite 47.

Syntax

Format 1: alias[␣**-t**][␣name]

Format 2: alias[␣**-x**][␣name**=**[wert]]...

Format 1 **alias**[␣**-t**][␣name]

Keine Option angegeben
alias schreibt die Alias-Tabelle in der Form *name=wert* auf die Standard-Ausgabe.

-t wird zum Setzen und Auflisten von mit Pfad versehenen Alias-Variablen (tracked alias) verwendet. Der *wert* einer solchen Variablen besteht aus dem vollen Pfadnamen, der zu *name* führt. Durch Ändern des Wertes von *PATH* wird *wert* undefiniert, *name* bleibt aber mit Pfad. Bei fehlender Option *-t* wird für jeden Namen ohne Wert in der Argumentliste das Paar *name=wert* ausgegeben.

-t nicht angegeben:
alias gibt für jeden Namen ohne Wert in der Argumentliste das Paar *name=wert* aus.

Format 2 **alias**[␣**-x**][␣name**=**[wert]]...

Ein Leerzeichen am Ende von *wert* bedeutet, dass bei der Kommandoabarbeitung auch das nächste Wort in der Kommandozeile auf Alias-Ersetzung überprüft wird.

-x wird zum Setzen und Ausgeben von exportierten Alias-Variablen verwendet. Eine exportierte Alias-Variable ist für Prozeduren definiert, die mit ihrem Namen aufgerufen werden.

Endestatus ungleich 0, wenn für *name* kein *wert* definiert ist.

Variable *PATH*
Standard-Variable der POSIX-Shell. Der Variablen sind die absoluten Pfadnamen der Dateiverzeichnisse zugewiesen, in denen die POSIX-Shell nach Kommandos suchen soll.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *alias*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Mit den folgenden Alias-Namen können Sie das *ls*-Kommando vereinfachen:

```
$ alias ll='ls -l'
$ alias la='ls -al'
```

Siehe auch *unalias*

ar Bibliotheken verwalten (create and maintain library archives)

ar verwaltet Bibliotheken.

Im Einzelnen können Sie mit *ar* folgende Verwaltungsaufgaben erledigen. Zu jeder Verwaltungsaufgabe ist/sind die entsprechende(n) Hauptoption(en) in der Liste angegeben:

- Bibliothek erstellen: -q oder -r
- Datei schnell in die Bibliothek eintragen : -q
- Datei ersetzen bzw. eintragen: -r
- Datei aus der Bibliothek entfernen: -d
- Datei in der Bibliothek verschieben: -m
- Dateiinhalt ausgeben: -p
- Inhaltsverzeichnis der Bibliothek ausgeben: -t
- Datei aus der Bibliothek herauskopieren : -x
- Symbole ausgeben: -S

Syntax **ar**[␣**-V**]␣hauptoption[zusatzoption]...[position]␣bibliothek[␣datei]...

-V *ar* gibt seine Versionsnummer auf die Standard-Fehlerausgabe aus.

Hauptoptionen

Bei einem *ar*-Aufruf müssen Sie genau eine Hauptoption angeben (*-d, -m, -p, -q, -r, -t*, *-x* oder *-S*). Der Hauptoption können eine oder mehrere Zusatzoptionen folgen.

-d (d - delete) *ar* entfernt die angegebenen Dateien aus der Bibliothek. Sind keine Dateien angegeben, wird *keine* Datei entfernt.

-m (m - move) *ar* verschiebt die angegebenen Dateien innerhalb der Bibliothek.

Mit *position*:
Die Dateien werden hinter (*a*) bzw. vor (*b* oder *i*) der Datei *posdatei* eingefügt (siehe *position*).

Ohne *position*:
Die Dateien werden am Ende der Bibliothek eingefügt.

-p (p - print) *ar* gibt den Inhalt der angegebenen Dateien aus. Sind keine Dateien angegeben, gibt *ar* den Inhalt aller Dateien aus.

-q (q - quickly) *ar* trägt die angegebenen Dateien „schnell“ in die Bibliothek ein, d.h.:

- Wenn die Bibliothek bereits existiert, hängt *ar* die Dateien ans Ende der Bibliothek an, ohne zu überprüfen, ob die Dateien bereits in der Bibliothek vorhanden sind.
- Wenn die Bibliothek noch nicht existiert, wird sie erstellt.

Angaben für *position* sind nicht erlaubt.

Die Option *-q* ist sinnvoll, wenn Sie große Bibliotheken schrittweise erstellen wollen.

-r (r - replace) Diese Option hat drei verschiedene Wirkungen, abhängig davon, ob die angegebene Bibliothek existiert und die angegebenen Dateien enthält:

- Wenn die Bibliothek existiert und die Dateien enthält, ersetzt *ar* die angegebenen Dateien.
- Wenn die Bibliothek existiert und eine der angegebenen Dateien nicht enthält, trägt *ar* diese Datei in die Bibliothek ein.
- Wenn die Bibliothek noch nicht existiert, erstellt *ar* die Bibliothek aus den angegebenen Dateien.

Mit Zusatzoption *u*:

ar ersetzt eine Datei in der Bibliothek nur dann, wenn die beim *ar*-Aufruf angegebene Datei neueren Datums ist als die Version, die in der Bibliothek steht.
(Das Datum bezieht sich auf den Zeitpunkt der letzten Änderung.)

Mit *position*:
Dateien, die noch nicht in der Bibliothek enthalten sind, werden hinter (*a*) bzw. vor (*b* oder *i*) der Datei *posdatei* eingefügt (siehe „position“ auf Seite 101).

Ohne *position*:
Dateien, die noch nicht in der Bibliothek enthalten sind, werden am Ende der Bibliothek eingefügt.

-t (t - table) *ar* gibt ein Inhaltsverzeichnis der Bibliothek aus. Sind keine Dateien angegeben, listet *ar* alle Dateien der Bibliothek auf, sonst nur die angegebenen Dateien.

-x (x - extract) *ar* kopiert die angegebenen Dateien aus der Bibliothek heraus. Sind keine Dateien angegeben, kopiert *ar* alle Dateien heraus. Die Bibliothek wird dadurch nicht verändert.

-S (S – Symbols) *ar* listet alle Symbole der angegebenen Dateien auf. Sind keine Dateien angegeben, listet *ar* alle Symbole auf, die in der Bibliothek enthalten sind. Die Bibliothek wird dadurch nicht verändert.

Zusatzoptionen

c (c - create) Die Standard-Meldung von *ar* beim Erstellen einer Bibliothek wird unterdrückt.

C (C - Create) Herauskopierte Dateien werden nicht durch gleichnamige Datein im Dateisystem überschrieben. Diese Option ist im Zusammenhang mit der Zusatzoption -T sinnvoll, damit dort verkürzte Dateinamen nicht mit dem gleichen Präfix im Dateisystem abgelegt werden.

T (T - truncate) Verkürzen des Dateinamens. Wenn Dateien aus einer Bibliothek herauskopiert werden, können ihre Namen länger sein, als das Dateisystem zulässt. Standardmäßig wird ein Fehler ausgegeben und die Datei nicht herauskopiert, wenn der Dateiname zu lang ist.

l (l - local) Kann angegeben werden, wird aber ignoriert.

s (s - symbol table) *ar* erstellt die Symboltabelle der Bibliothek neu, jedoch nur dann, wenn neue Objekte in die Bibliothek aufgenommen bzw. bestehende Objekte ersetzt wurden.

u (u - update) Siehe Hauptoption *-r*.

v (v - verbose) *ar* meldet, welche Datei es gerade in die Bibliothek einträgt, in der Bibliothek verschiebt, aus der Bibliothek entfernt usw.. Wenn Sie *v* zusammen mit *-t* verwenden, gibt *ar* ähnlich wie *ls -l* (siehe *ls*) ausführliche Informationen über die Dateien aus.

position

position legt fest, wo eine Datei in die Bibliothek eingefügt wird. *position* kann sein:

a␣posdatei
ar fügt die Dateien hinter *posdatei* ein.

b␣posdatei
i␣posdatei

ar fügt die Dateien vor *posdatei* ein.

posdatei ist der Name einer Datei, die in der Bibliothek steht.

bibliothek

Name der Bibliothek, die erstellt bzw. bearbeitet werden soll.

datei

Name der Datei, die aufgelistet, in die Bibliothek eingetragen, aus der Bibliothek entfernt, aus der Bibliothek herauskopiert, in der Bibliothek verschoben bzw. deren Inhalt ausgegeben werden soll.
Sie können mehrere Dateien angeben. Geben Sie bei einem *ar*-Aufruf mit Hauptoption *-q* oder *-r* dieselbe Datei mehrmals an, trägt *ar* diese Datei auch mehrmals in die Bibli-

othek ein. Wenn mehrere Dateien mit gleichem Namen in der Bibliothek enthalten sind, und Sie *ar* ohne Positionsangabe verwenden, wird auf die erste Datei mit diesem Namen zugegriffen (und nicht z.B. auf die neueste oder älteste etc).

Aufbau einer Bibliothek

Eine Bibliothek ist eine Datei, in der mehrere Dateien zusammengefasst sind. Gemäß Konvention endet der Name einer Bibliothek mit *.a*. Der Sinn einer Bibliothek besteht darin, Dateien zusammenzufassen, die bzgl. ihrer Verwendung eine Gruppe bilden. Die Verwendung von Bibliotheken erleichtert die Verwaltung der Dateien, da Sie oft nur die Bibliothek angeben müssen, statt alle Elemente einzeln aufzuführen.
Sehr häufig sind die Elemente einer Bibliothek Objektmodule, die üblicherweise zu einem Programm oder Programmsystem gehören.

Die Magic-Zeichenkette (magic string) und der Dateivorspann (header), die *ar* benutzt, bestehen aus druckbaren ASCII-Zeichen. Enthält die Bibliothek nur druckbare Dateien, so ist die ganze Bibliothek druckbar.
Wenn *ar* eine Bibliothek erzeugt, dann erzeugt es den Dateivorspann (header) in einem Format, das über alle Rechner portabel ist.

Enthält die Bibliothek mindestens eine Objektdatei, erstellt und verwaltet *ar* eine Symboltabelle. Der Binder *ld* benutzt die Symboltabelle, um bei mehrfach notwendigem Durchlaufen der Bibliothek dieses Durchlaufen zu beschleunigen. Die Symboltabelle steht in einer eigenen Datei, die immer die erste Datei in der Bibliothek ist. Auf diese Datei können Sie nicht zugreifen.

Es wird so weit als möglich versucht, nur im virtuellen Arbeitsspeicher zu arbeiten und das Auslagern von Elementen in temporäre Dateien zu vermeiden. Gelingt dies nicht, so werden temporäre Dateien angelegt, was aber nur im Falle der Hauptoption *–m* notwendig ist. Für den Ablageort der temporären Dateien gelten dann die folgenden Prioritäten:

1. das aktuelle Verzeichnis
2. das in der Variablen *TMPDIR* angegebene Verzeichnis
3. das Verzeichnis */tmp*

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *ar*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_TIME Legt das Format der Datums- und Zeitangaben bei der Auflistung des Archivinhalts mit der Option *v* fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Eine Bibliothek schnell erstellen:

```
$ ar -qv archiv.a atoi.o itoa.o
ar: creating archiv.a
a - atoi.o
a - itoa.o
```

ar erstellt die Bibliothek *archiv.a* aus den Dateien *atoi.o* und *itoa.o*. Die Zusatzoption *v* bewirkt, dass *ar* die Namen der eingetragenen Dateien ausgibt.

Beispiel 2 Eine neue Datei an einer angegebenen Stelle in eine Bibliothek einfügen:

```
$ ar -rvb atoi.o archiv.a atof.o
a - atof.o
```

ar kopiert *atof.o* vor *atoi.o* in die Bibliothek *archiv.a*.

Beispiel 3 Inhaltsverzeichnis der Bibliothek ausgeben:

```
$ ar -tv archiv.a
rw-r--r-- 104/     1      2276 Jul 13 12:17 2008 atof.o
rw-r--r-- 104/     1       759 Jul 13 12:17 2008 atoi.o
rw-r--r-- 104/     1      1280 Jul 13 12:17 2008 itoa.o
```

Beispiel 4 Eine Datei aus der Bibliothek holen:

```
$ ar -xv archiv.a atoi.o
x - atoi.o
```

Die Datei *atoi.o* wird aus der Bibliothek *archiv.a* in das aktuelle Dateiverzeichnis kopiert. Die Kopie heißt auch *atoi.o*.

Siehe auch *cpio*, *pax*, *tar*

asa Steuerzeichen für die Positionierung umsetzen (interpret carriage-control characters)

asa liest Eingabedateien, setzt Steuerzeichen für die Positionierung in Steuersequenzen für einen zeilenorientierten Drucker um und gibt die veränderten Dateien auf die Standard-Ausgabe aus. Als Steuerzeichen für die Positionierung wird jeweils das erste Zeichen jeder Zeile ausgewertet und aus der Eingabe entfernt. Folgende Zeichen werden von *asa* erkannt und in Druckeranweisungen umgesetzt:

leerzeichen	Die Zeile wird ohne Veränderung ausgegeben.
0	Vor der Zeile wird ein Neue-Zeile-Zeichen ausgegeben.
1	Vor der Zeile wird eine Steuerseqenz für den Seitenwechsel ausgegeben.
+	Das Neue-Zeile-Zeichen der vorhergehenden Zeile wird durch ein Wagenrücklauf-Zeichen ersetzt und danach die aktuelle Zeile ausgegeben. Ist + das erste Zeichen der Eingabe, wird es wie ein Leerzeichen behandelt.

Syntax **asa**[␣datei ...]

datei
Pfadname einer Textdatei, die als Eingabe verwendet wird.
Ist *datei* nicht angegeben, liest *asa* von der Standard-Eingabe.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *asa*:

LANG	Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.
LC_ALL	Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.
LC_CTYPE	Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).
LC_MESSAGES	Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.
NLSPATH	Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *lp*

at Kommandos zu einem späteren Zeitpunkt ausführen (execute commands at a later time)

Das Kommando *at*
- liest Kommandos von der Standard-Eingabe oder aus einem Shell-Skript und führt sie zu einem späteren Zeitpunkt, den Sie angeben, aus. Eine so entstandene Kommandoliste läuft unter einer Auftragsnummer (Format 1 und 2)
- gibt auf die Standard-Ausgabe aus, welche mit *at* oder *batch* (siehe *batch*) erteilten Kommandoaufträge noch nicht bearbeitet wurden (Format 3 und 4)
- löscht Kommandoaufträge, die mit *at* oder *batch* erteilt wurden (Format 5)

Wenn Sie die Standard-Ausgabe und Standard-Fehlerausgabe der auszuführenden Kommandos nicht umgelenkt haben, wird Ihnen die Ausgabe von Format 1 und Format 2 mit *mailx* geschickt.

Die Umgebungsvariablen, das aktuelle Dateiverzeichnis, die für neue Dateien gültigen Zugriffsrechte (siehe *umask* auf Seite 834) und die maximal zulässige Dateigröße (siehe *ulimit* auf Seite 831) bleiben erhalten. Offene Dateien und Prioritäten werden nicht vererbt. Das Kommando *trap* (eingebautes Shell-Kommando zum Abfangen von Signalen) wird aufgehoben.
at schreibt die Auftragsnummer und die angegebene Ausführungszeit auf die Standard-Fehlerausgabe.

Aufträge, die mit *at* erteilt werden, bleiben auch dann erhalten, wenn der Auftraggeber mit *exit* die POSIX-Shell beendet oder das POSIX-Subsystem beendet wird. Die Aufträge müssen nicht neu gestartet werden.

Vor dem Aufruf beachten

Die Benutzerkennung muss eine Standardabrechnungsnummer für den *rlogin*-Zugang haben. Diese kann mit den Kommandos ADD-USER, MODIFY-USER-ATTRIBUTES oder ADD-POSIX-USER zugewiesen werden.

Wenn die Datei */usr/lib/cron/at.allow* existiert, dann dürfen Sie das Kommando *at* nur dann aufrufen, wenn Ihre Benutzerkennung in dieser Datei steht.

Wenn die Datei */usr/lib/cron/at.allow* nicht existiert, dann dürfen Sie das Kommando *at* nur dann aufrufen, wenn Ihre Benutzerkennung *nicht* in der Datei */usr/lib/cron/at.deny* steht.

Wenn weder */usr/lib/cron/at.allow* noch */usr/lib/cron/at.deny* existieren, dann darf nur der POSIX-Verwalter *at* aufrufen.

Existiert z.B. nur die leere *deny*-Datei, so dürfen alle Benutzer *at* aufrufen.

Die *allow*/*deny*-Dateien darf nur der POSIX-Verwalter anlegen und ändern. Sie enthalten pro Zeile eine Benutzerkennung.

Der *cron*-Dämon wird über ein rc-Skript gestartet.

Syntax

Format 1: at[␣**-m**][␣**-f**␣skript][␣**-q**␣warteschlange]␣**-t**␣zeit

Format 2: at[␣**-m**][␣**-f**␣skript][␣**-q**␣warteschlange]␣zeitpunkt

Format 3: at␣**-l**[␣auftragsnummer]

Format 4: at␣**-l**␣**-q**␣warteschlange

Format 5: at␣**-r**␣auftragsnummer

Kommandos zu einem späteren Zeitpunkt ausführen

Format 1 **at**[␣**-m**][␣**-f**␣skript][␣**-q**␣warteschlange]␣**-t**␣zeit

-f␣skript

at liest die auszuführenden Kommandos aus dem angegebenen Shell-Skript.
Sie können mehrere Kommandos angeben, jeweils durch Strichpunkt ; oder Neue-Zeile-Zeichen getrennt. Eine so entstandene Kommandoliste läuft unter einer Auftragsnummer. Die Kommandoliste beenden Sie, abhängig vom verwendeten Terminal, mit @ @d oder END.

-f nicht angegeben:
at liest die auszuführenden Kommandos von der Standard-Eingabe.

-m Wenn sich ein Auftrag beendet hat, schickt *at* dem Benutzer über *mailx* eine entsprechende Nachricht. Dies geschieht jedoch nur dann, wenn der Auftrag nicht bereits das Senden einer Nachricht veranlasst hat.

-q␣warteschlange

Durch die Option *-q* wird die Warteschlange im Dateiverzeichnis */var/spool/cron* spezifiziert, in die der Auftrag eingereiht werden soll.

warteschlange kann sein:

a für die standardmäßige Warteschlange für Aufträge des Kommandos *at*.

b für die standardmäßige Warteschlange für Aufträge des Kommandos *batch*.

c für die standardmäßige Warteschlange für Aufträge des Kommandos *crontab*.

-t␣zeit

Gibt den Ausführzeitpunkt für die Kommandos an. *zeit* geben Sie wie folgt an:

[[CC]YY]MMDDhhmm[.SS]

CC Die ersten beiden Ziffern einer Jahreszahl (Jahrhundert).
CC nicht angegeben:
Ist die zweistellige Jahreszahl größer 68, wird das aktuelle Jahrhundert angenommen, sonst das folgende.

YY Jahreszahl zweistellig.
YY nicht angegeben:
Das aktuelle Jahr wird angenommen.

MM Monatsangabe zweistellig (01 bis 12)

DD Angabe des Tages zweistellig (01 bis 31)

hh Angabe der Stunde zweistellig (00 bis 23)

mm Angabe der Minute zweistellig (00 bis 59)

SS Angabe der Sekunde zweistellig (00 bis 61)
Die Werte 60 und 61 sind für Schaltsekunden vorgesehen..
SS nicht angegeben:
Es werden 0 Sekunden angenommen.

Format 2 **at**[␣**-m**][␣**-f**␣skript][␣**-q**␣warteschlange]␣zeitpunkt

Die Optionen sind unter Format 1 beschrieben.

zeitpunkt
Gibt den Ausführzeitpunkt für die Kommandos an. *zeitpunkt* geben Sie wie folgt an:

zeit[␣datum][␣+inkrement]

zeit: ziffern[suffix] oder sondername

ziffern:

[h]h Eine und zwei Ziffern werden als Stunden interpretiert.

hhmm Vier Ziffern werden als Stunden und Minuten interpretiert.

[h]h**:**[m]m Durch Doppelpunkt : getrennte Ziffern werden als Stunden und Minuten interpretiert.

suffix:

am Interpretation als vor 12 Uhr mittags (abhängig von der Umgebungsvariablen *LC_TIME*)

pm Interpretation als nach 12 Uhr mittags (abhängig von der Umgebungsvariablen *LC_TIME*)

ohne am, pm
Interpretation im 24-Stunden-Format

zulu[**am**][**pm**]
Interpretation als Greenwich Meantime

sondername
noon - mittags
midnight - mitternachts
now - jetzt

datum:
monat␣tag[,jahr] oder
wochentag[␣**nextweek**|**next**␣**week**] oder
spezieller tag

monat (abhängig von der Umgebungsvariablen *LC_TIME*)
jan[**uary**] (Januar), **feb**[**ruary**] (Februar), **mar**[**ch**] (März), **apr**[**il**] (April **may** (Mai), **jun**[**e**] (Juni), **jul**[**y**] (Juli), **aug**[**ust**] (August), **sep**[**tember**] (September), **oct**[**ober**] (Oktober), **nov**[**ember**] (November), **dec**[**ember**] (Dezember), **nextmonth** | **next**␣**month** (der auf den aktuellen Monat folgende Monat)

tag
Zahl zwischen 1 und 31, entsprechend der Länge des Monats

jahr
Jahreszahl, für die die Datumsangabe gelten soll
nextyear | **next**␣**year:** das auf das aktuelle Jahr folgende Jahr.
jahr nicht angegeben:
Wenn die Datumsangabe vor dem aktuellen Datum liegt, geht *at* vom nächsten Jahr, sonst vom aktuellen Jahr aus.

wochentag (abhängig von der Umgebungsvariablen *LC_TIME*)
mon[**day**] (Montag), **tue**[**sday**] (Dienstag), **wed**[**nesday**] (Mittwoch), **thu**[**rsday**] (Donnerstag), **fri**[**day**] (Freitag), **sat**[**urday**] (Samstag), **sun**[**day**] (Sonntag)

nextweek | **next**␣**week** -
die auf die aktuelle Woche folgende Woche

spezieller tag
today (heute), **tomorrow** (morgen), **nextday | next**␣**day** (der auf den aktuellen Tag folgende Tag

Die Angabe *nextday* (oder *next*␣*day*) bedeutet, dass der Auftrag einen Tag später ausgeführt wird.
Wenn die Zeitangabe vor der aktuellen Uhrzeit liegt, interpretiert *at* den nächsten Tag als aktuellen Tag. Wird z.B. am 1.7.95 um 11:00 Uhr der Auftrag *at 10 nextday* erteilt, so wird er erst am 3.7.95 um 10:00 Uhr ausgeführt. *at 14 nextday* bewirkt hingegen, dass die Ausführung am 2.7.95 um 14:00 Uhr erfolgt.

datum nicht angegeben:
– entspricht der Angabe *today*, wenn die Zeitangabe (gerundet auf die Minute) nach der aktuellen Uhrzeit liegt.

– entspricht der Angabe *tomorrow*, wenn die Zeitangabe (gerundet auf die Minute) vor der aktuellen Uhrzeit liegt.
– entspricht der Angabe *now*, wenn die Zeitangabe (gerundet auf die Minute) gleich der aktuellen Uhrzeit ist.

+inkrement

inkrement ist eine positive ganze Zahl, auf die eine der folgenden Zeiteinheiten folgen muss:

minute[**s**]	Minute(n)
hour[**s**]	Stunde(n)
day[**s**]	Tag(e)
week[**s**]	Woche(n)
month[**s**]	Monat(e)
year[**s**]	Jahr(e)

Beispiel *at* können Sie u.a. in folgenden Formen angeben:

```
at 0815am jan 24
at 8:15am jan 24
at 5pm friday
at now +1hour
```

Nicht bearbeitete Aufträge ausgeben

Format 3 **at**␣**-l**[␣auftragsnummer]

Format 4 **at**␣**-l**␣**-q**␣warteschlange

-l[␣auftragsnummer]
at listet den Auftrag mit *auftragsnummer* auf, wenn er noch nicht bearbeitet wurde. *auftragsnummer* ist die Nummer, die auf die Standard-Fehlerausgabe ausgegeben wird, wenn ein Kommandoauftrag mit *at*, *batch* oder *cron* erteilt wird.

auftragsnummer nicht angegeben:
at listet alle Aufträge, die noch nicht bearbeitet wurden, mit ihren Auftragsnummern auf.

-l␣**-q**␣warteschlange
Alternativ zur auftragsnummer kann die Warteschlange angegeben werden. *at* listet alle Aufträge auf, die in *warteschlange* stehen.

Aufträge löschen

Format 5 **at␣-r␣**auftragsnummer

-r␣auftragsnummer
at löscht den Auftrag *auftragsnummer*, den Sie zuvor mit *at* oder *batch* erteilt haben. *auftragsnummer* ist die Nummer, die auf die Standard-Fehlerausgabe ausgegeben wird, wenn ein Kommandoauftrag mit *at* oder *batch* erteilt wird. Sie können mehrere Auftragsnummern, getrennt durch Leerzeichen, angeben. Nur der POSIX-Verwalter hat das Recht, Aufträge von anderen Benutzern zu löschen.

Fehler Die häufigsten Fehlermeldungen sind:

`at: bad date specification`
Sie haben das Datum in einem falschen Format angegeben.

`at: too late`
Sie haben ein bereits abgelaufenes Datum angegeben.

`at: This job may not be executed at the proper time`
zeit liegt zwischen „now“ und „now+1hour“.

`at: you are not authorized to use at. Sorry.`
Sie dürfen *at* nicht aufrufen (siehe *Vor dem Aufruf beachten*).

Datei */usr/lib/cron/at.allow*
Liste der Benutzerkennungen mit Ausführrecht für *at*. In jeder Zeile steht jeweils eine Benutzerkennung.

/usr/lib/cron/at.deny
Liste der Benutzerkennungen ohne Ausführrecht für *at*. In jeder Zeile steht jeweils eine Benutzerkennung.

/var/spool/cron/atjobs
Dateiverzeichnis, in dem die noch nicht bearbeiteten *at*-Aufträge in einzelnen Dateien aufgelistet werden. Für jeden *at*-Auftrag gibt es eine eigene Datei mit dem Dateinamen *auftragsnummer.a*.

Variable *SHELL*
bestimmt den Namen eines Kommandointerpreters, um den *at*-Auftrags aufzurufen. Ist diese Variable nicht gesetzt oder hat den Wert 0, wird *sh* verwendet.

TZ
bestimmt die Zeitzone. Der *at*-Auftrag wird zum Zeitpunkt *-t␣zeit* oder *zeitpunkt* ausgeführt, relativ zu der in *TZ* bestimmten Zeitzone. Wenn *zeitpunkt* eine Zeitzone bestimmt, wird *TZ* überschrieben. Wenn *zeitpunkt* keine Zeitzone bestimmt und *TZ* nicht gesetzt ist oder den Wert 0 hat, wird der entsprechende Standardwert verwendet.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *at*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_TIME Legt das Format der Datums- und Zeitangaben fest, die von *at* geschrieben und akzeptiert werden.

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Es soll am 1. April um 13 Uhr an der Datensichtstation tty013 das aktuelle Datum und die Zeichenkette *April, April!* ausgegeben werden:

```
$ at 1pm apr 1 ↵
echo `date`":April, April!" >> /dev/tty013 ↵
END oder @@d
```

Siehe auch *batch*, *crontab*, *date*, *kill*, *mailx*, *nice*, *ps*, *sh*, *sort*

awk Programmierbare Bearbeitung von Textdateien (pattern scanning and processing language)

awk ist ein programmgesteuertes Textbearbeitungssystem.
Sie geben beim Aufruf des *awk* das auszuführende *awk*-Programm und die Namen der zu bearbeitenden Dateien an. *awk* führt dann die durch das Programm festgelegte Bearbeitung mit den angegebenen Dateien aus. *awk* verändert die Eingabedateien nicht. Standardmäßig schreibt *awk* das Ergebnis der Bearbeitung auf die Standard-Ausgabe.

Im Vergleich zu Textbearbeitungsprogrammen wie *egrep* und *sed* bietet *awk* folgende Vorteile:

– *awk* arbeitet satzweise. Ein Eingabesatz ist zwar wie bei *egrep* und *sed* standardmäßig eine Zeile, der Benutzer kann diese Einstellung aber ändern und andere Texteinheiten als Satz definieren.
– Jeder Eingabesatz ist in Felder aufgeteilt, die einzeln angesprochen werden können.
– Eine Auswahlbedingung kann eine aus erweiterten regulären Ausdrücken und Vergleichen zusammengesetzte Bedingung sein.
– Der Benutzer kann beliebige Aktionen programmieren. Die *awk*-Sprache ist eine höhere, C-ähnliche Programmiersprache.

Die Beschreibung des *awk* gliedert sich in folgende Abschnitte:

Syntax

Format 1: awk[␣**-F**␣ERE][␣**-v**␣initialisierung]...␣prog[␣initialisierung]...[␣datei]...

Format 2: awk[␣**-F**␣ERE][␣**-f**␣progdat][␣**-v**␣initialisierung]...[␣initialisierung]...[␣datei]

-F␣ERE

Trennzeichen zwischen den Feldern eines Eingabesatzes festlegen.

ERE — Erweiterter regulärer Ausdruck zur Bestimmung des Trennzeichens zwischen den Feldern eines Eingabesatzes. Das Trennzeichen gehört nicht zu einem Feld.

Wenn Sie als Trennzeichen zwischen den Feldern eines Eingabesatzes das Zeichen *t* verwenden wollen, so müssen Sie es beim *awk*-Aufruf oder im BEGIN-Teil des *awk*-Programms folgendermaßen angeben:
`awk -F"[t]"...` oder `BEGIN {FS="t"...}`

-F␣ERE nicht angegeben:
Leer- und Tabulatorzeichen gelten als Feldtrenner.

-v␣initialisierung

Wertzuweisung der Form *var=wert*.
awk initialisiert die Variable *var*, die im Programm vorkommt, mit *wert*.

var — Name der Variablen, die initialisiert werden soll.

wert — Wert, mit dem die Variable *var* initialisiert werden soll. *wert* kann genauso definiert werden, wie eine Umgebungsvariable in der Shell.

Die Wertzuweisung über *-v initialisierung* ist identisch mit der Wertzuweisung über *initialisierung* (siehe dort).

prog

Angabe des *awk*-Programms.

prog kann sein:
– 'awk-programm'; ein *awk*-Programm in der Kommandozeile
– *-f progdat*; der Name einer Datei, die ein *awk*-Programm enthält.

'awk_programm'

Ein *awk*-Programm in der Kommandozeile.
Sie sollten das *awk*-Programm immer in Hochkommata '...' einschließen, um Shell-Sonderzeichen vor ungewollter Auswertung durch die Shell zu schützen. Wenn das Programm länger als eine Zeile ist, dann müssen Sie das Neue-Zeile-Zeichen durch einen Gegenschrägstrich \ entwerten.

Beispiel

Alle Zeilen aus der Datei *eingabe* ausgeben, deren drittes Feld aus dem Zeichen '0' besteht:

```
$ awk '$3 == 0' eingabe
```

-f␣progdat
Das *awk*-Programm steht in der Datei *progdat*.

initialisierung
Wertzuweisung der Form: *var*=*wert*

awk initialisiert die Variable *var* (egal, ob diese im *awk*-Programm vorkommt oder nicht) mit *wert*. *initialisierung* und *datei* können in beliebiger Reihenfolge angegeben werden. Die Wertzuweisung erfolgt zu dem Zeitpunkt, zu dem normalerweise die an der Position stehende Datei geöffnet worden wäre.
Zuweisungen vor dem ersten Dateinamen werden nach dem BEGIN-Teil des *awk*-Programms (wenn vorhanden) ausgeführt; Zuweisungen nach dem letzten Dateinamen werden vor dem END-Teil des *awk*-Programms (wenn vorhanden) ausgeführt.

Ausnahme
Die $-Variablen (siehe *Grundelemente*) können nicht auf diese Weise initialisiert werden.

var — Name der Variablen, die initialisiert werden soll. Die Variable darf nicht mit $ beginnen.

wert — Wert, mit dem die Variable *var* initialisiert werden soll. *wert* kann genauso definiert werden wie eine Umgebungsvariable in der Shell.

datei
Name der Textdatei, die bearbeitet werden soll. Sie können mehrere Dateien angeben. Bei mehreren Angaben verarbeitet *awk* die Dateien in der angegebenen Reihenfolge. Wenn Sie für *datei* einen Bindestrich - angeben, liest *awk* von der Standard-Eingabe.

datei nicht angegeben:
awk liest von der Standard-Eingabe. *awk* liest die Eingabe satzweise ein, bearbeitet sie und gibt nach jeder Zeile das Ergebnis für diesen Satz aus. Terminalabhängig beenden Sie die Eingabe mit END oder CTRL D oder @@d.

Verwendungsmöglichkeiten für awk

awk ist ein Werkzeug, mit dem Sie Textbearbeitungsaufgaben bequem lösen können. Typische Anwendungen sind:

– Daten aus Dateien heraussuchen

– Dateiinhalte überprüfen

– Berechnungen mit den Daten in einer Datei durchführen

– Format der Eingabedaten ändern.

Dieser Abschnitt zeigt an vier einfachen Beispielen, wie Sie *awk* anwenden können.

Beispiel Die Datei *artikel* enthält eine Aufstellung von Büroartikeln. Angegeben sind jeweils der Artikelname, die Stückzahl und der Einzelpreis:

```
Bleistift  1500     0.60
Tisch         5   345.00
Lampe        20    79.80
Papier       75     1.00
Diskette   1000     2.40
Umschlag    100     0.20
```

Beispiel 1 Alle Artikel heraussuchen, deren Stückzahl größer als 100 ist:

```
$ awk '$2 > 100 {print}' artikel
Bleistift  1500     0.60
Diskette   1000     2.40
```

Mit *$2* sprechen Sie das zweite Feld einer Zeile an, das in diesem Beispiel die Stückzahl eines Artikels enthält. Wenn die Stückzahl größer als 100 ist, ist die Bedingung erfüllt und die Funktion *print* wird ausgeführt. Da für *print* keine Argumente angegeben sind, gibt *print* die ganze Zeile aus.

Beispiel 2 Für alle Artikel mit einer Stückzahl größer als 100 den Gesamtpreis berechnen und zusammen mit der Artikelbezeichnung ausgeben:

```
$ awk '$2 > 100 {print $1 "\t" $2*$3}' artikel
Bleistift       900
Diskette        2400
```

In diesem Beispiel hat die *print*-Funktion drei Argumente. Ausgegeben werden:

$1 Artikelbezeichnung (erstes Feld)

\t Tabulatorzeichen

$2*$3 Stückzahl (zweites Feld) mal Einzelpreis (drittes Feld)

Beispiel 3 Die Ausgabe mit einer Überschrift versehen:

```
$ awk 'BEGIN    {print "Artikelbezeichnung \tGesamtbetrag"} \
>      $2 > 100 {print $1 "\t\t" $2*$3}' artikel
Artikelbezeichnung       Gesamtbetrag
Bleistift                900
Diskette                2400
```

Dieses Beispiel zeigt die Verwendung des BEGIN-Teils. *awk* führt die Aktion hinter BEGIN nur einmal bei Start des Programms aus. Deshalb wird die Überschrift genau einmal am Anfang ausgegeben.

Beispiel 4 Am Ende die Summe aller Beträge ausgeben.
Dazu wird eine Variable *summe* verwendet, die im BEGIN-Teil mit 0 initialisiert wird. Für jede Zeile wird das Produkt aus zweiter und dritter Spalte aufaddiert:

```
$ awk 'BEGIN    {summe=0; print "Artikelbezeichnung \tGesamtbetrag"} \
>      $2 > 100 {print $1 "\t\t" $2*$3; summe += $2*$3} \
>      END      {print "\nSumme: " summe}' artikel
Artikelbezeichnung       Gesamtbetrag
Bleistift                900
Diskette                 2400

Summe: 3300
```

Dieses Beispiel zeigt die Verwendung des END-Teils. *awk* führt die Aktion hinter END nur einmal vor Beendigung des Programms aus. Deshalb wird die Summe aller Beträge genau einmal am Ende ausgegeben.

Struktur eines awk-Programms

Ein *awk*-Programm kann aus einem BEGIN-, Haupt- und END-Teil bestehen, die nach folgendem Schema aufgebaut sind:

Syntax

```
– BEGIN-Teil –

[ BEGIN {aktion} ]

– Hauptteil –

[[auswahlbedingung] {aktion}
| auswahlbedingung [{aktion}]
| funktionsdefinition
.
.
.
                                    ]

– END-Teil –

[ END {aktion} ]
```

auswahlbedingung
: Mit *auswahlbedingung* gibt der Benutzer an, welche Daten aus den Eingabedateien ausgewählt werden sollen (siehe „Auswahlbedingungen“ auf Seite 129).

aktion
: Mit *aktion* gibt der Benutzer an, was geschehen soll, wenn Daten entsprechend *auswahlbbedingung* auftreten (siehe „Aktionen“ auf Seite 131).

funktionsdefinition
: Mit *funktionsdefinition* hat der Benutzer die Möglichkeit, eigene Funktionen zu definieren (siehe „Funktionen“ auf Seite 138).

Mindestens einer der drei Teile *auswahlbedingung*, *aktion*, *funktionsdefinition* muss vorhanden sein.
Von einem Paar *auswahlbedingung {aktion}* kann entweder die Auswahlbedingung oder die Aktion fehlen. Fehlt *aktion*, so wird jeweils die durch *auswahlbedingung* erfasste Zeile ausgegeben. Fehlt *auswahlbedingung*, so wird *aktion* für jede Zeile ausgeführt.
Die Definition einer benutzerdefinierten Funktion kann an beliebiger Stelle im Hauptteil erfolgen.
Folgende Teile müssen jeweils am Anfang einer Zeile (nach beliebig vielen Leer- oder Tabulatorzeichen) stehen:

- der BEGIN-Teil
- die Paare *[auswahlbedingung]{aktion}* bzw. *auswahlbedingung [{aktion}]*
- die Funktionsdefinition
- der END-Teil

Arbeitsweise von awk

awk führt das vom Benutzer angegebene *awk*-Programm aus. Dabei geht *awk* im Einzelnen wie folgt vor:

1. Anfangsarbeiten
   Wenn Variablen angegeben wurden, so werden als erstes diese Variablen initialisiert. Falls ein *BEGIN*-Teil mit *aktion* vorhanden ist, führt *awk* dann die dort festgelegte Aktion aus. Die im *BEGIN*-Teil angegebene Aktion wird genau einmal ausgeführt und zwar vor der Bearbeitung der ersten Eingabezeile.
2. Dateibearbeitung
   Anschließend verarbeitet *awk* die angegebenen Eingabedateien. *awk* liest die Eingabesätze der Reihe nach ein. Für jeden Eingabesatz prüft *awk* jede Auswahlbedingung ab und zwar in der Reihenfolge, wie sie im *awk*-Programm angegeben sind. Wenn eine Auswahlbedingung zutrifft, wird die zugehörige Aktion ausgeführt.
   Wenn zu einer Aktion keine Auswahlbedingung angegeben ist, führt *awk* die Aktion für jeden Satz aus. Wenn zu einer Auswahlbedingung keine Aktion angegeben ist, ist die Standard-Aktion Ausgabe des Satzes.
   Mehrere Eingabedateien werden in der angegebenen Reihenfolge abgearbeitet.
3. Abschlussarbeiten
   Wenn alle angegebenen Dateien abgearbeitet sind und ein END-Teil vorhanden ist, dann führt *awk* zum Schluss die im END-Teil angegebene Aktion aus. Danach beendet sich *awk*.

Eingabedatei

Eine Eingabedatei besteht aus Sätzen, die in Felder eingeteilt sind.

– **Sätze**
  Die Sätze sind durch ein Satztrennzeichen getrennt. Die Satztrennzeichen sind nicht Bestandteil des Satzes. Standardmäßig ist ein Satz eine Zeile und das Satztrennzeichen ist das Neue-Zeile-Zeichen. Der Benutzer hat die Möglichkeit, diese Einteilung zu ändern. Dazu gibt es die vordefinierte Variable *RS* (RS - Record Separator), der Sie ein beliebiges Zeichen zuweisen können. Falls als Wert für *RS* eine Zeichenkette angegeben wird, wird nur das erste Zeichen dieser Zeichenkette berücksichtigt. Die Anzahl der aktuell verarbeiteten Sätze wird in der Variablen *NR* (NR - Number of Records) gezählt. Bei mehreren Eingabedateien wird *NR* über alle Dateien hochgezählt. Auf den aktuellen Satz können Sie mit der vordefinierten Variablen *$0* zugreifen. Näheres zu Variablen erfahren Sie im Abschnitt „Grundelemente der awk-Sprache“ auf Seite 122.

– **Felder**
  Jeder Satz ist in Felder unterteilt, die durch ein oder mehrere Feldtrennzeichen getrennt sind. Standardmäßig ist eine beliebig lange Folge von Leer- und Tabulatorzeichen ein Feldtrenner. Der Benutzer hat die Möglichkeit, diese Einteilung zu ändern. Dazu gibt es die vordefinierte Variable *FS* (FS - Field Separator), der Sie entweder beim Aufruf durch Angabe der Option *-F* oder im *awk*-Programm ein beliebiges anderes Zeichen zuweisen können. Der an *FS* zugewiesene Wert wird als erweiterter regulärer Ausdruck (siehe Abschnitt „Reguläre Ausdrücke der POSIX-Shell“ auf Seite 939) interpretiert.

Beispiel 1 Sie wollen die Zeichen *x* und *y* als alternative Feldtrenner definieren.
Syntax beim *awk*-Aufruf: `-F"[xy]"`
Syntax im *awk*-Programm: `FS=[xy]`

Beispiel 2 Sie wollen eine beliebig lange Folge des Zeichens *x* als Feldtrenner definieren.
Syntax beim *awk*-Aufruf: `-F"x+"`
Syntax im *awk*-Programm: `FS=x+`

Die Standard-Einstellung (beliebig lange Folge von Leer- und Tabulatorzeichen) kann also mit dem folgenden regulären Ausdruck dargestellt werden: `[␣\t]+` (␣ steht hier für das Leerzeichen, \t für das Tabulatorzeichen).
Beachten Sie, dass ein Neue-Zeile-Zeichen immer als Feldtrenner interpretiert wird, egal welchen Wert *FS* hat!

Die Anzahl der Felder des aktuellen Satzes wird in der Variablen *NF* (NF - Number of Fields) gespeichert. Auf die einzelnen Felder des aktuellen Satzes können Sie mit den vordefinierten Variablen *$1*, *$2* bis *$NF* zugreifen. Näheres zu Variablen erfahren Sie im Abschnitt „Grundelemente der awk-Sprache“ auf Seite 122.

Beispiel Standard-Einteilung

```
1.Feld  2. Feld   ...         5. Feld ...

Das     ist       der erste   Satz             <--- 1. Satz

und     das       ist der     zweite Satz.     <--- 2. Satz
```

Nicht-Standard-Einteilung: *RS*="%"; *FS*=":";

```
1.Feld   2. Feld        3. Feld

%Name  : Adresse      : Telefon          <--- 1. Satz

%SNI AG:81730 Muenchen:089-636-1         <--- 2. Satz
```

Regeln für Satz- und Feldtrennzeichen

- Standard-Einstellungen für Satztrenner
  - Standardmäßig ist das Neue-Zeile-Zeichen das Satztrennzeichen.
  - Wenn *RS* die leere Zeichenkette ist (*RS*=""),
    besteht die Datei aus einem einzigen Satz. Falls mehrere Dateien angegeben werden, besteht jede Datei aus einem einzigen Satz (*NR* hat am Ende die Anzahl der Dateien als Wert).
- Standard-Einstellungen für Feldtrenner
  - Wenn der Satztrenner das Neue-Zeile-Zeichen ist, dann gelten standardmäßig Leer- und Tabulatorzeichen als Feldtrenner.
  - Wenn der Satztrenner nicht das Neue-Zeile-Zeichen ist, dann gilt das Neue-Zeile-Zeichen **immer** als Feldtrenner, unabhängig davon, welches Zeichen als Feldtrenner definiert ist (siehe Felder, Beispiel 2 auf Seite 119).
  - Wenn Sie *FS* explizit das Leerzeichen zuweisen,
    durch Aufruf von *awk* mit -F" " oder mit der Zuweisung *FS*=" ",
    dann gelten Leer- **und** Tabulatorzeichen als Feldtrenner.
  - Wenn Sie dagegen *FS* explizit das Tabulatorzeichen zuweisen (*FS*="\t"), dann gilt nur noch das Tabulatorzeichen als Feldtrenner, das Leerzeichen nicht mehr.
- Führende Feldtrenner und Folgen von Feldtrennern
  - Für die Feldtrenner Leer-, Tabulator- und Neue Zeile-Zeichen gilt:
  - Führende Feldtrenner werden ignoriert.
  - Beliebige Folgen von Feldtrennern werden als ein Trennzeichen gezählt.
    (siehe Beispiel 9 auf Seite 152).
  - Bei jedem anderen Feldtrenner werden führende Feldtrenner gezählt. Bei einer Folge von Feldtrennern wird jedes Zeichen einzeln gezählt. Zwei aufeinanderfolgende Feldtrenner ergeben daher ein leeres Feld. (siehe Beispiel 10 auf Seite 153).
- Einteilungen verändern

  Sie können *RS* im *awk*-Programm verändern, wenn Sie für eine Datei verschiedene Satzeinteilungen benötigen. Der neue Satztrenner gilt, sobald die Zuweisung an *RS* ausgeführt wurde. Ebenso können Sie *FS* im *awk*-Programm verändern, wenn Sie für eine Datei verschiedene Feldeinteilungen benötigen. Der neue Feldtrenner gilt, sobald die Zuweisung an *FS* ausgeführt wurde.

Vordefinierte Variablen für die Eingabedatei

Die folgende Liste zeigt alle vordefinierten *awk*-Variablen, die die Eingabedatei betreffen und die jeweiligen Werte, mit denen *awk* diese Variablen standardmäßig belegt.

FILENAME
: Name der aktuellen Eingabedatei, - bei Standard-Eingabe

FS
: Feldtrenner für die Eingabe
(Standard: beliebig lange Folge von Leer- und Tabulatorzeichen)

NF
: Anzahl der Felder des aktuellen Satzes

NR
: Laufende Nummer des aktuellen Satzes im Input

FNR
: Laufende Nummer des aktuellen Satzes in der aktuellen Datei

RS
: Satztrenner für die Eingabe (Standard: Neue-Zeile-Zeichen)

$0
: Aktueller Satz

$1
: Erstes Feld des aktuellen Satzes

$2
: Zweites Feld des aktuellen Satzes

.
.
.

$NF
: Letztes Feld des aktuellen Satzes

Sie können die Variablen im *awk*-Programm verändern. Die Eingabedatei wird dadurch aber nicht verändert. Näheres zu Variablen erfahren Sie im Abschnitt „Grundelemente der awk-Sprache“ auf Seite 122.

Grundelemente der awk-Sprache

In diesem Abschnitt sind die Grundelemente der *awk*-Sprache zusammengestellt.
Die Grundelemente benötigen Sie bei der Formulierung von Auswahlbedingungen und Aktionen.

Kommentar
Sie können ein *awk*-Programm wie eine Shell-Prozedur kommentieren. Kommentar beginnt mit dem Zeichen # und geht bis zum Ende der Zeile.

Konstanten
Es gibt zwei Arten von Konstanten: zahl oder zeichenkette.

zahl
Eine Zahl (numerische Konstante) ist eine Ganzzahl oder eine Gleitkommazahl mit oder ohne Vorzeichen. *awk* überprüft das Format nicht. Wenn eine Zahl ungültige Zeichen enthält, versucht *awk* einen gültigen Teil herauszufiltern und ignoriert den Rest.

ganzzahl
Eine Ganzzahl ist eine Folge aus den Ziffern 0 bis 9.

gleitkommazahl
Eine Gleitkommazahl besteht aus Mantisse mit oder ohne Exponent.
Die Mantisse besteht aus einer Ganzzahl mit oder ohne Nachkommateil.
Der Nachkommateil besteht aus einem Dezimalpunkt und einer Ganzzahl.

zeichenkette
Eine Zeichenkette (alphanumerische Konstante) ist eine Folge von Zeichen, eingeschlossen in Anführungszeichen "...". Fehlen die Anführungszeichen, dann interpretiert *awk* die Zeichenkette als Variablennamen, Zahl oder Operator.

zeichen
Ein Einzelzeichen wird auch in Anführungszeichen "..." eingeschlossen, damit *awk* das Zeichen nicht als Variablennamen interpretiert. Ein Zeichen ist ein darstellbares Zeichen aus dem aktuell gültigen Zeichensatz (siehe Abschnitt „Zeichensatz EBCDIC (EDF04)“ auf Seite 956) oder eines der folgenden Sonderzeichen, die wie in C dargestellt werden:

\"	für "
\\	für \
\a	für Klingelzeichen
\n	für Neue-Zeile-Zeichen
\t	für Tabulatorzeichen
\v	für Vertikaltabulator
\b	für Rücksetzzeichen (backspace)
\r	für Wagenrücklauf (carriage return)
\f	für Seitenvorschub

Variablen

awk ermöglicht die Verwendung von einfachen Variablen und Arrays, um Werte abzuspeichern.
Es gibt vordefinierte und benutzerdefinierte Variablen.

Variablennname
Der Name einer benutzerdefinierten Variablen fängt entweder mit einem Unterstrich _ , einem Großbuchstaben oder einem Kleinbuchstaben an und besteht nur aus Unterstrichen, Groß- und Kleinbuchstaben sowie Ziffern.

Datentyp
Variablen haben keinen Datentyp. Sie können derselben Variablen sowohl eine Zahl als auch eine Zeichenkette zuweisen. In einem eindeutig numerischen Kontext werden Variablen als numerische Variablen behandelt, ansonsten gelten sie standardmäßig als alphanumerisch. Beispiel:

```
x = "Mueller";    # die Variable x enthält die Zeichenkette Mueller
x = "3"+4    ;    # die Variable x hat den Wert 7
```

Definition
Variablen werden nicht explizit definiert. Benutzerdefinierte Variablen sind mit der ersten Verwendung automatisch definiert.

Initialisierung
Die vordefinierten Variablen werden von *awk* wie vorgesehen belegt. Benutzerdefinierte Variablen werden von *awk* standardmäßig je nach Kontext mit der leeren Zeichenkette bzw. mit Null initialisiert. Bei Aufruf von *awk* können Sie andere Initialisierungen angeben.

Ausnahmen
– Für i>NF ist $i nicht unbedingt die leere Zeichenkette.
– $-Variablen können nicht bei Aufruf initialisiert werden.

Vordefinierte Variablen

awk kennt die in folgender Liste aufgeführten vordefinierten Variablen. In der Liste ist angegeben, welche Werte *awk* standardmäßig in diesen Variablen speichert. Der Benutzer kann den Variablen neue Werte zuweisen.

ARGC
Anzahl der Elemente im Array *ARGV*

ARGV
Array, das die Argumente der Kommandozeile enthält (ausgenommen Optionen und das Argument *prog*), nummeriert von 0 bis ARGC-1

CONVFMT
printf-Format, um Zahlen in Zeichenketten umzuwandeln

ENVIRON
: Array mit den Werten der Umgebungsvariablen, die Indices sind die Namen der Variablen

FILENAME
: Name der aktuellen Eingabedatei, - bei Standard-Eingabe

FS
: Feldtrenner für die Eingabe
(Standard: beliebig lange Folge von Leer- und Tabulatorzeichen)

NF
: Anzahl der Felder des aktuellen Satzes

NR
: Laufende Nummer des aktuellen Satzes im Input

FNR
: Laufende Nummer des aktuellen Satzes in der aktuellen Datei

OFS
: Feldtrenner für die Ausgabe (Standard: Leerzeichen)

ORS
: Satztrenner für die Ausgabe (Standard: Neue-Zeile-Zeichen)

OFMT
: Ausgabeformat für Gleitkommazahlen
(siehe *awk-Funktionen, Formatierte Ausgabe,)*
(Standard: %.6g, höchstens 6 Stellen hinter dem Komma)

RS
: Satztrenner für die Eingabe (Standard: Neue-Zeile-Zeichen)

RLENGTH
: Länge der Zeichenkette, die die *match*–Funktion als passend erkannt hat

RSTART
: Anfangsposition der Zeichenkette, die die *match*-Funktion als passend erkannt hat. Die Nummerierung fängt mit 1 an. Dieser Wert entspricht immer dem Return-Wert der *match*-Funktion.

SUBSEP
: Subscript-Zeichenketten-Trenner für mehrdimensionale Arrays.
Standard-Einstellung ist \034.

$0
: Aktueller Satz

$*n*
: *n*-tes Feld des aktuellen Satzes

$NF
letztes Feld des aktuellen Satzes

Wie wirkt sich die Änderung von vordefinierten Variablen aus?

Beispiel Durch die Zuweisung

```
$1 = "neu";
```

wird *$1* die Zeichenkette *neu* zugewiesen. Das erste Feld des aktuellen Eingabesatzes bleibt aber unverändert.

Das gilt auch für folgende *awk*-Einstellungen, die die Eingabedatei betreffen:

1. Die aktuelle Eingabedatei ändert sich nicht, wenn Sie FILENAME einen neuen Namen zuweisen.
2. Wenn Sie an eine Variable $i mit i>*NF* einen Wert zuweisen, bekommt *NF* den Wert *i* zugewiesen.
3. Wenn Sie *NR* einen neuen Wert zuweisen, wird nur die Zeilennummer verändert, aber die Einstellung, welcher Satz der aktuelle ist, bleibt unverändert.

Beispiel Der Inhalt von *$0* bleibt unverändert, auch wenn *NR* verändert wird.

```
{print NR, $0; NR=NR+34; print NR, $0}
```

Die Ausgabe sieht dann etwa so aus:

```
10 Dies ist die zehnte Zeile
44 Dies ist die zehnte Zeile
```

Wenn Sie einer Variablen einen neuen Wert zuweisen, wird der alte Inhalt gelöscht. Wenn Sie z.B. *NF* verändern, ist die Information über die Feld-Anzahl des aktuellen Satzes verloren.

Besonderheit bei $-Variablen:
Sie können die Nummer einer $-Variablen als Konstante angeben oder durch einen Ausdruck, der bei Auswertung die Nummer ergibt.

Beispiel Mit *$(NF-1)* sprechen Sie das vorletzte Feld an.

Array
Ein Array ist ein Feld von Konstanten oder Variablen.

Ein Array-Element wird angesprochen mit:

Syntax

```
array_name[index]
```

array_name
Variablenname.

index
einfache Variable.
Der Index kann numerisch oder alphanumerisch sein. Sie können für *index* daher eine Zahl, eine Zeichenkette oder einen Ausdruck angeben, der bei Auswertung den Indexwert ergibt.

awk bietet bezüglich Arrays zwei besondere Möglichkeiten:

- Arrays werden dynamisch angelegt:
  Arrays werden wie einfache Variablen nicht deklariert, insbesondere muss auch keine Dimension festgelegt werden. Bei Bedarf wird automatisch ein neues Array-Element angelegt.
- Arrays können assoziativ durchlaufen werden:
  Sie können einen alphanumerischen Index verwenden, um die Array-Elemente anzusprechen.
  Zum Durchlaufen aller Elemente eines assoziativen Arrays gibt es die spezielle Laufanweisung:

  for(index in array) anweisung

  index durchläuft in **unbestimmter** Reihenfolge die bisher vorhandenen Indexwerte. Für jedes Array-Element wird *anweisung* einmal ausgeführt (siehe Ablaufanweisung *for*).

Beispiel

In der Datei *ausgaben* sind Ausgaben erfasst. Für jede Ausgabe sind Tag, Monat, Betrag und eine Kurzbeschreibung angegeben. Die einzelnen Angaben sind durch Doppelpunkt **:** getrennt, z.B.:

```
01:Januar:   40.78:Buerobedarf
05:Januar: 6789.00:Laser-Drucker
23:Maerz:   240.32:Lampen
11:Januar:  478.00:Stuehle
01:Februar:  45.00:Literatur
```

Durch Verwendung eines assoziativen Arrays können Sie aus dieser Datei sehr einfach die Gesamtausgaben für jeden Monat berechnen. Das Beispielprogramm verwendet das Array *mausgaben* und die Monatsnamen als alphanumerischen Index. Für jede Zeile werden die Ausgaben im dritten Feld (*$3*) zu den Ausgaben des Monats addiert, der in dem zweiten Feld (*$2*) steht.

```
$ awk 'BEGIN {FS=":"}               \
>      {mausgaben[$2] += $3;}  \
>      END {for (i in mausgaben) print "Gesamtausgaben",\
>           i, mausgaben[i]}' ausgaben

Gesamtausgaben Maerz 240.32
Gesamtausgaben Januar 7307.78
Gesamtausgaben Februar 45
```

Ausdrücke

Ein Ausdruck kann sein:

Syntax

- konstante
- variable
- funktionsaufruf
- un_op ausdruck
- ausdruck bin_op ausdruck
- **(**ausdruck**)**
- ausdruck **?** ausdruck **:** ausdruck

konstante
: numerische oder alphanumerische Konstante (siehe *Grundelemente*).

variable
: Variable (siehe *Grundelemente*).

funktionsaufruf
: Aufruf einer vordefinierten Funktion (siehe *Funktionen*).

ausdruck
: Ausdruck.

un_op
: unitärer Operator (siehe *Operator-Tabelle*).

bin_op
: binärer Operator (siehe *Operator-Tabelle*).

Ausdrücke werden ausgewertet und liefern einen Wert. Sie können in Auswahlbedingungen und Aktionen vorkommen.

awk-Operatoren

awk kennt alle C-Operatoren und zusätzlich die Operatoren für Mustervergleich und Verkettung von Zeichenketten.
In der folgenden Liste sind alle *awk*-Operatoren nach steigender Priorität aufgeführt. Operatoren in einer Zeile haben die gleiche Priorität.

=	Zuweisung
+= -= *= /= %= ^=	zusammengesetzte Zuweisung wie in C
| |	logisches ODER
&&	logisches UND
~ !~	Mustervergleich
> >= < <= != ==	Vergleich
hintereinanderschreiben	Verkettung
+ -	plus, minus
* / %	multiplizieren, dividieren, modulo
!	logisches NICHT
^ **	Exponent
++ --	Inkrement, Dekrement

Auswertung von Ausdrücken

Für die Operanden ist kein Datentyp vorgeschrieben. Sie können numerische und alphanumerische Konstanten beliebig mischen. *awk* bestimmt aus dem Kontext, ob eine numerische oder alphanumerische Operation ausgeführt wird.
Beachten Sie, dass es wie in C keine speziellen Wahrheitswerte gibt. Bei *awk* gilt wie bei C Null als falsch und ein Wert ungleich Null als wahr. Dies bedeutet, als Argument einer logischen Operation wird jeder Wert ≠0 als wahr erkannt. Das Ergebnis einer einer solchen Operation wird, wenn es wahr ist, durch 1 dargestellt.

Beispiel `(2&&2)+3=4`

Auswahlbedingungen

Mit den Auswahlbedingungen gibt der Benutzer an, welche Daten aus den Eingabedateien ausgewählt werden sollen. Eine Auswahlbedingung kann sein:

Syntax

- **/**regulärer_ausdruck**/**
- vergleich
- mustervergleich
- bereichsangabe
- zusammengesetzte-auswahlbedingung

/regulärer_ausdruck**/**
Regulärer Ausdruck.
awk unterstützt erweiterte reguläre Ausdrücke (siehe Abschnitt „Reguläre Ausdrücke der POSIX-Shell“ auf Seite 939). Ein regulärer Ausdruck wird in Schrägstriche /.../ eingeschlossen.

Beispiel Regulärer Ausdruck, der für Folgen aus a, b oder c steht:

```
/[abc]+/
```

vergleich
Ein Vergleich ist ein Ausdruck (siehe *Ausdrücke*) mit Vergleichsoperatoren. Die Vergleichsoperatoren und ihre Bedeutung sind:

a **>** b	*a* größer als *b*?
a **>=** b	*a* größer als oder gleich *b*?
a **<** b	*a* kleiner als *b*?
a **<=** b	*a* kleiner als oder gleich *b*?
a **==** b	*a* gleich *b*?
a **!=** b	*a* ungleich *b*?

Die Operanden *a* und *b* können beliebige Ausdrücke sein. Wenn beide Operanden numerisch sind, wird ein numerischer Vergleich durchgeführt, ansonsten ein alphanumerischer.

mustervergleich

Ein Mustervergleich ist ein Ausdruck (siehe *Ausdrücke*) mit Mustervergleichsoperatoren. Bei einem Mustervergleich wird eine Zeichenkette mit einem regulären Ausdruck, genannt Muster, verglichen. Die Mustervergleichsoperatoren und ihre Bedeutung sind:

zk **~** m Zeichenkette *zk* passt zu Muster *m*

zk **!~** m Zeichenkette *zk* passt nicht zu Muster *m*

Mit einem Mustervergleich als Auswahlbedingung können Sie einzelne Felder auswählen.

Beispiel Alle Sätze auswählen, deren erstes Feld mit A oder a beginnt:

```
$1 ~ /^[Aa]/
```

Der reguläre Ausdruck ^[Aa] steht für Zeichenketten mit A oder a am Anfang. Das erste Feld des Satzes (*$1*) muss zu dem regulären Ausdruck passen (~), d.h. am Anfang muss ein A oder a stehen.

bereichsangabe

Die Bereichsangabe hat die Form:

/regulärer_ausdruck**/, /**regulärer_ausdruck**/**

Die Bereichsangabe bedeutet, dass die zugehörige Aktion für alle Sätze ausgeführt werden soll, die innerhalb des Bereichs liegen. Anfang und Ende des Bereichs werden durch zwei reguläre Ausdrücke festgelegt. Der Bereich beginnt mit dem ersten Satz, der eine Zeichenkette enthält, die zum ersten regulären Ausdruck passt. Der Bereich endet mit dem ersten Satz, der eine Zeichenkette enthält, die zum zweiten regulären Ausdruck passt.

Beispiel Sie wollen den Bereich von der ersten Zeile, die mit C beginnt, bis zur ersten Zeile, die mit K beginnt, auswählen und von jeder Zeile in diesem Bereich das erste Feld ausgeben lassen:

```
/^C/, /^K/  {print $1}
```

zusammengesetzte-auswahlbedingung

Mit den logischen Operatoren (siehe *Ausdrücke*) können Auswahlbedingungen negiert und mehrere Auswahlbedingungen zu einer Bedingung zusammengesetzt werden. Die logischen Operatoren und ihre Bedeutung sind:

!aw Negation der Auswahlbedingung *aw*

aw1 || aw2 Auswahlbedingung *aw1* oder *aw2*.

Die Bedingung ist erfüllt, wenn *aw1* oder *aw2* wahr (ungleich 0) ergibt.

aw1 **&&** aw2 Auswahlbedingung *aw1* und *aw2*.
Die Bedingung ist erfüllt, wenn *aw1* und *aw2* wahr (ungleich 0) ergeben.

(aw**)** Klammern

Die Auswertung einer zusammengesetzten Bedingung erfolgt von links nach rechts.

Beispiel Sie wollen alle Sätze auswählen, die eine gerade Feld-Anzahl haben und deren erstes Feld mit einem Buchstaben zwischen M (inklusive) und Q (exklusive) beginnt.

```
NF%2==0 && $1 >= "M" && $1 < "Q"
```

Es gibt im Allgemeinen mehrere Möglichkeiten, eine Auswahl durch eine Bedingung zu formulieren. Besteht in der aktuell gültigen Sortierreihenfolge der Bereich [M-Q] genau aus den Großbuchstaben M, N, O, P, Q, so erhält man dieselbe Auswahl durch folgenden Mustervergleich:

```
NF%2==0 && $1 ~ /^[MNOP]/
```

Die erste *awk*-Zeile kann in Abhängigkeit von der Sortierreihenfolge des aktuellen Zeichensatzes unterschiedliche Ergebnisse liefern. Die zweite *awk*-Zeile wählt immer nur die Sätze aus, deren erstes Feld mit den Buchstaben M, N, O oder P beginnt.

Aktionen

In den Aktionen geben Sie an, was geschehen soll, wenn die zugehörige Auswahlbedingung zutrifft. Dies kann z.B. die Bearbeitung einer ausgewählten Daten sein. Die Aktion muss auf derselben Zeile wie die zugehörige Auswahlbedingung beginnen. Ist dies nicht möglich, so ist ein Neue-Zeile-Zeichen mit \ zu entwerten. Leer- und Tabulatorzeichen zwischen Aktion und Auswahlbedingung werden ignoriert. Eine Aktion besteht aus einer oder mehreren Anweisungen und muss in geschweifte Klammern {...} eingeschlossen sein:

Syntax **{**anweisung␣[**;**anweisung]...**}**

Anweisung

Eine Anweisung kann sein:

Syntax
– ausdruck
– ablaufanweisung

ausdruck
Der Ausdruck wird ausgewertet, jedoch nur dann weiter verwertet, wenn *ausdruck* eine Zuweisung, ein Inkrement oder ein Dekrement ist (siehe Abschnitt „Ausdrücke“ auf Seite 127).

ablaufanweisung
: Mit *ablaufanweisung* können Sie den Ablauf des *awk*-Programms steuern (siehe Abschnitt „Ablaufanweisungen“ auf Seite 132).

Eine Anweisung kann mehrere Zeilen einnehmen. Dabei muss jede Zeile mit einem Gegenschrägstrich \ abgeschlossen werden. Hierdurch wird das Neue-Zeile-Zeichen entwertet.

Mehrere Anweisungen

Mehrere Anweisungen können durch geschweifte Klammern {} zusammengefasst werden. Anweisungen werden voneinander getrennt durch:
- Strichpunkt ;
- geschweifte Klammer zu }
- Neue-Zeile-Zeichen.

Ablaufanweisungen

Mit den Ablaufanweisungen können Sie den Ablauf des *awk*-Programms steuern. Bei *awk* gibt es folgende Ablaufanweisungen:

break	Schleife abbrechen
continue	Rest eines Schleifendurchgangs überspringen
exit	*awk*-Programm beenden
for	Gezählte Wiederholung und ARRAY-Durchlauf
if	Bedingte Anweisung
next	Mit dem nächsten Eingabesatz fortfahren
while	Schleife
do	Schleife
delete array[i]	Element *i* des Arrays *array* löschen
return x	Rücksprung aus einer Funktion mit Wertangabe
return	Rücksprung aus einer Funktion ohne Wert

Die Ablaufanweisungen sind im Folgenden alphabetisch beschrieben.

break - Schleife abbrechen

break kann im Schleifenrumpf einer *for*-, *while*- oder *do*-Schleife verwendet werden. *break* bewirkt, dass die Schleife beendet wird.

Syntax **break**

Beispiel Solange ein Satz mit . beginnt, soll der nächste Satz eingelesen werden. Wenn das 2. Feld des eingelesenen Satzes größer als 1000 ist, soll abgebrochen werden.

```
{ while($1 ~ /^\./)
    {
       getline;
       if($2 > 1000) break;
    }
}
```

continue - Rest eines Schleifendurchlaufs überspringen

continue kann im Schleifenrumpf einer *for*-, *while*- oder *do*-Schleife verwendet werden. *continue* bewirkt, dass der aktuelle Schleifendurchlauf beendet und mit dem nächsten Durchlauf weitergemacht wird.

Syntax **continue**

Beispiel Es sollen nur gerade Felder ausgegeben werden:

```
{
   i=1;
   while(i++ <= NF)
      {
        if(i%2) continue;
        else print $i
      }
}
```

do - Schleife

Mit der *do*-Schleife (oder *do-while*-Schleife) wird eine Anweisung wiederholt, solange eine Bedingung erfüllt ist. Im Gegensatz zur *while*-Schleife wird die Anweisung auf jeden Fall mindestens einmal ausgeführt.

Syntax

do anweisung **while (**ausdruck**)**

anweisung
: Anweisung, die bei jedem Schleifendurchgang ausgeführt wird. Wenn mehrere Anweisungen ausgeführt werden sollen, müssen sie mit geschweiften Klammern { } zusammengefasst und durch Strichpunkt ; oder Neue-Zeile-Zeichen getrennt werden.

ausdruck
: Ausdruck (siehe *Ausdrücke*), der die Bedingung angibt.

Beispiel

Die Felder eines Satzes sollen einzeln ausgegeben werden:

```
{ i=0; do {print $(++i)} while (i != NF) }
```

exit - awk-Programm beenden

Mit *exit* wird das *awk*-Programm beendet.

Wenn ein END-Teil vorhanden ist, dann führt *awk* noch die dort angegebene Abschluss-Aktion aus, ansonsten wird das Programm sofort beendet.

Syntax

exit

Beispiel

Wenn das Zeichen Klammeraffe in der Eingabe erscheint, soll das Ergebnis ausgegeben und die Bearbeitung beendet werden:

```
...
/@/ {exit}
...
END {print ergebnis}
```

for - Gezählte Wiederholung

Mit der *for*-Schleife wird eine Anweisung wiederholt, solange eine Bedingung erfüllt ist.

Syntax

for(ausdruck1**;** ausdruck2**;** ausdruck3**)** anweisung

ausdruck1
: Ausdruck (siehe *Ausdrücke*).
ausdruck1 wird einmal bei Start der *for*-Anweisung ausgewertet.
Beispiel: `i=1`

ausdruck2
: Ausdruck (siehe *Ausdrücke*).
ausdruck2 wird vor jedem Schleifendurchgang ausgewertet. *anweisung* wird nur ausgeführt, falls *ausdruck2* ungleich 0 (wahr) ergibt, ansonsten wird die Wiederholung beendet.
Beispiel: `i<10`

ausdruck3
: Ausdruck (siehe *Ausdrücke*).
ausdruck3 wird nach jedem Schleifendurchgang ausgewertet.
Beispiel: `i++`

anweisung
: Anweisung, die bei jedem Schleifendurchgang ausgeführt wird. Wenn mehrere Anweisungen ausgeführt werden sollen, müssen sie mit geschweiften Klammern {} zusammengefasst werden.

Beispiel

Die Felder des aktuellen Satzes sollen in umgekehrter Reihenfolge ausgegeben werden.

```
{for(i=NF; i>0; i--) print $i}
```

for - Array-Durchlauf

Diese Form der *for*-Anweisung ist eine *awk*-Besonderheit zum Durchlaufen eines Arrays.

Syntax

for(index␣**in**␣array**)**␣anweisung

index
: Variable (siehe *Grundelemente*), die in unbestimmter Reihenfolge alle Elemente des Arrays *array* durchläuft. Der Index kann numerisch oder alphanumerisch sein.

array
: Array, das durchlaufen wird.

anweisung
: Anweisung, die für jedes Array-Element ausgeführt wird. Wenn mehrere Anweisungen ausgeführt werden sollen, müssen sie mit geschweiften Klammern { } zusammengefasst werden.

Beispiel

Das Array *monate* enthält für jeden Monat die Anzahl von Tagen. Ein Array-Element wird mit dem Monatsnamen indiziert, z.B.
monate["Januar"]=31.
Durch folgendes *awk*-Programm wird für jeden Monat der Monatsname und die Anzahl von Tagen ausgegeben.

```
$ awk ' BEGIN { monate["Januar"]=31;    \
>               monate["Februar"]=28;   \
>               monate["Maerz"]=31;     \
>               monate["April"]=30;     \
>               monate["Mai"]=31;       \
>               monate["Juni"]=30;      \
>               monate["Juli"]=31;      \
>               monate["August"]=31 }   \
>       END { for(i in monate) print "Monat",i,"hat",monate[i],"Tage" } '

Monat Mai hat 31 Tage
Monat August hat 31 Tage
Monat Juli hat 31 Tage
Monat April hat 30 Tage
Monat Juni hat 30 Tage
Monat Januar hat 31 Tage
Monat Maerz hat 31 Tage
Monat Februar hat 28 Tage
```

if - Bedingte Anweisung

Bei der *if*-Anweisung wird eine Anweisung abhängig von einer Bedingung ausgeführt.

Syntax

if(ausdruck**)**␣anweisung1␣[**else**␣anweisung2]

ausdruck
: Ausdruck (siehe *Ausdrücke*), der die Bedingung angibt. Wenn *ausdruck* ungleich 0 (wahr) ergibt, wird *anweisung1* ausgeführt.

anweisung1
: Anweisung, die ausgeführt wird, falls *ausdruck* wahr ist. Wenn mehrere Anweisungen ausgeführt werden sollen, müssen sie mit geschweiften Klammern { } zusammengefasst werden.

anweisung2
: Anweisung, die ausgeführt wird, falls *ausdruck* falsch ist. Wenn mehrere Anweisungen ausgeführt werden sollen, müssen sie mit geschweiften Klammern { } zusammengefasst werden.

Beispiel Wenn Feld 1 größer als 2 ist, dann sollen die Felder 2 und 3 ausgegeben werden, ansonsten die Felder 4 und 5:

```
{ if($1 > 2) print $2, $3; else print $4, $5 }
```

next - Mit dem nächsten Eingabesatz fortfahren

awk unterbricht die Bearbeitung des aktuellen Satzes; die Anweisungen, die *next* folgen, werden nicht ausgeführt. Dann liest *awk* den nächsten Eingabesatz ein. *NR*, *NF*, *FNR*, *$0* und *$1* bis *$NF* werden neu gesetzt.

Unterschied zur *getline*-Funktion:

getline macht den nächsten Satz zum aktuellen Satz; die Anweisungen, die *getline* folgen, werden mit den Werten des nächsten Satzes für die $-Variablen und für *NR*, *NF* und *FNR* ausgeführt.

Syntax **next**

Beispiel Sätze, die mit . beginnen, sollen ignoriert werden:

```
{ if ($1 ~/^\./) next }
```

while - Schleife

Mit der *while*-Schleife wird eine Anweisung wiederholt, solange eine Bedingung erfüllt ist.

Syntax **while(**ausdruck**)**␣anweisung

ausdruck
: Ausdruck (siehe *Ausdrücke*), der die Bedingung angibt.

anweisung
: Anweisung, die bei jedem Schleifendurchgang ausgeführt wird. Wenn mehrere Anweisungen ausgeführt werden sollen, müssen sie mit geschweiften Klammern { } zusammengefasst werden.

Beispiel Alle Eingabefelder werden ausgegeben; jedes Feld wird in eine eigene Ausgabezeile geschrieben:

```
{ i = 1;
  while (i <= NF) {
      print $i
      i++
  }
}
```

Funktionen

awk stellt eine Reihe vordefinierter Funktionen zur Verfügung, bietet aber auch die Möglichkeit, eigene Funktionen zu definieren:

Syntax

function name**(**arg,...**)** **{**anweisungen**}**

Vor *{anweisungen}* darf ein Neue-Zeile-Zeichen stehen. Leerzeilen innerhalb der geschweiften Klammern *{...}* sind ebenfalls erlaubt. Eine Funktionsdefinition steht im Hauptteil eines *awk*-Programms gleichberechtigt neben *auswahlbedingung {aktion}*.

Funktionsaufrufe dürfen innerhalb eines Aktionsteils an beliebiger Stelle in einem Ausdruck stehen, nicht jedoch vor der Funktionsdefinition. Beim Aufruf darf zwischen dem Funktionsnamen und der öffnenden runden Klammer kein Leerzeichen stehen.
Funktionsaufrufe können geschachtelt werden, Funktionen dürfen rekursiv aufgerufen werden.

Bei den meisten Funktionen müssen Sie die Argumente nicht in Klammern einschließen. Klammern sind aber empfehlenswert, weil sie die Lesbarkeit verbessern. Übergeben Sie ein Array als Argument, dann wird ein Verweis auf das Array übergeben (call by reference) - Sie können aus der Funktion die Elemente des Arrays verändern. Bei skalaren Argumenten wird zur Übergabe der Wert der Variable kopiert (call by value) - Sie können den Wert der Variablen aus der Funktion heraus nicht ändern. Argumente haben einen lokal auf die Funktion beschränkten Geltungsbereich, während alle anderen Variablen immer einen globalen haben. Benötigen Sie lokale Variablen in einer Funktion, dann definieren Sie diese am Ende der Argumentliste in der Funktionsdefinition. Jede Variable der Argumentliste, für die kein aktuelles Argument existiert, ist eine lokale Variable, die mit dem Wert *0* vorbelegt ist.

Wie in C kann es Funktionen geben, die ein Ergebnis liefern (z.B. *exp*) und solche mit prozeduralem Charakter (z.B. Ausgabefunktionen).
Die Anweisung *return* kann mit oder ohne Wertangabe benutzt werden, oder ganz wegfallen - dann ist der Rückgabewert undefiniert, falls darauf zugegriffen werden sollte.

Beispiel

Die Funktion *suche* sucht in dem Array *allenamen* nach der Zeichenkette *wer* und gibt den Index oder *-1* zurück. Dabei wird das 3. Argument *lauf* als lokale Variable verwendet.

```
   ...
{ print $1, suche($1, allenamen) }
   ...
function suche(wer, allenamen, lauf)
{
   for (lauf=0; allenamen[lauf]; lauf++)
      if (index(allenamen[lauf], wer) == 1
          && length(allenamen[lauf]) == length(wer))
             return lauf
   return -1
}
```

Vordefinierte Funktionen

– **Eingabefunktion**

getline	Satz einlesen

– **Ausgabefunktionen**

print([arg,...])	Standard-Ausgabefunktion
printf(format [arg,...])	Formatierte Ausgabe

– **Arithmetische Funktionen**

atan2(y,x)	Arcustangens von *y/x*
cos(x)	Cosinus
exp(x)	Exponentialfunktion
int(x)	Ganzzahliger Anteil
log(x)	Natürlicher Logarithmus
rand()	Liefert eine Zufallszahl
sin(x)	Sinus
sqrt(x)	Quadratwurzel
srand([x])	Setzt den Anfangs-Berechnungswert für *rand()*

– **Zeichenketten-Funktionen**

gsub(re,repl[,in])	Globale Substitutionsfunktion
index(zk1,zk2)	Erstes Vorkommen einer Teilzeichenkette
length([zk])	Länge einer Zeichenkette
match(zk,re)	Prüft, ob *zk* zum regulären Ausdruck *re* passt
split(zk,array,[sep])	Aufteilen einer Zeichenkette
sprintf(format,e1,e2,...)	Formierte Ausgabe in eine Variable
sub(re, repl[,in])	Substitutionsfunktion
substr(zk,m,[n])	Teilzeichenkette bestimmen
tolower(s)	Umwandlung in Kleinbuchstaben
toupper(s)	Umwandlung in Großbuchstaben

– **Allgemeine Funktionen**

close(expr)	Datei oder Pipe schließen
system(expr)	Shell-Kommando aufrufen

Die Funktionen sind im Folgenden alphabetisch beschrieben. Bei jeder Funktion ist angegeben, welche Argumente Sie angeben können: Sie können als Argument eine Konstante angeben oder einen Ausdruck (siehe *Ausdrücke*). *awk* wertet die Argument-Ausdrücke zuerst aus und wendet dann die Funktion auf die berechneten Ergebnisse an.

atan2 - Arcustangens
atan2 berechnet den Arcustangens des Quotienten zweier Zahlen: *atan2(y,x)* liefert den Arcustangens von *y/x*.

Syntax **atan2**(y,x)

y,x Zahlen, für deren Quotient der Arcustangens berechnet werden soll.

close - Datei oder Pipe schließen
close schließt die angegebene Datei bzw. Pipe.

Syntax **close**(expr)

expr
Name der Datei oder der Pipe, die geschlossen werden soll (siehe bei der Beschreibung der Funktionen *print* und *printf* den Abschnitt über Ausgabeumlenkung).

cos - Cosinus
cos berechnet den Cosinus einer Zahl.

Syntax **cos**(x)

x Zahl, für die der Cosinus berechnet werden soll.

exp - Exponentialfunktion
exp berechnet *e* hoch *x*.

Syntax **exp**(x)

x Zahl, für die e^x berechnet werden soll.

getline - Einen Satz einlesen
awk liest einen Satz ein (siehe *next*, *Ablaufanweisung*).

getline hat mehrere Formate. Die Formate von *getline* haben folgende Rückgabewerte:

1 bei fehlerfreiem Lesevorgang
0 bei Dateiende
-1 bei einem Fehler

Syntax **getline**

awk liest in *$0* den nächsten Eingabesatz aus der Eingabedatei ein. *NR*, *NF*, *FNR*, *$0* und *$1* bis *$NF* werden neu gesetzt.

Beispiel Wenn ein Satz %%% enthält, wird der nächste Satz eingelesen, d.h. Eingabesätze, die %%% enthalten, werden ignoriert.

```
/%%%/ {getline}
```

Syntax **getline␣<␣**datei

awk liest in *$0* einen Satz aus der Datei *datei* ein. *NF*, *$0* und *$1* bis *$NF* werden neu gesetzt.

datei
Name der Datei, aus der gelesen werden soll.

Syntax **getline␣**var

awk liest in die Variable *var* den nächsten Eingabesatz aus der Eingabedatei ein. *NR* und *FNR* werden neu gesetzt.

var Variable, in die der nächste Satz eingelesen werden soll.

Syntax **getline␣**var**␣<␣**datei

awk liest in die Variable *var* einen Satz aus der Datei *datei* ein. *NR*, *NF*, *FNR*, *$0* und *$1* bis *$NF* werden nicht verändert.

var Variable, in die der Satz eingelesen werden soll.

datei
Name der Datei, aus der gelesen werden soll.

Syntax kommando␣|␣**getline**␣[var]

Die Ausgabe des Kommandos *kommando* wird nach *getline* umgelenkt. Mit jedem *getline*-Aufruf in diesem Format liest *awk* in *$0* bzw. *var* jeweils die nächste Zeile aus der Ausgabe von *kommando* ein.
Ist *var* angegeben, werden *NR*, *NF*, *FNR*, *$0* und *$1* bis *$NF* nicht verändert. Ist *var* nicht angegeben, werden *NF*, *$0* und *$1* bis *$NF* neu gesetzt.

Dieses Konstrukt wirkt wie der Aufruf der C-Funktion *popen()* mit Modus *r*.

var Variable, in die der Satz eingelesen werden soll.

var nicht angegeben:
Der Satz wird in *$0* eingelesen.

kommando
Name des Kommandos, aus dessen Ausgabe gelesen werden soll.

gsub - Globale Substitutionsfunktion
gsub ersetzt alle Zeichenketten in *$0* bzw. in *in*, die zu dem erweiterten regulären Ausdruck *re* passen, durch die Zeichenkette *repl*.
gsub liefert die Anzahl der Ersetzungen zurück.

Syntax **gsub**(re,repl[,in])

re Erweiterter regulärer Ausdruck, der als Muster für die Ersetzung dienen soll.

repl
Zeichenkette, die die zu *re* passenden Zeichenketten ersetzen soll.

in Zeichenkette, in der die Ersetzung stattfinden soll.

in nicht angegeben:
Die Zeichenketten werden in *$0* ersetzt.

index - Teilzeichenkette suchen
index sucht eine Teilzeichenkette in einer Zeichenkette. Wenn die Teilzeichenkette vorkommt, gibt *index* die Anfangsposition (in Zeichen, nummeriert ab 1) des ersten Vorkommens zurück. Wenn die Teilzeichenkette nicht vorkommt, gibt *index* 0 zurück.

Syntax **index**(zk1,zk2)

zk1
Zeichenkette, in der *index* die Teilzeichenkette sucht.

zk2
Teilzeichenkette, die *index* sucht.

Beispiel Vergleich der Zeichenketten „ToTo-LoTo“ mit „To“

```
index("ToTo-LoTo","To") ist 1.
```

int - Ganzzahliger Anteil
int gibt die größte ganze Zahl zurück, die kleiner oder gleich dem Argument ist.

Syntax **int**(x)

x Zahl, deren ganzzahliger Anteil bestimmt werden soll.

length - Länge bestimmen
length bestimmt die Länge einer Zeichenkette.

Syntax **length**[(zk)]

zk *length* bestimmt die Länge der Zeichenkette *zk*.

zk nicht angegeben:
length bestimmt die Länge des aktuellen Eingabesatzes *$0*.

log - Logarithmus
log berechnet den natürlichen Logarithmus zur Basis *e*.

Syntax **log**(x)

x Zahl, für die der natürliche Logarithmus berechnet werden soll.

match - Prüffunktion für reguläre Ausdrücke
match prüft, ob eine Zeichenkette in *zk* zu dem erweiterten regulären Ausdruck *re* passt. *match* liefert die Stelle in *zk* (in Zeichen, nummeriert ab 1) zurück, an der die zu *re* passende Zeichenkette anfängt; wenn keine Zeichenkette in *zk* passt, liefert *match* 0 zurück.
Die Variable *RSTART* wird auf den Return-Wert von *match* gesetzt. Die Variable *RLENGTH* wird auf die Länge der passenden Zeichenkette gesetzt bzw. auf -1, falls keine Zeichenkette passt.

Syntax **match**(zk,re)

zk Zeichenkette, in der die Mustersuche stattfinden soll.

re Erweiterter regulärer Ausdruck.

print - Standard-Ausgabefunktion

print ist die Standard-Ausgabefunktion. *print* gibt entweder den aktuellen Satz oder die angegebenen Argumente aus und schließt die Ausgabe mit dem Ausgabesatztrenner *ORS* (ORS - Output Record Separator) ab. Nähere Beschreibung des Ausgabeformats siehe Seite 145.

Syntax

print(arg1[[,]arg2]...)[umlenkung]

Kein Argument angegeben:
print gibt den aktuellen Eingabesatz auf die Standard-Ausgabe aus.

arg1arg2
Argumente, die ausgegeben werden sollen. *print* wertet die Argument-Ausdrücke aus und hängt die Ergebnisse in der Reihenfolge der Argumente hintereinander.

arg1,arg2
Argumente, die ausgegeben werden sollen. *print* gibt die ausgewerteten Argument-Ausdrücke in der angegebenen Reihenfolge aus und trennt sie durch den Ausgabe-Feldtrenner *OFS* (OFS - Output Field Separator), sofern sie durch Kommata getrennt in der *print*-Anweisung stehen.

umlenkung
Sie können die Ausgabe in eine Datei umlenken oder an ein Programm weiterreichen. Maximal können Sie 10 Ausgabedateien verwenden.

umlenkung kann sein:

>, >>, Name eines Programms

>datei
Die Ausgabe wird in die Datei *datei* geschrieben. Der alte Inhalt von *datei* wird beim ersten *print*-Aufruf gelöscht. Alle weiteren *print*- oder *printf*-Ausgaben auf *datei* in demselben *awk*-Programm werden angehängt. *datei* bleibt bis zum Ende des *awk*-Programms geöffnet, falls sie nicht explizit geschlossen wird.

>>datei
Die Ausgabe wird an den bisherigen Inhalt der Datei *datei* angehängt. *datei* bleibt bis zum Ende des *awk*-Programms geöffnet, falls sie nicht explizit geschlossen wird.

Iprog
Die Ausgabe wird über eine Pipe an das Programm *prog* geschickt.

Sie dürfen innerhalb eines *awk*-Programms nur eine Pipe zu *prog* öffnen. Auf diese Pipe können mehrere *print*- oder *printf*-Ausgaben erfolgen.
Dieses Konstrukt wirkt wie der Aufruf der C-Funktion *popen()* [4] mit Modus *w*.
Die Pipe bleibt bis zum Ende des *awk*-Programms geöffnet, falls sie nicht explizit geschlossen wird.

Den Dateinamen bzw. Programmnamen können Sie direkt angeben, eingeschlossen in "...", oder durch eine Variable, die den Dateinamen enthält.

Achtung!
Wenn Sie auf die Eingabedatei umlenken, wird die Eingabedatei ohne Warnung zerstört!

Ausgabeformat

print gibt ganze Zahlen dezimal aus und Zeichenketten in ihrer vollen Länge. Ansonsten ist das Ausgabeformat durch folgende vordefinierte Variablen festgelegt:

OFS - Ausgabe-Feldtrenner
OFS ist standardmäßig ein Leerzeichen. Sie können *OFS* ein beliebiges Zeichen zuweisen und damit den Ausgabe-Feldtrenner ändern.

ORS - Ausgabe-Satztrenner
ORS ist standardmäßig das Neue-Zeile-Zeichen. Sie können *ORS* ein beliebiges Zeichen zuweisen und damit den Ausgabe-Satztrenner ändern.

OFMT - Gleitkomma-Ausgabeformat
(OFMT - output format) *OFMT* enthält das Ausgabeformat für Gleitkommawerte. *OFMT* ist standardmäßig "%.6g". Das bedeutet, dass Gleitkommazahlen mit maximal 6 Stellen hinter dem Dezimalpunkt ausgegeben werden. Sie können *OFMT* ein anderes *printf*-Format für Gleitkommazahlen zuweisen (siehe *printf - Formatierte Ausgabe*).

Beispiel Erstes und zweites Feld, getrennt durch Leerzeichen, ausgeben:

```
{print $1,$2}
```

Beispiel Erstes und zweites Feld ohne Ausgabe-Feldtrenner aneinanderhängen:

```
{print $1$2}
```

oder

```
{print $1 $2}
```

printf - Formatierte Ausgabe
printf ist die Ausgabefunktion für formatierte Ausgabe. Sie können das Ausgabeformat wie bei der Standard-C-Funktion *printf()* angeben.

Syntax **printf**(format,arg,...)[umlenkung]

format
Zeichenkette, die das Ausgabeformat enthält. Das Ausgabeformat besteht aus Zeichen und Formatangaben. Die darstellbaren Zeichen werden unverändert ausgegeben. Die im Abschnitt Grundelemente angegebenen Sonderzeichen werden gleich umgesetzt.

So wird z.B. durch *n* auf den Anfang der nächten Zeile positioniert.
Eine Formatangabe wird mit dem Prozent-Zeichen % eingeleitet. Die wichtigsten Formatelemente sind in der folgenden Liste zusammengestellt.

%c	Einzelnes Zeichen
%d	Ganzzahl dezimal
%e	Gleitkommazahl in Exponent-Darstellung, z.B. 5.234e+2
%f	Gleitkommazahl, z.B. 52.34
%g	%e oder %f, je nachdem, welche Darstellung kürzer ist
%o	Ganzzahl oktal (Basis 8)
%s	Zeichenkette
%u	Ganzzahl dezimal ohne Vorzeichen
%x	Ganzzahl hexadezimal (Basis 16)

arg
Argumente, die ausgegeben werden sollen.
printf wertet die Argument-Ausdrücke aus, ordnet sie den Formatangaben in *format* in der angegebenen Reihenfolge zu und gibt sie entsprechend formatiert aus.

- Wenn Formatangabe und Argument nicht zusammenpassen (Formatangabe numerisch, Argument alphanumerisch), wird 0 ausgegeben.
- Wenn mehr Argumente vorhanden sind als Formatangaben, werden die überzähligen Argumente ignoriert, d.h. nicht ausgegeben.
- Wenn mehr Formatangaben vorhanden sind als Argumente, wird eine Fehlermeldung ausgegeben.

umlenkung
Bezüglich Umlenkung gilt dasselbe wie bei *print*.

umlenkung nicht angegeben:
printf gibt auf die Standard-Ausgabe aus.

Beispiel Feld 1 wird als Dezimalzahl mit mindestens 2 Stellen ausgegeben, danach, getrennt durch **, Feld 2 als Zeichenkette mit mindestens 5 Zeichen und danach Neue Zeile:

```
{ printf("%2d**%5s\n", $1,$2) }
```

rand - Zufallszahl liefern
rand liefert eine Zufallszahl *r* zurück, für die gilt: 0 <= *r* < 1.

Syntax **rand**

Siehe auch *srand*.

sin - Sinus
sin liefert den Sinus einer Zahl.

Syntax **sin**(x)

x Zahl, deren Sinus berechnet werden soll.

split - Zeichenkette aufteilen
split teilt eine Zeichenkette in Teilzeichenketten auf und speichert die Teilzeichenkette als Elemente eines Arrays ab. Die Array-Elemente werden beginnend mit 1 aufsteigend indiziert.
split liefert die Anzahl der Elemente des Arrays zurück.

Syntax **split(**zk,array[,sep]**)**

zk Zeichenkette, die aufgeteilt werden soll.

array
Name des Ergebnis-Arrays.

sep
Erweiterter regulärer Ausdruck, der die Zeichen bestimmt, die in *zk* als Trennzeichen zwischen den Teilzeichenketten gelten sollen.

sep nicht angegeben:
FS gilt als Trennzeichen.

Beispiel Die Eingabe

```
{
      s=split("januar:februar:maerz", monate, ":");
      for(i=1; i<s; i++) print monate[i];
   }
```

erzeugt die Ausgabe:

```
januar
februar
maerz
```

sprintf - Formatierte Ausgabe in Zeichenkette
Bei *sprintf* ist die Formatierung wie bei *printf*. Es erfolgt aber keine Ausgabe. Stattdessen gibt *sprintf* die formatierte Ausgabe als Ergebnis-Zeichenkette zurück. Diese Zeichenkette können Sie z.B. einer Variablen zuweisen.

Syntax **sprintf**(format,arg,...)

format
Zeichenkette, die das Ausgabeformat enthält (siehe *Formatierte Ausgabe*).

arg
Argumente, die ausgegeben werden sollen (siehe *Formatierte Ausgabe*).

Beispiel Das folgende *awk*-Programmstück erzeugt diesselbe Ausgabe wie das Beispiel bei *printf*.

```
{ x = sprintf("%2d**%5s\n", $1,$2); print x }
```

sqrt - Quadratwurzel berechnen
sqrt berechnet die Quadratwurzel zu einer Zahl.

Syntax **sqrt**(x)

x Zahl, deren Quadratwurzel berechnet werden soll.

srand - Anfangs-Berechnungswert für die Funktion rand setzen
srand setzt den Anfangs-Berechnungswert für die Funktion *rand* auf die Zahl *x* bzw. auf die aktuelle Uhrzeit, falls kein Argument angegeben ist.

Syntax **srand**([x])

x Zahl, die als Anfangs-Berechnungswert für *rand* dienen soll.

sub - Globale Substitutionsfunktion
sub ersetzt die erste Zeichenkette in *$0* bzw. in *in*, die zu dem erweiterten regulären Ausdruck *re* passt, durch die Zeichenkette *repl*.
sub liefert die Anzahl der Ersetzungen zurück.

Syntax **sub**(re,repl[,in])

re Erweiterter regulärer Ausdruck, der als Muster für die Ersetzung dienen soll.

repl
Zeichenkette, die die zu *re* passenden Zeichenketten ersetzen soll.

in Zeichenkette, in der die Ersetzung stattfinden soll.
in nicht angegeben:
Die Zeichenketten werden in *$0* ersetzt.

substr - Teilzeichenkette bestimmen
substr liefert eine Teilzeichenkette einer Zeichenkette.

Syntax **substr(**zk,m[,n]**)**

zk Zeichenkette, aus der eine Teilzeichenkette ausgewählt werden soll.

m Position in *zk*, bei der die Teilzeichenkette beginnt. Die Zeichenpositionen werden beginnend mit 1 von links nach rechts aufsteigend durchnummeriert.

n Maximale Länge der Teilzeichenkette.

n nicht angegeben:
Die Teilzeichenkette geht bis zum Ende von *zk*.

Beispiel Die Eingabe

```
{
   x = substr("060795",3,2); print "Monat = "x
}
```

erzeugt die Ausgabe:

```
Monat = 07
```

system - Shell-Kommando ausführen
system führt das angegebene Shell-Kommando aus und liefert den Ende-Status des Kommandos zurück.

Syntax **system**(kommando)

kommando
Name des Shell-Kommandos, das ausgeführt werden soll.

Fehler Wenn das *awk*-Programm fehlerhaft ist, gibt *awk* Fehlermeldungen aus und beendet sich sofort. Die Fehlermeldungen enthalten die Fehlerursache, falls *awk* diese erkennt, sowie die Zeile des *awk*-Programms, in der *awk* den Fehler vermutet.
Typische Fehlermeldungen sind:

`awk: syntax error at source line xxx`
In der Zeile xxx des *awk*-Programms liegt ein Syntaxfehler vor.

`awk: illegal statement source line number xxx`
In der Zeile xxx des *awk*-Programms steht eine falsche Anweisung.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *awk*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt in geklammerten regulären Ausdrücken die Bedeutung von Zeichenbereichen, Äquivalenzklassen und Zeicheneinheiten fest, während *LC_CTYPE* die Bedeutung von Zeichenklassen festlegt. *LC_COLLATE* bestimmt das Verhalten von Vergleichsoperatoren beim Vergleich von Zeichenketten.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_NUMERIC Legt die Darstellung von Dezimalpunkt, Exponentzeichen und Tausendertrennzeichen fest.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

PATH Legt den Suchpfad fest, wenn ein Kommando durch *system(kommando)*, Ein- oder Ausgabepipes ausgeführt werden soll.

Beispiele zu awk

Beispiel 1 Sie wollen alle Eingabezeilen ausgeben, bei denen Feld 3 größer ist als Feld 5:

```
$ awk '$3 > $5' datei
```

Da keine Aktion angegeben ist, gibt *awk* standardmäßig die ausgewählten Zeilen aus.

Beispiel 2 Sie wollen jede 10. Zeile der Datei *datei* ausgeben:

```
$ awk '(NR % 10) == 0' datei
```

Beispiel 3 Sie wollen für jede Zeile das vorletzte und letzte Feld, getrennt durch Doppelpunkt, ausgeben:

```
$ awk 'BEGIN {OFS=":"}  \
>      {print $(NF-1), $NF}' datei
```

Bei Zeilen, die aus einem einzigen Feld bestehen, wird die ganze Zeile zweimal, getrennt durch Doppelpunkt, ausgegeben (zuerst *$0*, dann *$1*).

Beispiel 4 Sie wollen die Werte des ersten Feldes von allen Zeilen addieren. Zum Schluss sollen Summe und Mittelwert ausgegeben werden.

```
$ awk '{s += $1} \
>      END {print "Summe: ", s, "Mittelwert: ", s/NR}'\
>      datei
```

Beispiel 5 Präprozessor-*if*-Anweisung heraussuchen, d.h. einen Bereich auswählen, dessen erste Zeile mit #if und dessen letzte Zeile mit #endif beginnt:

```
$ awk '/^#if/, /^#endif/' datei
```

Beispiel 6 Sie wollen alle Zeilen ausgeben, deren erstes Feld sich von dem ersten Feld der vorhergehenden Zeile unterscheidet:

```
$ awk '$1 != vorher { print; vorher = $1 }' datei
```

Beispiel 7 *datei* enthält eine Tabelle mit Daten über Jugendliche. In dem zweiten Feld steht entweder *Schueler*, *Student*, *Lehrling* oder *Sonstige*. Für eine Statistik sollen Schüler und Studenten gezählt werden:

```
$ awk '$2 ~ /Schueler/ {haeuf["Schueler"]++} \
>     $2 ~ /Student/ {haeuf["Student"]++} \
>     END {print "Schueler:" haeuf["Schueler"]; \
>          print "Student:" haeuf["Student"]} ' datei
```

Beispiel 8 In der Datei *inhalt* steht ein mit Dezimalklassifikation gegliedertes Inhaltsverzeichnis eines Textes. Die Liste hat folgendes Format:

```
1. Vorwort
2. Einleitung
3. Das Schachspiel
3.1. Geschichte
3.2. Regeln
3.2.1 Aufstellen der Figuren
.
.
.
```

```
4. Das Damespiel
4.1. Geschichte
.
.
.
8. Register
```

Mit folgendem *awk*-Programm bringen Sie diese Liste in ein übersichtlicheres Format.

```
$ awk '{$1=$1"       ";    \
>      $1=substr($1,1,6);  \
>      print $0}' inhalt  >> inh.form
```

Die Aufbereitung der Ausgabezeilen erfolgt in folgenden Schritten: Zuerst werden an das erste Feld sechs Leerzeichen angehängt (*$1=$1"␣␣␣␣␣␣"*). Dann wird das erste Feld auf Länge sechs gekürzt. Damit ist in jeder Zeile das erste Feld 6 Zeichen lang und Feld 2 beginnt immer auf Spalte sieben. Sie erhalten folgende Ausgabe in der Datei *inh.form*:

```
1.    Vorwort
2.    Einleitung
3.    Das Schachspiel
3.1.  Geschichte
3.2.  Regeln
3.2.1 Aufstellen der Figuren
.
.
.
4.    Das Damespiel
4.1.  Geschichte
.
.
.
8.    Register
```

Beispiel 9 Das folgende *awk*-Programm in der Datei *prog* gibt für jeden Satz die Feldanzahl und die Felder aus. Der Satztrenner wird auf das Dollarzeichen *$* umdefiniert. Feldtrenner sind daher Leer-, Tabulator- und Neue-Zeile-Zeichen:

```
BEGIN { RS="$"; printf "Satz\tAnz" }
      { printf ("\n%4d\t%3d\t", NR, NF);
        for(i=1; i<=NF; i++) printf "%s:", $i }
END {print"\n"}
```

Die Datei *text* enthält folgenden Text:

```
erster Satz$  zweiter   Satz     $
$
vierter     und  letzter
Satz$
```

Der Aufruf:

```
$ awk -f prog text
```

liefert:

```
Satz    Anz
   1      2      erster:Satz:
   2      2      zweiter:Satz:
   3      0
   4      4      vierter:und:letzter:Satz:
   5      0
```

Beispiel 10 Sie ändern die Datei *text* wie folgt:

```
&&
erster&&Satz$zweiter Satz$$vierter
und&
letzter

Satz&
```

und rufen *awk* auf, dieses Mal mit der Option *-F*, um den Feldtrenner auf & zu ändern.

```
$ awk -F"&"  -f prog text
```

Die Ausgabe ergibt:

```
Satz    Anz
   1      6    :::erster::Satz:
   2      1    zweiter Satz:
   3      0
   4      8    vierter:und::letzter::Satz:::
```

Dieses Beispiel verdeutlicht die Feldeinteilung bei einem Nicht-Standard-Feldtrenner. Die erste Zeile (&&) der Datei *text* ist Teil des ersten Satzes und ergibt z.B. 3 Felder, weil bei einer Folge von Trennern jeder Trenner zählt und Neue Zeile implizit auch als Trenner gilt (2 & + 1 Neue Zeile = 3).

Siehe auch *egrep, fgrep, grep, lex, sed*

basename Dateinamen vom Pfad trennen (return non-directory portion of pathname)

Mit *basename* können Sie
– aus dem Pfadnamen einer Datei den einfachen Dateinamen dieser Datei erzeugen
– aus Dateinamen beliebige Endungen entfernen

basename entfernt aus einer beim Aufruf angegebenen Zeichenkette alle Zeichen bis einschließlich des letzten vorkommenden Schrägstrichs /. Den Rest gibt *basename* auf die Standard-Ausgabe aus. Sie können damit den einfachen Dateinamen einer Datei vom Pfad-Präfix trennen. Wenn Sie beim Aufruf von *basename* zusätzlich die Endung der Zeichenkette angeben, entfernt *basename* auch diese Endung. *basename* findet nützliche Anwendung in Shell-Prozeduren.

Syntax **basename**[␣zeichenkette[␣endung]]

zeichenkette
zeichenkette ist eine beliebige Zeichenkette.
basename entfernt aus *zeichenkette* alle Zeichen bis einschließlich des letzten Schrägstrichs /. Den Rest gibt *basename* auf die Standard-Ausgabe aus. Zeichenketten, die das Zeichen / nicht enthalten, werden unverändert ausgegeben.

zeichenkette nicht angegeben:
es wird nur ein Punkt auf die Standard-Ausgabe ausgegeben.

endung
endung ist eine beliebige Zeichenkette.
Wenn *endung* mit dem Ende von *zeichenkette* übereinstimmt, wird *zeichenkette* ohne *endung* ausgegeben.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *basename*:

LANG	Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.
LC_ALL	Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.
LC_CTYPE	Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
: Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH
: Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Aus */home/katharina/programm* soll der Name *prog* erzeugt werden:

```
$ basename /home/katharina/programm ramm
prog
```

Beispiel 2 Die folgende Shell-Prozedur übersetzt ein C-Quellprogramm. *basename* erzeugt den Namen des übersetzten Programms aus dem Dateinamen, der beim Aufruf der Shell-Prozedur übergeben wird. Das übersetzte Programm wird in einer ausführbaren Datei im aktuellen Dateiverzeichnis abgelegt. Die Shell-Prozedur steht in der Datei *uebersetze*.

Inhalt von *uebersetze*:

```
c89 -o 'basename $1 .c' $1
```

Wenn Sie *uebersetze* wie folgt aufrufen:

```
$ uebersetze /home/anna/cprogs/tab.c
```

wird dem Kommando *c89* (Steuerprogramm zum Übersetzen und Binden von C-Programmen, siehe *c89* [5]) im Stellungsparameter $1, der Name der C-Quelldatei übergeben.

Die Shell ersetzt den Operanden für die Option *-o* von *c89* durch das Ergebnis des Aufrufs von *basename*. Der Name der ausführbaren Datei ist *tab*.

Siehe auch *dirname*, *ed*

batch Kommandos zu einer späteren Zeit ausführen (execute commands at a later time)

batch liest Kommandos von der Standard-Eingabe, setzt sie in eine Warteschlange und führt sie aus, sobald die Systembelastung es zulässt.

Wenn Sie die Standard-Ausgabe und Standard-Fehlerausgabe der auszuführenden Kommandos nicht umgelenkt haben, wird Ihnen die Ausgabe mit *mailx* geschickt. Die Umgebungsvariablen, das aktuelle Dateiverzeichnis, die für neue Dateien gültigen Zugriffsrechte (siehe *umask*) und die maximal zulässige Dateigröße (siehe *ulimit*) bleiben erhalten. Offene Dateien und Prioritäten werden nicht vererbt. Das Kommando *trap* (eingebautes Shell-Kommando zum Abfangen von Signalen) wird aufgehoben.

batch schreibt die Auftragsnummer und die berechnete Ausführungszeit auf die Standard-Fehlerausgabe.

Aufträge, die mit *batch* erteilt werden, bleiben auch dann erhalten, wenn der Auftraggeber mit *exit* die POSIX-Shell beendet oder das POSIX-Subsystem beendet wird. Die Aufträge müssen nicht neu gestartet werden.

batch verhält sich genau wie *at -qb* ohne zusätzliche Option.

Vor dem Aufruf beachten

Die Benutzerkennung muss eine Standardabrechnungsnummer für den *rlogin*-Zugang haben. Diese kann mit den Kommandos ADD-USER, MODIFY-USER-ATTRIBUTES oder ADD-POSIX-USER zugewiesen werden.

Wenn die Datei */usr/lib/cron/at.allow* existiert, dann dürfen Sie das Kommando *batch* nur dann aufrufen, wenn Ihre Benutzerkennung in dieser Datei steht.

Wenn die Datei */usr/lib/cron/at.allow* nicht existiert, dann dürfen Sie das Kommando *batch* nur dann aufrufen, wenn Ihre Benutzerkennung *nicht* in der Datei */usr/lib/cron/at.deny* steht.

Wenn weder */usr/lib/cron/at.allow* noch */usr/lib/cron/at.deny* existieren, dann darf nur der POSIX-Verwalter *batch* aufrufen.

Existiert z.B. nur die leere deny-Datei, so dürfen alle Benutzer *batch* aufrufen.

Die allow/deny-Dateien darf nur der POSIX-Verwalter anlegen und ändern. Sie enthalten pro Zeile eine Benutzerkennung.

Syntax

batch ↵
kommando ... ↵
END oder @@d

kommando
: Beliebiges Kommando oder Shell-Prozedur. Sie können *kommando* mehrmals, jeweils durch Strichpunkt ; oder Neue-Zeile-Zeichen getrennt, angeben. Eine so entstandene Kommandoliste läuft unter einer Auftragsnummer.

Fehler

`at: you are not authorized to use at. Sorry.`
Sie dürfen *batch* nicht aufrufen. Siehe „Vor dem Aufruf beachten“ auf Seite 156.

Datei

/usr/lib/cron/at.allow
Liste der Benutzerkennungen mit Ausführrecht für *batch*. In jeder Zeile steht jeweils eine Benutzerkennung.

/usr/lib/cron/at.deny
Liste der Benutzerkennungen ohne Ausführrecht für *batch*. In jeder Zeile steht jeweils eine Benutzerkennung.

/var/spool/cron/atjobs
Dateiverzeichnis, in dem die noch nicht bearbeiteten *batch*-Aufträge in einzelnen Dateien aufgelistet werden. Für jeden *batch*-Auftrag gibt es eine eigene Datei mit dem Dateinamen *auftragsnummer.b*.

Variable

SHELL
bestimmt den Namen eines Kommandointerpreters, um den *at*-Auftrags aufzurufen. Ist diese Variable nicht gesetzt oder hat den Wert 0, wird *sh* verwendet. Ist diese Variable auf einen Wert als *sh* gesetzt, ist das weitere Verhalten implementierungsabhängig.

TZ
bestimmt die Zeitzone. Der *at*-Auftrag wird zum Zeitpunkt *-t␣zeit* oder *zeitpunkt* ausgeführt, relativ zu der in *TZ* bestimmten Zeitzone. Wenn *zeitpunkt* eine Zeitzone bestimmt, wird *TZ* überschrieben. Wenn *zeitpunkt* keine Zeitzone bestimmt und *TZ* nicht gesetzt ist oder den Wert 0 hat, wird der entsprechende Standardwert verwendet.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *batch*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_TIME Legt das Format und Inhalt der Datums- und Zeitangaben fest, die von *batch* geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Im folgenden Beispiel wird die Standard-Eingabe umgelenkt, so dass *batch* seinen Auftrag aus der Datei *auftraege* liest:

```
$ batch < auftraege
job 604763316.b at Mon Mar  9 14:48:36 2009
```

batch führt die Aufträge, die in der Datei *auftraege* stehen, der Reihe nach als Hintergrundprozesse aus. Nach Beendigung können Sie sich das Ergebnis mit *mailx* auf dem Bildschirm ausgeben lassen.

Siehe auch *at*, *crontab*, *mailx*, *ulimit*

bc Arithmetische Sprache (arbitrary-precision arithmetic language)

Mit *bc* können Sie rechnen. *bc* ist ein interaktives Programm für eine C-ähnliche Eingabe-Sprache.

Syntax **bc**[␣**-l**][␣datei...]

-l *-l* steht für */usr/lib/lib.b*. In dieser Bibliothek sind *bc*-Programme für verschiedene mathematische Funktionen enthalten. Sie müssen die Option *-l* angeben, wenn Sie eine der folgenden Funktionen verwenden:

s(x)	Sinus
c(x)	Cosinus
e(x)	Exponentialfunktion mit Basis e
l(x)	natürlicher Logarithmus
a(x)	Arcustangens
j(n,x)	Bessel-Funktion n-ter Ordnung

datei
Name der Datei mit einem *bc*-Programm. Sie können mehrere Dateien angeben. Nachdem alle Anweisungen aus allen Dateien abgearbeitet sind, liest *bc* von der Standard-Eingabe. Sie können dann weitere Anweisungen eingeben.

datei nicht angegeben:
bc liest von der Standard-Eingabe.

Elemente von bc-Programmen

Ein *bc*-Programm besteht aus
– Definitionen
– Anweisungen
– Kommentaren

Bei der Beschreibung des Aufbaus von *bc*-Programmen werden folgende Bezeichnungen verwendet:

L (L - letter) steht für einen der Buchstaben a-z

E (E - expression) steht für einen Ausdruck

S (S - statement) steht für eine Anweisung

Kommentare

Kommentare werden wie in C-Programmen in /*...*/ eingeschlossen.

Anweisungen

Anweisungen in *bc*-Programmen können sein:

- Ausdrücke (siehe *Ausdrücke*)

  Das Ergebnis einer Anweisung, die ein Ausdruck ist, wird ausgegeben, es sei denn, der Hauptoperator ist ein Zuweisungsoperator.

- Blöcke: {S; ...; S}

- Auswahlanweisung: if (E) S

  Wenn der Ausdruck E wahr ist, d.h. einen Wert ungleich 0 hat, dann wird die Anweisung S ausgeführt.

- Wiederholungsanweisungen:

  – while (E) S
  Der Ausdruck E wird ausgewertet und wenn sein Wert ungleich 0 ist, dann wird die Anweisung S ausgeführt. Danach wird E erneut ausgewertet und S ggf. wiederum ausgeführt. Dies wiederholt sich, solange der Wert von E ungleich 0 ist.

  – for (E; E; E) S
  Zuerst wird der erste Ausdruck ausgewertet. Danach wird der zweite Ausdruck ausgewertet und falls dessen Wert ungleich 0 ist, wird die Anweisung S ausgeführt. Schließlich wird der dritte Ausdruck ausgewertet. Anschließend wird wieder der zweite Ausdruck ausgewertet und gegebenenfalls Anweisung S ausgeführt usw. Anders als in C-Programmen muss eine for-Anweisung immer drei Ausdrücke enthalten.

- Sprunganweisung: break

  Die Anweisung break darf nur innerhalb einer Wiederholungsanweisung benutzt werden. Sie sorgt für den Abbruch der nächstgelegenen while- oder for-Anweisung. Das Programm wird mit der Anweisung fortgesetzt, die auf die abgebrochene Wiederholungsanweisung folgt.

- Abbruchanweisung: quit

  Die Anweisung quit beendet die Ausführung des *bc*-Programms. quit wird sofort beim Einlesen interpretiert, nicht erst bei der Ausführung des *bc*-Programms.

Anweisungen können Sie durch einen Strichpunkt oder ein Neue-Zeile-Zeichen voneinander trennen.

Beispiel

Das folgende *bc*-Programm beendet sich sofort, ohne dass der Wert von a ausgegeben wird:

```
a=5
if (a>10) quit
a)
```

Ausdrücke

Ausdrücke bestehen aus Operanden und Operatoren.

Operanden sind Namen oder beliebig lange Zahlen, wahlweise mit Vorzeichen und Dezimalpunkt.

Namen

L	einfache Variablen
L	Funktionsnamen
L[E]	Feldelemente
ibase	Basis für eingelesene Zahlen, Standard (keine Angabe): 10
obase	Basis für ausgegebene Zahlen, Standard (keine Angabe): 10
scale	Anzahl der Nachkommastellen, Standard (keine Angabe): 0

Wenn Felder als Argumente einer Funktion verwendet oder als auto-Variablen definiert werden, so müssen Sie hinter den Namen des Feldes leere eckige Klammern schreiben.

Eine einfache Variable, ein Feld und eine Funktion können denselben Namen haben. Alle Variablen sind global im ganzen *bc*-Programm.

Weitere Operanden

(E)	Ergebnis von E
sqrt(E)	Quadratwurzel von E
length(E)	Zahl der signifikanten Dezimalziffern von E
scale(E)	Zahl der Nachkommastellen von E
L(E, ...,E)	

Operatoren

+ – * /	Addition, Subtraktion, Multiplikation, Division
^	Potenzieren

%	Rest bei ganzzahliger Division (Modulo), kann jedoch auch für Gleitkommazahlen verwendet werden.
++ --	Inkrement- und Dekrement-Operator, die sowohl in Präfix- wie in Postfix-Notation auf Namen angewendet werden können
< <= == >= > !=	Vergleichsoperatoren (echt kleiner, kleiner gleich, gleich, größer gleich, echt größer, ungleich)
=	Zuweisungsoperator
=@	zusammengesetzte Zuweisungsoperatoren; dabei bedeutet a=@b dasselbe wie a=a@b. Für @ können Sie einen der Operatoren + – * / ^ oder % angeben.

Die logischen Operatoren *&&* und || stehen in *bc nicht* zur Verfügung.

Definition von Funktionen

```
define L (L, ...,L) {
          auto L, ...,L
          S; ...;S
          return (E)
  }
```

Beispiel

```
   define p(x) {
         auto q
         q = p * p
         return (q)
     }
```

Indem die Ergebnisparameter einer Funktion als auto vereinbart werden, wird ihr Gültigkeitsbereich auf diese Funktion beschränkt. Alle Funktionsargumente werden als Werte übergeben.

Funktionen in der mathematischen Bibliothek /usr/lib/lib.b

In */usr/lib/lib.b* sind Definitionen der folgenden mathematischen Funktionen enthalten. Sie stehen Ihnen zur Verfügung, wenn Sie *bc* mit der Option *-l* aufrufen:

```
s(x)    Sinus
c(x)    Cosinus
e(x)    Exponentialfunktion mit Basis e
l(x)    natürlicher Logarithmus
a(x)    Arcustangens
j(n,x)  Bessel-Funktion n-ter Ordnung
```

Die Werte für *x* müssen im absoluten Bogenmaß angegeben werden.

Wenn Sie Schreibrecht für die Datei */usr/lib/lib.b* haben, können Sie die Definitionen weiterer Funktionen hinzufügen oder vorhandene Definitionen abändern oder löschen.

Basis für Ein- und Ausgaben festlegen

Mit *ibase* und *obase* legen Sie fest, in welchem Zahlensystem eingegebene bzw. ausgegebene Werte interpretiert werden. Dabei gelten folgende Regeln:

1. Wenn Sie *ibase* oder *obase* nicht explizit einen Wert zuweisen, werden eingegebene Zahlen im Dezimalsystem interpretiert und Ergebnisse im Dezimalsystem ausgegeben.
2. Wenn Sie mit der Anweisung *ibase* = *n* eine Basis für Eingaben festgelegt haben, müssen Sie die gewünschte Basis für Ausgaben in der Anweisung *obase* = *m* schon in der Eingabe-Basis *n* darstellen.

Beispiel

Basis für Eingaben soll 2, Basis für Ausgaben soll 16 sein:

```
$ bc
ibase=2
obase=10000
10100000/1010
10
```

Nachkommastellen

Jedem Ausdruck *E* in *bc* ist eine Zahl von Nachkommastellen zugeordnet. Diese Zahl können Sie mit der Variablen *scale* ändern und abfragen oder mit der Funktion *scale(E)* abfragen.

Beispiel

Im folgenden Beispiel werden zunächst die Werte zweier Operanden *a* und *b* dividiert, ohne dass die Variable *scale* gesetzt wurde: das Ergebnis hat keine Nachkommastellen. Dann wird *scale* der Wert 8 zugewiesen: das Ergebnis der Division ist nun auf 8 Stellen nach dem Komma genau.
Anschließend wird die Anzahl der Nachkommastellen des Ergebnisses und der Wert von *scale* abgefragt.

```
$ bc
a=15.0
b=7.8
a/b
1
scale=8
```

```
a/b
1.92307692
scale(a/b)
8
scale
8
@@d oder END
$
```

Wenn Sie zwei Ausdrücke durch einen Operator verknüpfen, wird die dem Ergebnis zugeordnete Anzahl von Nachkommastellen durch eine Regel bestimmt, die für den Operator spezifisch ist. im Folgenden werden die Regeln für die *bc*-Operatoren beschrieben. Dabei werden einige Abkürzungen verwendet:

a = erster Operand
b = zweiter Operand
R = Anzahl der Nachkommastellen des Ergebnisses einer Rechenoperation
A = scale(a)
B = scale(b)

-
++
--

Das unäre Minuszeichen sowie die Inkrement- und Dekrement-Operatoren ++ und -- (in Präfix- und Postfix-Notation) ändern die Zahl der Nachkommastellen nicht.

Regel: `scale(E) = scale (-E) = scale(--E) = scale(++E) ...`

Beispiel

a erhält einen Wert mit drei Nachkommastellen. Die Funktionsabfrage *scale(a)* ergibt hier immer die 3, unabhängig davon, ob *a* mit einem der Operatoren -, -- oder ++ versehen wird und unabhängig davon, dass *scale* vorher ein anderer Wert zugewiesen wurde:

```
$ bc
scale=1
a=1.123
scale(a)
3
scale(-a)
-3
scale(a++)
3
scale(--a)
3
@@d oder END
$
```

+
-

Für die binären Operatoren + und - ist *R* gleich der Anzahl der Stellen des Operanden, der die meisten Nachkommastellen hat. Dabei spielt es keine Rolle, ob Sie *scale* zuvor einen anderen Wert zugewiesen haben.

Regel: `R = max(A,B)`

Beispiel

Die Variable *scale* erhält den Wert 1. *a* wird ein Wert mit zwei Nachkommastellen, dem Operanden *b* ein Wert mit drei Nachkommastellen zugewiesen. *b* hat also mehr Stellen als *a*, aber auch mehr als *scale*. Die Funktionsabfrage ergibt für beide Operanden 3, das ist die Anzahl der Nachkommastellen von *b*.

```
$ bc
scale=1
a=0.12
b=0.123
scale(a+b)
3
scale(b-a)
3
@@d oder END
$
```

* Bei einer Multiplikation spielt ein zuvor gesetzter Wert von *scale* eine Rolle: *bc* ermittelt zunächst den höchsten Wert *max* aus den Werten von *scale*, *A* und *B*. Anschließend bildet *bc* die Summe von *A* und *B*. *R* ist dann der kleinere Wert aus dem Vergleich von *max* und dieser Summe.

Regel: `R = min (A+B, max (scale, A, B))`

Beispiel

scale hat den Wert 9, *A* und *B* haben jeweils den Wert 1. Der höchste Wert ist also 9. Die Summe der Nachkommastellen der Operanden ist 2. Die Anzahl Nachkommastellen des Ergebnisses der Multiplikation ist der kleinere Wert aus *max* und der Summe, also 2.

```
$ bc
scale=9
a=0.1
b=0.1
scale(a*b)
2
@@d oder END
$
```

/ Bei einer Division bestimmt ausschließlich der Wert von scale die Genauigkeit des Ergebnisses:

Regel: `R = scale`

Beispiel

Zunächst erhält *scale* den Wert 8. Anschließend werden den Operanden ganze Zahlen zugewiesen. Das Ergebnis ist wieder eine ganze Zahl. Trotzdem ergibt die Funktionsabfrage 8, und das Ergebnis wird auf 8 Stellen hinter dem Komma genau ausgegeben.

```
$ bc
scale=8
a=16
b=4
scale(a/b)
8
a/b
4.00000000
@@d oder END
$
```

^ Beim Potenzieren berechnet sich R wie folgt:

– wenn der ganzzahlige Exponent e >= 0 ist:
bc ermittelt den höchsten Wert *max* aus den Werten von *scale* und A. Anschließend multipliziert es A mit dem Betrag m des Exponenten, vergleicht das Ergebnis mit *max* und nimmt von beiden den kleineren Wert.

Regel: `R = min (A*m, max (scale, A))`

– wenn der ganzzahlige Exponent e < 0 ist:
Die Anzahl Nachkommastellen des Ergebnisses entspricht dem Wert von *scale*.

Regel: `R = scale`

Beispiel 1

scale erhält den Wert 7. *a* hat eine Nachkommastelle, und der Betrag des Exponenten *e* ist 4, also größer als 0. Der höchste Wert von *scale* und *a* ist 7. Das Ergebnis der Multiplikation aus *a* und *m* ist jedoch 4. Die Anzahl der Nachkommastellen des Potenzierens ist also 4:

```
$ bc
scale=7
a=3.1
e=4
scale(a^e)
4
a^e
92.3512
@@d oder END
$
```

Beispiel 2

Wenn Sie für den Exponenten *e* jedoch -4 angeben, ist die Anzahl der Nachkommastellen der Wert von *scale*:

```
$ bc
scale=7
a=3.1
e=-4
scale(a^e)
7
a^e
.0108281
@@d oder END
$
```

=
=@

Bei den Zuweisungsoperatoren entspricht der Wert von *R* demjenigen von *A* nach der Zuweisung. Dabei gilt für einen zusammengesetzten Operator =@ jeweils die gleiche Regel zur Berechnung der Anzahl von Nachkommastellen, wie für den einfachen Operator @.

Regeln: `R = scale(b)` bzw. `R = scale(a@b)`

Beispiel 1

a wird eine Zahl mit einer Nachkommastelle, *b* eine mit zwei Nachkommastellen zugewiesen: *scale(a)* ist 1 und *scale(b)* ist 2. Fragen Sie nach der Zuweisung der Operanden *scale(a)* ab, so gibt *bc* den Wert 2 aus, also *scale(b)*. Wenn Sie anschließend noch *a* abfragen, erhalten Sie den zugewiesenen Wert von *b*:

```
$ bc
a=0.1
b=0.12
scale(a)
1
scale(b)
2
a=b
scale(a)
2
a
.12
@@d oder END
$
```

Beispiel 2

a wird eine Zahl mit zwei und *b* eine Zahl mit drei Nachkommastellen zugewiesen. Beim Funktionsaufruf wird *a* der Wert der Addition der beiden Operatoren zugewiesen. Für *bc* ist die Anzahl Nachkommastellen eines Additionsergebnisses gleich der Anzahl Stellen desjenigen Operanden, der die meisten Stellen hat. *a* hat nach der Zuweisung ebensoviele Stellen, also drei:

```
$ bc
a=0.12
a=0.123
scale(a)
2
scale(b)
3
a=+b
scale(a)
3
a
.243
@@d oder END
$
```

% Ist bei einer %-Berechnung der Wert von scale ungleich 0, so errechnet sich das Ergebnis folgendermaßen:

```
a%b = a – (a / b) * b
```

Dabei wird zunächst die Division mit der Genauigkeit von *scale* berechnet.
Die Genauigkeit der Multiplikation mit *b* beträgt:

```
scale + B
```

Das heißt, dass die Multiplikation exakt ausgeführt wird. Der %-Operator kann hier also als Maß für die Genauigkeit verwendet werden, mit der die Division durchgeführt wird.

R schließlich ist der höchste Wert aus *A* und dem Additionsergebnis von *scale* und *B*.

Regel: `R = max ((scale + B), A)`

Beispiel

scale hat den Wert 4, *a* und *b* haben jeweils eine Nachkommastelle. Das Ergebnis der %-Berechnung hat fünf Nachkommastellen:

```
$ bc
scale=4
a=1.2
b=1.1
scale(a%b)
5
a%b
.00001
@@d oder END
$
```

Datei *lusr/lib/lib.b*
Mathematische Bibliothek

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *bc*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedatein).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Zahlen addieren, subtrahieren, multiplizieren, dividieren. Nicht ganzzahlige Ergebnisse sollen mit zwei Nachkommastellen ausgegeben werden:

```
$ bc
scale=2
3+7
10
8-15
-7
7*6
42
3/5
.60
quit
$
```

Beispiel 2 Definition einer Funktion, die angenähert den Wert der Exponentialfunktion berechnet.

```
scale=20
define e(x){
      auto a, b, c, i, s
      a = 1
      b = 1
      s = 1
      for(i=1;1==1;i++){
              a = a*x
              b = b*i
              c = a/b
              if(c==0) return (s)
              s = s+c
        }
  }
```

Das Abbruchkriterium der for-Schleife ist im Schleifenrumpf enthalten (*if(c==0)*) und wird nicht, wie sonst üblich, durch den zweiten Ausdruck der for-Anweisung gegeben. In einem C-Programm würde man den zweiten Ausdruck der for-Anweisung in einem solchen Fall einfach weglassen. Die for-Anweisung in einem *bc*-Programm muss jedoch immer drei Ausdrücke enthalten. Daher wird hier der Ausdruck *1==1* eingefügt, der immer wahr ist.

Beispiel 3 Ausgeben der angenäherten Werte der Exponentialfunktion für die ersten 10 natürlichen Zahlen.

```
for(i=1;i<=10;i++) e(i)
```

Beispiel 4 Die Quadrate der ersten vier natürlichen Zahlen ausgeben:

```
$ bc
for(i=1;i<5;i++) {i*i}
1
4
9
16
quit
```

Siehe auch *expr, let*

bg Jobs in den Hintergrund schicken (run jobs in the background)

Mit dem in die POSIX-Shell *sh* eingebauten Kommando *bg* können Sie Aufträge in den Hintergrund schicken.

Syntax **bg**[␣job-id...]

job-id
Jeder der angegebenen Aufträge wird in den Hintergrund geschickt. Der Abschnitt „Aufträge“ auf Seite 66 enthält eine Beschreibung des Formats von *job-id*.

job-id nicht angegeben:
Der aktuelle Auftrag wird in den Hintergrund geschickt.

Internationale Umgebung
Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *bg* aus:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Der Job %1 soll in den Hintergrund geschickt werden:

```
$ bg %1
```

Siehe auch *fg, jobs, kill, wait*

bs2cmd BS2000-Kommando ausführen *(BS2000)* (execute BS2000 command)

Das POSIX-Kommando *bs2cmd* führt ein BS2000-Kommando aus.

Syntax **bs2cmd**[␣**-h**]␣cmd

-h Ausgabe der Kommandosyntax mit Erläuterung der Optionen

cmd

BS2000-Kommando:
Angabe des BS2000-Kommandos gemäß der BS2000-Syntax im SDF-Format (Das ISP-Format ist nur noch aus Kompatibilitätsgründen zulässig).

Sonderzeichen im BS2000-Kommando (z.B. $, *) sind durch einen Backslash ('\') zu entwerten oder die gesamte cmd-Zeichenkette ist in Hochkommas einzuschließen.

Der Kommandoname, die Operanden und Operandenwerte werden in Großbuchstaben umgewandelt. Bei Kommandos im ISP-Format wird nur der Kommandoname in Großbuchstaben umgewandelt.

Hinweis Das Kommando *bs2cmd* wird auch bei *rlogin*- und *telnet*-Zugängen in das POSIX unterstützt.

SYSFILE-Umgebung:

BS2000-Kommandos, die bei der Ausführung die SYSFILE-Umgebung benötigen, können derzeit nur in der Basisshell erfolgreich aufgerufen werden, da die SYSFILE-Umgebung in einer Subshell nicht vollständig initialisiert ist.

Entladen der Shell:

Bei Aufruf von *bs2cmd* aus der Basisshell wird das Entladen der Shell durch die Angabe eines entsprechenden BS2000-Kommandos (z.B. START-PROGRAM oder benutzerdefinierte Kommandos, deren CALL-Prozedur das laufende Programm terminiert) verhindert. *bs2cmd* gibt eine Meldung aus (SDP0250) und das BS2000-Kommando wird nicht ausgeführt. Bei Aufruf von *bs2cmd* aus einer Subshell kann derzeit das Entladen der Basisshell nicht verhindert werden.

Beispiel Katalogeinträge der SYSRME-Dateien auf der BS2000-Kennung $QM212 ausgeben:

```
/home/user1> bs2cmd 'show-file-attributes $qm212.sysrme.*'

%      15 :1OSN:$QM212.SYSRME.ARCHIVE.090.D
%      15 :1OSN:$QM212.SYSRME.ARCHIVE.090.E
  .
  .
  .
%      12 :1OSN:$QM212.SYSRME.POSIX-BC.070.D
%      12 :1OSN:$QM212.SYSRME.POSIX-BC.070.E
%      18 :1OSN:$QM212.SYSRME.POSIX-BC.080.D
%      18 :1OSN:$QM212.SYSRME.POSIX-BC.080.E
%:1OSN: PUBLIC:    15 FILES RES=      210  FREE=       18  REL=       0 PAGES
%:1OSN: PUB/S2:     2 FILES RES=      123  FREE=        1  REL=       0 PAGES

/home/user1>
```

Siehe auch *bs2cp*, *bs2file*

bs2cp POSIX-/BS2000-Dateien kopieren *(BS2000)* (copy POSIX/BS2000 files)

bs2cp kopiert Dateien vom POSIX-Dateisystem ins BS2000 und umgekehrt.
Das Kommando hat vier Formate:

- Eine einzelne POSIX-Datei ins BS2000 kopieren und umgekehrt (siehe unten)
- POSIX-Dateien als DVS-Dateien ins BS2000 kopieren (siehe Seite 178)
- POSIX-Dateien in eine BS2000-PLAM-Bibliothek kopieren (siehe Seite 180)
- BS2000-Dateien mit Wildcard-Syntax in ein POSIX-Dateiverzeichnis kopieren (siehe Seite 181)

Das Kommando erstellt physische Kopien. Das heißt, die Dateien sind nach dem Kopiervorgang sowohl im POSIX-Dateisystem als auch im BS2000 physisch vorhanden.

Format 1 Eine einzelne POSIX-Datei ins BS2000 kopieren und umgekehrt

Syntax

bs2cp[␣**-kl**␣**-t**␣tabelle][␣**-f**][␣**-l**][␣**-h**]␣**bs2:**datei␣dateikopie

bs2cp[␣**-kl**␣**-t**␣tabelle][␣**-f**][␣**-l**][␣**-h**]␣datei␣**bs2:**dateikopie

-k Beim Kopieren wird der Inhalt der Datei konvertiert:

- von ASCII nach EBCDIC, falls *datei* eine Datei des POSIX-Dateisystems ist
- von EBCDIC nach ASCII, falls *datei* eine BS2000-Datei ist (**bs2:** angegeben).

Wenn die Originaldatei oder die Dateikopie ein PLAM-Bibliothekselement vom Typ *L* (*LLM*) ist, wird diese Option ignoriert und keine Konvertierung durchgeführt.

Im Gegensatz zur Steuerung mit der Umgebungsvariablen *IO_CONVERSION* wird mit *-k* unabhängig vom Typ des POSIX-Dateisystems konvertiert, d.h. auch dann, wenn das POSIX-Dateisystem ein EBCDIC-Dateisystem ist (also kein ASCII-Dateisystem oder ein mit NFS eingehängtes Dateisystem).

-k ist nur bei Textdateien sinnvoll. Eine inhaltliche Plausibilitätskontrolle, ob die zu konvertierende Eingabedatei sinnvoll zu konvertieren ist, findet durch *bs2cp* nicht statt.

-t tabelle
Beim Kopieren der Datei wird ihr Inhalt konvertiert, wobei die Datei *tabelle* als Konvertierungstabelle verwendet wird.

Die Optionen *-k* und *-t* schließen sich gegenseitig aus.

Analog zur Option *-k* wird diese Option ignoriert und keine Konvertierung durchgeführt, wenn die Originaldatei oder die Dateikopie ein PLAM-Bibliothekselement vom Typ *L* (*LLM*) ist.

Eigenschaften der Datei *tabelle*:

- EBCDIC-Format.
- Es müssen genau 256 Zeichenpaare enthalten sein, die sich aus folgenden Zeichen zusammensetzen: *0* bis *9*, *a* bis *f* und *A* bis *F*.
- Zwischen den Zeichenpaaren dürfen alle EBCDIC-Zeichen von *X'00'* bis *X'40'* stehen. Dies sind z. B. Leerzeichen, Tabulatoren oder Zeilenwechsel.
- Die Datei kann beliebig groß sein. Es werden jedoch maximal die ersten 8172 Zeichen ausgewertet.

Sofern die Datei *tabelle* nicht mit dem absoluten Pfadnamen angegeben wird, sucht sie das Kommando standardmäßig unter *$BS2CPTABS*. Ist diese Variable nicht gesetzt oder leer, so wird die Datei *tabelle* im Verzeichnis */usr/lib/bs2cp* gesucht.

Bei der Konvertierung wird zunächst aus der Datei *tabelle* eine 256 Byte große Umsetzungstafel erstellt, indem jedes der 256 Zeichenpaare zu einem Hexadezimalzeichen verdichtet wird. Danach wird jedes Zeichen aus der zu kopierenden Datei durch ein Zeichen der Umsetzungstafel ersetzt, das über den Binärwert des Eingabezeichens adressiert wird.

So wird z. B. das Eingabezeichen *M* durch das 212. Byte der Umsetzungstafel ersetzt, weil *M* den EBCDIC-Wert *X'D4'* und damit den Dezimalwert *212* hat.

Ein Beispiel für den Aufbau einer Codier-Tabelle finden Sie unter „Beispiel-Tabelle zur Option -t“ auf Seite 183.

-f Beim Kopieren vom POSIX ins BS2000 erfolgt standardmäßig eine Abfrage nach *stderr*, ob bereits existierende BS2000-Dateien/Elemente überschrieben werden sollen oder nicht. Mit *-f* wird diese Dialogabfrage unterdrückt und bereits existierende Dateien werden überschrieben.

Beim Kopieren von BS2000 nach POSIX wird diese Option (sofern angegeben) ignoriert; bereits existierende POSIX-Dateien werden generell überschrieben.

Falls die Umgebungsvariable *OV* existiert, entfällt die Dialogabfrage unabhängig von der Angabe *-f*. Ist *OV* = "*Y*", werden bereits existierende BS2000-Dateien/Elemente überschrieben, andernfalls werden sie nicht überschrieben und es wird stattdessen eine Meldung ausgegeben.

-l Jede erfolgreich kopierte Datei wird wie folgt protokolliert:
`bs2cp: copy from` *pfadname* `to` *pfadname* `done`

-h Ausgabe der Kommandosyntax mit Erläuterung der Optionen.

bs2:

Die bei dieser Option angegebene *datei* oder *dateikopie* ist eine BS2000-Datei.

Handelt es sich dabei um eine DVS-Datei, wird sie über ihren Namen angesprochen. Sonderzeichen im Dateinamen (z. B. $ in der BS2000-User-ID) sind durch einen vorangestellten Backslash zu entwerten oder die Zeichenketten **bs2:***datei* bzw. **bs2:***dateikopie* sind in Hochkommata einzuschließen.

Handelt es sich dabei um ein Bibliothekselement, wird es in der folgenden Form angesprochen (siehe auch Beispiel 3):

'lib(elem[,[type][,vers]])'

Dabei bedeuten

lib Name der PLAM-Bibliothek im BS2000.

elem Name des Elements.

type Typ des Elements. Unterstützt werden die Elementtypen *S*, *M*, *J*, *P*, *D*, *X* und *L*. Standardwert ist *S*.

vers Version des Elements. Standard ist **HIGH*.

Genau eine der Dateien (*datei* oder *dateikopie*) muss eine BS2000-Datei oder ein Bibliothekselement sein. POSIX-Dateiverzeichnisse können nicht ins BS2000 kopiert werden. Wildcard-Syntax und Konstruktionsangaben werden nicht unterstützt.

datei

Name der Datei, die kopiert werden soll.

dateikopie

Name der Kopie. Ist *dateikopie* eine bereits existierende BS2000-Datei oder ein Bibliothekselement, so wird abgefragt, ob überschrieben werden soll. Die Option *-f* und die Umgebungsvariable *OV=Y* schalten diese Abfrage aus.

Ist *dateikopie* eine BS2000-DVS-Datei, so gilt:

– Gibt es bereits eine BS2000-DVS-Datei mit dem Namen *dateikopie*, so verwendet *bs2cp* die Dateiattribute aus dem Dateikatalog.

– Gibt es noch keine BS2000-DVS-Datei mit dem Namen *dateikopie*, so wird sie neu angelegt. Der Dateityp der neuen Datei kann vor Aufruf von *bs2cp* mit dem Kommando *ftyp* festgelegt werden. Wurde kein Dateityp festgelegt, wird eine SAM-Datei mit variabler Satzlänge angelegt.

Ist *dateikopie* ein Element einer BS2000-PLAM-Bibliothek, so gilt:

- Gibt es bereits ein Bibliothekselement mit dem Namen *elem*, so wird es überschrieben, wenn die Option *-f* angegeben ist oder die Umgebungsvariable *OV* den Wert *Y* hat.
- Gibt es noch kein Bibliothekselement mit dem Namen *elem*, so wird es neu angelegt. Die zusätzlichen Einflussmöglichkeiten auf den Dateityp über die Kommandos *ftyp* und *bs2file* gelten dabei auch für Elemente ungleich Typ *L*.
- Existiert keine PLAM-Bibliothek mit dem Namen *lib*, so wird sie neu angelegt.

Format 2 POSIX-Dateien als DVS-Dateien ins BS2000 kopieren

Syntax **bs2cp**␣**-x**[␣**-kl**␣**-t**␣tabelle][␣**-f**][␣**-l**][␣**-h**][␣**-p**␣präfix][␣**-s**␣suffix]␣datei . . . ␣**bs2:**

-x Erweitertes (extended) Format des Kommandos *bs2cp*.

-k -t
siehe Format 1 (Seite 175).

-f Analog zu Format 1 erfolgt beim Kopieren standardmäßig eine Abfrage nach *stderr*, ob bereits existierende BS2000-Dateien überschrieben werden sollen. Die Abfrage enthält jedoch mehrere Antwortmöglichkeiten:
`bs2cp: overwrite A ?` *y* (yes), *n* (no), *a* (all) oder *q* (quit)

Bei Angabe von *q* wird das Kommando *bs2cp* mit einem Exit-Status ungleich Null abgebrochen.

Mit *-f* wird diese Dialogabfrage unterdrückt und bereits existierende Dateien werden generell überschrieben.

Falls die Umgebungsvariable *OV* existiert, entfällt die Dialogabfrage unabhängig von der Angabe *-f*. Ist *OV* = "*Y*", werden bereits existierende BS2000-Dateien/Elemente überschrieben, andernfalls werden sie nicht überschrieben und es wird stattdessen eine Meldung ausgegeben.

-l Jede erfolgreich kopierte Datei wird wie folgt protokolliert:
`bs2cp: copy from` *pfadname* `to` *pfadname* `done`

-h Ausgabe der Kommandosyntax mit Erläuterung der Optionen.

-p präfix
Die Zeichenkette *präfix* wird den Namen der Kopien vorangestellt.

-s suffix
Die Zeichenkette *suffix* wird an die Namen der Kopien angehängt.

datei ...

ist eine Liste aus einer oder mehreren POSIX-Dateien, wobei jeweils Shell-Sonderzeichen für Dateinamen-Erzeugung genutzt werden können. *datei* darf kein Dateiverzeichnis sein.

bs2:

Die Kopien werden im BS2000 als DVS-Dateien abgelegt. Die Kopien erhalten jeweils denselben einfachen Dateinamen wie die Originale in Großbuchstaben, können jedoch über *präfix* und *suffix* erweitert werden. Eventuell vorhandene Unterstriche werden in Dollarzeichen umgewandelt.

Bei bereits bestehenden Dateien wird abgefragt, ob überschrieben werden soll. Zusätzlich kann bestimmt werden, ob alle folgenden Dateien ohne Rückfrage überschrieben werden sollen. Es besteht auch die Möglichkeit zum Abbruch.

Die Dateien werden standardmäßig in die BS2000-Benutzerkennung des POSIX-Benutzers auf dem Home-Pubset kopiert. Das mit *-p* angegebene Präfix kann z.B. dazu genutzt werden, eine andere Cat-ID und/oder User-ID anzugeben.

Siehe Beispiel 4 (Seite 187).

Format 3 POSIX-Dateien in eine BS2000-PLAM-Bibliothek kopieren

Syntax

bs2cp␣**-x**[␣**-kl**␣**-t**␣tabelle][␣**-f**][␣**-l**][␣**-h**][␣**-p**␣präfix][␣**-s**␣suffix]
␣datei . . .␣**'bs2:**lib**(**[,[type][,vers]]**)'**

-x Erweitertes (extended) Format des Kommandos *bs2cp*.

-k -t
siehe Format 1 (Seite 175).

-f Beim Kopieren erfolgt standardmäßig eine Abfrage nach *stderr*, ob bereits existierende BS2000-Elemente überschrieben werden sollen oder nicht. Mit *-f* wird diese Dialogabfrage unterdrückt und bereits existierende Elemente werden überschrieben.

Falls die Umgebungsvariable *OV* existiert, entfällt die Dialogabfrage unabhängig von der Angabe *-f*. Ist *OV* gleich "*Y*", werden bereits existierende BS2000-Dateien/Elemente überschrieben, andernfalls werden sie nicht überschrieben und es wird stattdessen eine Meldung ausgegeben.

-l Jede erfolgreich kopierte Datei wird wie folgt protokolliert:
`bs2cp: copy from` *pfadname* `to` *pfadname* `done`

-h Ausgabe der Kommandosyntax mit Erläuterung der Optionen.

-p präfix
Die Zeichenkette *präfix* wird den Namen der Kopien vorangestellt.

-s suffix
Die Zeichenkette *suffix* wird an die Namen der Kopien angehängt.

datei ...
ist eine Liste aus einer oder mehreren POSIX-Dateien, wobei jeweils Shell-Sonderzeichen für Dateinamen-Erzeugung genutzt werden können. *datei* darf kein Dateiverzeichnis sein.

'bs2:lib**(**[,[type][,vers]]**)'**
Die Kopien werden als Elemente der BS2000-PLAM-Bibliothek *lib* abgelegt. Zu *type* und *vers* siehe Format 1, Seite 177. Die Namen der Elemente werden jeweils aus den einfachen Dateinamen der Originale in Großbuchstaben gebildet, können jedoch über *präfix* und *suffix* erweitert werden.

Siehe Beispiel 5 (Seite 188).

Format 4 BS2000-Dateien mit Wildcard-Syntax in ein POSIX-Dateiverzeichnis kopieren

Syntax

bs2cp ␣ **-x** [␣ **-kl** ␣ **-t** ␣ tabelle] [␣ **-l**] [␣ **-h**] [␣ **-p** ␣ präfix] [␣ **-s** ␣ suffix] ␣ **'bs2:**datei' |
␣ **'bs2:**lib**(**[elem][**,**[type][**,**vers]]**)'** ␣ dateiverzeichnis

-x Erweitertes (extended) Format des Kommandos *bs2cp*.

-k -t
siehe Format 1 (Seite 175).

-l Jede erfolgreich kopierte Datei wird wie folgt protokolliert:
`bs2cp: copy from` *pfadname* `to` *pfadname* `done`

-h Ausgabe der Kommandosyntax mit Erläuterung der Optionen.

-p präfix
Die Zeichenkette *präfix* wird den Namen der Kopien vorangestellt.

-s suffix
Die Zeichenkette *suffix* wird an die Namen der Kopien angehängt.

'bs2:datei'
datei Ist ein vollqualifizierter DVS-Dateiname oder ein teilqualifizierter DVS-Dateiname mit Wildcardsyntax (Sonderzeichen „*“ des BS2000-Kommandos SHOW-FILE-ATT bzw. FS). Es ist nur ein Operand **bs2:***datei* zulässig.

Achtung: Wenn der Name nur aus Leerzeichen besteht oder ganz weggelasssen wird, so entspricht dies der Wildcard-Angabe „*“

'bs2:lib**(**[elem][**,**[type][**,**vers]]**)'**
elem ist der vollqualifizierte Name eines PLAM-Elements oder der teilqualifizierte Name mehrerer PLAM-Elemente mit LMS-Wildcardsyntax. Für Wildcard werden folgende Sonderzeichen unterstützt: * < : >. Andere Sonderzeichen für LMS-Wildcards werden nicht garantiert.

Sonderfall: Wird für *elem* nichts angegeben, so werden alle Elemente von entsprechendem Typ und Version genommen (würde der Wildcard-Angabe „*“ entsprechen).

Minimalangabe: **bs2:***lib***()** alle Elemente vom Typ *S* und jeweils höchster Version werden kopiert.

dateiverzeichnis
Die BS2000-Dateien/Elemente werden in das angegebene POSIX-Dateiverzeichnis kopiert. Die in der Shell üblichen Notationen für Dateiverzeichnisse sind zulässig, z.B:

. (Punkt)	aktuelles Verzeichnis
~ (Tilde)	Home-Verzeichnis
/dvz1/dvz2	absoluter Pfad
dvz1/dvz2	relativer Pfad ab aktuellem Verzeichnis

Siehe Beispiele 6 und 7 (Seite 188).

Hinweise

EXIT-Status bei den erweiterten Formaten 2 bis 4

Wenn pro *bs2cp*-Aufruf mehrere Dateien kopiert werden, erhält man den Exit-Status *0* nur dann, wenn alle Kopieraktionen erfolgreich waren. Beim ersten auftretenden Fehler wird Exit ungleich *0* geliefert und die restlichen Dateien werden nicht mehr kopiert.

Kopieren von Bibliothekselementen

Beim Kopieren von Bibliothekselementen in POSIX-Dateien oder umgekehrt ist Voraussetzung, dass das Software-Produkt LMS installiert ist.

Unterstützte Attribute von DVS-Dateien

bs2cp unterstützt nur Dateiattribute, die auch vom C-Laufzeitsystem bei STREAM I/O unterstützt werden, d.h. SAM-Dateien mit fester Satzlänge können nur binär (*ftyp binary*) geöffnet werden.

Folgende Dateiattribute werden mit *ftyp binary* unterstützt:

FCB-TYPE	REC-FORM	BLKCTRL	BLKSIZE (STD,n)	RECSIZE (r Byte)	Max. Anz. Datenbyte
SAM	F	PAMKEY	1<=n<=16	1<=r<=n*2048	RECSIZE
SAM	F	DATA(2K)	1<=n<=16	1<=r<=n*2048-16	RECSIZE
SAM	F	DATA(4K)	2<=n<=16	1<=r<=n*2048-16	RECSIZE
SAM	V	PAMKEY	1<=n<=16	4<=r<=n*2048-4	RECSIZE - 4
SAM	V	DATA(2K)	1<=n<=16	4<=r<=n*2048-16	RECSIZE - 4
SAM	V	DATA(4K)	2<=n<=16	4<=r<=n*2048-16	RECSIZE - 4
SAM	U	PAMKEY	1<=n<=16		BLKSIZE
SAM	U	DATA(2K)	1<=n<=16		BLKSIZE - 16
SAM	U	DATA(4K)	2<=n<=16		BLKSIZE - 16
PAM		PAMKEY	1<=n<=16		BLKSIZE
PAM		DATA(2K)	1<=n<=16		BLKSIZE - 12
PAM		DATA(4K)	2<=n<=16		BLKSIZE - 12
PAM		NO(2K)	1<=n<=16		BLKSIZE
PAM		NO(4K)	2<=n<=16		BLKSIZE

Folgende Dateiattribute werden mit *ftyp text* unterstützt:

FCB-TYPE	REC-FORM	BLKCTRL	BLKSIZE (STD,n)	RECSIZE (r Byte)	Max. Anz. Datenbyte
SAM	V	PAMKEY	1<=n<=16	4<=r<=n*2048-4	RECSIZE - 4
SAM	V	DATA(2K)	1<=n<=16	4<=r<=n*2048-16	RECSIZE - 4
SAM	V	DATA(4K)	2<=n<=16	4<=r<=n*2048-16	RECSIZE - 4
SAM	U	PAMKEY	1<=n<=16		BLKSIZE
SAM	U	DATA(2K)	1<=n<=16		BLKSIZE - 16
SAM	U	DATA(4K)	2<=n<=16		BLKSIZE - 16
ISAM		PAMKEY	1<=n<=16	12<=r<=n*2048	BLKSIZE
ISAM		DATA(2K)	1<=n<=16	12<=r<=n*2048	BLKSIZE - 12
ISAM		DATA(4K)	2<=n<=16	12<=r<=n*2048	BLKSIZE - 12

Bei ISAM-Dateien werden die Satzschlüssel nicht übertragen. Ein Kopieren von temporären Dateien aus dem BS2000 sowie über Remote File Access ist nicht möglich.

Beispiel-Tabelle zur Option -t

Aufbau einer EBCDIC-Standardtabelle, bei deren Zuweisung als Zieldatei eine 1:1 Kopie der Quelle entsteht:

```
00  01  02  03  04  05  06  07  08  09  0A  0B  0C  0D  0E  0F
10  11  12  13  14  15  16  17  18  19  1A  1B  1C  1D  1E  1F
20  21  22  23  24  25  26  27  28  29  2A  2B  2C  2D  2E  2F
30  31  32  33  34  35  36  37  38  39  3A  3B  3C  3D  3E  3F
40  41  42  43  44  45  46  47  48  49  4A  4B  4C  4D  4E  4F
50  51  52  53  54  55  56  57  58  59  5A  5B  5C  5D  5E  5F
60  61  62  63  64  65  66  67  68  69  6A  6B  6C  6D  6E  6F
70  71  72  73  74  75  76  77  78  79  7A  7B  7C  7D  7E  7F
80  81  82  83  84  85  86  87  88  89  8A  8B  8C  8D  8E  8F
90  91  92  93  94  95  96  97  98  99  9A  9B  9C  9D  9E  9F
A0  A1  A2  A3  A4  A5  A6  A7  A8  A9  AA  AB  AC  AD  AE  AF
B0  B1  B2  B3  B4  B5  B6  B7  B8  B9  BA  BB  BC  BD  BE  BF
C0  C1  C2  C3  C4  C5  C6  C7  C8  C9  CA  CB  CC  CD  CE  CF
D0  D1  D2  D3  D4  D5  D6  D7  D8  D9  DA  DB  DC  DD  DE  DF
E0  E1  E2  E3  E4  E5  E6  E7  E8  E9  EA  EB  EC  ED  EE  EF
F0  F1  F2  F3  F4  F5  F6  F7  F8  F9  FA  FB  FC  FD  FE  FF
```

Fehler

pfadname: Permission denied
Sie haben kein Leserecht für die Datei bzw. kein Schreibrecht für das Dateiverzeichnis.

pfadname: no such file or directory
Die angegebene Datei bzw. das Dateiverzeichnis existieren nicht.

pfadname: is a directory
Der angegebene Name bezeichnet keine Datei sondern ein Dateiverzeichnis.

pfadname: DMS FSTAT error *xxx*
Zu viele Dateien angegeben.

invalid character pair: *xx* (valid only 0, ..,9,A,..,F)
Bei der Option *-t* wurde eine fehlerhafte Konvertierungstabelle angegeben.

incorrect number of character pairs: *xxx* (must be 256)
Bei der Option *-t* wurde eine fehlerhafte Konvertierungstabelle angegeben.

more than 256 character pairs given
Bei der Option *-t* wurde eine fehlerhafte Konvertierungstabelle angegeben.

do not use options -k and -t at the same time
Die Optionen *-k* und *-t* können nicht gleichzeitig benutzt werden.

file name *name* invalid in this case
Beim Format 2 wurde hinter **bs2:** ein Name angegeben.

element name *name* invalid in this case
Beim Format 3 wurde ein Elementname angegeben.

invalid target name *pfadname*
Im Namen der Zieldatei ist ein * enthalten

lib(*elem*,*type*,*vers*) already exists
Das Ausgabe-Bibliothekselement existiert bereits. (Option *-f* wurde nicht angegeben, die Umgebungsvariable *OV* existiert nicht, und die Eingabe erfolgt von *stdin* (-).)

lib(*elem*,*type*,*vers*) not accessible
Auf das Ausgabe-Bibliothekselement kann nicht zugegriffen werden. Entweder haben Sie kein Schreibrecht auf die Bibliothek oder auf das Element, oder *lib* ist keine PLAM-Bibliothek.

datei already exists
Die BS2000-Ausgabedatei existiert bereits. (Option *-f* wurde nicht angegeben, die Umgebungsvariable *OV* existiert nicht, und die Eingabe erfolgt von *stdin* (-).)

pfadname: is a directory
Der angebene Pfadname bezeichnet ein Dateiverzeichnis.

pfadname is not an executable file
Es wurde versucht, eine POSIX-Datei, die nicht das Format (UFS-LLM) einer ausführbaren Datei hat, als L-Element in eine Bibliothek zu kopieren.

closing bracket missing
In der Elementangabe fehlt die schließende Klammer. (Eine öffnende Klammer wurde gefunden.)

Copy from DMS file to DMS file not supported - use command /COPY-FILE *f1*,*f2*
Das Kopieren einer BS2000-Datei in eine BS2000-Datei wird nicht unterstützt; hierzu ist das BS2000-Kommando /COPY-FILE zu verwenden.

`Copy from UFS file to UFS file not supported - use command cp` *f1 f2*
Das Kopieren einer POSIX-Datei in eine POSIX-Datei wird nicht unterstützt; hierzu ist das POSIX-Kommando *cp* zu verwenden.

`Copy of UFS directories is not supported`
Als *datei* wurde `.` oder `..` angegeben.

`DMS FSTAT error D`*xxx*
Die Systemfunktion FSTAT für eine BS2000-Datei liefert den Fehlercode `D`*xxx*.

`element name` *elem* `too long`
Der als Quelle angegebene Elementname (Format 2, 3, oder 4) ist länger als 64 Zeichen.

`FILE command for "`*dateiname*`" returned error`
Das FILE-Kommando für die BS2000-Ausgabedatei lieferte einen Fehler. Unmittelbar vorher wurde eine Meldung des FILE-Kommandos ausgegeben. (Die Meldung steht im Zusammenhang mit einem vorher vom Anwender abgesetzten Kommando *bs2file* oder *ftyp*.)

`file` *dateiname* `not found`
Die BS2000-Eingabedatei wurde nicht gefunden.

`Invalid BS2000 filename:` *datei*
Der angegebene Dateiname endet mit einem `.` (Punkt).

`invalid character in` *elem*
Der angegebene Elementname enthält unzulässige Zeichen.

`invalid element name` *elem*
Der Elementname *elem* beginnt oder endet mit `.` oder `-`.

`LMS error - LMS`*nnnn*`[, PLA`*mmmm*`][, DMS or macro RC` *xxxx*`]`
`(fct=`*fct*`, lib=`*lib*`, elem=`*elem*`, type=`*type*`, ver=`*vers*`, file=`*filename*`)`
Beim Zugriff auf ein Bibliothekselement meldet LMSUP einen Fehler mit dem Fehlercode LMS*nnnn* zurück, ggf. ergänzt um den PLAM-Fehlercode und/oder den DMS- oder Makro-Fehlercode (Bedeutung der Fehlercodes kann mit `bs2cmd help LMS`*nnnn* etc. abgefragt werden).
Die zweite Zeile wird nur ausgegeben, wenn die Option `-l` oder `-x` angegeben wurde. *fct* ist `ADD`, `SEL` oder `TOC`. Bei *fct*=`TOC` bleibt die Angabe *filename* leer.

`LMS end errorcode` *xx*
Das zum Kopieren von Bibliothekselementen benötigte LMSUP konnte nicht ordnungsgemäß beendet werden.

`LMS init errorcode` *xx*
Das zum Kopieren von Bibliothekselementen benötigte LMSUP konnte nicht initialisiert werden.

`LMS toc errorcode` *xx*
Das Inhaltsverzeichnis der Bibliothek konnte nicht weiter gelesen werden.

`LMS tocprim errorcode` *xx*
Das Inhaltsverzeichnis der Bibliothek konnte nicht gelesen werden.

`No POSIX file involved - use utility LMS`
Als Quelle und Ziel wurden ein Bibliothekselement und eine BS2000-Datei oder umgekehrt angegeben.

`No write of` *file*`. OV is set to` *value*
Die angegebene BS2000-Datei wurde nicht überschrieben, weil die Option `-f` nicht angegeben und die Umgebungsvariable *OV* gesetzt und nicht gleich *"Y"* ist.

`No write of` *lib*`(`*elem*`,`*type*`,`*vers*`). OV is set to` *value*
Das angegebene Element wurde nicht überschrieben, weil die Option `-f` nicht angegeben und die Umgebungsvariable *OV* gesetzt und nicht gleich *"Y"* ist.

`opening bracket missing`
In der Elementangabe fehlt die öffnende Klammer. (Eine schließende Klammer wurde gefunden.)

`PLAM element` *elem*`, type` *type*`, version` *vers* `not accessible`
Auf das als Quelle angegebene Eingabe-Element kann nicht zugegriffen werden. Entweder haben Sie kein Leserecht auf die Bibliothek oder auf das Element, oder *lib* ist keine PLAM-Bibliothek.

`PLAM element` *elem*`, type` *type*`, version` *vers* `not found`
Das als Quelle angegebene Eingabe-Element wurde nicht gefunden.

`PLAM element name longer than 64`
Der angegebene Elementname ist länger als 64 Zeichen.

`PLAM element name missing`
In der Elementangabe für Format 1 fehlt der Elementname.

`PLAM element version longer than 24`
Die angegebene Elementversion ist länger als 24 Zeichen.

`PLAM element version` *vers* `invalid`
Die Versionsangabe *vers* enthält ungültige oder Wildcard-Zeichen.

`PLAM element type not supported`
Der angegebene Typ ist nicht `S`, `M`, `J`, `P`, `D`, `X` oder `L`.

`PLAM error - PLA`*mmmm*`[, DMS or macro RC` *xxxx*`]`
`(fct=`*fct*`, lib=`*lib*`, elem=`*elem*`, type=`*type*`, ver=`*vers*`, file=`*filename*`)`
Beim Zugriff auf ein Bibliothekselement meldet PLAM einen Fehler mit dem Fehlercode `PLA`*mmmm* zurück, ggf. ergänzt um den DMS- oder Macro-Fehlercode (Bedeutung des Fehlercodes kann mit `bs2cmd help PLA`*mmmm* etc. abgefragt werden).
Die zweite Zeile wird nur ausgegeben, wenn die Option `-l` oder `-x` angegeben wurde. *fct* ist `ADD` oder `SEL`.

`PLAM library name missing`
In der Elementangabe fehlt der Bibliotheksname.

`PLAM library name longer than 54`
Der Bibliotheksname ist länger als 54 Zeichen.

`PLAM library` *lib* `not found`
Die angegebene PLAM-Bibliothek existiert nicht.

`PLAM names must end with single closing bracket`
Die Elementangabe endet mit mehr als einer schließenden Klammer.

`too many opening brackets`
Die Elementangabe enthält mehr als eine öffnende Klammer.

`too many tokens within brackets`
Die Elementangabe enthält mehr als die drei durch Kommata getrennten Angaben *elem*, *type*, *vers*.

Beispiel 1 Die BS2000-Datei *fachliteratur* soll mit gleichem Namen in das Dateiverzeichnis */usr/fl* kopiert werden. Der Zeichensatz soll von EBCDIC nach ASCII umcodiert werden (Option *-k*).

```
$ bs2cp -k bs2:fachliteratur /usr/fl/fachliteratur
```

Beispiel 2 Die Datei *fachliteratur* im POSIX-Dateisystem soll mit Namen *flcopy* ins BS2000 kopiert werden. Der vorhandene Zeichensatz soll unverändert beibehalten werden (keine Option *-k*).

```
$ bs2cp fachliteratur bs2:flcopy
```

Beispiel 3 Die Datei *dokumentation* ist ein Element vom Typ *D* in der BS2000-PLAM-Bibliothek *produkt*. Das Element soll in das Dateiverzeichnis */usr/produkt* kopiert werden.

```
$ bs2cp 'bs2:produkt(dokumentation,D)' /usr/produkt/dokumentation
```

Beispiel 4

```
$ bs2cp -x -p posix. -s .sich /home/do/sich/dat* bs2:
bs2cp: overwrite DATEI1 ? [y=yes/n=no/a=all/q=quit] a

$ ls /home/do/sich
datei1 datei2 datei3

$ bs2cmd fstat posix.
       12  :ABCD:$USER.POSIX.DATEI1.SICH
        9  :ABCD:$USER.POSIX.DATEI2.SICH
       18  :ABCD:$USER.POSIX.DATEI3.SICH
```

Beispiel 5

```
$ ls
genpos.c   hrcv.c      hrcv.s     hrmgt.c   hrupos.c   hsdax.c

$ bs2cp -x -l -p p hr*.c 'bs2:clib(,s,300)'
bs2cp:     copy from hrcv.c to CLIB(PHRCV.C,S,300) done
bs2cp:     copy from hrmgt.c to CLIB(PHRMGT.C,S,300) done
bs2cp:     copy from hrupos.c to CLIB(PHRUPOS.C,S,300) done
```

Erzeugt wird die PLAM-Bibliothek CLIB mit folgendem Inhalt:

```
   TYP NAME      VER (VAR#) DATE
   (S) PHRCV.C   300 (0001) 2008-05-27
   (S) PHRMGT.C  300 (0001) 2008-05-27
   (S) PHRUPOS.C 300 (0001) 2008-05-27
```

Beispiel 6

```
$ bs2cmd fstat '$user2.*kvh*'
        9  :ABCD:$USER2.AKVH7
       12  :ABCD:$USER2.KVHMEM
       12  :ABCD:$USER2.ZKVH

$ bs2cp -x -s .c 'bs2:$user2.*kvh*' /home/tag/kvh

$ ls -l /home/tag/kvh
total 12
-rw-r--r--  1 kvh       prod5       2321 May 27 13:56 akvh7.c
-rw-r--r--  1 kvh       prod5      18549 May 27 13:56 kvhmem.c
-rw-r--r--  1 kvh       prod5        971 May 27 13:56 zkvh.c
```

Beispiel 7 Aus der BS2000-PLAM-Bibliothek *$USER2.DOCLIB* sollen alle Elemente vom Typ *D*, deren Namen mit *KVH* beginnen und die die Version *300* haben, in das Dateiverzeichnis */home/usr/doku* kopiert werden, wobei den Namen der Kopien das Präfix *doc.* vorangestellt werden soll.

```
$ bs2cp -xl -p doc. 'bs2:$user2.doclib(kvh*,d,300)' /home/usr/doku
bs2cp: copy from $USER2.DOCLIB(KVHGEN,D,300) to /home/usr/doku/doc.kvhgen
done
bs2cp: copy from $USER2.DOCLIB(KVHPROD,D,300) to /home/usr/doku/doc.kvhprod
done
bs2cp: copy from $USER2.DOCLIB(KVHZ,D,300) to /home/usr/doku/doc.kvhz done

$ ls -l /home/usr/doku
total  12
-rw-r--r--   1 kvh      prod5      21738 May 31 06:21 doc.kvhgen
-rw-r--r--   1 kvh      prod5       7461 May 31 06:21 doc.kvhprod
-rw-r--r--   1 kvh      prod5      11729 May 31 06:21 doc.kvhz
```

Siehe auch *bs2file*, *ftyp*

bs2do Aufruf von BS2000-Prozeduren aus der POSIX-Shell *(BS2000)* (call BS2000 procedures from the POSIX shell)

Die auszuführende BS2000-Prozedur wird in einem ENTER-JOB mit dem Kommando CALL-PROCEDURE gestartet. *bs2do* wartet synchron auf die Beendigung des ENTER-JOB.

Syntax

bs2do[␣**-DV**][␣**-o**␣outfile]␣procedure␣[[**(**]parameter[**)**]]

-D Debugmode - temporäre Dateien werden nicht gelöscht und stehen für Diagnose zur Verfügung.

-V Verbose - Einzelschritte werden auf *stdout* protokolliert.

-o die beim Start des ENTER-JOB geöffnete BS2000-Systemdatei SYSOUT wird in die POSIX-Datei *outfile* kopiert.

outfile

Der Name der POSIX-Datei, die eine Kopie der beim Start des ENTER-JOB geöffneten BS2000-Systemdatei SYSOUT aufnimmt. *outfile* ist ein beliebiger POSIX-Pfadname.

procedure

Der Name der Datei, die die auszuführende BS2000-Prozedur enthält.
procedure ist ein beliebiger POSIX-Pfadname oder ein String **bs2:***BS2Name*.
BS2Name ist ein BS2000-Prozedurname entsprechend der SDF-Syntax des Operanden FROM-FILE des BS2000-Kommandos /CALL-PROC.

parameter

Die Parameter der auszuführenden BS2000-Prozedur. Die Angabe erfolgt entsprechend der SDF-Syntax des Operanden PROCEDUR-PARAMETERS des BS2000-Kommandos CALL-PROC, wobei die einschließenden runden Klammern optional sind.

Für *procedure* und *parameter* gilt: Sonderzeichen (z. B. (,), $) sind für die Shell zu entwerten. *bs2do* interpretiert den Operanden nicht, u. a. werden z. B. Schlüsselwörter in Schlüsselwortparametern nicht in Großbuchstaben umgesetzt.

Datei

Eingabe-Dateien:
- *stdin*: nicht benutzt
- Prozedurdatei *procedure*
- Optionsdatei *optionfile,* wobei *optionfile* ein beliebiger POSIX-Pfadname sein kann. Standardmäßig ist */opt/BS2DO/options/bs2doopt.std* installiert.

Ausgabe-Dateien:
- *stdout*: Ausgabe von Meldungen bei Angabe von Option *-V* (verbose).
- *stderr*: Fehlermeldungen von *bs2do*, Inhalt der in der Optionsdatei *optionfile* vereinbarten Jobvariablen.
- BS2000-Systemdatei SYSOUT nach POSIX-Datei *outfile* bei Angabe von Option *-o*.

Variable

BS2DOOPT: POSIX-Pfadname einer Optionsdatei. Ist die Umgebungsvariable *BS2DOOPT* nicht definiert, so wird die standardmäßig installierte Optionsdatei *optionfile* verwendet.

Hinweis

Ausführliche Beschreibung

bs2do baut im BS2000 den Stapelauftrag #T.*pid*.ENTER auf, wobei *pid* die POSIX-Prozessnummer des Prozesses ist, der *bs2do* ausführt). Eingeleitet wird der Stapelauftrag durch

```
/.DOpid LOGON
```

Liegt die Prozedur *procedure* im POSIX-Dateisystem, wird diese in die BS2000-Datei T.*pid*.PRO kopiert. Dann wird der Stapelauftrag gestartet:

```
/ENTER-JOB   FROM-FILE=#T.pid.ENTER,-
PROCESSING-ADMISSION=SAME,-
HOST=*STD,-
JOB-CLASS=*STD,-
JOB-PRIORITY=*STD
```

Die Operanden PROCESSING-ADMISSION, HOST, JOB-CLASS und JOB-PRIORITY sind über die Optionsdatei *optionfile* einstellbar.

In dem Stapelauftrag wird die Systemdatei SYSOUT der Datei #T.<pid>.OUT zugeordnet, die Jobvariable *#BS2DOJV* eingerichtet und anschließend die Prozedur *procedure* aufgerufen, der Name der Jobvariablen *#BS2DOJV* ist über die Optionsdatei *optionfile* einstellbar:

```
/CALL-PROC  FROM-FILE=T.pid.PRO,-
       PROC-PAR=parameter,-
       LOGGING=NO
```

Der Operand LOGGING ist über die Optionsdatei *optionfile* einstellbar. Der Operand FROM-FILE wird nur angegeben, wenn *procedure* im POSIX-Dateisystem liegt, ansonsten wird der String *procedure* unverändert übernommen. Sonderzeichen (z.B. (,),$) sind für die Shell zu entwerten.

Nach Abschluss der Prozedur *procedure* wird die Systemdatei *SYSOUT* geschlossen und die POSIX-Shell gestartet.

```
/START-PROG SHELL
```

Der Operand SHELL ist über die Optionsdatei *optionfile* einstellbar.

Wurde in der Prozedur *procedure* keine Fehleranzeige gesetzt (/EXIT-PROCEDURE ERROR=NO), so wird an den Vaterprozess des Stapelauftrages das Signal SIGUSR1 als Endenachricht gesendet.

Wurde in der Prozedur *procedure* die Fehleranzeige gesetzt (/EXIT-PROCEDURE ERROR=YES), wird der Inhalt der Jobvariablen *#BS2DOJV* und anschließend das Signal SIGUSR2 als Endenachricht an den Vaterprozess des Stapelauftrages gesendet.

In beiden Fällen wird vorher bei angegebener Option *-o outfile* die Datei #T.*pid*.OUT in die POSIX-Datei *outfile* kopiert. Nach Beendigung der Shell werden alle Dateien T.*pid* gelöscht und der Stapelauftrag beendet.

Alle temporären Dateien #T.*pid* werden ebenfalls gelöscht. Im Debugmode (Option *-D*) werden keine temporären Dateien, sondern ausschließlich Dateien T.*pid* angelegt, die bei Ende des Stapelauftrages nicht gelöscht werden.

Die Optionsdatei

Die Optionsdatei ist aufgebaut nach dem Schema

```
Schlüsselwort keyword = value
```

Zeilen, die nicht mit einem gültigen Schlüsselwort *keyword* beginnen, werden ignoriert. Ausgenommen sind Fortsetzungszeilen. *value* kann über Zeilengrenzen mit dem Zeichen - fortgesetzt werden. Die folgende Liste zeigt die unterstützten Schlüsselwörter und die zugehörigen *value*-Angaben:

BS2DOJV
: Jobvariablenname (optional)
Wenn das Produkt Jobvariablen nicht zur Verfügung stellt, darf für dieses *keyword* kein *value* angegeben werden.

SHELL
: Operanden von START-PROG zum Laden der POSIX-Shell

PROCESSING-ADMISSION
HOST
JOB-CLASS
JOB-PRIORITY
: entsprechen den gleichnamigen Operanden von ENTER-JOB

LOGGING
entspricht dem gleichnamigen Operanden von CALL-PROC

Die Standard-Optionsdatei */opt/BS2DO/options/bs2doopt.std*:

```
#
#  bs2do controlling job variable
#  (temporarily in order to allow parallel calls within one userid)
#
BS2DOJV = #BS2DOJV
#
#  BS2000 enter-job operands
#
PROCESSING-ADMISSION = SAME
HOST = *STD
JOB-CLASS = *STD
JOB-PRIORITY = *STD
#
#  BS2000 call-proc operands
#
LOGGING = NO
#
#  BS2000 shell configuration  (POSIX-BC shell with
#                               correction level >= A12
#                               POSIX-SH shell with correction level
#                               level >= A52)
#
SHELL = *M($TSOS.SINLIB.POSIX-BC.030.SHELL, SH, -
RUN-MODE=ADVANCED, PROG-MODE=ANY)
#
```

Einschränkungen

Da *bs2do* nicht die BS2000-Sicherheitsmechanismen umgehen kann, sind Einschränkungen zu beachten, wenn PROCESSING-ADMISSION <> SAME in der Optionsdatei eingestellt ist:

- Die Benutzerkennung, in welcher der ENTER-JOB ausgeführt werden soll, muss POSIX-Benutzer-Rechte haben.
- Bei Angabe der Option *-o* muss die Benutzerkennung, in der der ENTER-JOB ausgeführt werden soll, in dem aktuellen Verzeichnis, in dem *bs2do* aufgerufen wurde, such- und schreibberechtigt sein. Ansonsten wird trotz der Option *-o* die SYSOUT-Datei des ENTER-JOBS nicht ins UFS kopiert.
- Die auszuführende Prozedur wird nicht aus dem UFS in eine andere Benutzerkennung kopiert, d.h. das Ausführen einer Prozedur in einer anderen Benutzerkennung wird nur unterstützt, wenn *procedure* gleich **bs2:***BS2Name* ist.

Beispiel In den folgenden Beispielen repräsentiert ein vorangestelltes DO: das Shellprompt, hinter dem Eingaben an die Shell folgen. Zeilen zwischen den Shellprompts sind Ausgaben des jeweiligen Kommandos. Beispielprozedur ist die Prozedur *proc*:

```
/BEGIN-PROCEDURE PAR=YES(PROC-PAR=(&CMD='STA',&CMDPAR='L'))
/
/      WRITE-TEXT T='-----> ''&CMD &CMDPAR'''
/      &CMD &CMDPAR
/      SKIP-COMMAND TO-LABEL=OK
/
/      SET-JOB-STEP
/      MOD-JV JV-CONTENTS=#BS2DOJV,-
/             SET-VALUE=C'Command ''&CMD &CMDPAR'' failed'
/      SKIP-COMMAND TO-LABEL=ERR
/
/.ERR  REMARK
/      WRITE-TEXT T='-----> DOING EXIT-PROCEDURE ERROR=YES'
/      EXIT-PROCEDURE ERROR=YES
/
/.OK   REMARK
/      WRITE-TEXT T='-----> DOING EXIT-PROCEDURE ERROR=NO'
/      EXIT-PROCEDURE ERROR=NO
/
END-PROCEDURE
```

Beispiel 1 Prozedur *proc* in POSIX, einfache Auswertung des Endestatus.

```
DO: bs2do proc
%   JMS0066 JOB OD0492CACCEPTED ON 98-04-23 AT 11:28, TSN=57EE
DO: echo $?
0
```

Beispiel 2 Prozedur *proc* in BS2000-Datei, Ausgabe von SYSOUT in POSIX

```
DO: bs2do -o beispiel2.out bs2:proc
%   JMS0066 JOB OD0494CACCEPTED ON 98-04-23 AT 11:29, TSN=57EG
DO: cat beispiel2.out
/CREATE-JV JV=#BS2DOJV
/SET-JOB-STEP
/MODIFY-JV JV=#BS2DOJV,SET-VALUE=C' '
/SET-JOB-STEP
/CALL-PROC FROM-FILE=proc,LOGGING=NO
-----> 'STA L'
NAME        TSN TYPE       PRI     CPU-USED CPU-MAX ACCOUNT#
DO494      57EG 2 BATCH    9 220     0.6044   32000 1
RLOGIN     57C7 2 DIALOG   0 210    14.3916   9999  1
%  SPS0171 NO LOCAL SPOOLOUT JOB PRESENT
%  SPS0420 RSO WARNING : SOME RSO PRINT-JOBS CANNOT BE DISPLAYED
```

```
%  SCP1095 DPRINTSV WARNING : SOME DPRINT PRINT-JOBS CANNOT BE ..
-----> DOING EXIT-PROCEDURE ERROR=NO
/SKIP-COMMANDS TO-LABEL=NOERR
/.NOERR  REMARK
/SYSFILE SYSOUT=*DUMMY
```

Beispiel 3 Prozedur *proc* in BS2000-PLAM-Bibliothek, fehlerhafter Prozedurparameter

```
DO: bs2cp proc 'bs2:BS2LIB(PROC,J)'  # proc in PLAM Bibliothek
# kopieren
DO: bs2do -o beispiel3.out bs2:\(BS2LIB,PROC\) "CMDPAR='x x'"
%  JMS0066 JOB 'DO501' ACCEPTED ON 98-04-23 AT 12:51, TSN = 57EN
BS2DOJV: Command 'STA x x' failed
DO:echo $?
1
DO: cat beispiel3.out
/CREATE-JV JV=#BS2DOJV
/SET-JOB-STEP
/MODIFY-JV JV=#BS2DOJV,SET-VALUE=C'  '
/SET-JOB-STEP
/CALL-PROC FROM-FILE=(BS2LIB,PROC),PROC-PAR=(CMDPAR='x x'),LOGGING= ..
-----> 'STA x x'
%  EXC0898 INVALID OPERAND IN COMMAND. COMMAND REJECTED
%  CMD0205 ERROR IN PRECEDING COMMAND OR PROGRAM AND PROCEDURE ..
-----> DOING EXIT-PROCEDURE ERROR=YES
%  CMD0205 ERROR IN PRECEDING COMMAND OR PROGRAM AND PROCEDURE ..
/SET-JOB-STEP
/SKIP-COMMANDS TO-LABEL=ERROR
/.ERROR  REMARK
/SYSFILE SYSOUT=*DUMMY
```

Beispiel 4 Prozedur *proc* in POSIX, fehlerhafte Prozedurparameter, Umlenkung von *stderr*

```
DO: bs2do proc "CMD='STA P',CMDPAR=',TYPE=x " 2>beispiel4.err
DO: echo $?
1
DO: cat beispiel4.err
%  JMS0066 JOB 'DO620' ACCEPTED ON 98-04-23 AT 15:43, TSN = 57H1
BS2DOJV: Command 'STA P ,TYPE=x' failed
```

bs2file Dateiattribute für BS2000-Dateien festlegen *(BS2000)* (set BS2000 file attributes)

bs2file legt die BS2000-Dateiattribute einer zu übertragenden Datei fest.
bs2file wird auf das BS2000-FILE-Kommando abgebildet.

Syntax **bs2file**[␣**-h**]␣datei**,**operandenliste

-h Ausgabe der Kommandosyntax mit Erläuterung der Optionen.

datei
Vollqualifizierter Dateiname im BS2000. Für diese Datei werden die Dateiattribute festgelegt und im zugehörigen *bs2cp*-Kommando verwendet.

Die Angabe „*“ („\“ dient zur Entwertung) anstatt eines Dateinamens legt fest, dass die Einstellungen des *bs2file*-Kommandos für alle Dateien im nächsten *bs2cp*-Kommando gelten. Die Dateiattribute gelten dann für die im *bs2cp*-Kommando angegebene(n) Datei(en).

Wird ein vollqualifizierter Dateinamen angegeben, so verliert der *bs2file*-Eintrag erst dann seine Gültigkeit, wenn in einem *bs2cp*-Aufruf bei einem übereinstimmenden Dateinamen die Dateiattribute erfolgreich ausgewertet wurden. Bei Angabe eines Sterns (*) verliert der *bs2file*-Eintrag seine Gültigkeit bei Beendigung des *bs2cp*-Kommandos.

operandenliste
Liste der Dateiattribute. Das Format der *operandenliste* muss dem Format des FILE-Kommandos (BS2000) entsprechen.
Der Operand „LINK=“ des FILE-Kommandos darf nicht verwendet werden.

Hinweis Werden mehrere *bs2file*-Kommandos für eine Datei gegeben, so sind nur die zuletzt eingestellten Attribute gültig.

Bei ISAM-Dateien werden für die Schlüsselposition und die Schlüssellänge nur die Einstellungen KEYPOS=5 und KEYLEN=8 unterstützt.

Fehler Fehlermeldungen des FILE-Kommandos

Beispiel Die BS2000-Datei *flcopy* soll als PAM-Datei eingerichtet werden.

```
$ bs2file flcopy,fcbtype=pam
```

Siehe auch *bs2cp*, *ftyp*

bs2lp Dateien ausdrucken (send files to a printer) *(BS2000)*

Es werden Druckaufträge an den BS2000 Spool über den Makro PRNT erteilt. Für den Druckauftrag wird keine ID vergeben. Die Verwaltung des Druckauftrags ist nur über den BS2000 Spool möglich.

Syntax **bs2lp**[␣option]...[␣datei] ...

option

-c Diese Option ist implizit immer gesetzt. Die angegebenen Dateien werden kopiert und die Kopien werden ausgedruckt.

-d Ausgabemedium. Es kann ein Druckername oder ein Poolname (Druckerklassenname) angegeben werden. Die Umgebungsvariablen *LPDEST* und *PRINTER* werden nicht unterstützt.

-ncopies
Mit dieser Option legen Sie fest, wie oft die Dateien ausgedruckt werden. Die größte wirksame Angabe für *copies* ist 255, die kleinste ist 1.

-ncopies nicht angeben:
Für copies wird 1 angenommen.

-ttitel
(t - title) Auf der Kopfseite des Ausdrucks wird *titel* ausgedruckt.

-m, -o, -s, -w
werden ignoriert.

datei
Name der Datei, die ausgedruckt werden soll. Sie können auch mehrere Dateien angeben. Sie werden dann in der beim Aufruf gegebenen Reihenfolge ausgedruckt.
Alle Optionen, die für den Ausdruck nur einer Datei gelten sollen, müssen Sie vor dieser Datei angeben.

datei nicht angegeben:
bs2lp liest von der Standardeingabe.

Arbeitsweise

Die POSIX-Datei wird als temporäre SAM-Datei kopiert, d.h. der Operand ERASE ist implitzt immer gesetzt bei Aufruf von SPOOL.

Angabe zusätzlicher Operanden für den Druckauftrag

Alle Schlüsselwort-Operanden des Macros PRNT können implizit über die Datei .lprc angegeben werden, welche im aktuellen HOME-Dateiverzeichnis gesucht wird. Pro Zeile muss mindestens ein PRNT-Operand vollständig definiert sein. Durch Komma getrennt können auch mehrere Operanden in einer Zeile stehen. Die Einträge der Datei .lprc werden ungeprüft an PRNT übergeben.

Beispiel für die Datei .lprc:

```
FORM=STD
LINES=80
ROT=90, CHARS=R01
```

Die gleichzeitige Angabe eines Operanden und einer Option, die auf diesen Operanden abgebildet wird, ist nicht möglich.

Abbildung der Optionen:
- die Option *-d* wird abgebildet auf den BS2000-Operanden DEST,
- die Option *-n* wird abgebildet auf den BS2000-Operanden COPIES,
- die Option *-t* wird abgebildet auf den BS2000-Operanden TEXT.

Fehler `bs2lp: ERROR: No (or empty) input files`

Endestatus 0, wenn Spoolaufträge erfolgreich gestartet wurden.

Beispiel Das Kommando *bs2lp -n2 datei* und die Einträge aus obiger .lprc ergeben folgenden Spool-Aufruf für das BS2000:

```
PRNT datei, COPIES=2,FORM=STD.LINES=80,ROT=90,CHARS=R01
```

bs2pkey P-Tasten belegen (set pkeys) *(BS2000)*

Sie können die P-Tasten P3, P4 und P5 durch den Aufruf des Kommandos *bs2pkey* folgendermaßen belegen:

P3 mit @@c ([CTRL] C)
P4 mit @@d ([CTRL] D)
P5 mit @@z ([CTRL] Z)

Das Kommando hat keine Optionen.

Syntax

bs2pkey

Das Programm wird entweder in der POSIX-Shell (ohne Optionen) aufgerufen oder kann in die Datei */etc/profile* aufgenommen werden. Dann wird das Programm bei jedem Aufruf der Shell aktiviert.

cal Kalender ausgeben (print calendar)

cal gibt einen Kalender auf die Standard-Ausgabe aus.

Syntax **cal**[[␣monat]␣jahr]

Kein Argument angegeben
cal gibt den Kalender für den aktuellen Monat aus.

monat
cal gibt den Kalender für den angegebenen Monat aus.

Für *monat* können Sie Werte von 1 bis 12 angeben.
Wenn Sie einen Wert für *monat* angeben, dann müssen Sie auch einen Wert für *jahr* angeben.

monat nicht angegeben:
cal gibt den Kalender für das ganze Jahr aus.

jahr
cal gibt den Kalender für das angegebene Jahr aus.

Für *jahr* können Sie Werte von 1 bis 9999 angeben.

Beachten Sie: Wenn Sie z.B. *cal 10* eingeben, dann erhalten Sie den Kalender für das Jahr 10 und nicht für das Jahr 2010.

Wenn Sie *cal* mit dem Argument *1752* oder den Argumenten *9 1752* aufrufen, erhalten Sie für den Monat September eine verkürzte Ausgabe. Im September 1752 gab es eine Zeitkorrrektur von 11 Tagen, die von *cal* berücksichtigt wird.

Endestatus Ungleich 0, falls die angegebenen Werte für *jahr* bzw. *monat* außerhalb des zulässigen Bereichs liegen.

Fehler `Bad argument [ for month | for year ]`
Die Werte, die Sie für *monat* bzw. *jahr* angegeben haben, liegen außerhalb des zulässigen Bereichs.

Variable *TZ*
bestimmt die Zeitzone, die zur Berechnung des aktuellen Monatswertes verwendet wird.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cal*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_TIME Legt das Format und Inhalt des Kalenders fest.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Ausgabe des Kalenders für Januar 2009 in der Locale C:

```
$ cal 1 2009
    January 2009
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

cancel Druckaufträge löschen (cancel line printer requests)

Mit *cancel* können Sie Druckaufträge abbrechen oder löschen, die Sie mit den Kommandos *lp* erteilt haben.

Nichtprivilegierte Benutzer können nur Druckaufträge abbrechen oder löschen, die von der eigenen Benutzerkennung aus gestartet wurden.

Mit *lpstat* können Sie sich Informationen über Druckaufträge ausgeben lassen.

Syntax

Format 1: cancel[␣auftragsnr...]␣drucker...

Format 2: cancel␣auftragsnr...␣[drucker...]

Sie müssen für mindestens einen der Operanden einen Wert angeben.

auftragsnr

Für *auftragsnr* geben Sie die Auftragsnummer eines Druckauftrags an. Diese Nummer erhalten Sie auf die Standard-Ausgabe, nachdem Sie mit *lp* den Druckauftrag erteilt haben.
Sie können mehrere Auftragsnummern angeben, jeweils getrennt durch ein Leerzeichen.

Die Auftragsnummern aller anstehenden Druckaufträge können Sie mit dem Kommando *lpstat* abfragen. Nichtprivilegierte Benutzer können nur Druckaufträge abfragen, die von der eigenen Benutzerkennung aus gestartet wurden.
cancel gibt eine Meldung über gelöschte bzw. abgebrochene Druckaufträge auf die Standard-Ausgabe aus.

drucker

Für *drucker* geben Sie den Namen eines RSO-Druckers oder einer Druckergruppe an. Sie können mehrere Namen angeben. Ein auf *drucker* laufender Druckauftrag wird abgebrochen, und der Drucker wird frei für den nächsten Auftrag. *cancel* gibt eine Meldung über abgebrochene Druckaufträge auf die Standard-Ausgabe aus.
Das Kommando *lpstat* gibt aus, auf welchen Druckern gerade Aufträge der eigenen Benutzerkennung laufen.

Fehler

```
cancel: printer "G005" was not busy
```

Sie haben beim *cancel*-Aufruf für *drucker* den Namen einer Druckergruppe angegeben, in diesem Fall G005, auf der kein Druckauftrag lief. Durch die Angabe des Druckergruppennamens können Sie nur einen Druckauftrag abbrechen, der gerade auf einem Drucker dieser Druckergruppe läuft. Sie können aber nicht Druckaufträge löschen, die zwar für diese Druckergruppe gestellt wurden, aber noch auf ihre Ausführung warten. Um das zu erreichen, müssen Sie die Auftragsnummer angeben.

```
cancel: request "TSN-AB35" non existent
```

Sie haben beim *cancel*-Aufruf eine Auftragsnummer angegeben, in diesem Fall TSN-AB35, für die es keinen Druckauftrag gibt. Einen Überblick über alle laufenden und anstehenden Druckaufträge erhalten Sie mit den Kommandos *lpstat*.

```
UX:cancel: "XY" is not a request id or a printer
           Cancel requests by id or by
           name of printer where printing.
```

Sie haben beim *cancel*-Aufruf ein Argument XY angegeben, das weder eine Auftragsnummer noch der Name einer Druckergruppe ist.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cancel*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Sie starten versehentlich einen Druckauftrag doppelt. Deswegen möchten Sie den zuletzt gestarteten Druckauftrag löschen.

```
$ lp test test
request id is TSN-6J1V (test)
request id is TSN-6J1W (test)
$ cancel tsn-6j1w
request “TSN-6J1W“ cancelled
```

Beispiel 2 In Ihrem System gibt es zwei Druckergruppen, G001 und G002, zu denen jeweils ein Drucker gehört.

Der auf der Druckergruppe G001 laufende Auftrag soll abgebrochen werden:

```
$ cancel G001
request "TSN-1234" cancelled
```

Der abgebrochene Druckauftrag hatte die Auftragsnummer TSN-1234.

Die Druckaufträge mit den Auftragsnummern TSN-9WJ7 und TSN-9WAS sollen gelöscht werden:

```
$ cancel TSN-9WJ7 TSN-9WAS
request "TSN-9WJ7" cancelled
request "TSN-9WAS" cancelled
```

Siehe auch *lp*, *lpstat* [12]

cat Dateien aneinanderfügen und ausgeben (concatenate and print files)

Das Kommando *cat* liest Dateien sequentiell und gibt diese auf der Standard-Ausgabe aus. Die Reihenfolge der Zeichen innerhalb der Dateien und deren Format bleiben dabei unverändert.
Werden beim Aufruf von *cat* mehrere Dateien angegeben, so werden diese in der gleichen Reihenfolge nacheinander ausgegeben.
Wenn Sie beim Aufruf keine Datei angeben, so liest *cat* von der Standard-Eingabe.

Syntax **cat**[␣**-s**][␣**-u**]␣datei]...

Keine Option angegeben
: Die Ausgabe erfolgt gepuffert in Blöcken von *BUFSIZ* byte. Der Wert von *BUFSIZ* ist abhängig von der Maschine, auf der Sie arbeiten. Er wird in der Datei */usr/include/stdio.h* definiert und beträgt 8192 byte. Wenn beim Aufruf angegebene Dateien nicht existieren, gibt *cat* eine entsprechende Meldung aus.

-s Meldungen über nicht vorhandene Dateien werden unterdrückt.

-u Byteweise Ausgabe ohne Zwischenspeicherung (ungepuffert).

datei
: Name der Datei, die ausgegeben werden soll. Sie können mehrere Dateien angeben. Wenn Sie für *datei* einen Bindestrich - angeben, liest *cat* von der Standard-Eingabe.

datei nicht angegeben:
cat liest von der Standard-Eingabe.

Achtung!
Wenn Sie die Ausgabe von *cat* auf eine Datei umlenken, von der gelesen wird, dann hat das den Verlust des originalen Dateiinhalts zur Folge. Der Inhalt von *datei1* geht im folgenden Beispiel verloren:

```
cat datei1 datei2 datei3 > datei1
```

Fehler
```
cat >aus_datei aus_datei: cannot create
```
Sie haben kein Schreibrecht für die Ausgabedatei *aus_datei* oder für das Dateiverzeichnis, in dem *aus_datei* enthalten ist.

```
cat ein_datei cat: cannot open ein_datei: Permission denied
```
Sie haben kein Leserecht für die Eingabedatei *ein_datei*.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cat*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Aneinanderhängen und Umleiten der Ausgabe von zwei Dateien:

```
$ echo Montag Dienstag Mittwoch >datei1
$ echo Donnerstag Freitag Samstag >datei2
$ cat datei1 datei2 > datei3
$ cat datei3
Montag Dienstag Mittwoch
Donnerstag Freitag Samstag
```

Beispiel 2 Ausgabe des Inhalts von *datei1*

```
$ cat datei1
Jeder weiß, was so ein Mai-
käfer für ein Vogel sei.
```

In *datei2* schreiben Sie nun zwei Zeilen Text:

```
$ cat > datei2
In den Bäumen hin und her
kriecht und fliegt und krabbelt er.
```

Abhängig vom verwendeten Terminaltyp beeenden Sie die Eingabe mit END oder mit @@d.

Nun bringen Sie den Inhalt von *datei1* und *datei2*, ergänzt um zwei weitere Zeilen, die Sie von der Standard-Eingabe eingeben, in *datei3*. Anschließend lassen Sie den Inhalt von *datei3* ausgeben:

```
$ cat datei1 datei2 - > datei3
Max und Moritz immer munter
schütteln sie vom Baum herunter.
```

Abhängig vom verwendeten Terminaltyp beeenden Sie die Eingabe mit END oder mit @@d.

```
$ cat datei3
Jeder weiß, was so ein Mai-
käfer für ein Vogel sei.
In den Bäumen hin und her
kriecht und fliegt und krabbelt er.
Max und Moritz immer munter
schütteln sie vom Baum herunter.
```

Siehe auch *cp*, *pr*

cd Aktuelles Dateiverzeichnis wechseln (change working directory)

Das in die POSIX-Shell *sh* eingebaute Kommando *cd* macht das angegebene Dateiverzeichnis zu Ihrem aktuellen Dateiverzeichnis.
In einer eingeschränkten Shell wird das Kommando *cd* abgewiesen.

Syntax

Format 1: cd[␣dateiverzeichnis]

Format 2: cd␣-

Format 3: cd␣alt␣neu

Format 1 Dateiverzeichnis mit CDPATH wechseln

cd[␣dateiverzeichnis]

dateiverzeichnis

Name des Dateiverzeichnisses, das Ihr aktuelles Dateiverzeichnis werden soll. Für dieses Dateiverzeichnis brauchen Sie Ausführrecht. Wenn Sie für *dateiverzeichnis* einen relativen oder absoluten Pfadnamen angeben, brauchen Sie Ausführrecht für alle Dateiverzeichnisse, aus denen sich dieser Pfadname zusammensetzt.

Das angegebene Dateiverzeichnis wird ohne Zugriff auf die Umgebungsvariable *CDPATH* (siehe *Variablen*) gesucht, falls der Name mit folgenden Zeichen beginnt:

/ bedeutet, dass die Suche im Dateiverzeichnis / (root) beginnt.
./ bedeutet, dass die Suche im aktuellen Dateiverzeichnis beginnt.
../ bedeutet, dass die Suche im übergeordneten Dateiverzeichnis beginnt.

Beginnt der Name des angegebenen Dateiverzeichnisses mit keinem dieser Zeichen, so wertet *cd* den Inhalt der Umgebungsvariablen *CDPATH* aus:

– Ist die Variable *CDPATH* nicht definiert oder leer, so wird das angegebene Dateiverzeichnis relativ zum aktuellen Dateiverzeichnis gesucht.
– Ist der Variablen *CDPATH* ein Wert zugewiesen, so wird das angegebene Dateiverzeichnis der Reihe nach in den Dateiverzeichnissen gesucht, deren Pfad in der Variablen *CDPATH* enthalten ist. Wenn *cd* das Dateiverzeichnis gefunden hat, schreibt es vor dem Wechsel den absoluten Pfadnamen dieses Dateiverzeichnisses auf die Standard-Ausgabe.

dateiverzeichnis nicht angegeben:
Das Kommando *cd* macht Ihr HOME-Dateiverzeichnis zum aktuellen Dateiverzeichnis. Das HOME-Dateiverzeichnis ist identisch mit dem Login-Dateiverzeichnis, falls Sie der Shell-Variablen *HOME* keinen anderen Pfadnamen zugewiesen haben.

Format 2 **cd␣-**

- Der Bindestrich - als Operand ist gleichbedeutend mit dem Kommando `cd „$OLDPWD“ && pwd`, das zum vorgehenden aktuellen Dateiverzeichnis wechselt und dessen Namen schreibt.

Format 3 Dateiverzeichniswechsel mit Textersetzung

cd␣alt␣neu

cd ersetzt die Zeichenkette *alt* durch *neu* im Namen des aktuellen Verzeichnisses (PWD) und versucht, in dieses neue Verzeichnis zu wechseln.

Fehler `sh: `*datei*`: not found`
Das angegebene Dateiverzeichnis existiert nicht. Das können Sie mit *ls -l* prüfen.

`sh: `*datei*`: not a directory`
Die angegebene Datei ist kein Dateiverzeichnis. Das können Sie mit *ls -l* prüfen.

datei`: permission denied`
Sie haben für das angegebene Dateiverzeichnis kein Ausführrecht.
Wenn Sie für *dateiverzeichnis* einen relativen oder absoluten Pfadnamen angegeben haben, haben Sie kein Ausführrecht für eines der Dateiverzeichnisse, aus denen sich dieser Pfadname zusammensetzt.

`rsh: cd: restricted`
Die aktuelle Shell ist eingeschränkt, deshalb wird *cd* abgewiesen.

Variable *HOME*
enthält den absoluten Pfadnamen Ihres HOME-Dateiverzeichnisses.

CDPATH
Sie können dieser Variablen die absoluten Pfadnamen der Dateiverzeichnisse zuweisen, die *cd* durchsuchen soll.
Standardmäßig ist diese Variable nicht definiert.

OLDPWD
Pfadname des vorhergehenden Dateiverzeichnisses, der von *cd -* benutzt wurde.

PWD
Pfadname des aktuellen Dateiverzeichnisses, der von *cd* nach dem Wechsel in dieses Verzeichnis gesetzt wird.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cd*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Mit der folgenden Eingabe wird das Unterdateiverzeichnis *termine* zum aktuellen Dateiverzeichnis:

```
$ cd termine
$ pwd
/home/rosa/termine
```

Beispiel 2 Die Benutzerin *rosa* hat die Umgebungsvariable *CDPATH* neu definiert. Sie will in ihr Unterdateiverzeichnis *usr* wechseln, landet aber mit den folgenden Angaben im Dateiverzeichnis */usr*:

```
$ echo $CDPATH
/:/home/rosa/termine:.
$ pwd
/home/rosa
$ ls -l
drwx--x--x 2 ROSA        144 Feb 28 12:32 usr
drwx--x--x 2 ROSA        192 Feb 28 11:51 termine
-rw------- 1 ROSA      11734 Mar  7 16:22 probe
   .
   .
   .
$ cd usr
$ pwd
/usr
```

Das Dateiverzeichnis *usr* wird zuerst in den Dateiverzeichnissen gesucht, deren Pfadnamen der Variablen *CDPATH* zugewiesen sind. Hier enthält *CDPATH* als ersten Pfadnamen / für das Root-Dateiverzeichnis. Das aktuelle Dateiverzeichnis durchsucht *cd* als letztes.

Mit der folgenden Eingabe kann die Benutzerin *rosa* verhindern, dass *cd* die Umgebungsvariable *CDPATH* auswertet:

```
$ cd ./usr
$ pwd
/home/rosa/usr
```

Siehe auch *pwd*
chdir() [4]

chgrp Gruppennummer einer Datei ändern (change file group ownership)

chgrp weist einer Datei oder einem Dateiverzeichnis eine neue Benutzergruppe zu.

Nur der POSIX-Verwalter darf die Benutzergruppe für jede Datei beliebig verändern. Mit der Option *_POSIX_CHOWN_RESTRICTED* des Betriebssystems kann für Benutzer ohne POSIX-Administratorrechte die Möglichkeit, eine Benutzergruppe zu verändern, eingeschränkt werden. Ist diese Option gesetzt, darf die Gruppe nur für eigene Dateien geändert werden. Der Benutzer muss dann

– in der Datei */etc/group* als Mitglied der beim chgrp-Aufruf angegebenen neuen Gruppe eingetragen sein (siehe *Datei*),
– aktuell zu dieser neuen Gruppe gehören, d.h., er muss vor dem Aufruf von *chgrp* mit dem Kommando *newgrp* in die neue Benutzergruppe wechseln (siehe *newgrp* und *Beispiel*).

Wird *chgrp* von einem Benutzer ohne POSIX-Verwaltungsrecht aufgerufen, werden für die angegeben Dateien alle gesetzten s-Bits (set-user-ID und set-group-ID, siehe *chmod*) zurückgesetzt.

Syntax **chgrp**[␣**-h**][␣**-R**]␣gid␣datei␣...

-h Ist *datei* ein symbolischer Link, ändert *chgrp* dessen Gruppennummer. Ohne diese Option wird die Gruppe der Datei verändert, auf die der symbolische Link verweist.

-R (R - rekursiv) *chgrp* ändert die Gruppennummer rekursiv in allen angegebenen Dateiverzeichnissen und deren Unterverzeichnissen. Dabei werden auch symbolische Links durchlaufen.

gid
(group id). Neuer Gruppenname oder neue Gruppennummer.
gid muss in der Datei */etc/group* eingetragen sein.

datei
Name der Datei, die eine neue Benutzergruppe erhalten soll. *datei* kann auch ein Dateiverzeichnis sein. Pro Aufruf können Sie mehrere Namen angeben.

Fehler *datei*`: Not super-user`
Sie dürfen die Benutzergruppe der angegebenen Datei nicht ändern, da Sie nicht Eigentümer dieser Datei sind, nicht als Mitglied der angegebenen Gruppe eingetragen sind oder nicht aktuell zu dieser Gruppe gehören. Nur der POSIX-Verwalter darf die Gruppe für alle Dateien beliebig ändern.

`chgrp: unknown group:` *gid*
Sie haben für *gid* einen Gruppennamen angegeben, der nicht in der Datei */etc/group* eingetragen ist.

Datei */etc/group*

Die Datei */etc/group* enthält alle eingerichteten Benutzergruppen. Jede Zeile dieser Datei besteht aus vier Feldern, die durch Doppelpunkte getrennt sind:

```
gruppenname:gruppennummer:benutzer,benutzer
```

Nur der POSIX-Verwalter darf neue Benutzergruppen einrichten und neue Gruppenmitglieder eintragen.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *chgrp*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Sie arbeiten unter der Benutzerkennung *BERTA*. Diese Kennung ist in der Datei */etc/group* als Mitglied der Benutzergruppen *ag* und *prog* eingetragen. Aktuell gehören Sie zu der Benutzergruppe *ag*. Das können Sie daran erkennen, dass bei neu angelegten Dateien für „Gruppe“ der Name *ag* eingetragen wird:

```
$ >datei
$ ls -l datei
-rw-------   1 BERTA    ag          0   Feb 17 15:48 datei
```

Sie möchten nun für *datei* die Benutzergruppe ändern. Die neue Gruppe soll *prog* sein. Dazu wechseln Sie mit dem Kommando *newgrp* in die Gruppe *prog* und ändern dann mit *chgrp* die Gruppe für *datei*:

```
$ newgrp prog
$ chgrp prog datei
$ ls -l datei
-rw-------   1 BERTA    prog        0   Feb 17 15:48 datei
```

Siehe auch *chmod*, *chown*, *id*, *newgrp*
chown() [4]

chmod Zugriffsrechte ändern (change file modes)

chmod ändert die Zugriffsrechte für Dateien.

Nur der Eigentümer der Datei oder der POSIX-Verwalter darf die Zugriffsrechte ändern. Das s-Bit für die Gruppe darf nur ein Benutzer setzen, dessen aktuelle Gruppennummer mit der Gruppennummer der Datei übereinstimmt, deren Zugriffsrechte verändert werden (siehe *chgrp* und *newgrp*).

Syntax **chmod** ␣[**-R**]␣modus␣datei␣...

-R (R - rekursiv) *chmod* ändert rekursiv die Zugriffsrechte aller Dateien in allen angegebenen Verzeichnissen und deren Unterverzeichnissen.

modus

Mit *modus* geben Sie an, wie Sie die Zugriffsrechte für die angegebenen Dateien ändern wollen. Sie können *modus* auf zwei Arten definieren:

– durch eine symbolische Angabe
– durch eine absolute Angabe

datei

Name der Datei, für die Sie die Zugriffsrechte definieren oder ändern wollen. *datei* kann auch ein Dateiverzeichnis sein. Pro Aufruf können Sie mehrere Namen angeben.

Symbolische Angabe

```
[wer]darf[was][,[wer]darf[was]]...
```

wer

Mit *wer* geben Sie an, für wen Sie die Zugriffsrechte ändern wollen.
wer kann sein:

u (u - user) für Eigentümer

g (g - group) für Gruppe

o (o - other) für andere Benutzer

a (a - all) für *ugo*, d.h. für alle Benutzer

oder eine Kombination aus den Buchstaben *u*, *g*, *o*.

wer nicht angegeben:
für *wer* gilt *ugo*, d.h. alle Benutzer. Die Zugriffsrechte für *ugo* werden mit Ausnahme der Bits der Schutzbit-Maske (siehe *umask*) gesetzt.

darf

Mit *darf* geben Sie an, ob Sie die angegebenen Zugriffsrechte erteilen, unverändert lassen oder entziehen wollen.
darf kann sein:

+ Zugriffsrechte neu erteilen

- Zugriffsrechte entziehen

= Zugriffsrechte absolut setzen, d.h., es werden genau die angegebenen Zugriffsrechte erteilt, alle anderen werden entzogen.

was Mit *was* geben Sie an, welche Zugriffsrechte Sie erteilen bzw. entziehen wollen.
was kann eine Kombination folgender Optionen sein:

r (r - read) für Leserecht

w (w - write) für Schreibrecht

x (x - execute) für Ausführrecht bzw. für Recht zum Durchlaufen von Dateien und Dateiverzeichnissen.

X (X - execute) für Ausführrecht bzw. für Recht zum Durchlaufen für eine Datei, wenn mindestens ein x-Bit gesetzt ist, oder von Dateiverzeichnissen. Wenn Datei kein Dateiverzeichnis ist oder wenn für Datei nicht mindestens ein x-Bit gesetzt ist, wird diese Option ignoriert.

s für s-Bit (set-user-ID-Bit bzw. set-group-ID-Bit).
Die Angabe *s* beim *chmod*-Aufruf ist nur zusammen mit *u*, *g* oder *ug* wirksam (wird *wer* nicht angegeben, so wird hierfür *ug* angenommen). Ein gesetztes s-Bit ist nur für ausführbare Binärdateien (nicht für Shell-Prozeduren!) wirksam (siehe *Das s-Bit*).

t für Sticky-Bit (t-Bit).
Nur der POSIX-Verwalter kann das Sticky-Bit setzten. Versucht ein nichtprivilegierter Benutzer das Sticky-Bit zu setzen, so wird dies ignoriert. Die Angabe *t* beim *chmod*-Aufruf ist nur zusammen mit *u*, *a* oder ohne Angabe für *wer* wirksam. Ein gesetztes Sticky-Bit ist nur für ausführbare Dateien wirksam (siehe *Das Sticky-Bit*). Werden die Zugriffsrechte einer Datei mit gesetztem Sticky-Bit geändert, so wird das Sticky-Bit automatisch gelöscht.

l (l - lock) für obligatorische Sperre von Dateien, Dateiverzeichnissen oder Datensätzen. Hierbei können Schreib- und Lesezugriff gesperrt werden, solange ein Programm auf *datei* zugreift.
Für *datei* wird das l-Bit gesetzt, wenn für *datei* das Ausführrecht für Gruppe nicht gesetzt ist und für *datei* das s-Bit für die Gruppe gesetzt ist.

Folgende Beispiele sind deshalb *nicht* korrekt und führen zu Fehlermeldungen:

chmod g+x,+l datei
chmod g+s,+l datei

u Die Zugriffsrechte des aktuellen Eigentümers sollen übernommen werden.

g Die Zugriffsrechte der aktuellen Gruppe sollen übernommen werden.

o Die Zugriffsrechte der aktuellen anderen Benutzer sollen übernommen werden.

was nicht angegeben:
Dies ist nur sinnvoll in Kombination mit dem Gleichheitszeichen = ; dem betreffenden *wer* werden dann alle Zugriffsrechte entzogen.

Wie oben angegeben, können Sie mehrere "wer-darf-was"-Angaben, getrennt durch Kommas, aneinanderreihen, z.B.

```
chmod g-w,o-rw datei
```

chmod arbeitet die Zeichenkette, die Sie für *modus* angeben, von links nach rechts ab. So erhält z.B. durch die Angabe a-w,u+w der Eigentümer Schreibrecht, während es allen anderen entzogen wird.

Absolute Angabe

Eine absolute Angabe für *modus* ist eine drei- oder vierstellige Oktalzahl. Die zulässigen Oktalzahlen erhalten Sie, wenn Sie die untenstehenden oktalen Modi im Binärsystem mit dem bitweisen logischen ODER verknüpfen. Dasselbe Ergebnis erhalten Sie, wenn Sie die Modi (im Oktal- oder Dezimalsystem) addieren. Eine führende Null (weder s-Bit noch Sticky-Bit) können Sie weglassen.
Die angegebenen Zugriffsrechte werden erteilt, alle anderen Zugriffsrechte werden entzogen.

4000 s-Bit für Eigentümer

20#0 s-Bit für Gruppe, wenn # gleich 7, 5, 3 oder 1 ist. Eine obligatorische Sperre wird gesetzt, wenn # gleich 6, 4, 2 oder 0 ist. Der Wert von # wird ignoriert, wenn *datei* ein Dateiverzeichnis ist. In diesem Fall dürfen Sie nur die symbolische Angabe verwenden.

1000 Sticky-Bit (t-Bit)

0400 Leserecht für Eigentümer

0200 Schreibrecht für Eigentümer

0100 Ausführrecht (bzw. Recht zum Durchlaufen von Dateiverzeichnissen) für Eigentümer

0040	Leserecht für Gruppe
0020	Schreibrecht für Gruppe
0010	Ausführrecht (bzw. Recht zum Durchlaufen von Dateiverzeichnissen) für Gruppe
0004	Leserecht für andere Benutzer
0002	Schreibrecht für andere Benutzer
0001	Ausführrecht (bzw. Recht zum Durchlaufen von Dateiverzeichnissen) für andere Benutzer

Beispiel

Wenn Sie für den Eigentümer das Lese-, Schreib- und Ausführrecht setzen wollen, und für Gruppe das Lese- und Ausführrecht, dann geben Sie für *modus* 750 an:
`400 + 200 + 100 + 40 + 10 = 750`

Für die Datei gelten dann die Zugriffsrechte *rwx r-x* ---.

Das s-Bit

Ist für ein ausführbares Programm das s-Bit für den Eigentümer gesetzt, dann ist beim Aufruf des Programms die *effektive* Benutzernummer des zugehörigen Prozesses gleich der Benutzernummer des *Eigentümers* der Datei (und nicht gleich der Benutzenummer des Aufrufers). Das bedeutet, dass der Prozess unter der Benutzernummer des Programm-Eigentümers abläuft. Dadurch kann er auch auf Dateien zugreifen, für die der Programm-Aufrufer direkt kein Zugriffsrecht hat.
Die *reale* Benutzernummer des Prozesses bleibt die des Programm-*Aufrufers*.

Ist das s-Bit für die Gruppe gesetzt, dann ist die *effektive* Gruppennummer des Prozesses gleich der Gruppennummer des Programm-*Eigentümers*. Der Prozess läuft also unter der Gruppennummer des Programm-Eigentümers ab.
Die *reale* Gruppennummer des Prozesses bleibt die des Programm-*Aufrufers*.

Das s-Bit wirkt nur bei ausführbaren Binärdateien (ausführbaren Programmen), nicht aber bei Shell-Prozeduren. Für Dateien, die eine Shell-Prozedur enthalten, kann das s-Bit zwar mit *chmod* gesetzt werden, es bleibt jedoch wirkungslos.

Bei Dateiänderungen werden die s-Bits aus Sicherheitsgründen zurückgesetzt.

Setzt der POSIX-Verwalter für ein Programm, das ihm gehört, das s-Bit, erlaubt er allen Benutzern, die das Programm aufrufen dürfen, alle Operationen, die er selbst mit Hilfe dieses Programms ausführen darf. Er darf das s-Bit also nur dann setzen, wenn sichergestellt ist, dass dadurch keine Sicherheitslücke entsteht, z.B. Daten gefährdet werden.

Beispiel für die Anwendung des s-Bits

Ein Beispiel für die Anwendung des s-Bits ist das Kommando *mailx*, das Nachrichten an den Benutzer USER1 in die Datei `/var/mail/USER1` schreibt.

Die Datei gehört zur Gruppe `MAIL`, Eigentümer ist `USER1`. Sowohl Gruppe als auch Eigentümer haben Lese- und Schreibrecht, alle anderen haben keine Rechte für diese Datei:

```
-rw-rw----      USER1      MAIL    /var/mail/USER1
```

Somit könnte ein anderer Benutzer (USER2) keine Nachrichten in diese Datei schreiben. Da für das Kommando *mailx* jedoch das s-Bit für die Gruppe `MAIL` gesetzt ist, wird USER2 nach Aufruf von *mailx* kurzzeitig Mitglied der Gruppe `MAIL` und hat somit Schreibrecht für die Datei `/var/mail/USER1`.

Das Sticky-Bit (t-Bit)

Nur der POSIX-Verwalter kann das Sticky-Bit (t-Bit) setzen. Versucht ein nichtprivilegierter Benutzer, das Sticky-Bit zu setzen, so wird dies ignoriert.

Das Sticky-Bit wirkt nur bei Dateiverzeichnissen und ausführbaren Dateien. Für andere Dateien kann es zwar mit *chmod* gesetzt werden, es ist jedoch wirkungslos.

Ist das Sticky-Bit bei ausführbaren Dateien gesetzt, kann beim Programmstart das Einlesen des Programms aus der Programmdatei und das Auslagern in den Swap-Bereich teilweise vermieden werden.

Wenn ein Dateiverzeichnis schreibbar ist, aber das Sticky-Bit gesetzt hat, muss eine der folgenden Bedingungen erfüllt sein, um eine Datei unter diesem Dateiverzeichnis zu löschen, zu verschieben oder einen Verweis auf diese Datei zu erzeugen:

– die Datei muss dem Benutzer gehören
– das Dateiverzeichnis muss dem Benutzer gehören
– der Benutzer muss Schreibberechtigung für die Datei haben
– der Benutzer muss ein privilegierter Benutzer sein

Bei der Ausgabe des Kommandos *ls -l* erscheint ein gesetztes Sticky-Bit an der letzten Stelle der Zugriffsrechte. Ist gleichzeitig das x-Bit für „andere Benutzer“ gesetzt, so wird ein *t* ausgegeben, andernfalls ein *T*.

Das l-Bit

Mit Hife der Funktion *lockf()* kann ein Programm eine Datei sperren, solange das Programm auf diese Datei zugreift. Ist für diese Datei das l-Bit gesetzt, bewirkt der Funktionsaufruf eine obligatorische Sperre (mandatory locking) der Datei (siehe *lockf()* [4]).

Fehler

`chmod: ERROR: Invalid mode`
Sie haben für *chmod* einen unzulässigen Modus angegeben.

`chmod: WARNING: Locking not permitted on` *datei*`, a group executable file`
Sie wollten eine Datei sperren, obwohl für diese Datei das Ausführrecht für Gruppe gesetzt ist.

`chmod: WARNING: Execute permission requiered for set-ID on execution for` *datei*
Sie wollten eine Datei sperren, obwohl für diese Datei das s-Bit gesetzt ist.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *chmod*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die folgenden Beispiele beziehen sich alle auf eine Datei mit den Zugriffsrechten *rw- --- ---*. Die ersten beiden Spalten der Tabelle enthalten Angaben für *modus*, die letzte Spalte enthält das Ergebnis des jeweiligen *chmod*-Aufrufs.

Symbolische Angabe	Absolute Angabe	Ergebnis
u-w	400	r--------
-w	400	r--------
go+r	644	rw-r--r--
go=r	644	rw-r--r--
go+rw	666	rw-rw-rw-
=rw	666	rw-rw-rw-
+rx	755	rwxr-xr-x
=r	444	r--r--r--
ug=rw,o=r	664	rw-rw-r--
u=rwx,g=rx,o=	750	rwxr-x---
+x,u+s	4711	rws--x--x
+xt	1711	rwx--x--t

Das Sticky-Bit (letztes Beispiel) kann nur der POSIX-Verwalter setzen. Versucht ein nicht-privilegierter Benutzer, das Sticky-Bit zu setzen, so wird dies ignoriert.

Siehe auch *chgrp*, *ls*, *newgrp*, *umask*
chmod(), *chown()* [4]

chown Eigentümer einer Datei ändern (change file ownership)

chown weist einer Datei oder einem Dateiverzeichnis einen neuen Eigentümer zu.
Nur der POSIX-Verwalter darf den Eigentümer einer Datei beliebig verändern.
Benutzer ohne POSIX-Verwaltungsrecht dürfen hingegen nur den Eigentümer ihrer eigenen Dateien verändern. Ist die Option *_POSIX_CHOWN_RESTRICTED* des Betriebssystems gesetzt, besteht auch diese Möglichkeit nicht.
Wird *chown* von einem Benutzer ohne POSIX-Verwaltungsrecht aufgerufen, dann wird das s-Bit für Eigentümer, 4000, zurückgesetzt.

Syntax

chown[␣**-h**][␣**-R**]␣uid[:gid]␣datei␣...

-h Ist *datei* ein symbolischer Link, ändert *chown* dessen Eigentümer. Ist diese Option nicht gesetzt, wird der Eigentümer der Datei verändert, auf die der symbolische Link verweist.

-R (R - rekursiv) *chown* ändert den Eigentümer rekursiv in allen angegebenen Dateiverzeichnissen und deren Unterverzeichnissen. Dabei werden auch symbolische Links durchlaufen.

uid[:gid]
Benutzerkennung oder Benutzernummer des neuen Eigentümers. Die Gruppennummer kann optional angegeben werden.

datei
Name der Datei, die einen neuen Eigentümer erhalten soll. *datei* kann auch ein Dateiverzeichnis sein. Pro Aufruf können Sie mehrere Namen angeben.

Fehler

`chown: `*datei*`: Not super-user`
Sie dürfen den Eigentümer der Datei *datei* nicht ändern, da Sie nicht Eigentümer von *datei* sind oder in Ihrem System die Variable *_POSIX_CHOWN_RESTRICTED* gesetzt ist.

`chown: Unknown user: `*neueeigentümer*
Sie haben für *neueeigentümer* eine Benutzerkennung angegeben, die nicht in der user table eingetragen ist.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *chown*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Sie arbeiten als POSIX-Verwalter und wollen den Eigentümer einer Datei ändern. Die Datei *text1* der Benutzerin *CARLA* soll als neuen Eigentümer den Benutzer *MARKUS* erhalten. Dazu geben Sie ein:

```
# ls -l text1
-rw-------   1 CARLA    ag          2426   Feb 17 15:48 text1
# chown markus text1
# ls -l text1
-rw-------   1 MARKUS   ag          2426   Feb 17 15:48 text1
```

Siehe auch *chgrp*, *chmod*
chown() [4]

cksum Prüfsummen und Größen von Dateien schreiben (write file checksums and sizes)

Das Kommando *cksum* führt für jede Eingabedatei eine CRC-Prüfung (cyclic redundancy check) durch und schreibt die Anzahl der in den einzelnen Dateien enthaltenen Oktetts in die Standardausgabe. Welche CRC-Prüfung verwendet wird, hängt vom Polynom ab, das im Ethernet-Standard, auf den verwiesen wird, zur CRC-Fehlerprüfung verwendet wird.

Syntax **cksum**[␣**-C**][␣datei...]

Optionen

-C berechnet die Prüfsumme wie in früheren Versionen

datei
Der Pfadname einer zu prüfenden Datei. Die Dateien können von beliebigen Dateityp sein.

datei nicht angegeben:
Die Standardeingabe wird verwendet. Die Standardeingabe muss mit END oder @@d abgeschlossen werden.

Codierung

Die Codierung der CRC-Prüfsumme wird von dem Polynom definiert, das die Prüfsumme erstellt:

$G(x) = x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$

Mathematisch wird der CRC-Wert für eine bestimmte Datei mit folgender Prozedur definiert:

1. Die auszuwertenden n Bits werden als Koeffizienten eines Modulo 2 Polynom (Mx) des Grades n-1 aufgefasst. Diese n Bits stammen aus der angegebenen Datei. Das bedeutendste Bit ist dabei das bedeutendste Bit des ersten Oktetts der Datei, das letzte Bit ist das unbedeutendste Bit des letzten Oktetts. Dieses wird falls erforderlich mit Null-Bits aufgefüllt, damit eine ganze Zahl Oktetts erreicht wird. Ihm folgen ein oder mehrere Oktetts, die die Länge der Datei als Binärwert darstellen, wobei das unbedeutendste Oktett an erster Stelle steht. Es wird die kleinste Anzahl Oktetts verwendet, mit der die Ganzzahl dargestellt werden kann.
2. M(x) wird mit x^{32} multipliziert (d.h. um 32 Bits nach links verschoben) und unter Verwendung der Modulo 2 Division durch G(x) dividiert. Das Ergebnis ist ein Rest R(x) des Grades ≤ 31.
3. Bei den Koeffizienten von R(x) handelt es sich um 32-Bit-Folgen.
4. Die Bit-Folge wird vervollständigt und das Ergebnis ist die CRC-Summe.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cksum*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Stdout Standardausgabe

Für jede erfolgreich verarbeitete Datei erzeugt das Kommando *cksum* eine Ausgabe in folgendem Format:

```
"%u %d %s\n", <prüfsumme>, <# des oktetts>, <pfadname>
```

Wurde keine Datei angegeben, werden der Pfadname und sein führendes Leerzeichen ausgelassen.

Endestatus

0 Alle Dateien wurden erfolgreich verarbeitet.

>0 Ein Fehler ist aufgetreten.

Hinweis Das Kommando *cksum* wird in der Regel für den schnellen Vergleich einer verdächtigen Datei mit einer „sicheren“ Version dieser Datei verwendet. Dadurch kann beispielsweise geprüft werden, ob Dateien, die über eine durch Geräusche gestörte Leitung übertragen wurden, intakt sind. Dieser Vergleich ist jedoch keineswegs kryptografisch sicher. Die Wahrscheinlichkeit, dass eine beschädigte Datei dieselbe Prüfsumme übergibt wie die Originaldatei ist jedoch minimal. Eine beabsichtigte Täuschung ist äußerst schwierig, aber keineswegs undenkbar.

Das Kommando *cksum* kann zwar Eingabedateien eines beliebigen Typs verwenden, die Ergebnisse müssen jedoch nicht dem entsprechen, was bei einer zeichenorientierten Datei oder bei einem nicht in den XSH-Spezifikationen beschriebenen Dateityp erwartet wird. Da dieses Dokument nicht die bei der Eingabe verwendete Blockgröße angibt, müssen bei der CRC-Prüfung von zeichenorientierten Dateien nicht alle Daten in diesen Dateien verarbeitet werden.

Der Algorithmus ist ein in Oktetts unterteilter Bitstrom. Wird eine Datei zwischen zwei Systemen übertragen und werden die Daten dabei geändert (z.B. bei der Übertragung von 8-Bit-Zeichen in 9-Bit-Bytes), können keine identischen CRC-Werte erwartet werden. Implementierungen, die solche Umformungen durchführen, können *cksum* so erweitern, dass auch diese Fälle berücksichtigt werden.

Beispiel Für die Datei *test*, die eine Aufzählung der Wochentage enthält, wird die Prüfsumme berechnet.

```
$ cat test
Montag Dienstag Mittwoch Donnerstag Freitag Samstag Sonntag
$ cksum test
219535433 60 test
```

Siehe auch *sum*

cmp Dateien zeichenweise vergleichen (compare two files)

Mit *cmp* können Sie zwei Dateien zeichenweise vergleichen. Wenn es Unterschiede zwischen den beiden Dateien gibt, dann gibt *cmp* die Unterschiede auf die Standard-Ausgabe aus.
Wenn die beiden Dateien identisch sind, gibt *cmp* nichts aus.

Syntax **cmp**[␣**-l**|␣**-s**]␣datei1␣datei2

Keine Option angegeben
Wenn die Dateien identisch sind, gibt *cmp* nichts aus.

Wenn sich die Dateien unterscheiden, gibt *cmp* die Zeichennummer und die Zeilennummer des *ersten* Unterschieds zwischen *datei1* und *datei2* in folgender Form aus:

```
datei1 datei2 differ: char zeichennummer, line zeilennummer
```

-l Alle Unterschiede werden in folgender Form ausgegeben:

```
zeichennummer   zeichen(datei1)   zeichen(datei2)
```

zeichennummer ist die Position der abweichenden Zeichen ab Dateianfang. Dabei erhält das erste Zeichen einer Datei die Nummer 1, Leerzeichen werden mitgezählt. *zeichennummer* wird dezimal ausgegeben.

zeichen sind die voneinander abweichenden Zeichen in *datei1* und *datei2*. *zeichen* werden oktal ausgegeben. Eine ASCII-Tabelle mit der Aufschlüsselung der oktalen Werte finden Sie in *Tabellen und Verzeichnisse*. Wenn die Dateien identisch sind, wird nichts ausgegeben.

-s *cmp* gibt nichts aus. Der Wert des Endestatus wird zurückgeliefert, aber nicht automatisch am Bildschirm ausgegeben. Der Endestatus kann mit *echo $?* abgefragt werden.

datei1␣datei2
Namen der Dateien, die Sie vergleichen wollen.

- Wenn Sie für *datei1* den Bindestrich - angeben, dann liest *cmp* von der Standard-Eingabe und vergleicht Ihre Eingaben mit *datei2*.
- Wenn eine der beiden Dateien endet, ohne dass *cmp* einen Unterschied feststellen konnte, dann meldet *cmp* mit folgendem Text, dass in der kürzeren Datei EOF (Dateiende) erkannt wurde:

  ```
  cmp: EOF on datei
  ```

- Wenn in einer von zwei sonst gleichen Dateien ein Zeichen fehlt, dann meldet *cmp -l* wegen der Verschiebung der Positionen alle folgenden Zeichen als unterschiedlich.

– Dem ersten Zeichen einer Datei ist als *zeichennummer* die 1 und nicht die 0 zugeordnet.

– Leerzeichen und Neue-Zeile-Zeichen zählen bei den *zeichennummern* mit.

Endestatus

0 Dateien sind identisch

1 Dateien sind unterschiedlich

>1 Fehler: Auf Datei kann nicht zugegriffen werden, oder Argument fehlt.

Fehler `cmp: cannot open` *datei*
Sie haben für eine Datei kein Leserecht oder eine der angegebenen Dateien existiert nicht.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cmp*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Vergleich zweier Dateien mit Ausgabe der unterschiedlichen Zeichen und ihrer Position *zeichennummer*.

```
$ echo 1 2 3 4 5 7 8 a >dat1
$ echo 1 2 3 4 5 6 9 a >dat2
$ cmp -l dat1 dat2
    11 367 366
    13 370 371
$
```

Beispiel 2 Die Shell-Prozedur *loesche.gl* vergleicht zwei Dateien und löscht bei Gleichheit eine der beiden.

```
if cmp -s $1 $2
then
  rm $2
fi
```

Beim Aufruf der Prozedur mit

```
$ loesche.gl dat1 dat2
```

übergeben Sie *dat1* und *dat2* als Stellungsparameter an die Prozedur. *cmp* liefert mit der Option *-s* den Endestatus zurück. Wenn sein Wert gleich 0 ist, wird *dat2* gelöscht, sonst nicht.

Siehe auch *comm, diff*

comm Gleiche Zeilen in zwei sortierten Dateien suchen (select or reject lines common to two files)

comm vergleicht zwei Dateien, deren Zeilen nach der aktuell gültigen Sortierreihenfolge sortiert sind. Zum Sortieren können Sie das Kommando *sort* (siehe *sort*) verwenden.

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

Keine Option angegeben

comm gibt drei Spalten aus, die folgende Bedeutung haben:

Spalte 1: Zeilen, die nur in *datei1* stehen

Spalte 2: Zeilen, die nur in *datei2* stehen

Spalte 3: Zeilen, die in beiden Dateien stehen

option

-1 Spalte 1 wird nicht ausgegeben.

-2 Spalte 2 wird nicht ausgegeben.

-3 Spalte 3 wird nicht ausgegeben.

Es sind auch Kombinationen der Optionen *1, 2* und *3* zulässig, z.B.:

-12 *comm* gibt alle Zeilen aus, die beiden Dateien gemeinsam sind.

-23 *comm* gibt alle Zeilen aus, die nur in *datei1* stehen.

-13 *comm* gibt alle Zeilen aus, die nur in *datei2* stehen.

-123 *comm* gibt nichts aus.

datei1␣datei2

Namen der beiden sortierten Dateien, die Sie vergleichen wollen.
Das Kommando *comm* arbeitet nur korrekt, wenn die beiden zu vergleichenden Dateien sortiert vorliegen. Wenn Sie für einen der beiden Namen den Bindestrich - eingeben, liest *comm* von der Standard-Eingabe.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *comm*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die Sortierreihenfolge fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel In der Datei *buch* sind Buchtitel und die dazugehörenden Autoren erfasst. In jeder Zeile steht ein Buchtitel und dahinter, durch ein Leerzeichen getrennt, der jeweilige Autor. Sie möchten die Datei *buch* nach mehreren Autoren durchsuchen, die Sie in der Datei *autoren1* erfasst haben. Die Dateien *buch* und *autoren1* haben folgenden Inhalt:

```
buch                                                autoren1

"Mann_aus_Apulien" Stern                            Duras
"Wir_sind_noch_einmal_davongekommen" Wilder         Goethe
"Tod_in_Venedig" Mann                               Joyce
"Ulysses" Joyce                                     Mann
                                                    Schiller
                                                    Wilder
```

Sie gehen in folgenden Schritten vor:

– Herausfiltern der Autoren aus *buch* mit *awk*.
– Sortieren der Autoren aus *buch* mit *sort*.
– Umlenken der Ausgabe von *sort* in die neue Datei *autoren2*.
– Vergleichen der beiden Dateien *autoren1* und *autoren2* mit *comm -2*.

```
$ awk '{printf"%s\n",$2}' buch | sort > autoren2
```

Die Datei *autoren2* hat folgenden Inhalt:

```
Joyce
Mann
Stern
Wilder
```

```
$ comm -2 autoren1 autoren2
Duras
Goethe
      Joyce
      Mann
Schiller
      Wilder
```

Ausgegeben werden in der ersten Spalte alle Autoren, die nur in *autoren1* stehen. Darauf folgt der Inhalt der dritten Spalte, das heißt die Namen der Autoren, die in beiden Dateien stehen.

Siehe auch *cmp*, *diff*, *sort*, *uniq*

command einfaches Kommando ausführen (execute a simple command)

command teilt der Shell mit, dass die Argumente als einfache Kommandos auszuführen sind. Die Auswertung der Shell-Funktionen wird unterdrückt.

command liefert auch Informationen darüber, wie der Name eines Kommandos von der Shell interpretiert wird (siehe Format 2).

Syntax

Format 1: command [␣**-p**]␣kommando_name [␣argument ...]

Format 2: command [␣**-v** |␣**-V**]␣kommando_name

Format 1 **command** [␣**-p**]␣kommando_name [␣argument ...]

option

-p Die Kommandosuche wird mit einem Standardwert für *PATH* durchgeführt, mit dem alle Standard-Kommandos sicher gefunden werden.

kommando_name
Der Name eines Kommandos oder eines speziellen eingebauten Kommandos.

argument
Eine der Zeichenketten als Argument für *kommando_name.*

Format 2 **command**[␣**-v** |␣**-V**]␣kommando_name

option

-v Es wird eine Zeichenkette auf die Standard-Ausgabe geschrieben, die den Pfadnamen oder das Kommando angibt, mit dem die Shell in der aktuellen Shell-Umgebung *kommando_name* aufruft.

- Kommandos, reguläre eingebaute Kommandos, *kommando_namen*, die einen Schrägstrich enthalten sowie implementationsspezifische Funktionen, die durch die Variable *PATH* angegeben werden, werden als absolute Pfadnamen geschrieben.
- Shell-Funktionen, spezielle eingebaute Kommandos und reguläre eingebaute Kommandos, die nicht über die Definition in der *PATH*-Variablen erreichbar sind, sowie reservierte Begriffe der Shell werden als einfache Namen angegeben.
- Ein Alias-Name wird als Aufrufzeile geschrieben, die die Alias-Definition darstellt.
- Andernfalls wird nicht in die Ausgabe geschrieben. Der Endestatus gibt an, dass der Name nicht gefunden werden konnte.

Die Standard-Ausgabe wird folgendermaßen formatiert:
„%s\n“, <pfadname oder kommando>

-V Es wird eine Zeichenkette auf die Standard-Ausgabe geschrieben, die angibt, wie der Name im Operanden *kommando_name* in der aktuellen Shell-Umgebung von der Shell interpretiert wird. Obwohl das Format der Zeichenkette nicht angegeben ist, wird trotzdem festgelegt, in welche der folgenden Kategorien *kommando_name* gehört. Außerdem sind dadurch die zugehörigen Informationen festgelegt:

- Kommandos, reguläre eingebaute Kommandos sowie implementationsspezifische Funktionen, die über die Variable *PATH* gefunden werden, werden als solche gekennzeichnet. Sie enthalten den absoluten Pfadnamen in der Zeichenkette.
- Andere Shell-Funktionen werden als Funktionen gekennzeichnet.
- Alias-Namen werden als Alias-Namen gekennzeichnet. Die Definitionen werden in der Zeichenkette angegeben.
- Spezielle eingebaute Kommandos werden als spezielle eingebaute Kommandos gekennzeichnet.
- Reguläre eingebaute Kommandos, die nicht über die Definition in der *PATH*-Variablen erreichbar sind, werden als reguläre eingebaute Kommandos gekennzeichnet. (Der Begriff „regulär“ ist optional.)
- Reservierte Begriffe der Shell werden als reservierte Begriffe gekennzeichnet.

Die Standard-Ausgabe wird folgendermaßen formatiert:
„%s\n“, <unbestimmt>

kommando_name
Der Name eines Kommandos oder eines speziellen eingebauten Kommandos.

Anwendungsgebiet

Über die Reihenfolge für die Kommandosuche können Funktionen reguläre eingebaute Kommandos und die Suchreihenfolge für Pfade außer Kraft setzen. Dies ist notwendig, damit Funktionen, die für Kommandos denselben Namen haben, dieses Kommando aufrufen können (statt eines rekursiven Aufrufs der Funktion).

Der Standard-Systempfad ist über *getconf* verfügbar. Da für *getconf* vor einem Aufruf unter Umständen *PATH* bereits eingerichtet sein muss, steht die folgende Option zur Verfügung:

```
command -p getconf _CS_PATH
```

Gelegentlich kann es von Vorteil sein, die Eigenschaften spezieller eingebauter Kommandos zu unterdrücken. Zum Beispiel:

```
command exec > nicht_beschreibbare_Datei
```

Bei dieser Eingabe wird eine nicht-interaktive Prozedur nicht abgebrochen. Der Ausgabe-Status kann von der Prozedur abgefragt werden.

Für *command, env, nohup, time* und *xargs* wird der Rückgabecode 127 verwendet, wenn ein Fehler auftritt oder *kommando_name* nicht gefunden werden konnte. So können Anwendungen unterscheiden, ob ein Kommando nicht gefunden wurde oder ob das Kommando mit einem Fehler beendet wurde. Der Wert 127 wurde deshalb ausgewählt, weil er in der Regel noch nicht für andere Bedeutungen belegt ist. Meistens werden für „normale Fehlerbedingungen“ kleine Werte verwendet. Die Werte über 128 könnten mit einer Beendingung aufgrund des Empfangs eines Signals verwechselt werden. Ähnlich wurde der Wert 126 ausgewählt um anzuzeigen, dass das Kommando zwar gefunden wurde, aber nicht aufgerufen werden konnte. Manche Prozeduren erzeugen aussagekräftige Fehlermeldungen, die zwischen 126 und 127 unterscheiden. Die Unterscheidung zwischen den Rückgabecodes 126 und 127 beruht auf Erfahrungen mit der Korn-Shell, bei der folgendes gilt: 127 wird verwendet, wenn alle Versuche, das Kommando *exec* auszuführen, mit [ENOENT] fehlschlagen; 126 wird verwendet, wenn ein Versuch, *exec* auszuführen, aus anderen Gründen fehlschlägt.

Da mit den Optionen *-v* und *-V* die Ausgabe für *command* je nach der aktuellen Shell-Umgebung erzeugt wird, wird *command* in der Regel als reguläres eingebautes Kommando der Shell zur Verfügung gestellt. Erfolgt der Aufruf in einer Subshell oder einer besonderen Ausführungsumgebung wie zum Beispiel:

```
(PATH=foo command –v)
nohup command –v
```

so führt dies nicht in jedem Fall zu korrekten Ergebnissen. So können die meisten Implementationen zum Beispiel in besonderen Ausführungsumgebungen keine Alias-Namen, Funktionen oder spezielle eingebaute Kommandos erkennen, wenn der Aufruf mit der Funktion *nohup* oder *exec* erfolgt.

In einem System können zwei Arten regulärer eingebauter Kommandos vorgefunden werden. Beide werden unter *command* separat beschrieben. Die Beschreibung der Kommandosuche berücksichtigt die Implementation eines Standard-Kommandos als reguläres eingebautes Kommando, solange es an der richtigen Stelle in einer *PATH*-Suche gelesen wird. So könnte zum Beispiel *command -v* true den Pfadnamen */bin/true* oder einen ähnlichen Pfadnamen zum Ergebnis haben. Andere, implementationsspezifische Kommandos, die nicht in dieser Dokumentation definiert werden, können allenfalls als eingebaute Kommandos vorkommen. Ihnen wird kein Pfadname zugewiesen. Die Ausgabe wird als (reguläre) eingebaute Kommandos gekennzeichnet. Anwendungen können nicht in jedem Fall das Kommando *exec* ausführen, *nohup* verwenden, sie mit einem anderen *PATH* außer Kraft setzen etc.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *command*:

LANG Gibt einen Standardwert für die Internationalisierungsvariablen an, die nicht gesetzt oder null sind. Ist *LANG* nicht gesetzt oder null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als wäre keine der Variablen definiert.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation von Byte-Folgen als Zeichen fest (z. B. Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt die Position der Meldungskataloge für die Verarbeitung von *LC_MESSAGES* fest.

PATH Legt den bei der Kommandosuche verwendeten Suchpfad fest, außer wenn mit *-p* ein Standardwert gesetzt wurde.

Beispiel 1 Es wird eine Version von *cd* erstellt, die das neue aktuelle Dateiverzeichnis genau ein Mal ausgibt:

```
cd() {
       command cd "$@" >/dev/null
       pwd
      }
```

Beispiel 2 Es wird eine „sichere Shell-Prozedur“ gestartet, bei der die Prozedur nicht vom Vater beeinflusst werden kann:

```
IFS='
'
#   Der vorstehende Wert sollte <space><tab><newline> sein.
#   IFS wird auf den Standardwert gesetzt.

\unalias -a
#   Alle möglichen Alias-Namen werden zurückgesetzt.
#   unalias wird entwertet, damit kein Alias-Name
#   verwendet wird.

unset -f command
#   Das Kommando darf keine Benutzerfunktion sein.

PATH="$(command -p getconf _CS_PATH):$PATH"
# PATH-Präfix einführen.

#   ...
```

Unter der Voraussetzung, dass die Zugriffsrechte für die von *PATH* aufgerufenen Verzeichnisse richtig gesetzt sind, kann die Prozedur an dieser Stelle sicherstellen, dass die aufgerufenen Kommandos auch den Kommandos entsprechen, die angesprochen werden sollten. Dabei wird sehr vorsichtig vorgegangen, weil angenommen wird, dass implementationsspezifische Erweiterungen vorliegen können, die bei Aufruf Benutzerfunktionen zulassen würden. Diese Möglichkeit wird in dieser Dokumentation nicht weiter beschrieben, wird aber als Erweiterung durchaus zugelassen. Zum Beispiel steht die Variable *ENV* vor dem Aufruf einer Prozedur mit einer benutzereigenen Start-Prozedur. Eine solche Prozedur könnte zum Beispiel Funktionen definieren, die die Anwendung beeinflussen.

Siehe auch *sh, type*

compress Dateien komprimieren (compress files)

compress komprimiert Dateien mittels adaptiver Lempel-Ziv-Codierung: Zeichenketten, die sich im Text wiederholen, werden durch eindeutige Codes von 9 bis maximal 16 bit Länge abgekürzt.

Eigentümer, Zugriffsrechte sowie Zugriffs- oder Änderungsdatum der angegebenen Dateien werden nicht verändert, wenn der aufrufende Prozess die passenden Privilegien hat.

Jede angegebene Datei wird ersetzt durch eine Datei gleichen Namens mit dem Suffix *.Z* .

Der Umfang der Komprimierung hängt ab von der Größe der Eingabedatei, vom Wert für *bits* (siehe unten, Option *-b*) sowie von der Verteilung gleicher Zeichenketten.
Dateien, die nur Text oder Quellcode enthalten, werden in der Regel um 50-60% komprimiert. Die verwendete Lempel-Ziv-Codierung erreicht im allgemeinen eine bessere Komprimierung als die Codierung nach Huffman und verbraucht auch weniger Rechenzeit.

Eine Komprimierung wird nicht durchgeführt, wenn
– die zu komprimierende Datei keine einfache Datei ist
– auf die zu komprimierende Datei Verweise bestehen
– die zu komprimierende Datei bereits ein *.Z*-Suffix hat
– die anzulegende *.Z*-Datei bereits existiert und *compress* im Hintergrund abläuft
– die Komprimierung keine Platzersparnis erwarten lässt

Mit *uncompress* können Sie eine komprimierten Datei wieder dekomprimieren.

Mit *zcat* können Sie komprimierte Dateien im Originalzustand auf die Standard-Ausgabe ausgeben. Die komprimierte Datei wird dabei nicht verändert.

Syntax

Format 1: compress [␣**-fv**][␣**-b**␣bits][␣datei ...]

Format 2: compress [␣**-cfv**][␣**-b**␣bits][␣datei]

Keine Option angegeben
Die angegebenen Dateien werden komprimiert, wenn dadurch Speicherplatz gespart werden kann.

option

-c Die Ausgabe von *compress* wird nur auf die Standard-Ausgabe geschrieben. Es wird keine Datei geändert oder angelegt. Bei *-c* darf nur eine Datei angegeben werden.

-f (f - force) Die Komprimierung wird erzwungen, auch wenn dadurch kein Speicherplatz gespart wird oder die anzulegende *.Z*-Datei bereits existiert. Diese Datei wird überschrieben.

-f nicht angegeben:
compress fragt nach, ob eine existierende *.Z*-Datei überschrieben werden soll oder nicht. Diese Nachfrage erfolgt jedoch nicht, wenn *compress* im Hintergrund abläuft.

-v (v - verbose) Die prozentuale Platzersparnis für jede komprimierte Datei wird angezeigt:
`datei Compression: xx.xx% -- replaced with datei.Z`

-b␣bits
Die maximale Größe für den Abkürzungs-Code gleicher Zeichenketten wird auf *bits* Bits festgelegt. Der Wert für *bits* muss zwischen 9 und 16 liegen. Eine Herabsetzung des Wertes bewirkt, dass die Dateien weniger komprimiert werden.
Der Parameter *bits* wird verschlüsselt in der komprimierten Datei abgelegt, zusammen mit einem Kennzeichen (magic number), das sicherstellt, dass eine mehrfache Komprimierung nicht möglich ist.

-b nicht angegeben:
Für *bits* wird 16 angenommen.

Achtung!
Komprimierte Dateien sind nur kompatibel zwischen Maschinen mit großem Prozessdatenbereich. Für Datenübertragungen auf Architekturen mit kleinerem Prozessdatenbereich (64 Kbyte oder weniger) sollte für die Komprimierung die Option *-b␣12* angegeben werden.

datei
Name der Datei, die komprimiert werden soll. Sie können mehrere Dateien angeben. Wenn Sie für eine der Dateien einen Bindestrich – angeben, liest *compress* an dieser Stelle von der Standard-Eingabe.
datei darf kein Dateiverzeichnis sein, und es dürfen keine Verweise auf *datei* bestehen.

Die komprimierte Datei erhält den Namen *datei.Z*, *datei* wird nach erfolgreicher Komprimierung gelöscht. *datei.Z* hat dieselben Zugriffsrechte, dasselbe Zugriffs- bzw. Änderungsdatum wie *datei*.
Die Länge des Namens von *datei* ist abhängig vom verwendeten Dateisystem. Die maximal zulässige Länge für *datei* errechnet sich aus der maximal im verwendeten Dateisystem zulässigen Dateinamenslänge abzüglich zwei Zeichen. Damit ist die Namenserweiterung auf *datei.Z* noch möglich. Ist der Name länger, so wird *datei* nicht komprimiert.

datei nicht angegeben oder für *datei* – angegeben:
Die Daten der Standard-Eingabe werden in komprimierter Form auf die Standard-Ausgabe geschrieben.

Endestatus

0 Komprimierung erfolgreich

1 Fehler

2 Ein oder mehrere Dateien wurden nicht komprimiert, da die Komprimierung die Datei vergrößert hätte.

>2 Fehler

Fehler

datei: filename too long to tack on .Z
Der Name der zu komprimierenden Datei ist zu lang. *compress* komprimiert nicht.

datei -- not a regular file: unchanged
Die angegebene Datei ist keine einfache Datei. *compress* komprimiert nicht.

datei: -- has xx other links: unchanged
Auf die angegebene Datei bestehen *xx* Verweise. *compress* komprimiert nicht.

datei unchanged
Es kann keine Einsparung durch die Komprimierung erzielt werden. *compress* komprimiert nicht. Wollen Sie dennoch komprimieren, so geben Sie die Option *-f* an.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *compress*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Datei *filme*, die in unkomprimiertem Zustand 4862 Byte belegt, wird komprimiert.

```
$ ls -l
total 10
-rw-------  1  FELIX    gruppe1    4862 Aug 19 09:27 filme
$ compress -v filme
compress: filme: 50.78% Compression -- replaced with filme.Z
$ ls -l
total 6
-rw-------  1  FELIX    gruppe1    2393 Aug 19 09:27 filme.Z
```

Siehe auch *uncompress*, *zcat*

cp Dateien kopieren (copy files)

cp kopiert Dateien. Kopieren heißt: die Datei ist nachher physisch zweimal vorhanden.

cp hat vier Formate. Das Kommando kopiert
- entweder eine Datei in eine Datei mit anderem Namen (Format 1)
- oder eine oder mehrere Dateien in ein anderes Dateiverzeichnis, wobei die Kopien dieselben Dateinamen haben (Format 2).
- jede Datei in der Dateihierarchie aus *datei* in einen im Folgenden angegebenen Zielpfad (Formate 3 und 4)

Syntax

Format 1: cp[␣**-fip**]␣datei␣dateikopie

Format 2: cp[␣**-fip**]␣datei␣...␣dateiverzeichnis

Format 3: cp␣**-R**[␣**-fip**]␣datei␣...␣dateiverzeichnis

Format 4: cp␣**-r**[␣**-fip**]␣datei␣...␣dateiverzeichnis

Format 1 **Eine Datei kopieren: cp**[␣**-fip**]␣datei␣dateikopie

-f Kann kein Dateideskriptor für *dateikopie* erhalten werden (Schritt 3.a.ii.), wird versucht, *unlink()* für *dateikopie* aufzurufen. Die Bearbeitung wird fortgesetzt.

-i (i - interaktiv). Wenn *dateikopie* bereits existiert, erwartet *cp* eine Bestätigung, dass diese Datei überschrieben werden darf. Bestätigen Sie, wird kopiert. Jede andere Eingabe verhindert ein Überschreiben.
Handelt es sich bei der Standard-Eingabe nicht um ein Terminal, wird diese Option ignoriert und nicht kopiert.

-p Die folgenden Eigenschaften jeder *datei* werden in der entsprechenden *dateikopie* dupliziert:

1. Der Zeitpunkt der letzten Datenmodifikation sowie der Zeitpunkt des letzten Zugriffs. Schlägt dieser Vorgang aus irgendeinem Grund fehl, so schreibt *cp* eine Meldung auf die Standard-Fehlerausgabe.
2. Die Benutzernummer und die Gruppennummer. Schlägt dieser Vorgang aus irgendeinem Grund fehl, so ist nicht vorhersehbar, ob *cp* eine Meldung auf die Standard-Fehlerausgabe schreibt.
3. Die Bits für die Zugriffsrechte sowie die Bits S_ISUID und S_ISGID. Schlägt dieser Vorgang aus irgendeinem Grund fehl, so schreibt *cp* eine Meldung auf die Standard-Fehlerausgabe.

Können Benutzernummer oder Gruppennummer nicht dupliziert werden, so werden die Bits für die S_ISUID und S_ISGID gelöscht.

Die Reihenfolge für die Duplizierung der oben genannten Eigenschaften ist nicht festgelegt. *dateikopie* wird nicht gelöscht, wenn diese Eigenschaften nicht beibehalten werden können.

datei
Dateiname des Originals.

dateikopie
Dateiname der Kopie.
Wenn es noch keine Datei mit dem Namen *dateikopie* gibt, wird sie neu angelegt.

Ist die Option *-p* nicht gesetzt, so erhält die Kopie die gleichen Zugriffsrechte wie das Original sowie die Benutzer- und Gruppennummer des Benutzers, der *cp* aufgerufen hat. Das Änderungsdatum wird jedoch nicht dupliziert, d.h., dass das aktuelle Datum gesetzt wird.

Achtung!
Wenn es bereits eine Datei mit dem Namen *dateikopie* gibt, wird der Inhalt dieser Datei ohne Rückfrage überschrieben, falls sie nicht die Option *-i* angegeben haben; Zugriffsrechte, Eigentümer und Gruppe bleiben aber unverändert.

Wenn *dateikopie* ein Verweis auf eine Datei ist, bleiben alle Verweise erhalten. Der Inhalt der Datei *dateikopie* wird mit dem Inhalt von *datei* überschrieben.

Format 2 **Dateien in ein anderes Dateiverzeichnis kopieren**

cp[␣**-fip**]␣datei␣...␣dateiverzeichnis

-fip siehe Format 1

datei
Dateiname des Originals. Sie können mehrere Namen angeben und so auf einmal mehrere Dateien kopieren. Die Kopien erhalten jeweils denselben Dateinamen wie die Originale.

Achtung!
Wenn es in *dateiverzeichnis* bereits eine Datei gibt, die denselben einfachen Dateinamen hat wie die Originaldatei, so wird der Inhalt der Datei ohne Rückfrage überschrieben, falls Sie nicht die Option *-i* angegeben haben.

dateiverzeichnis
Name des Dateiverzeichnisses, in das die Kopien eingetragen werden sollen. Es darf nicht das Dateiverzeichnis sein, in dem die Originale stehen.

Ist die Option *-p* nicht gesetzt, so erhalten die Kopien die gleichen Zugriffsrechte wie die Originale sowie die Benutzer- und Gruppennummer des Benutzers, der *cp* aufgerufen hat. Das Änderungsdatum wird jedoch nicht dupliziert, d.h., dass das aktuelle Datum gesetzt wird.

Format 3 **cp**␣**-R**[␣**-fip**]␣datei␣...␣dateiverzeichnis

-fip siehe Format 1

-R Kopiert Dateihierarchien.

Wurde die Option *-R* gesetzt, so werden die folgenden Schritte durchgeführt:

i. *dateikopie* wird mit demselben Dateityp erstellt wie *datei*.

ii. Für *dateikopie* werden die Zugriffsrechte so gesetzt, wie sie auch für *datei* gelten. Anschließend erfolgt eine Modifikation durch die Dateierstellungsmaske des Benutzers, wenn die Option *-p* nicht gesetzt wurde.

Schlägt dieser Vorgang aus irgendeinem Grund fehl, so schreibt *cp* eine Meldung auf die Standard-Fehlerausgabe. *datei* wird nicht weiter bearbeitet. Gegebenenfalls werden aber noch weitere Dateien bearbeitet.

datei␣...␣dateiverzeichnis
siehe Format 2

Format 4 **cp**␣**-r**[␣**-fip**]␣datei␣...␣dateiverzeichnis

-fip siehe Format 1

-r Kopiert Dateihierarchien. Die Behandlung von Gerätedateien ist implementationsabhängig.

Wurde die Option *-r* gesetzt, so ist die Behandlung von special files implementationsabhängig.

datei␣...␣dateiverzeichnis
siehe Format 2

Vorgehensweise

Wenn *dateiverzeichnis* bereits existiert und ein Verzeichnis ist, so ist der Name des Zielpfades für die einzelnen Dateien in der Dateihierarchie eine Verkettung aus *dateiverzeichnis*, einem Schrägstrich und dem relativen Pfadnamen der Datei zu dem Verzeichnis, das *datei* enthält.

Wenn es *dateiverzeichnis* noch nicht gibt und zwei Operanden angegeben sind, so ist der Name des entsprechenden Zielpfades für *datei* das Verzeichnis *dateiverzeichnis*. Der Name des entsprechenden Zielpfades für alle anderen Dateien in der Dateihierarchie ist eine Verkettung aus *dateiverzeichnis*, einem Schrägstrich und dem relativen Pfadnamen der Datei zu *datei*.

Es tritt ein Fehler auf, wenn *dateiverzeichnis* nicht existiert und mehr als zwei Operanden angegeben werden. Es tritt ebenfalls ein Fehler auf, wenn *dateiverzeichnis* existiert und eine Datei eines in der XSH-Spezifikation definierten Typs ist, aber kein Dateiverzeichnis.

In der folgenden Beschreibung ist *datei* die Datei, die kopiert wird, egal ob die Angabe über einen Operanden erfolgt oder ob sie als Datei in einer Dateihierarchie in einem Operanden *datei* angegeben wird. Der Begriff *dateikopie* verweist auf die Datei, die durch den Zielpfad angegeben wird.

Für jede *datei* werden die folgenden Schritte durchgeführt:

1. Wenn *datei* und *dateikopie* auf die gleiche Datei verweisen, schreibt *cp* gegebenenfalls eine Meldung auf die Standard-Fehlerausgabe. *datei* wird nicht weiter bearbeitet. Gegebenenfalls werden aber noch weitere Dateien bearbeitet.
2. Ist *datei* ein Dateiverzeichnis, so werden die folgenden Schritte durchgeführt:
   a) Wurde weder die Option **-R** noch die Option **-r** angegeben, so schreibt *cp* eine Meldung auf die Standard-Fehlerausgabe. *datei* wird nicht weiter bearbeitet. Gegebenenfalls werden aber noch weitere Dateien bearbeitet.
   b) Wurde *datei* nicht als Operand angegeben, und ist *datei* entweder „." oder „..", so wird *datei* nicht weiter bearbeitet. Gegebenenfalls werden aber noch weitere Dateien bearbeitet.
   c) Falls *dateikopie/dateiverzeichnis* vorhanden ist und ein Dateityp ist, der nicht durch die XSH-Spezifikation angegeben wird, so ist das Verhalten implementationsabhängig.
   d) Existiert *dateikopie/dateiverzeichnis*, und es handelt sich nicht um ein Verzeichnis, so schreibt *cp* eine Meldung auf die Standard-Fehlerausgabe. *datei* sowie alle Dateien, die in der Dateihierarchie unterhalb von *datei* angeordnet sind, werden nicht weiter bearbeitet. Gegebenenfalls werden aber noch weitere Dateien bearbeitet.
   e) Existiert das Verzeichnis *dateikopie/dateiverzeichnis* nicht, so wird es mit den Zugriffsrechten erstellt, die auch für *datei* gelten. Anschließend erfolgt eine Modifikation durch die Dateierstellungsmaske des Benutzers, wenn die Option *-p* nicht gesetzt wurde, sowie eine bitweise, inklusive ODER-Operation mit S_IRWXU. Wenn *dateikopie* nicht erstellt werden kann, so schreibt *cp* eine Meldung auf die Standard-Fehlerausgabe. *datei* wird nicht weiter bearbeitet. Gegebenenfalls werden aber noch weitere Dateien bearbeitet. Es ist nicht vorhersehbar, ob *cp* versucht, Dateien in der Dateihierarchie von *datei* zu kopieren.
   f) Dateien im Verzeichnis *datei* werden in das Verzeichnis *dateikopie/dateiverzeichnis* kopiert. Dabei werden die Schritte 1 bis 4 mit den *dateien* durchgeführt.
   g) War *dateikopie/dateiverzeichnis* bereits erstellt, so werden die Zugriffsrechte (gegebenenfalls) geändert und an die Zugriffsrechte von *datei* angepasst. Anschließend erfolgt eine Modifikation durch die Dateierstellungsmaske des Benutzers, wenn die Option *-p* nicht angegeben wurde.
   h) *cp* führt keine weitere Bearbeitung von *datei* durch. Gegebenenfalls werden aber noch weitere Dateien bearbeitet.

3. Ist *datei* eine reguläre Datei, so werden die folgenden Schritte durchgeführt:
   a) Liegt *dateikopie* vor, so werden die folgenden Schritte durchgeführt:
      i. Ist die Option *-i* wirksam, so schreibt *cp* eine Eingabeaufforderung auf die Standard-Fehlerausgabe und liest eine Zeile von der Standard-Eingabe. Stellt die Eingabe keine Bestätigung dar, so führt *cp* keine weitere Bearbeitung von *datei* durch. Gegebenenfalls werden aber noch weitere Dateien bearbeitet.
      ii. Es wird ein Dateideskriptor für *dateikopie* erhalten, indem Schritte entsprechend der Funktion *open()* gemäß der XSH-Spezifikation mit *dateikopie* als Argument für den Pfad und der bitweisen, inklusiven ODER-Operation von O_WRONLY und O_TRUNC als Argument *oflag* durchgeführt werden.
      iii. Misslingt der Versuch, einen Dateideskriptor zu erhalten, und die Option *-f* ist wirksam, so versucht *cp*, die Datei zu entfernen, indem Schritte entsprechend der Funktion *unlink()* gemäß der XSH-Spezifikation mit *dateikopie* als Argument für den Pfad durchgeführt werden. Ist dieser Versuch erfolgreich, so setzt *cp* die Bearbeitung mit dem Schritt 3b fort.
   b) Liegt *dateikopie* nicht vor, so wird ein Dateideskriptor erhalten, indem Schritte entsprechend der Funktion *open()* gemäß der XSH-Spezifikation mit *dateikopie* als Argument für den Pfad und der bitweisen, inklusiven ODER-Operation von O_WRONLY und O_CREAT als Argument *oflag* durchgeführt werden. Die Zugriffsrechte für *datei* werden dem Argument *mode* entnommen.
   c) Misslingt der Versuch, einen Dateideskriptor zu erhalten, schreibt *cp* eine Meldung auf die Standard-Fehlerausgabe. *datei* wird nicht weiter bearbeitet. Gegebenenfalls werden aber noch weitere Dateien bearbeitet.
   d) Der Inhalt von *datei* wird in den Dateideskriptor geschrieben. Treten Schreibfehler auf, so schreibt *cp* eine Meldung auf die Standard-Fehlerausgabe und setzt die Bearbeitung mit Schritt 3e fort.
   e) Der Dateideskriptor wird geschlossen.
   f) *cp* bearbeitet *datei* nicht weiter. Trat in Schritt 3d ein Schreibfehler auf, so ist nicht vorhersehbar, ob *cp* gegebenenfalls noch weitere Dateien bearbeitet. Trat in Schritt 3d kein Schreibfehler auf, so bearbeitet *cp* noch weitere Dateien.

Fehler

`cp: Cannot access` *datei*
datei existiert nicht.

`cp: Cannot open` *datei* `: Permission denied`
Sie haben kein Leserecht für *datei*.

`cp: cannot create` *datei*
Sie haben kein Schreibrecht für das Dateiverzeichnis, in dem *datei* angelegt werden soll, bzw. dieses Dateiverzeichnis existiert nicht.

`cp: dvz directory`
dvz ist ein Dateiverzeichnis und kann nicht kopiert werden (Format 1), bzw. Sie haben *-r* nicht gesetzt (Format 4).

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cp*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Hat diese Variable einen Wert, d.h. sie ist nicht leer, dann überschreibt dieser Wert die Werte alle übrigen Variablen für die internationale Umgebung.

LC_COLLATE Legt die internationale Umgebung für das Verhalten von Bereichen, Äquivalenzklassen und Zeicheneinheiten in erweiterten regulären Ausdrücken für ja/nein-Abfragen fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien) sowie der Zeichenklassen in erweiterten regulären Ausdrücken für ja/nein-Abfragen.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die Datei *fachliteratur* soll kopiert werden, bevor man sie verändert. Die Kopie soll *fl* heißen und im gleichen Dateiverzeichnis stehen wie *fachliteratur*.

```
$ cp fachliteratur fl
```

Beispiel 2 Alle Dateien aus dem aktuellen Dateiverzeichnis, deren Namen mit *dat* beginnen, sollen ins Dateiverzeichnis */home/do/sicher* kopiert werden. Dabei soll auch das Datum der letzten Änderung dupliziert werden

```
$ cp -p dat* /home/do/sicher
$ ls -l /home/do/sicher
total 4
-rw------  1  DOBER  gr1        37 Nov 11 11:11  datei1
-rw------  1  DOBER  gr1        97 Apr 01 13:24  datei2
-rw------  1  DOBER  gr1       116 Dec 31 12:13  datei3
-rw------  1  DOBER  gr1       381 Feb 16 08:08  datei4
```

Siehe auch *chmod*, *ln*, *mv*, *rm*

cpio Dateien und Dateiverzeichnisse ein- und auslagern (copy in and out)

Das Kommando *cpio* hat 3 Funktionen:
- Es kopiert eine oder mehrere Dateien in eine Archivdatei (Format 1)
- Es holt Dateien aus einem zuvor erstellten Archiv (Format 2)
- Es kopiert Dateien in ein Dateiverzeichnis (Format 3)

Das Kommando *cpio* wurde aus dem XPG4-Standard zurückgezogen. Es wird durch das Kommando *pax* abgelöst.

In Zukunft sollte nur noch das Kommando *pax* verwendet werden, da *cpio* nur noch aus Kompatibilitätsgründen zu SINIX-(Prozeduren) unterstützt wird.

Syntax

Format 1: cpio␣-o[**Bacv**][␣**-D**␣archiv]

Format 2: cpio␣-i[**Bcdmrtuvf**][␣**-D**␣archiv][␣muster...]

Format 3: cpio␣-p[**adlmruv**]␣dvz

Bei der Darstellung der drei Formate werden nur die Hauptoptionen **-o**, **-i** und **-p** beschrieben. Die übrigen Optionen sind im Abschnitt „Zusatz- und Einzeloptionen“ auf Seite 248 erläutert. Bei den jeweiligen Formaten ist angegeben, welche Zusatz- und Einzeloptionen Sie verwenden können.

Format 1 **Dateien auslagern**

cpio␣-o[**Bacv**][␣**-D**␣archiv]

Zusatzoption
Bacv

Einzeloption
-D␣archiv

-o (o - out) *cpio* liest von der Standard-Eingabe eine Reihe von Pfadnamen einfacher Dateien und kopiert diese Dateien zusammen mit Status-Informationen in einem speziellen Archiv-Format auf die Standard-Ausgabe oder in das mit der Einzeloption *-D* angegebene Archiv. Das Archiv-Format ist das gleiche wie es vom Kommando *pax -x cpio* erstellt wird. Dabei wird die Ausgabe auf ein Vielfaches von 512 Byte aufgefüllt. Die Anzahl der 512-Byte-Blöcke gibt *cpio* auf die Standard-Fehlerausgabe aus.

Format 2 **Dateien einlagern**

cpio␣**-i**[**Bcdmrtuvf**][␣**-D**␣archiv][␣muster...]

Zusatzoption
Bcdmrtuvf

Einzeloption
-D␣archiv

-i (i - in) *cpio* liest von der Standard-Eingabe oder von dem mit der Einzeloption *-D* angegebenen Archiv, das zuvor mit *cpio -o* erzeugt worden sein muss und holt aus der Eingabe diejenigen Dateien, deren Name zu *muster* (siehe unten) passen.

Diese Dateien werden abhängig von den angegebenen Optionen in den aktuellen Dateiverzeichnis-Baum kopiert. Die so eingelagerten Dateien erhalten die gleichen Zugriffsrechte wie die zuvor mit *cpio -o* ausgelagerten Dateien. Eigentümer und Gruppennummer sind die des Benutzers, der *cpio - i* aufruft.
Nur wenn der Systemverwalter *cpio -i* aufruft, erhalten die eingelagerten Dateien dieselbe Eigentümer- und Gruppennummer, wie sie die mit *cpio -o* ausgelagerten Dateien hatten.

Gerätedateien dürfen nur von Systemverwalter extrahiert werden.

muster
Mit *muster* legen Sie fest, welche Dateien aus dem Archiv extrahiert werden.
Zur Angabe von *muster* können Sie alle Shell-Sonderzeichen zur Generierung von Dateinamen verwenden (siehe Abschnitt „Dateinamen-Erzeugung“ auf Seite 57).

Ist *muster* nicht angebenen, werden alle Dateien aus dem Archiv extrahiert.

Format 3 **Dateien in ein Dateiverzeichnis kopieren**

cpio␣**-p**[**adlmruv**]␣dvz

Zusatzoption
adlmruv

-p (p - pass) *cpio* liest von der Standard-Eingabe eine Reihe von Pfadnamen einfacher Dateien und kopiert sie abhängig von den angegebenen Optionen in das Dateiverzeichnis *dvz*.

Die Anzahl der kopierten Blöcke von 512 Byte gibt *cpio* auf die Standard-Fehlerausgabe aus.

dvz
Name des Dateiverzeichnisses, in das die Dateien kopiert werden.
Dieses Dateiverzeichnis muss auch dann bereits vorhanden sein, wenn die Zusatzoption *-d* angegeben wird.

Zusatz- und Einzeloptionen

Die folgenden Zusatzoptionen können in beliebiger Reihenfolge angegeben werden. Mit Ausnahme der Einzeloption *-D* müssen sie ohne Leerzeichen direkt an die Hauptoption *-o*,*-i* oder *-p* angefügt werden.

a (a - access time) Nach dem Kopieren wird das Zugriffsdatum für die Eingabedateien neu gesetzt. Wenn auch die Zusatzoption *l* gesetzt ist, dann wird das Zugriffsdatum für die Dateien, auf die es Verweise gibt, nicht neu gesetzt.

B (B - block) Bei der Ein- und Ausgabe wird mit einer Blockgröße von 512 Byte gearbeitet. Diese Option ist nur für Gerätedateien mit direktem Zugriff (raw devices) sinnvoll.

Diese Zusatzoption darf nicht gemeinsam mit der Hauptoption *-p* verwendet werden.

c (c- compatible) *cpio* schreibt oder liest die Information im Header des Archivs aus Kompatibilitätsgründen in Zeichenform.

d (d - directory) Unterverzeichnisse werden automatisch angelegt, wenn notwendig.

f Es werden nur Dateien extrahiert, auf die das angebene *muster* nicht passt.

l (l - link) Diese Option ist nur zusammen mit der Hauptoption *-p* zu verwenden. Die Dateien werden nicht kopiert, sondern es werden nur Verweise für sie eingetragen.

m (m - modification time) Das alte Änderungsdatum der Datei bleibt erhalten. Diese Zusatzoption ist für Dateiverzeichnisse unwirksam. Die Zusatzoptionen *m* und *a* dürfen Sie nicht gemeinsam angeben, da sie sich gegenseitig ausschließen.

r (r - rename) Die Dateien können beim Extrahieren interaktiv umbenannt werden. *cpio* fragt bei jeder Datei nach einem neuen Namen.

Wenn Sie mit einer Leerzeile antworten (ENTER-Taste drücken), wird diese Datei nicht umbenannt.

Wenn die Antwort aus einem Punkt (**.**) besteht, wird die Datei ohne Namensänderung kopiert.

Ansonsten wird der Name der Datei auf den Inhalt der Antwort geändert.

t (t - table of contents) *cpio* gibt ein Inhaltsverzeichnis des Archivs aus. Es werden keine Dateien kopiert.

u (u - unconditional) Eine bereits vorhandene Datei wird beim Kopieren ersetzt, auch wenn die vorhandere Datei ein neueres Datum hat als die zu kopierende Datei.

Wenn die Zusatzoption *u* nicht angegeben ist, wird die bereits vorhandene Datei nicht überschrieben.

v (v - verbose) *cpio* gibt die Namen der bearbeiteten Dateien aus.
Wird *v* zusammen mit der Zusatzoption *t* angegeben, so werden zusätzliche Informationen über die bearbeiteten Dateien ausgegeben.

␣**-D**␣archiv
archiv bezeichnet die Archivdatei, in die Dateien eingelagert bzw. aus der Dateien ausgelagert werden sollen. Diese Archivdatei wird anstelle von Standard-Eingabe bzw. Standard-Ausgabe verwendet.

Fehler

`Old format cannot support expanded types on file`
Sie haben die Option *-c* nicht angegeben. *cpio* verwaltet aus Kompatibilitätsgründen die Indexnummer (I-Node) einer zu kopierenden Datei so, dass Indexnummern über 64 Kbyte nur bearbeitet werden können, wenn Sie die Option *-c* angeben.

`directories are not being created`
Bei Format 2 haben Sie die Option *-d* nicht angegeben.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cpio*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Hat diese Variable einen Wert, d.h. sie ist nicht leer, dann überschreibt dieser Wert die Werte alle übrigen Variablen für die internationale Umgebung.

LC_COLLATE Legt die internationale Umgebung für das Verhalten von Bereichen, Äquivalenzklassen und Zeicheneinheiten in erweiterten regulären Ausdrücken für ja/nein-Abfragen fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien) sowie der Zeichenklassen in erweiterten regulären Ausdrücken für ja/nein-Abfragen.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Alle Dateien im aktuellen Dateiverzeichnis, deren Namen mit *mb* beginnen, sollen in eine Archivdatei *mailarchiv* ausgelagert werden:

```
$ find . -name 'mb*' -print | cpio -o -D mailarchiv
15776 blocks
```

Das *find*-Kommando durchsucht das aktuelle Verzeichnis nach Dateinamen, die mit *mb* beginnen und schreibt diese auf Standard-Ausgabe, die durch das Pipe-Zeichen | mit der Standard-Eingabe von *cpio* verbunden ist. *cpio -o* liest die Dateinamen und kopiert die entsprechenden Dateien zusammen mit Status-Informationen in die Archivdatei *mailarchiv* (Option *-D*). Nach der Ausführung gibt *cpio* die Anzahl der kopierten Blöcke auf die Standard-Fehlerausgabe aus.

Beispiel 2 Das Inhaltsverzeichnis der Archivdatei *mailarchiv* soll ausgegeben werden.

```
$ cpio -it -D mailarchiv
mbox
mbox.new
mbox.old
15776 blocks
```

Beispiel 3 Es soll eine Kopie des Dateiverzeichnisses *dir.old* mit allen Unterverzeichnissen erstellt werden. Die Kopie soll den Namen *dir.new* enthalten:

```
$ cd dir.old
$ find . -print | cpio -pdl ../dir.new
```

Durch den *cd*-Aufruf wechseln Sie in das Dateiverzeichnis *dir.old*. Das *find*-Kommando durchsucht dann dieses Dateiverzeichnis und alle untergeordneten Dateiverzeichnisse und gibt die Namen aller gefundenen Dateien auf die Standard-Ausgabe aus. Durch das Pipe-Zeichen | ist die Standard-Ausgabe von *find* mit der Standard-Eingabe von *cpio* verbunden. *cpio* liest die Namen und kopiert die Dateien in das Dateiverzeichnis *dir.new*, das bereits vorhanden sein muss (im Beispiel auf gleicher Ebene im Dateibaum wie *dir.old*). Wenn es erforderlich ist, werden Unterverzeichnisse angelegt (*-d*), d.h. rekursiv werden auch alle Unterverzeichnisse von *dir.old* kopiert. Da die Option *-l* angegeben ist, werden Dateien wenn möglich nicht kopiert, sondern es werden Verweise eingerichtet.

Siehe auch *ar*, *cat*, *echo*, *find*, *ls*, *pax*, *tar*

crontab Kommandos regelmäßig zu bestimmten Zeitpunkten ausführen (schedule periodic background work)

Mit *crontab* können Sie veranlassen, dass Kommandos, Shell-Prozeduren oder ausführbare Programme regelmäßig zu bestimmten Zeitpunkten ausgeführt werden.

Sie können mit *crontab*:
– Kommandoaufträge erteilen (Format 1)
– erteilte Aufträge ausgeben lassen (Format 2)
– erteilte Aufträge löschen (Format 3)
– erteilte Aufträge editieren (Format 4).

Aufträge, die mit *crontab* erteilt werden, bleiben auch dann erhalten, wenn der Auftraggeber mit *exit* die POSIX-Shell beendet oder das POSIX-Subsystem beendet wird. Die Aufträge müssen nicht neu gestartet werden.

Vor dem Aufruf beachten

Die Benutzerkennung muss eine Standardabrechnungsnummer für den *rlogin*-Zugang haben. Diese kann mit den Kommandos ADD-USER, MODIFY-USER-ATTRIBUTES oder ADD-POSIX-USER zugewiesen werden.

Wenn die Datei */usr/lib/cron/cron.allow* existiert, dann dürfen Sie das Kommando *crontab* nur dann aufrufen, wenn Ihre Benutzerkennung in dieser Datei steht.
Wenn die Datei */usr/lib/cron/cron.allow* nicht existiert, dann dürfen Sie das Kommando *crontab* nur dann aufrufen, wenn Ihre Benutzerkennung *nicht* in der Datei */usr/lib/cron/cron.deny* steht.
Wenn weder */usr/lib/cron/cron.allow* noch */usr/lib/cron/cron.deny* existieren, dann darf nur der POSIX-Verwalter *crontab* aufrufen.

Existiert z.B. nur die leere deny-Datei, so dürfen alle Benutzer *crontab* aufrufen.

Die allow/deny-Dateien darf nur der POSIX-Verwalter anlegen und ändern. Sie enthalten pro Zeile eine Benutzerkennung.

Der POSIX-Verwalter muss vor dem Aufruf des *crontab*-Kommandos den Cron-Dämonen starten (*/etc/init.d/cron␣start*).

Achtung!
Wenn Sie *crontab* versehentlich ohne Argumente aufrufen, dürfen Sie das Kommando *nicht* mit CTRL D abbrechen! Denn in diesem Fall würden sämtliche Einträge in Ihrer *crontab*-Datei gelöscht. Benutzen Sie stattdessen DEL.

Syntax

Format 1: crontab[␣datei]

Format 2: crontab␣-l[␣benutzerkennung]

Format 3: crontab␣-r[␣benutzerkennung]

Format 4: crontab␣-e[␣benutzerkennung]

Format 1 **Kommandoaufträge erteilen**

crontab[␣datei]

Wenn Sie mit *crontab* einen Kommandoauftrag erteilen wollen, geben Sie an:
– das Kommando (bzw. die Shell-Prozedur/das Programm), das ausgeführt werden soll,
– den Zeitpunkt, zu dem das Kommando ausgeführt werden soll (z.B. jeden Freitag oder jeden 15. Januar).

Rufen Sie *crontab* im obigen Format auf, dann schreibt *crontab* diese Angaben in Ihre *crontab*-Datei */var/spool/cron/crontabs/benutzerkennung*. Der Prozess */usr/sbin/cron* überprüft jede Minute, ob Kommandos in den jeweiligen *crontab*-Datei ausgeführt werden sollen, und veranlasst ggf. die Ausführung.

Um *kommando* auszuführen, ruft *crontab* von Ihrem Home-Verzeichnis aus eine neue Shell (*sh*) auf. *crontab* richtet zu jeder Shell eine Standardumgebung ein, mit den Variablen *HOME*, *LOGNAME*, *SHELL* (/bin/sh) und *PATH* (:/bin:/usr/bin:/usr/lbin). Wenn Sie Ihre *.profile*-Datei ausführen lassen wollen, müssen Sie dies in Ihrer *crontab*-Datei explizit angeben.

Werden in der *crontab*-Datei die Standard-Ausgabe und Standard-Fehlerausgabe der Kommandos nicht umgelenkt, dann erhalten Sie sowohl die Standard-Ausgabe als auch die Standard-Fehlerausgabe über *mailx*.

Pro Benutzerkennung gibt es höchstens eine *crontab*-Datei. Existiert die *crontab*-Datei */var/spool/cron/crontabs/benutzerkennung* noch nicht, so wird sie beim *crontab*-Aufruf neu angelegt. Existiert sie bereits, so wird sie überschrieben. Wenn Sie also mit *crontab* zusätzliche Kommandoaufträge erteilen wollen, dann verwenden Sie die Option *-e*.

datei

Name der Datei, die die auszuführenden Kommandos und die Zeitpunkte der Ausführung enthält.
datei muss dem unten beschriebenen Aufbau entsprechen (siehe *Aufbau einer crontab-Datei*).

datei nicht angegeben:
crontab liest den Eingabetext von der Standard-Eingabe. Der Text, den Sie in diesem Fall eingeben, muss dem unten beschriebenen Aufbau entsprechen (siehe *Aufbau einer crontab-Datei*).

Format 2 **Erteilte Aufträge ausgeben**

crontab␣**-l**[␣benutzerkennung]

-l (l - list) *crontab* gibt den Inhalt Ihrer *crontab*-Datei */var/spool/cron/crontabs/benutzerkennung* aus.
Ist diese Datei nicht vorhanden, erhalten Sie eine Fehlermeldung (siehe *Fehlermeldungen*).

benutzerkennung
Ist *benutzerkennung* angegeben, wird die *crontab*-Datei des zugehörigen Benutzers gezeigt. Die Angabe von *benutzerkennung* ist jedoch nur dem POSIX-Verwalter erlaubt.

Format 3 **Erteilte Aufträge löschen**

crontab␣**-r**[␣benutzerkennung]

-r (r - remove) *crontab* löscht Ihre *crontab*-Datei */var/spool/cron/crontabs/benutzerkennung*.

benutzerkennung
Ist *benutzerkennung* angegeben, wird die *crontab*-Datei des zugehörigen Benutzers gelöscht. Die Angabe von *benutzerkennung* ist jedoch nur dem POSIX-Verwalter erlaubt.

Format 4 **Erteilte Aufträge editieren**

crontab␣**-e**[␣benutzerkennung]

-e (e - edit) Mit dieser Option können Sie eine Kopie Ihrer *crontab*-Datei oder, falls diese nicht existiert, eine leere Datei editieren und als aktuelle *crontab*-Datei speichern. Die Umgebungsvariable *VISUAL* legt fest, welcher Editor aufgerufen wird. Ist *VISUAL* nicht definiert, so wird die Umgebungsvariable *EDITOR* überprüft. Ist auch *EDITOR* nicht definiert, so wird *ed* aufgerufen.

benutzerkennung
Ist *benutzerkennung* angegeben, wird die *crontab*-Datei des zugehörigen Benutzers bearbeitet. Die Angabe von *benutzerkennung* ist jedoch nur dem POSIX-Verwalter erlaubt.

Aufbau einer crontab-Datei

Eine *crontab*-Datei bzw. der Eingabetext, den Sie beim *crontab*-Aufruf übergeben (siehe *Format 1*), muss folgendermaßen aufgebaut sein:

Der Text besteht
– aus Zeilen, die einen Kommandoauftrag enthalten
– evt. aus Kommentarzeilen.

Leerzeilen sind nicht erlaubt.

Zeilen mit einem Kommandoauftrag

Die Zeilen, die einen Kommandoauftrag enthalten, bestehen jeweils aus sechs Feldern, die durch Leer- oder Tabulatorzeichen voneinander getrennt sind. Die Felder enthalten folgende Angaben:

```
1        2        3               4       5           6
Minute   Stunde   Tag im  Monat   Monat   Wochentag   Kommando
```

Die Felder 1 bis 5 legen den Zeitpunkt fest, zu denen das in Feld 6 angegebene Kommando ausgeführt werden soll.

Minute	Mögliche Angaben: 0-59 oder *muster*
Stunde	Mögliche Angaben: 0-23 oder *muster*
Tag im Monat	Mögliche Angaben: 1-31 oder *muster*
Monat	Mögliche Angaben: 1-12 oder *muster*
Wochentag	Mögliche Angaben: 0-6 (0=Sonntag) oder *muster*
Kommando	Name des Kommandos, das zu dem angegebenen Zeitpunkt ausgeführt werden soll. Mit Hilfe des Prozentzeichens % stellen Sie dem Kommando die Standard-Eingabe zur Verfügung. Die Shell interpretiert jedes Prozentzeichen im Kommandofeld, das nicht durch einen Gegenschrägstrich \ entwertet ist, als Neue-Zeile-Zeichen. Sie führt den Inhalt des Kommandofeldes nur bis zum ersten nicht entwerteten Prozentzeichen oder Neue-Zeile-Zeichen aus; der Rest des Feldes wird dem Kommando als Standard-Eingabe übergeben (siehe *Beispiel 3*).

muster kann sein:
- ein Stern * (steht für alle zulässigen Werte)
- ein Bereich *zahl-zahl*
- eine Liste von zulässigen Zahlen oder Bereichen; die Elemente der Liste sind durch Kommas getrennt.
  Beispiel: 1,3,5 oder 1-3,5

Angabe des Tages

Für die Angabe des Tages stehen zwei Felder zur Verfügung: das 3. Feld (Tag im Monat) und das 5. Feld (Wochentag).

Wenn Sie das 3. Feld und zugleich das 5. Feld mit einer Zahl, einem Bereich oder einer Liste besetzen, sind beide Angaben unabhängig voneinander gültig.

Zum Beispiel bewirkt der Eintrag

```
0 0 1,15 3 1 kommando
```

in der *crontab*-Datei, dass *kommando* sowohl am 1. und 15. März als auch an jedem Montag im März aufgerufen wird.

Wenn Sie für die Angabe des Tages nur ein Feld besetzen wollen, müssen Sie in das andere Feld einen Stern schreiben.
Zum Beispiel bewirkt der Eintrag

```
0 0 * 3 1 kommando
```

in der *crontab*-Datei, dass *kommando* nur an jedem Montag im März aufgerufen wird.

Kommentarzeilen

Eine Kommentarzeile enthält in Spalte 1 das Nummernzeichen #. Darauf kann beliebiger Text folgen. Kommentarzeilen können Sie verwenden, um die Kommandoaufträge zu erläutern oder um die Aufträge zur besseren Übersicht in Abschnitte zu gliedern. Beachten Sie, dass der Eingabetext, den Sie an *crontab* übergeben, beliebig viele Kommentarzeilen, aber keine Leerzeilen enthalten darf.

Fehler

```
crontab: you are not authorized to use cron. Sorry.
```
Sie sind nicht berechtigt, *crontab* aufzurufen. Siehe *Vor dem Aufruf beachten*.

Format 1

```
crontab: can't open your crontab file.
```
Sie haben *crontab* mit dem Namen einer Datei aufgerufen, die nicht existiert oder auf die nicht zugegriffen werden kann.

```
crontab: error in previous line; ...
```
Hat eine Zeile des Eingabetextes, den Sie an *crontab* übergeben haben, nicht den richtigen Aufbau, dann gibt *crontab* diese Zeile aus, gefolgt von obiger Fehlermeldung. Anstelle der drei Punkte steht die genauere Beschreibung des Fehlers, z.B.

```
unexpected character found in line
```
Ungültiges Zeichen in der Zeile gefunden

```
number out of bounds
```
Eine angegebene Zahl liegt außerhalb der zulässigen Grenzen.

Format 2

```
crontab: can't open your crontab file.
```
Ihre *crontab*-Datei */var/spool/cron/crontabs/benutzerkennung* existiert nicht. Sie können sie also auch nicht mit *crontab -l* lesen.

Datei *`/usr/lib/cron/cron.allow`*
Liste der Benutzerkennungen mit Ausführrecht für *crontab*. In jeder Zeile steht jeweils eine Benutzerkennung.

`/usr/lib/cron/cron.deny`
Liste der Benutzerkennungen ohne Ausführrecht für *crontab*. In jeder Zeile steht jeweils eine Benutzerkennung.

`/var/spool/cron/crontabs/benutzerkennung`
crontab-Datei des Benutzers *benutzerkennung*.

Variable Folgende Umgebungsvariablen werden von *crontab* benutzt:

VISUAL Die Umgebungsvariable *VISUAL* legt bei Verwendung der Option *-e* fest, welcher Editor benutzt wird.

EDITOR Falls *VISUAL* nicht gesetzt ist, legt *EDITOR* bei der Verwendung der Option *-e* fest, welcher Editor benutzt wird.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *crontab*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Der Rechner soll Sie jedes Jahr am 15. Mai an den Geburtstag von Tante Emma erinnern. Dazu legen Sie eine Datei *geburtstag* mit folgendem Inhalt an:

```
0 0 15 5 * echo Tante Emma hat heute Geburtstag !!!
```

Beachten Sie, dass Sie nur für den Tag im Monat eine Angabe machen wollen, nicht für den Wochentag. Deshalb müssen Sie in das Feld für den Wochentag einen Stern schreiben.

Nun übergeben Sie Ihre *crontab*-Datei *geburtstag*:

```
$ crontab geburtstag
```

Sie erhalten dann jeden 15. Mai um 0 Uhr die betreffende Nachricht über *mailx*. Das funktioniert natürlich nur dann, wenn der Rechner in dieser Nacht nicht abgeschaltet ist.

Beispiel 2 Sie möchten eine Kollegin und sich selbst jeden Montag um 13 Uhr 30 an die Skigymnastik am Abend erinnern. Die Nachricht soll auf den Bildschirm ausgegeben werden. Sie sitzen am Bildschirm tty007, Ihre Kollegin am Bildschirm tty014. Sie können in den Eingabetext Kommentarzeilen einfügen, damit Sie ihn später mit *crontab -l* besser lesen können. Ihre *crontab*-Datei könnte dann z.B. so aussehen:

```
# *   = alle zulaessigen Werte
# n-n = Bereich
# n,n = Liste
# Wochentag: 0 = Sonntag
#
# Min  Std  Tag(Mon)  Mon  Tag(Wo)  Kommando
#
30   13      *       *    1      echo Heute abend Skigymnastik ! >/dev/tty007
30   13      *       *    1      echo Heute abend Skigymnastik ! >/dev/tty014
```

Ihre Kollegin und Sie selbst erhalten dann jeden Montag um 13 Uhr 30 die Meldung „Heute abend Skigymnastik !“ auf den Bildschirm (und nicht in den Postkasten).

Beispiel 3 Sie möchten der Benutzerin *doro*, die am Rechner *brummbox* arbeitet, jeden Montag mit *mailx* die folgende Nachricht schicken:

```
Guten Morgen,
schoene Woche...
```

Dazu schreiben Sie den folgenden Eintrag in Ihre *crontab*-Datei:

```
0 0 * * 1 mailx doro@brummbox %Guten Morgen, %schoene Woche...
```

Siehe auch *at*, *mailx*, *sh*, *vi*

csplit Datei nach bestimmten Kriterien unterteilen (split files based on context)

csplit teilt den Inhalt einer Datei oder den Eingabetext, den *csplit* von der Standard-Eingabe liest, in kleinere Abschnitte auf. Die Abschnitte, oder nur einige dieser Abschnitte, schreibt *csplit* in je eine Ausgabedatei. Die ursprüngliche Datei bleibt erhalten.

Mit Hilfe von Argumenten geben Sie an, wie *csplit* die Datei aufteilen und für welche Abschnitte *csplit* Ausgabedateien erstellen soll.

Ohne Angabe von *-n* erstellt *csplit* höchstens 100 Ausgabedateien.

Syntax **csplit**[␣**-ks**][␣**-f**␣name][␣**-n**␣zahl]␣datei␣arg1...argn

Keine Option angegeben
Die Ausgabedateien heißen *xx00, xx01* usw.

Auf die Standard-Ausgabe gibt *csplit* für jede erstellte Ausgabedatei die Anzahl der Zeichen aus, die in dieser Datei stehen.
Tritt ein Fehler auf, so entfernt *csplit* alle bereits erstellten Dateien.

option

-f␣name
Die Ausgabedateien heißen *name00, name01* usw.

-f nicht angegeben: Die Ausgabedateien heißen *xx00, xx01* usw.

-k Tritt ein Fehler auf, so bleiben die bereits erstellten Dateien erhalten.

-n␣zahl
Die laufende Nummer der Ausgabedateien besteht aus *zahl* Ziffern, wobei $1 \leq zahl \leq 9$.

Beispiel

Bei *-n␣4* heißen die Ausgabedateien *xx0000*, *xx0001* usw.

-n nicht angegeben: Die laufende Nummer besteht aus 2 Ziffern.

-s Die Ausgabe der Zeichenanzahl wird unterdrückt.

datei
Name der Eingabedatei.
Wenn Sie für *datei* einen Bindestrich - eingeben, liest *csplit* von der Standard-Eingabe.

arg1...argn
Sie können mehrere Argumente angeben. Jedes Argument verweist auf eine Zeile der Eingabedatei. Diese Zeilen bilden die Trennlinien zwischen den Abschnitten, in die *csplit* die Datei aufteilt: Jede solche Zeile ist die erste Zeile eines Abschnitts.
Geben Sie *n* Argumente an, dann teilt *csplit* die Eingabedatei in *n*+1 Abschnitte auf.

Normalerweise schreibt *csplit* jeden Abschnitt in eine Ausgabedatei.

Ausnahme: siehe das Argument %*regulärer_ausdruck*%[+*zahl*][-*zahl*]. Der letzte Abschnitt (Abschnitt *n*) wird auf jeden Fall in eine Ausgabedatei geschrieben.

csplit arbeitet die Argumente der Reihe nach ab. Zu Beginn ist die erste Zeile der Eingabedatei die aktuelle Zeile. Ist ein Argument abgearbeitet, so wird die Zeile, auf die dieses Argument verweist, zur aktuellen Zeile. Die Zeile, auf die das darauffolgende Argument verweist, muss sich in dem Bereich zwischen der aktuellen Zeile (ausschließlich) und dem Ende der Eingabedatei befinden. Z.B. muss die Zeile, auf die das zweite Argument verweist, in der Eingabedatei nach der Zeile stehen, auf die das erste Argument verweist.

argument kann sein:

/regulärer_ausdruck**/**[**+**zahl][**-**zahl]

Ein Argument der Form */regulärer_ausdruck/* verweist, ausgehend von der aktuellen Zeile, auf die nächste Zeile, die zu dem regulären Ausdruck passt. *csplit* schreibt den Abschnitt von der aktuellen Zeile bis (ausschließlich) zu der Zeile, die zu dem regulären Ausdruck passt, in eine Ausgabedatei. Dann wird die zum regulären Ausdruck passende Zeile zur aktuellen Zeile.

Zusammen mit *+zahl* oder *-zahl* verschiebt sich die Abschnittsgrenze um *zahl* Zeilen hinter (+) oder vor (-) die Zeile, die zu dem regulären Ausdruck passt. Entsprechend wird die Zeile, die *zahl* Zeilen hinter (+) bzw. vor (-) der zum regulären Ausdruck passenden Zeile steht, zur aktuellen Zeile.

Zulässig sind einfache reguläre Ausdrücke (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Enthält das Argument Leerzeichen oder Sonderzeichen der Shell (siehe *Tabellen und Verzeichnisse, Sonderzeichen der POSIX-Shell*), dann müssen Sie diese Zeichen mit einem Gegenschrägstrich \ entwerten oder das Argument in Hochkommata '...' einschließen. Der reguläre Ausdruck darf keine Neue-Zeile-Zeichen enthalten.

%regulärer_ausdruck**%**[**+**zahl][**-**zahl]

Ein Argument der Form %*regulärer_ausdruck*% verweist, ausgehend von der aktuellen Zeile, auf die nächste Zeile, die zu dem regulären Ausdruck passt. Die zum regulären Ausdruck passende Zeile wird zur aktuellen Zeile. *csplit* erstellt für den betreffenden Abschnitt **keine** Ausgabedatei.

Zusammen mit *+zahl* oder *-zahl* wird die Zeile, die *zahl* Zeilen hinter (+) bzw. vor (-) der zum regulären Ausdruck passenden Zeile steht, zur aktuellen Zeile.

Zulässig sind einfache reguläre Ausdrücke (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Enthält das Argument Leerzeichen oder Sonderzeichen der Shell (siehe *Tabellen und Verzeichnisse, Sonderzeichen der POSIX-*

Shell), dann müssen Sie diese Zeichen mit einem Gegenschrägstrich \ entwerten oder das Argument in Hochkommata '...' einschließen. Der reguläre Ausdruck darf keine Neue-Zeile-Zeichen enthalten.

nr — Dieses Argument verweist auf die Zeile mit der Nummer *nr*. *csplit* schreibt den Abschnitt von der aktuellen Zeile bis (ausschließlich) zur *nr*-ten Zeile in eine Ausgabedatei. Dann wird die *nr*-te Zeile zur aktuellen Zeile.

{n} — Dieses Argument steht abkürzend für *n* Argumente der oben beschriebenen Formen und bedeutet: "Vorhergehendes Argument *n* **weitere** Male verwenden". *n* ist eine ganze Zahl größer gleich 1.

Das Argument *{n}* kann, getrennt durch Leerzeichen, auf jedes der obengenannten Argumente folgen.
Folgt es auf ein Argument der Form */regulärer_ausdruck/[+zahl][-zahl]* oder *%regulärer_ausdruck%[+zahl][-zahl]*, so wird dieses Argument *n* **weitere** Male verwendet.

Beispiel

```
'/reg_ausdruck/' {2} ist eine Abkürzung für
'/reg_ausdruck/' '/reg_ausdruck/' '/reg_ausdruck/'
```

Folgt *{n}* auf ein Argument der Form *nr*, so wird die Datei ab der *nr*-ten Zeile *n*-mal alle *nr* Zeilen unterteilt.

Beispiel : `100 {2}` ist eine Abkürzung für `100 200 300`

Fehler

`argument – out of range`
Die Zeile, auf die das Argument *argument* verweist, befindet sich außerhalb des zulässigen Bereichs. Der zulässige Bereich erstreckt sich von der aktuellen Zeile (ausschließlich) bis zum Dateiende.

`100 file limit reached at arg ...`
Sie haben so viele Argumente angeben, dass *csplit* mehr als 100 Ausgabedateien anlegen müsste.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *csplit*:

LANG — Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL — Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für das Verhalten von Bereichen, Äquivalenzklassen und Zeicheneinheiten in regulären Ausdrücken fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien) und das Verhalten von Zeichenklassen in regulären Ausdrücken.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die Datei *buch* enthält einen Text, der in drei Kapitel gegliedert ist. Vor dem ersten Kapitel steht ein Vorwort, nach dem letzten ein Anhang. Jedes Kapitel beginnt mit der Überschrift „KAPITEL ...“; der Anhang hat die Überschrift „ANHANG“.
Sie wollen nun das Vorwort, die einzelnen Kapitel und den Anhang jeweils in eine eigene Datei schreiben. Die Ausgabedateien sollen *kap.*. heißen.

```
$ csplit -f kap buch '/KAPITEL/' '/KAPITEL/' '/KAPITEL/' '/ANHANG/'
1636
15124
32743
20344
2576
$ ls
buch         kap00         kap01        kap02        kap03
kap04
```

Die Datei *kap00* enthält das Vorwort; sie besteht aus 1636 Zeichen. Der Anhang steht in der Datei *kap04*.
Der folgende, kürzere *csplit*-Aufruf hat die gleiche Wirkung:

```
$ csplit -f kap buch '/KAPITEL/' {2} '/ANHANG/'
:
```

Nun können Sie die Abschnitte getrennt editieren. Danach setzen Sie sie mit *cat* wieder zusammen:

```
$ cat kap0[0-4] > buch
```

Beispiel 2 Eine Eingabedatei *datei* soll bei jeder hundertsten Zeile unterteilt werden. Dazu geben Sie ein:

```
$ csplit datei 100 {98}
```

Das Argument `{98}` steht für 98 Argumente: 200 300 ... 9900.

Enthält *datei* 9900 oder mehr Zeilen, dann erstellt *csplit* 100 Ausgabedateien. Die erste Ausgabedatei *xx00* enthält Zeile 1 bis 99 (einschließlich), die letzte Ausgabedatei *xx99* enthält den Rest von *datei* ab Zeile 9900.
Enthält *datei* weniger als 9900 Zeilen, dann gibt *csplit* die Fehlermeldung „`{98} - out of range`“ aus und bricht ab. Geben Sie beim Aufruf die Option *-k* an, dann bleiben die bereits erstellten Dateien erhalten:

```
$ csplit -k datei 100 {98}
```

Enthält *datei* z.B. nur 9830 Zeilen, dann ist *xx98* die zuletzt erstellte Ausgabedatei und enthält die Zeilen 9800 bis 9830.

Beispiel 3 Die Datei *prog.c* enthält ein C-Quellprogramm. Das Programm enthält die Funktion *main* sowie höchstens 20 weitere Funktionen. Jede Funktion endet gemäß den C-Konventionen mit einer schließenden geschweiften Klammer am Anfang der Zeile (1.Spalte). Innerhalb einer Funktion stehen schließende geschweifte Klammern nicht in der 1.Spalte einer Zeile.
Jede Funktion soll in eine eigene Datei geschrieben werden. Dazu geben Sie ein:

```
$ csplit -k prog.c '%main(%' '/^}/+1' {19}
```

Enthält das Programm außer der Funktion *main* genau 20 weitere Funktionen, dann teilt *csplit* die Datei in 22 Abschnitte auf.

Abschnitt 0 enthält alle Zeilen vom Beginn der Datei bis ausschließlich zum Beginn der Funktion *main*. Dieser Abschnitt wird in **keine** Ausgabedatei geschrieben (Argument *%main(%*).

Abschnitt 1 enthält die Funktion *main* und wird in die Ausgabedatei *xx00* geschrieben (Argument `/^}/+1`).

Entsprechend werden sukzessive die Funktionen 1 bis 19 in Ausgabedateien geschrieben (Argument {19}). Zum Schluss wird Abschnitt 22, der den Rest der Eingabedatei enthält (das ist gerade die 20. Funktion), in die Ausgabedatei *xx20* geschrieben.

Enthält das Programm weniger Funktionen, so bricht *csplit* bei der letzten Funktion mit der Fehlermeldung „`{19} - out of range`“ ab. Da jedoch Option *-k* gesetzt ist, bleiben die erstellten Ausgabedateien erhalten.

Siehe auch *ed*, *sh*, *split*

cut Bytes, Zeichen oder Felder aus den Zeilen einer Datei ausschneiden (cut out selected fields of each line of a file)

cut liest einen Eingabetext zeilenweise aus Dateien oder von der Standard-Eingabe und schneidet aus den Zeilen bestimmte Bytes (Format 1), Zeichen (Format 2) oder Felder (Format 3) aus. Die ausgeschnittenen Elemente gibt *cut* auf die Standard-Ausgabe aus.

Syntax

Format 1: cut␣-b␣liste[␣**-n**][␣datei..]

Format 2: cut␣-c␣liste[␣datei]...

Format 3: cut␣-f␣liste[␣**-d␣**zeichen][␣**-s**][␣datei]...

Format 1 **Bytes ausschneiden**

cut␣-b␣liste␣[**-n**][␣datei..]

-b␣liste
(b - bytes) *cut* schneidet aus jeder Eingabezeile die Bytes aus, die in der bei *liste* angegebenen Position stehen und gibt sie auf die Standard-Ausgabe aus.
liste ist eine Liste von Zahlen oder Zahlenbereichen. Die Elemente der Liste müssen durch Kommas getrennt und in aufsteigender Reihenfolge angeordnet sein. Ein Bereich *n1-n2* bezieht sich auf alle Zahlen von *n1* bis *n2* einschließlich.

Bei Bereichsangaben sind folgende Kurzformen zulässig:
-*n* für 1-*n* und *n*- für *n*-„letzte Spalte“

Beispiel

1,3,5	*cut* schneidet jeweils die 1., 3. und 5. Spalte aus.
1-3,5	*cut* schneidet jeweils die 1., 2., 3. und 5. Spalte aus.
-3,5	ist Kurzform für 1-3,5
3-	ist Kurzform für 3-„letzte Spalte“

-n Ein Zeichen, das aus mehreren Bytes bestehen kann, wird nicht byteweise ausgeschnitten. Jedes Zeichen von *liste*, das als Bereich *n1-n2* angegeben wird, wird folgendermaßen behandelt:
Ist *n1* nicht das erste Byte des Zeichens, wird *n1* herabgesetzt, um das erste Byte des Zeichens auszuschneiden. Ist *n2* nicht das letzte Byte des Zeichens, wird *n2* herabgesetzt. Es wird das letzte Zeichen vor *n2* ausgewählt. *n2* wird auf Null gesetzt, wenn es kein vorhergehendes Zeichen gibt.

datei
Name der Eingabedatei. Sie können mehrere Dateien angeben.

datei nicht angegeben: *cut* liest von der Standard-Eingabe.

Format 2 **Zeichen ausschneiden**

cut␣**-c**␣liste[␣datei]...

-c␣liste
(c - character) *cut* schneidet aus jeder Eingabezeile die Zeichen aus, die in der bei *liste* angegebenen Position stehen und gibt sie auf die Standard-Ausgabe aus. Eine Spalte ist genau ein Zeichen breit.

liste hat die unter *Format 1* beschriebene Form.

datei
Name der Eingabedatei.
Sie können mehrere Dateien angeben.

datei nicht angegeben: *cut* liest von der Standard-Eingabe.

Format 3 **Felder ausschneiden**

cut␣**-f**␣liste[␣**-d**␣zeichen][␣**-s**][␣datei]...

-f␣liste
(f - field) *cut* schneidet aus jeder Eingabezeile die in *liste* angegebenen Felder aus und gibt sie auf die Standard-Ausgabe aus.
Ein Feld besteht aus den Zeichen, die zwischen zwei Feldtrennzeichen stehen. Zwei aufeinanderfolgende Feldtrennzeichen begrenzen ein leeres Feld. Standard-Feldtrennzeichen ist das Tabulatorzeichen. Mit der Option *-d* können Sie das Feldtrennzeichen umdefinieren.
Die ausgegebenen Felder sind durch je ein Feldtrennzeichen voneinander getrennt.
Eingabezeilen ohne Feldtrennzeichen werden normalerweise vollständig ausgegeben (Ausnahme: Option *-s*). Dies ist bei Tabellentiteln nützlich.

liste hat die unter *Format 1* beschriebene Form.

-d␣zeichen
Feldtrennzeichen ist das Zeichen *zeichen*.
Ist *zeichen* das Leerzeichen oder ein Shell-Sonderzeichen (siehe *Tabellen und Verzeichnisse, Sonderzeichen der POSIX-Shell sh*), dann müssen Sie *zeichen* in Hochkommata einschließen: -d␣'*zeichen*'.

-d nicht angegeben: Feldtrennzeichen ist das Tabulatorzeichen.

-s Zeilen ohne Feldtrennzeichen werden nicht ausgegeben.

-s nicht angegeben: Zeilen ohne Feldtrennzeichen werden vollständig ausgegeben.

datei
Name der Eingabedatei. Sie können mehrere Dateien angeben.

datei nicht angegeben: *cut* liest von der Standard-Eingabe.

Fehler

`ERROR: line too long`
Eine Zeile kann maximal 1023 Zeichen oder Felder lang sein. Möglicherweise kann auch das Neue-Zeile-Zeichen fehlen.

`cut: Bad list for b/c/f option`
Falsch angegebene *liste* oder fehlende Option *-b, -c* oder *-f*. Es wird kein Fehler gemeldet, wenn die Zeile weniger Felder hat, als *liste* fordert.

`cut: No delimiter`
Das Feldtrennzeichen zur Option *-d* fehlt.

`ERROR: cannot handle multiple adjacent backspaces`
Angrenzende Rückschritt-Zeichen können nicht richtig abgearbeitet werden.

`cut: Cannot open` *datei*
Entweder existiert die Datei *datei* nicht oder sie kann nicht gelesen werden. Haben Sie mehrere Dateien angegeben, dann wird mit deren Bearbeitung fortgefahren.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *cut*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die ersten 72 Zeichen jeder Eingabezeile ausgeben:

```
$ cut -c1-72 datei
```

Beispiel 2 Aus den Zeilen der Datei */etc/group* sollen jeweils das erste (Gruppenname) und dritte Feld (Gruppennummer) ausgeschnitten und ausgegeben werden. Die Felder in dieser Datei sind durch einen Doppelpunkt getrennt.

```
$ cut -f1,3 -d: /etc/group
```

Beispiel 3 Aus der Datei *rechnung* eines Versandhauses sollen die Namen der Kunden, deren Rechnung im Mai 05 fällig ist, zusammen mit dem Rechnungsbetrag ausgefiltert werden.
Die Datei hat folgenden Inhalt:

```
Becker          Muenchen        10.000          13.05.05
Brauer          Nuernberg        7.000          01.07.05
Drechsler       Hamburg          8.000          07.05.05
Ebersbusch      Stuttgart          450          20.06.05
```

Die Felder der Tabelle sind durch genau ein Tabulatorzeichen voneinander getrennt und mit Leerzeichen aufgefüllt.

```
$ grep '\.05\.05' rechnung | cut -f1,3 > namen
$ cat namen
Becker          10.000
Drechsler        8.000
```

Erläuterung:
grep schreibt alle Zeilen der Datei, die die Zeichenkette .05.05 enthalten, auf die Standard-Ausgabe. Diese Zeilen erhält *cut* als Eingabe und schneidet aus ihnen das erste und dritte Feld aus. Die Ausgabe von *cut* wird in die Datei *namen* geschrieben.

Wollen Sie den Kundennamen in der Datei *namen* zusätzlich das Datum voranstellen und dann das Ergebnis nach dem Datum sortieren, dann geben Sie ein:

```
$ grep '\.05\.05' rechnung | cut -f 4 > datum
$ paste datum namen | sort
07.05.05        Drechsler        8.000
13.05.05        Becker          10.000
```

Erläuterung:
Mit der ersten Kommandozeile schreiben Sie das Datum, das zu den ausgewählten Kundennamen gehört, in die Datei *datum*. Der *paste*-Aufruf fügt die Zeilen der Dateien *datum* und *namen* horizontal, getrennt durch ein Tabulatorzeichen, zusammen. *sort* sortiert die Ausgabezeilen aufsteigend nach dem Datum.

Siehe auch *awk*, *grep*, *paste*

date Datum und Uhrzeit ausgeben (write the date and time)

date schreibt das aktuelle Datum und die Uhrzeit auf die Standard-Ausgabe.

Das Format, in dem *date* Datum und Uhrzeit ausgibt, hängt vom Wert der Umgebungsvariablen *LC_TIME* bzw., wenn diese leer oder nicht definiert ist, vom Wert der Umgebungsvariablen *LANG* ab. Sind *LC_TIME* und *LANG* beide leer bzw. nicht definiert, ist die entsprechende Datenbasis nicht vorhanden oder hat eine der NLS-Umgebungsvariablen einen ungültigen Wert, dann verhält sich *date*, als wäre das System nicht internationalisiert, d.h., es gibt Datum und Uhrzeit im amerikanischen Format aus.

Arbeitsweise

Das System arbeitet mit der Weltzeit UTC (Universal Time Coordinated; gleichbedeutend mit GMT - Greenwich Mean Time). *date* wandelt die UTC in die Ortszeit um und umgekehrt. Ist die Umgebungsvariable *TZ* definiert und die Option *-u* nicht gesetzt, so wird sie zur Bestimmung der Zeitzone bzw. zur Umrechnung der UTC in die Ortszeit verwendet.

Syntax

Format 1: date[␣**-u**][␣**+**format]

Format 2: date␣**-a**[␣**-**]␣sss.fff

Format 1 **date**[␣**-u**][␣**+**format]

Kein Argument angegeben
date gibt das aktuelle Datum und die Uhrzeit in der aktuell gültigen internationalisierten Umgebung aus.

-u *date* gibt das aktuelle Datum und die Uhrzeit in Greenwich-Zeit aus.
Wenn die Option *-u* angegeben ist, dann wird die Angabe *+format* ignoriert.

+format
Mit dem Argument *format* bestimmen Sie das Ausgabeformat von *date*. Enthält *format* Leer- bzw. Tabulatorzeichen oder sonstige Sonderzeichen der Shell, die die Shell nicht interpretieren soll, dann schließen Sie *format* in Hochkommas ein:
+'format'.

format ist dem Format des ersten Arguments der C-Funktion bzw. der *awk*-Funktion *printf()* ähnlich (siehe *awk*, *printf* und C-Funktion *printf()* [4]).

format erlaubt es, mit Hilfe von Feldbezeichnern die Ausgabe von *date* zu formatieren. Feldbezeichner sind von der Form %*buchstabe* (siehe Seite 268). Sie werden bei der Ausgabe durch ihren Wert ersetzt. Alle Zeichen, die nicht Teil eines Feldbezeichners sind, werden unverändert ausgegeben. Am Schluss der Ausgabe wird auf jeden Fall ein Neue-Zeile-Zeichen ausgegeben.

Format 2 **date**␣**-a**[␣**-**]␣sss.fff

-a[␣**-**]␣sss.fff
Die Uhr des Systems wird *sss* Sekunden und *fff* Sekundenbruchteile nachgestellt. Die Uhr kann vor oder zurück (-) gestellt werden. Die Korrektur wird durchgeführt, indem die Systemuhr so lange beschleunigt oder verlangsamt wird, bis die angegebene Differenz ausgeglichen ist.

Feldbezeichner

In der folgenden Übersicht sind die möglichen Feldbezeichner aufgelistet. Zwischen den Feldbezeichnern *%h* und *%b* besteht kein Unterschied; aus Kompatibilitätsgründen wurden jedoch beide Bezeichner beibehalten.

%n	Neue-Zeile-Zeichen
%t	Tabulatorzeichen
%c	Datum und Uhrzeit im Default-Format der aktuell gültigen internationalisierten Umgebung
%C	Jahrhundert (2stellig, die ersten beiden Ziffern der Jahreszahl, 00-99)
%D	Datum im Format %m/%d/%y
%x	Datum im Format der aktuell gültigen internationalisierten Umgebung
%y	Jahr (2stellig, die letzten beiden Ziffern der Jahreszahl, 00-99)
%Y	Jahr (4stellig, alle Ziffern der Jahreszahl)
%m	Monat (01 bis 12)
%h	Monat (in Buchstaben, abgekürzt) im Format der aktuell gültigen internationalisierten Umgebung
%b	wie %h
%B	Monat (in Buchstaben, ausgeschrieben) im Format der aktuell gültigen internationalisierten Umgebung
%W	Woche im Jahr (00 bis 53, Montag ist der erste Tag in der Woche)
%V	Woche im Jahr (01 bis 53, Montag ist der erste Tag in der Woche). Die erste Woche im Januar wird nur mitgezählt, wenn sie mindestens 4 Tage enthält. Andernfalls zählt diese Woche zum Vorjahr (nach ISO 8601).
%U	Woche im Jahr (00 bis 53, Sonntag ist der erste Tag in der Woche)
%j	Tag im Jahr (001 bis 366)
%d	Tag im Monat (01 bis 31)
%e	Tag im Monat (1 bis 31), bei einstelligen Datumsangaben wird ein Blank vorangestellt
%a	Wochentag (in Buchstaben, abgekürzt) im Format der aktuell gültigen internationalisierten Umgebung
%A	Wochentag (in Buchstaben, ausgeschrieben) im Format der aktuell gültigen internationalisierten Umgebung
%w	Wochentag (0 bis 6, Sonntag = 0)
%u	Wochentag (1 bis 7, Montag = 1)
%R	Uhrzeit im Format %H:%M

%T Uhrzeit im Format %H:%M:%S
%X Uhrzeit im Format der aktuell gültigen internationalisierten Umgebung
%r Uhrzeit in 12-Stunden-Notation: %I:%M:%S %p
%H Stunde (00 bis 23)
%I Stunde (01 bis 12)
%p Ante-meridiem- bzw. Post-meridiem-Affix im Format der aktuell gültigen internationalisierten Umgebung
%M Minute (00 bis 59)
%S Sekunde (00 bis 61)
%Z Name der Zeitzone oder keine Ausgabe, falls keine Zeitzone vorhanden ist (abhängig von der Umgebungsvariablen *TZ*, siehe *POSIX-Shell-Variablen*)

Modifizierte Feldbezeichner

Falls in Ihrer lokalen Umgebung eine alternative Darstellung (z.B. vor und nach Christus) definiert ist, können Sie diese über modifizierte Feldbezeichner abrufen. Modifizierte Feldbezeichner sind von der Form *%Ebuchstabe* oder *%Obuchstabe*. In der folgenden Übersicht sind die möglichen modifizierten Feldbezeichner aufgelistet.

Ist keine alternative Darstellung definiert, geben alle modifizierten Feldbezeichner den Wert des jeweiligen nicht modifizierten Feldbezeichners aus.

%Ec Datum und Uhrzeit im alternativen Format
%EC Name des Zeitabschnitts in der alternativen Darstellung
%Ex Datum im alternativen Format
%EX Uhrzeit im alternativen Format
%Ey Jahr, mit dem der Zeitabschnitt in der alternativen Darstellung beginnt.
%EY Jahr in der alternativen Darstellung
%Od Tag im Monat in alternativer Zifferndarstellung
%Oe Tag im Monat in alternativer Zifferndarstellung
%OH Stunde (24-Stunden Uhr) in alternativer Zifferndarstellung
%OI Stunde (12-Stunden Uhr) in alternativer Zifferndarstellung
%Om Monat in alternativer Zifferndarstellung
%OM Minuten in alternativer Zifferndarstellung
%OS Sekunden in alternativer Zifferndarstellung
%Ou Wochentag im alternativen Format (Montag = 1)
%OU Woche im Jahr in alternativer Zifferndarstellung (Regel wie %U)
%OV Woche im Jahr in alternativer Zifferndarstellung (Regel wie %V)
%Ow Wochentag im alternativen Format (Sonntag = 0)
%OW Wochen im Jahr in alternativer Zifferndarstellung (Regel wie %W)
%Oy Jahr im alternativen Format

Variable *TZ*

Die Umgebungsvariable *TZ* enthält, falls sie definiert ist, Informationen über die Zeitzonen. *date* benutzt die Variable *TZ* zur Bestimmung der Zeitzone bzw. zur Umrechnung der Weltzeit UTC (Universal Time Coordinated) in die Ortszeit und umgekehrt.

Der Standardwert "`MEZ-1MSZ-2,M3.5.0/02:00:00,M10.5.0/03:00:00`" der Variablen *TZ* ist wie folgt zu interpretieren:

– Standard-Zeitzone
  Name MEZ
  Zeitdifferenz Zone - 01:00:00 = UTC
– alternative Zeitzone
  Name MSZ
  Zeitdifferenz Zone - 02:00:00 = UTC
– Umschaltzeitpunkt auf die alternative Zeitzone
  Monat 3 = März
  Woche 5 (oder 4)
  Wochentag 0 = Sonntag
  Uhrzeit 02:00:00
– Umschaltzeitpunkt auf die Standard-Zeitzone
  Monat 10 = Oktober
  Woche 5 (oder 4)
  Wochentag 0 = Sonntag
  Uhrzeit 03:00:00

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *date*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_TIME Legt fest, in welcher Sprache und in welchem Format das Datum und die Uhrzeit ausgegeben werden bzw. der Wert der Datums- und Zeitkonstanten ausgegeben wird.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Datum und Uhrzeit ausgeben lassen
Wenn Sie am 19. Juni 2009 um 17 Uhr MESZ *date* ohne Argument aufrufen, erhalten Sie, falls die Systemuhr die richtige Zeit angibt, die Ausgabe

```
Fri Jun 19 17:00:00 MSZ 2009
```

Mit der Kommandozeile

```
$ date '+DATE: %m/%d/%y%nTIME: %H:%M:%S'
```

erzeugen Sie folgende Ausgabe:

```
DATE: 06/19/09
TIME: 17:00:00
```

Siehe auch *cal*
ctime(), printf() [4]

dd Dateien kopieren und konvertieren (convert and copy a file)

Mit *dd* können Sie Dateien kopieren und dabei konvertieren, z.B von EBCDIC nach ASCII oder umgekehrt.

Da das Kommando *dd* in beliebigen Block-Größen lesen und schreiben kann, eignet es sich gut für die Ein- und Ausgabe über im Raw-Modus betriebene Geräte (raw devices). Allerdings ist darauf zu achten, dass raw devices nur mit Blockgrößen beschrieben werden können, die für das betreffende Gerät spezifisch sind (z.B. 512 Byte für Diskettenlaufwerke). Wenn die zu konvertierende Dateigröße nicht ein Vielfaches der spezifischen Blockgröße ist, müssen Sie die Option *conv=sync* angeben.

Nach beendeter Bearbeitung gibt *dd* die Zahl der ganzen Blöcke und der Teil-Blöcke, die eingelesen bzw. ausgegeben wurden, auf die Standard-Fehlerausgabe aus.

Achtung!
Verwenden Sie *dd* nie zum Kopieren von Dateien zwischen Dateisystemen mit unterschiedlicher Blockgröße.
Wenn Sie ein blockorientiertes Gerät (block device) zum Kopieren der Datei verwenden, dann wird der letzte Block möglicherweise durch Null-Bytes bis zur Blockgröße aufgefüllt.

Syntax **dd**[␣option...]

Keine Option angegeben:
dd liest von der Standard-Eingabe und gibt auf die Standard-Ausgabe aus. *dd* führt keine Konvertierungen durch. Die Größe der Eingabe-Blöcke und der Ausgabe-Blöcke beträgt jeweils 512 Byte.

option

if=eingabedatei	Name der Eingabe-Datei
of=ausgabedatei	Name der Ausgabe-Datei
ibs=n	Größe der Eingabe-Blöcke in Byte
obs=n	Größe der Ausgabe-Blöcke in Byte
bs=n	Größe der Ein-und Ausgabe-Blöcke in Byte
cbs=n	Größe des Konvertierungspuffers in Byte
skip=n	Überspringen der ersten *n* Blöcke der Eingabe-Datei
iseek=n	Überspringen der ersten *n* Blöcke der Eingabe-Datei
oseek=n	Kopie der Eingabe-Datei ab *n+1* Blöcken der Ausgabe-Datei
seek=n	Kopie der Eingabe-Datei ab *n+1* Blöcken der Ausgabe-Datei

count=n Die ersten *n* Blöcke der Eingabe-Datei kopieren/konvertieren

conv=wert[,wert...] Angabe mehrere Werte für *conv*

conv=ascii Konvertierung von EBCDIC nach ASCII

conv=ebcdic Konvertierung von ASCII nach EBCDIC

conv=siemens Konvertierung von ASCII nach Siemens-spezifischen EBCDIC

conv=ibm Konvertierung von ASCII nach IBM-spezifischen EBCDIC

conv=block Konvertierung von Sätzen, die mit Neue-Zeile-Zeichen abgeschlossen sind in Sätze mit fester Satzlänge

conv=unblock Konvertierung von Sätzen mit fester Satzlänge in Sätze, die mit Neue-Zeile-Zeichen abgeschlossen sind

conv=lcase Umwandlung von Großbuchstaben in Kleinbuchstaben

conv=ucase Umwandlung von Kleinbuchstaben in Großbuchstaben

conv=swab Vertauschen zweier aufeinanderfolgender Byte

conv=noerror Bearbeitung trotz Fehler fortsetzen

conv=sync Auffüllen der Eingabe-Blöcke auf die in *ibs=n* festgelegte Anzahl Zeichen

conv=notrunc Ende der Ausgabedatei bleibt erhalten

Wenn eine Option Größenangaben verlangt, können Sie diese auf folgende Weise machen:

n	bedeutet *n* mal 1
*n***b**	bedeutet *n* mal 512
*n***k**	bedeutet *n* mal 1024
*n***w**	bedeutet *n* mal 2
*n***x***m*	bedeutet *n* mal *m*

if=eingabedatei
(if - input file) Für *eingabedatei* geben Sie den Namen der Eingabe-Datei an, aus der *dd* lesen soll.

if=eingabedatei nicht angegeben:
dd liest von der Standard-Eingabe.

of=ausgabedatei
(of - output file) Für *ausgabedatei* geben Sie den Namen der Ausgabe-Datei an, in die *dd* schreiben soll.

of=ausgabedatei nicht angegeben:
dd schreibt auf die Standard-Ausgabe.

ibs=n
(ibs - input block size) Für *n* geben Sie die Größe der Eingabe-Blöcke in Byte an.

ibs=n nicht angegeben:
Die Größe der Eingabe-Blöcke ist 512 Byte.

obs=n
(obs - oputput block size) Für *n* geben Sie die Größe der Ausgabe-Blöcke in Byte an.

obs=n nicht angegeben:
Die Größe der Ausgabe-Blöcke ist 512 Byte.

bs=n
(bs - block size) Für *n* geben Sie die Größe der Eingabe- und Ausgabe-Blöcke in Byte an. Die Angabe von *bs* hebt die Angaben für *ibs* und *obs* auf.

cbs=n
(cbs - conversion buffer size) Für *n* geben Sie die Größe des Konvertierungspuffers in Byte an. Diese Angabe ist nur in folgenden Fällen wirksam:
– bei der Konvertierung von EBCDIC in ASCII und umgekehrt
– bei der Konvertierung von und zu fester Satzlänge.

(siehe *conv=ascii*, *conv=ebcdic conv=ibm* und *conv=block*, *conv=unblock*).

skip=n
Die ersten *n* Blöcke der Eingabe-Datei werden übersprungen und die Bearbeitung beginnt erst beim Block *n*+1.
Diese Option ist nur für Magnetbandgeräte geeignet, für die *iseek* nicht definiert ist.

iseek=n
Die ersten *n* Blöcke der Eingabe-Datei werden übersprungen und die Bearbeitung beginnt erst beim Block *n*+1. Diese Option ist speziell für die Bearbeitung von Plattendateien gedacht, da *skip* hier zu langsam ist. Diese Option ist nur für Geräte zulässig, die den *lseek*-Systemaufruf unterstützen (z.B. Plattengeräte).
skip und *iseek* schließen sich gegenseitig aus.

oseek=n
Die ggf. konvertierte Kopie der Eingabe-Datei beginnt erst beim *n*+1-ten Block der Ausgabe-Datei. Die ersten *n* Blöcke der Ausgabe-Datei bleiben unverändert erhalten. Diese Option ist nur für Geräte zulässig, die den *lseek*-Systemaufruf unterstützen (z.B. Plattengeräte).

seek=n
seek hat die gleiche Funktion wie *oseek*.

count=n
Es werden nur die ersten *n* Blöcke der Eingabe-Datei kopiert und, falls angegeben, konvertiert.

conv=wert[,wert...]
Sie können für *conv* mehrere Werte, durch Kommas voneinander getrennt, angeben und so mehrere Konvertierungen herbeiführen.

Die Werte *ascii*, *ebcdic* und *ibm* können nicht gleichzeitig angegeben werden. Dasselbe gilt für die Wertepaare *block* und *unblock*, sowie *lcase* und *ucase*.

conv=ascii
Konvertierung von EBCDIC nach ASCII.

Die durch *cbs=n* festgelegte Anzahl von Zeichen wird jeweils in den Konvertierungspuffer kopiert und eventuell angegebene Zeichenumwandlungen werden ausgeführt. Leerzeichen am Zeilenende werden gelöscht und ein Neue-Zeile-Zeichen wird eingefügt. Damit wird die Größe der Ausgabeblöcke der Datei an den durch *cbs=n* festgelegten Wert angepasst. Haben Sie die *cbs*-Option nicht oder mit *cbs=0* angegeben, wird die Eingabe-Datei nur konvertiert und am Zeilenende werden keine Leerzeichen gelöscht.

conv=ebcdic
Konvertierung von ASCII nach EBCDIC.

Die ASCII-Zeichen werden in den Konvertierungspuffer gelesen und nach EBCDIC konvertiert. Die Ausgabeblockgröße wird durch Hinzufügen von Leerzeichen an den durch *cbs=n* festgelegten Wert angepasst. Haben Sie die Option *cbs* nicht oder mit *cbs=0* angegeben, wird die Eingabe-Datei nur konvertiert, ihre Blockstruktur wird jedoch nicht verändert.

conv=siemens
Konvertierung von ASCII nach EBCDIC wie oben beschrieben, wobei eine Siemens-spezifische EBCDIC-Tabelle verwendet wird. Das Newline-Zeichen wird wiefolgt konvertiert: X'0A' <-> X'05' .

conv=ibm
Konvertierung von ASCII nach EBCDIC wie oben beschrieben, wobei eine IBM-spezifische EBCDIC-Tabelle verwendet wird.

conv=block
Durch Neue-Zeile-Zeichen abgeschlossene ASCII-Sätze werden in Sätze mit fester Satzlänge konvertiert. Die ASCII-Zeichen werden in den Konvertierungspuffer gelesen. Die Satzlänge oder Ausgabeblockgröße wird durch Hinzufügen von Leerzeichen an den durch *cbs=n* festgelegten Wert angepasst. Haben sie die Option *cbs* nicht oder mit *cbs=0* angegeben, wird die Eingabe-Datei nur konvertiert.

conv=unblock
ASCII-Sätze mit fester Satzlänge werden in durch Neue-Zeile-Zeichen abgeschlossene Sätze konvertiert. Die Satzlänge wird durch den mit *cbs=n* festgelegten Wert bestimmt. Diese Anzahl Zeichen wird jedesmal in den Konvertierungspuffer geladen, am Zeilenende überschüssige Leerzeichen werden gelöscht und ein Neue-Zeile-Zeichen wird eingefügt, bevor die Zeichen ausgegeben werden. Haben Sie die Option *cbs* nicht oder mit *cbs=0* angegeben, wird die Eingabe-Datei nur konvertiert.

conv=lcase
Großbuchstaben werden in die entsprechenden Kleinbuchstaben verwandelt.

conv=ucase
Kleinbuchstaben werden in die entsprechenden Großbuchstaben verwandelt.

conv=swab
Jeweils zwei aufeinanderfolgende Byte werden vertauscht. Enthält ein Eingabeblock eine ungerade Anzahl Bytes, wird das letzte Byte ignoriert.

conv=noerror
Die Bearbeitung wird beim Auftreten eines Fehlers nicht beendet.

conv=sync
Alle Eingabe-Blöcke werden auf die durch *ibs=n* festgelegte Anzahl von Zeichen aufgefüllt. Bei Angabe von *block* oder *unblock* werden Leerzeichen verwendet, sonst Null-Bytes.

conv=notrunc
Die Ausgabedatei wird nicht neu erzeugt. Alle Blöcke, die nicht explizit von *dd* überschrieben werden, bleiben unverändert erhalten.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *dd*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien) sowie die Einteilung der Zeichen in Groß- und Kleinbuchstaben und deren Übereinstimmung.

LC_MESSAGES
: Legt das Format und den Inhalt von Meldungen fest, die auf die Standard-(Fehler)Ausgabe geschrieben werden.

NLSPATH
: Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Der Inhalt eines EBCDIC-Magnetbandes mit einer Blockstruktur von 10 mal 80 Byte wird in die ASCII-Datei *xy* geschrieben:

```
$ dd if=/dev/rmt32 of=xy ibs=800 cbs=80 conv=ascii
```

Siehe auch *cp*

debug Testen von POSIX-Programmen (program debugging in forked tasks)

Das Kommando *debug* ermöglicht das Testen von POSIX-Programmen, die aus einer Shell heraus gestartet werden.

Syntax

Format 1: debug[␣**-e**]␣program[␣arguments]...

Format 2: debug[␣**-p**]␣pid

Format 1 **debug**[␣**-e**]␣program[␣arguments]...

Das zu testende Programm wird in einer von der Shell durch *fork* erzeugten Task (Fork-Task) geladen. Nach Abschluss des Ladevorgangs vor der ersten Befehlsausführung erhält der Anwender die Kontrolle auf BS2000-Kommando-Ebene. Als Bereitzeichen wird die Prozess-Id der Fork-Task ausgegeben (siehe auch *Beispiel 1*).

-e Das Programm *program* wird ohne Symboltabelle geladen.

program
Name des Programms, das getestet werden soll. Das Programm muss ausführbar sein.

arguments
Argumente von *program*.

Format 2 **debug**[␣**-p**]␣pid

Das in der Fork-Task mit der Prozess-Id *pid* laufende Programm wird unterbrochen. Der Anwender erhält die Kontrolle auf BS2000-Kommando-Ebene. Als Bereitzeichen wird die Prozess-Id der unterbrochenen Fork-Task ausgegeben (siehe auch *Beispiel 2*).

-p Unterbrechen des Programms in der Task mit der Prozess-Id *pid*.

pid Prozess-Id der Task, die unterbrochen werden soll.

Beispiel 1 Das folgende Beispiel zeigt den Ablauf bis zum Laden des Programms, welches getestet werden soll. Nach dem Prompt *pid/* ist das Programm wie nach einem /LOAD-PROGRAM-Kommando bereit zum Start. Der weitere Ablauf kann durch AID-Kommandos beeinflusst werden. Nach /RESUME-PROGRAM startet das Programm in der Fork-Task.

```
/START-POSIX-SHELL
$ cd myprogdir
$ ls -l myprog
-rwxr-xr-x  1  MYUSER    MYGROUP 1150976 Thu May 28 11:56:03 MSZ 2009 myprog
$ debug myprog
% AID0348 Program stopped due to EXEC event (PID=0000000055)
0000000055/
```

Beispiel 2 Das folgende Beispiel zeigt den Ablauf bei Unterbrechung eines bereits laufenden Programms in einer Fork-Task. Nach dem Prompt *pid/* ist das Programm wie nach K2 unterbrochen. Der weitere Ablauf kann durch AID-Kommandos beeinflusst werden. Nach RESUME-PROGRAM läuft das Programm in der unterbrochenen Fork-Task weiter.

```
/START-POSIX-SHELL
$ ps -e
    UID  PID  PPID  C     STIME TTY        TIME  CMD
SYSROOT  243     1  0  16:32:38 term/001   0:01  [sh]
SYSROOT  245   243  0  16:33:44 term/001   0:00  [loop]

$ debug -p 245
% AID0492 %STOP was send for fork task (PID=0000000245)
$
% AID0348 Program stopped due to STOP event (PID=0000000245)
0000000245/
```

df Anzahl der freien und belegten Plattenblöcke und I-Nodes ausgeben (report free disk space)

Mit *df* können Sie ermitteln, wieviel freie Plattenblöcke und I-Nodes noch auf montierten bzw. nicht montierten lokalen Dateisystemen vorhanden sind, und sich über die Dateisystem-Codierung (ASCII oder EBCDIC) informieren.

Syntax

Format 1: df[␣**-F**␣FSType]␣ **-P**[␣**-k**][␣datei]

Format 2: df[␣**-F**␣FSType][␣**-begkIntVv**][␣**-o**␣ufs_options]...[␣datei]...

Format 3: df␣**-c**[␣datei]...

Format 1 **df**[␣**-F**␣FSType]␣ **-P**[␣**-kIV**][␣datei]...

option

-F␣FSType
Das Dateisystem, mit dem gearbeitet werden soll, ist vom Typ *FSType*. Die Angabe dieser Option ist nur notwendig, wenn das Dateisystem nicht montiert ist.

FSType kann sein:

ufs	Berkeley-Dateisystem
bs2fs	bs2fs-Dateisystem
proc	Pseudo-Dateisystem zum Zugriff auf Prozessdaten
fdfs	Pseudo-Dateisystem zum Zugriff auf Dateikennzahlen

-P Mit dieser Option erhalten Sie spaltenweise folgende Informationen über alle montierten oder die angegebenen Dateisysteme:

filesystem	Gerätedatei des Dateisystems
bytes	Gesamt-Speicherplatz in 512-Byte-Blöcken
used	belegter Speicherplatz in 512-Byte-Blöcken
availabe	verfügbarer Speicherplatz in 512-Byte-Blöcken
capacity	belegter Speicherplatz in Prozent
mounted on	Name des Dateiverzeichnisses, an das das Dateisystem montiert ist

-k Mit dieser Option erhalten Sie spaltenweise Informationen über alle montierten oder die angegebenen Dateisysteme (analog zur Option *-P)*. Der Speicherplatz wird in 1024-Byte-Blöcken (anstelle 512-Byte-Blöcken) ausgegeben.

Bei der Angabe der Speicherplatzes wird der reservierte Speicherplatz berücksichtigt, der nur vom POSIX-Verwalter belegt werden kann.

-l *df* gibt für alle lokal montierten oder die angegebenen Dateisysteme die Anzahl der freien Blöcke und I-Nodes aus. *-l* ist die standardmäßig voreingestellte Option.
Diese Option kann nicht mit *-e* oder *-o* benutzt werden.

-V *df* ergänzt eine unvollständig eingegebene *df*-Kommandozeile und gibt das ergänzte Kommando auf die Standard-Ausgabe aus. Das Kommando *df* wird jedoch nicht ausgeführt. *df* orientiert sich für die Ergänzung an den entsprechenden Einträgen in den Dateien */etc/mnttab* und */etc/vfstab*.

Ergänzt wird die Option *-F␣FSType* und/oder die zugehörige Gerätedatei, je nachdem, ob Sie eine der beiden Komponenten bereits beim Kommandoaufruf angegeben haben.
Wenn Sie *df␣-V* ohne andere Optionen und Argumente eingeben, dann erhalten Sie eine Liste von *df*-Kommandozeilen, die Ihnen alle Typen der an Ihrem Rechner montierten Dateisysteme und die entsprechenden Gerätedateien anzeigt.

datei

datei ist entweder der Name eines Dateiverzeichnisses oder der Name einer Gerätedatei, die ein Dateisystem enthält. Geben Sie ein Dateiverzeichnis an, so bezieht sich *df* auf das Dateisystem, welches das Dateiverzeichnis enthält. Sie können mehrere Namen angeben.

datei nicht angegeben:
df gibt die Informationen für alle montierten Dateisysteme bzw. alle Dateisysteme vom Typ *FSType* aus.

Format 2 **df**[␣**-F**␣FSType][␣**-begklntVv**][␣**-o**␣ufs_options]...[␣datei]...

Keine Option angegeben:

df gibt die Anzahl der freien Blöcke und I-Nodes aller montierten bzw. angegebenen Dateisysteme aus.

option

-F␣FSType

Das Dateisystem, mit dem gearbeitet werden soll, ist vom Typ *FSType*. Die Angabe dieser Option ist nur notwendig, wenn das Dateisystem nicht montiert ist.

FSType kann sein:

ufs	Berkeley-Dateisystem
bs2fs	bs2fs-Dateisystem
proc	Pseudo-Dateisystem zum Zugriff auf Prozessdaten
fdfs	Pseudo-Dateisystem zum Zugriff auf Dateikennzahlen
nfs	an fernem Rechner montiertes Dateisystem (network file system, Berkeley)

-b Der freie Speicherplatz (in KByte) wird ausgegeben. Diese Option kann nicht mit *-o* benutzt werden.

-e Die Anzahl der freien I-Nodes wird ausgegeben. Diese Option kann nicht mit *-o* benutzt werden und setzt die Option *-l* außer Kraft.

-g *df* gibt für alle montierten Dateisysteme oder für *FSType* bzw. *datei* die gesamte *statvfs*-Struktur aus. Sie erhalten eine vierzeilige Ausgabe mit folgenden Informationen:

dvz	Name des Montier-Dateiverzeichnisses (die Zeichenkette *dvz* erscheint nicht in der Ausgabe)
gerät	entsprechende Gerätedatei (die Zeichenkette *gerät* erscheint nicht in der Ausgabe)
block size	Blockgröße des Dateisystems,
frag size	Fragmentgröße des Dateisystems
total blocks	gesamte Anzahl der Blöcke in Einheiten von *frag size*
free blocks	gesamte Anzahl der freien Blöcke
available	Anzahl der freien Blöcke für Nicht-POSIX-Verwalter
total files	gesamte Anzahl der I-Nodes
free files	Anzahl der freien I-Nodes
filesys id	Nummer des Dateisystems
name	Dateisystem-spezifische Bezeichnung (die Zeichenkette *name* erscheint nicht in der Ausgabe)
fstype	Typ des Dateisystems
flag	Flags des Dateisystems
filename length	maximale Länge von Dateinamen

Diese Option kann nicht mit *-o* benutzt werden und setzt die Optionen *-b, -e, -k, -l, -n, -r* und *-v* außer Kraft.

-k Mit dieser Option erhalten Sie spaltenweise folgende Informationen über alle montierten oder die angegebenen Dateisysteme:

filesystem	Gerätedatei des Dateisystems
bytes	Gesamt-Speicherplatz in 1024-Byte-Blöcke
used	belegter Speicherplatz in 1024-Byte-Blöcke
available	verfügbarer Speicherplatz in 1024-Byte-Blöcke
capacity	belegter Speicherplatz in Prozent
mounted on	Name des Dateiverzeichnisses, an das das Dateisystem montiert ist

Bei der Angabe der Speicherplatzes wird der reservierte Speicherplatz berücksichtigt, der nur vom POSIX-Verwalter belegt werden kann.

Diese Option kann nicht mit *-o* benutzt werden und setzt die Optionen *-b*, *-e*, *-k*, *-l*, *-n*, *-r* und *-v* außer Kraft.

-l *df* gibt für alle lokal montierten oder die angegebenen Dateisysteme die Anzahl der freien Blöcke und I-Nodes aus. *-l* ist die standardmäßig voreingestellte Option.
Diese Option kann nicht mit *-e* oder *-o* benutzt werden.

-n *df* gibt den Dateisystem-Typ aus.
Kombinieren Sie *-n* mit *-o* oder *-f*, so erhalten Sie eine Fehlermeldung.

-t *df* gibt pro montiertem oder angegebenem Dateisystem die Anzahl freier Blöcke und Dateien sowie die Gesamtanzahl der zur Verfügung stehenden Blöcke und Dateien aus.
-t setzt die Option *-b* außer Kraft. Die Kombination von *-t* mit *-o i* ergibt eine Total-Zeile aller *-o i* Ausgaben.

-V *df* ergänzt eine unvollständig eingegebene *df*-Kommandozeile und gibt das ergänzte Kommando auf die Standard-Ausgabe aus. Das Kommando *df* wird jedoch nicht ausgeführt. *df* orientiert sich für die Ergänzung an den entsprechenden Einträgen in den Dateien */etc/mnttab* und */etc/vfstab*.

Ergänzt wird die Option *-F␣FSType* und/oder die zugehörige Gerätedatei, je nachdem, ob Sie eine der beiden Komponenten bereits beim Kommandoaufruf angegeben haben.
Wenn Sie *df␣-V* ohne andere Optionen und Argumente eingeben, dann erhalten Sie eine Liste von *df*-Kommandozeilen, die Ihnen alle Typen der an Ihrem Rechner montierten Dateisysteme und die entsprechenden Gerätedateien anzeigt.

-v Mit dieser Option erhalten Sie spaltenweise folgende Informationen über alle montierten oder die angegebenen Dateisysteme:

Mount Dir	Name des Dateiverzeichnisses, an das das Dateisystem montiert ist
Filesystem	Gerätedatei des Dateisystems
blocks	Gesamt-Speicherplatz in 512-Byte-Blöcken
used	belegter Speicherplatz in 512-Byte-Blöcken
free	noch nicht belegter Speicherplatz in 512-Byte-Blöcken
%used	belegter Speicherplatz in Prozent

Diese Option kann nicht mit *-o* benutzt werden und setzt die Optionen *-b*, *-e*, *-k*, *-l*, *-n*, *-r* und *-v* außer Kraft.

-o␣ufs_options
Ist das Dateisystem vom Typ *ufs*, dann können Sie eine *ufs*-spezifische Option angeben:

i Sie erhalten spaltenweise folgende Informationen:

Filesystem	Name der Gerätedatei
iused	Anzahl der belegten I-Nodes
ifree	Anzahl der freien I-Nodes
%iused	belegte I-Nodes in Prozent
Mounted on	Name des Dateiverzeichnisses, an das das Dateisystem montiert ist

-o␣*i* ist sinnvoll nur mit den Optionen *-F* und *-t* kombinierbar.

datei
datei ist entweder der Name eines Dateiverzeichnisses oder der Name einer Gerätedatei, die ein Dateisystem enthält. Geben Sie ein Dateiverzeichnis an, so bezieht sich *df* auf das Dateisystem, welches das Dateiverzeichnis enthält. Sie können mehrere Namen angeben.

datei nicht angegeben:
df gibt die Informationen für alle montierten Dateisysteme bzw. alle Dateisysteme vom Typ *FSType* aus.

Format 3 **df␣-c**[␣datei]...

option

-c Das Kommando gibt für alle lokal montierten oder für die mit *datei* angegebenen Dateisysteme die Art der Codierung (ASCII oder EBCDIC) aus. Diese Option kann nicht mit anderen Optionen kombiniert werden.

Hinweis In UFS-Dateisystemen ist der verfügbare Speicherplatz in der Regel kleiner als der freie Speicherplatz. Das liegt daran, dass ein Teil (10 Prozent) des Speicherplatzes für privilegierte Anwendungen reserviert ist und somit für normale Anwendungen nicht zur Verfügung steht.

Fehler Bei einigen der folgenden Fehlermeldungen wird zusätzlich die korrekte Kommando-Syntax ausgegeben. Diese Syntax weicht von einigen Elementen der obigen Beschreibung wie folgt ab:

Syntaxelement der Fehlermeldung	entspricht in der obigen Beschreibung
specific_options	ufs_option
directory | special	datei
generic options	-begkltvV

`df: Cannot access` *datei*
Bei *df␣-V* wurde der Name einer Gerätedatei oder eines Dateiverzeichnisses unvollständig oder falsch angegeben.

`df ufs: Usage: df [-F ufs] [generic options] [-o i] [directory|special]`
Zur Option *-o* wurde für *ufs_option* ein ungültiges Argument angegeben. *ufs_option* kann nur *i* sein.

`df: operation not applicable for FSType DStyp`
Bei *df␣-F␣FSType␣-o␣i* wurde für *FSType* ein ungültiger Dateisystem-Typ (*proc* oder *fdfs*) angegeben. Im Zusammenhang mit der Option *-o* kann nur der Typ *ufs* angegeben werden.

Datei */dev/dsk/**
Gerätedateien der Dateisysteme

/etc/mnttab
Tabelle der montierten Dateisysteme

/etc/vfstab
Liste der Standardparameter für jedes Dateisystem

*/etc/fs/DStyp/**
Spezifische Kommandos für die Dateisystem-Typen

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *df*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Für alle Benutzerverzeichnisse sollen die zugehörigen Dateisystem-Typen ausgegeben werden:

```
$ df -n /h*
/home    : ufs
/home2   : ufs
/home3   : ufs
```

Beispiel 2 Für das Dateisystem */stand* soll die gesamte *statvfs*-Struktur ausgegeben werden:

```
$ df -g /stand
/stand           (/dev/dsk/c0d0s10):     512 block size     512 frag size
  10710 total blocks  7577 free blocks   7577 available      104 total file
    100 free files      10 filsys id                           /stand
    bfs fstype  0x00000000 flag            14 filename length
```

Beispiel 3 Für die Dateiverzeichnisse */home* und */var* soll die Dateisystem-Codierung ausgegeben werden:

```
$ df -c /home /var
/home    : EBCDIC
/var     : ASCII
```

Siehe auch *du*

diff Dateien zeilenweise vergleichen (compare two files)

Das Kommando *diff* vergleicht die Inhalte von *datei1* und *datei2* und schreibt nach Standard-Ausgabe eine Liste von Änderungen (mit *ed*-ähnlichen Kommandos), die notwendig sind, um *datei1* nach *datei2* zu konvertieren. Diese Liste sollte minimal sein. Wenn die Dateien identisch sind, wird keine Ausgabe erzeugt.

Syntax **diff**[␣option]␣datei1␣datei2

Keine Option angegeben

Wenn die verglichenen Dateien gleich sind, gibt *diff* nichts aus. Wenn die verglichenen Dateien Unterschiede aufweisen, gibt *diff* aus:

```
1. Zeile[nbereich] aus    ed-Kommando    Zeile[nbereich] aus
   datei1                                datei2
2. Zeilen, die nur in datei1 stehen
3. Zeilen, die nur in datei2 stehen
```

Die Ausgabe hat folgendes Format:

```
                a
1. n1[,n2]  d n1[,n2]
            c

2. < text einer Zeile aus datei1
        :
        - - -
3. > text einer Zeile aus datei2
          :
```

a, d und *c* sind *ed*-ähnliche Kommandos. Sie bedeuten:

a (append) anfügen

d (delete) löschen

c (change) ändern

Die Ausgabe lesen Sie folgendermaßen:

Die *ed*-Kommandos *a, d* und *c* mit den davorstehenden Zeilen(bereichs)angaben zeigen, wie *datei1* in *datei2* umzuwandeln ist.

Wenn Sie *a* durch *d* und *d* durch *a* ersetzen und die rechts stehenden Zeilen(bereichs)-angaben verwenden, sehen Sie, wie *datei2* in *datei1* umzuwandeln ist. Zeilen aus *datei1* sind mit < gekennzeichnet, aus *datei2* mit >.

option

-a diff gibt alle Zeilen von *datei1* und *datei2* aus. Zeilen, die nur in *datei1* vorkommen, werden mit einem Strich - gekennzeichnet. Zeilen, die nur in *datei2* vorkommen, werden mit einem Pluszeichen + gekennzeichnet. Zeilen, die in beiden Dateien identisch sind, werden mit einem Leerzeichen ␣ gekennzeichnet.

-b *diff* berücksichtigt weder Leerzeichen und Tabulatorzeichen am Zeilenende, noch unterschiedlich lange Folgen von Leerzeichen innerhalb von Zeilen an derselben Stelle. Leerzeichen am Zeilenanfang werden als Differenz ausgegeben, ebenso Leerzeilen.

-i nicht mit Option *-h* zu verwenden.
diff berücksichtigt Groß- und Kleinschreibung nicht, z.B. wird 'A' nicht von 'a' unterschieden.

-t *diff* expandiert Tabulatorzeichen in der Ausgabe. Bei Option *-c* oder normaler Ausgabe fügt *diff* gelegentlich am Zeilenanfang Zeichen ein, die die Einrückung der ursprünglichen Zeilen verfälschen und dadurch die Interpretation der Ausgabe erschweren. Diese Option erhält die ursprüngliche Einrückung aufrecht.

-w *diff* berücksichtigt keine Leer- und Tabulatorzeichen und unterscheidet nicht zwischen verschieden langen Folgen von Leer- und Tabulatorzeichen. Z.B. wird die Zeichenkette 'if␣(␣a␣=␣=␣b␣)' als identisch mit '(a==b)' angesehen.

Die folgenden Optionen schließen sich gegenseitig aus:

-c *diff* erzeugt eine dreiteilige Liste, wobei das Ausgabeformat leicht abgeändert ist. Zuerst werden Namen und Entstehungsdatum von *datei1* und *datei2* angezeigt. Dann werden die Kontrastzeilen ausgegeben, wobei die Zeilen, die nicht in *datei2* vorkommen, mit einem Minuszeichen - versehen sind, die Zeilen, die nicht in *datei1* vorkommen, mit einem Pluszeichen + versehen sind und die Zeilen, die sich in *datei1* und *datei2* unterscheiden, mit einem Ausrufezeichen ! versehen sind. Zusätzlich werden die drei Zeilen vor und die drei Zeilen hinter den jeweiligen Kontrastzeilen ausgegeben.

-C␣zahl
diff gibt im gleichen Format aus wie bei der Option *-c*, zeigt jedoch außer den sich unterscheidenden Stellen auch jeweils *zahl* Zeilen davor und *zahl* Zeilen dahinter an.

-e nicht mit Option *-h*, *-l* oder *-s* zu verwenden!
diff erzeugt ein *ed*-Skript und gibt es aus. Das *ed*-Skript enthält die Kommandos *a,d* und *c*, sowie die zugehörigen Textzeilen, mit denen der Editor *ed datei1* in *datei2* umwandeln kann. Hierfür muss das *ed*-Skript an den Editor *ed* als Eingabe übergeben werden. Vorher müssen allerdings die Anweisungen *w* und *q* an das Ende des *ed*-Skripts geschrieben werden. Außerdem müssen Sie an den Anfang das Kommando *e datei1* setzen (siehe *ed*).

-f nicht mit Option *-h* zu verwenden!
diff erzeugt ein ähnliches Skript wie bei Option *-e*, nur in der umgekehrten Richtung. Dieses Skript eignet sich aber **nicht** als Eingabe für *ed*.

Die unter *-e* und *-f* erstellten ed-Skripts sind möglicherweise nicht korrekt, wenn die verglichenen Zeilen nur aus einem einzelnen Punkt . bestehen.

Die folgenden Optionen beeinflussen die Arbeitsweise von *diff*:

-h nicht mit Option *-e*, *-f*, *-i*, *-c*, *-C*, *-n* und *-D* zu verwenden!
diff arbeitet schneller, und die Dateien können beliebig lang sein. Allerdings ist das Ergebnis bei Option *-h* nicht zuverlässig!

-n *diff* erzeugt, ähnlich wie bei der Option *-e*, ein Skript. Dort stehen die *ed*-Kommandos jedoch in umgekehrter Reihenfolge. Außerdem steht hinter jedem *Insert*- oder *Delete*-Kommando die Anzahl der zu ändernden Zeilen.

-D␣zeichenkette
datei1 und *datei2* sollten in diesem Fall C-Quellprogramme oder C-Quellprogrammteile enthalten. *diff* erzeugt ein C-Quellprogramm aus *datei1* und *datei2*, in das es Angaben für den C-Präprozessor einfügt. Wird dieses Programm übersetzt, dann entsteht eine übersetzte *datei1*, wenn *zeichenkette* nicht definiert ist. Es ergibt sich eine übersetzte *datei2*, wenn *zeichenkette* definiert ist.

Die folgenden Optionen werden zum Vergleich von Dateiverzeichnissen verwendet:

-l *diff* gibt im langen Format aus. Vor der Ausführung von *diff* wird jede Textdatei in Seiten zerteilt, indem sie durch eine Pipeline zu *pr* gesendet wird. Weitere Unterschiede werden gesammelt und am Stück ausgegeben, nachdem alle textspezifischen Unterschiede angezeigt sind.

-r *diff* arbeitet rekursiv alle gemeinsamen Unterverzeichnisse ab.

-s *diff* zeigt an, welche Dateien übereinstimmen. Standardmäßig unterbleibt dies.

-S␣name
diff bearbeitet ein Verzeichnis erst ab der Datei *name*.

datei1␣datei2
Namen der Dateien, die *diff* vergleichen soll. Ist *datei1* ein Dateiverzeichnis, dann wird aus diesem Verzeichnis die Datei *datei2* für den Vergleich mit *datei2* herangezogen. Ist *datei2* ein Dateiverzeichnis, dann wird mit *datei1* aus diesem Verzeichnis verglichen.

Wenn sowohl *datei1* als auch *datei2* Verzeichnisse sind, sucht *diff* in beiden Verzeichnissen nach Dateien gleichen Namens und vergleicht diese miteinander. Folgende Dateien werden bei diesem Vergleich jedoch nicht berücksichtigt: blockorientierte Gerätedateien, zeilenorientierte Gerätedateien und FIFO-Dateien. Außerdem vergleicht *diff* in diesem Fall keine regulären Dateien mit Verzeichnissen.

Wenn Sie für *datei1* oder *datei2* einen Bindestrich - angeben, liest *diff* die entsprechende Datei von der Standard-Eingabe. Die Eingabedateien müssen in diesem Fall Textdateien sein.

Endestatus

0	Dateien sind identisch
1	Dateien sind nicht identisch
>1	Eingabefehler

Fehler

`diff: two filename arguments required`
Sie haben eine falsche Anzahl Dateien angeben. Es können nur zwei Dateien verglichen werden.

`diff: No such file or directory`
Eine der angegebenen Dateien existiert nicht.

`diff: files too big, try -h`
Sie müssen die Option *-h* angeben, da die zu vergleichenden Dateien zu groß sind.

`diff: Permission denied`
Sie haben kein Leserecht für eine der angegebenen Dateien.

Variable

TZ
bestimmt die Zeitzone, wenn der Dateizeitstempel berechnet wird, der bei den Optionen *-C* und *-c* geschrieben wird.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *diff*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_TIME Legt das Format der Dateizeitstempels bei den Optionen *-C* und *-c* fest.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die Dateien *datei1* und *datei2* haben folgenden Inhalt:

```
datei1                datei2

Amsel Drossel         Amsel und Drossel
Fink und Star         Fink Star
und die ganze         Vogelschar
Vogelschar
```

Mit dem Aufruf von *diff* können Sie genau feststellen, in welchen Zeilen sich die beiden Dateien unterscheiden:

```
$ diff datei1 datei2
1,3c1,2
<Amsel Drossel
<Fink und Star
<und die ganze
- - -
>Amsel und Drossel
>Fink Star
```

Das bedeutet: Um aus *datei1 datei2* zu erzeugen, müssen die Zeilen 1 bis 3 aus *datei1* (1,3) ersetzt werden (c) durch die Zeilen 1 bis 2 (1,2) aus *datei2*. Was in diesen Zeilen jeweils steht, ist an den mit < oder > beginnenden Ausgabezeilen zu sehen: mit < beginnen die Zeilen, die nur in *datei1*, mit > die Zeilen, die nur in *datei2* stehen.

Beispiel 2 Dateien vergleichen und ein *ed*-Skript erstellen:

```
Inhalt von datei1:     Inhalt von datei2:

heute ist Montag       heute ist Dienstag
es ist kalt            es ist Herbst
                          es ist kalt
```

Nach folgendem Aufruf gibt *diff* die *ed*-Kommandos aus, mit denen *ed datei1* in *datei2* umwandeln kann. Um das Resultat dieses Aufruf als Eingabe für den *ed* benutzen zu können, müssen Sie noch die Anweisungen *w* und *q* anfügen (siehe *ed*).

```
$ diff -e datei1 datei2
1c
heute ist Dienstag
es ist Herbst
.
$
```

Siehe auch *cmp*, *comm*, *ed*, *pr*

dirname Pfad-Präfix vom Dateinamen trennen (return directory portion of pathname)

Mit *dirname* können Sie das Pfad-Präfix vom einfachen Dateinamen trennen. *dirname* entfernt aus einer Zeichenkette alle Zeichen ab dem letzten Schrägstrich / einschließlich und gibt den Rest auf die Standard-Ausgabe aus. *dirname* ist nützlich in Shell-Prozeduren.

Syntax **dirname**[␣zeichenkette]

zeichenkette

zeichenkette ist eine beliebige Zeichenkette.
dirname löscht alle Zeichen ab dem letzten / einschließlich dieses Schrägstrichs und schreibt den Rest auf die Standard-Ausgabe. Enthält *zeichenkette* keinen Schrägstrich, so wird nur ein Punkt auf die Standard-Ausgabe ausgegeben.

zeichenkette nicht angegeben:
es wird nur ein Punkt auf die Standard-Ausgabe ausgegeben.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *dirname*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Im folgenden Beispiel wird der Variablen *NAME* als Wert das Pfad-Präfix */usr/src/cmd* zugewiesen:

```
NAME=`dirname /usr/src/cmd/xyz.c`
```

Siehe auch *basename*, *ed*

du Belegten Speicherplatz ausgeben (estimate file space usage)

du gibt die Speicherplatz-Belegung durch Dateiverzeichnisse, Unter-Dateiverzeichnisse und einfache Dateien mit der Anzahl der belegten Blöcke von 512 Bytes aus.

Syntax **du**[␣**a** |␣**s**][␣**-kx**][␣**-r**][␣datei...]

Keine Option angegeben
: Wenn *datei* ein Dateiverzeichnis ist, listet *du* den Speicherplatz auf, den das Dateiverzeichnis und alle seine Unterverzeichnisse belegen. Der von den einfachen Dateien im angegebenen Dateiverzeichnis belegte Speicherplatz ist enthalten, wird aber nicht einzeln aufgelistet.

option

-a Wenn *datei* ein Dateiverzeichnis ist, listet *du* den belegten Speicherplatz für alle Dateien dieses Dateiverzeichnisses einzeln auf. Wenn *datei* kein Dateiverzeichnis ist, listet *du* den von *datei* belegten Speicherplatz auf. Die Option *-a* ist nicht mit der Option *-s* kombinierbar.

-k *du* gibt die Speicherplatz-Belegung mit der Anzahl der belegten Blöcke von 1024 Bytes aus.

-r *du* gibt eine Fehlermeldung aus, wenn *datei* ein Dateiverzeichnis ist, für das Sie kein Leserecht haben oder eine Datei, die nicht geöffnet werden kann. Die Option *-r* ist immer gesetzt.

-s *du* gibt nur die Gesamtsumme des Speicherplatzes aus, den der Teil-Dateibaum oder die Datei belegt. Die Option *-s* ist nicht mit der Option *-a* kombinierbar.

-x Bei der Berechnung von Dateigrößen werden nur die Dateien berechnet, die dieselben Gerätedateien wie *datei* benutzen.

datei
: Name der Datei bzw. des Dateiverzeichnisses, für die/das die Speicherplatz-Belegung ausgegeben werden soll. Eine Datei, auf die zwei oder mehr Verweise vorhanden sind, wird nur einmal gezählt. Eine Datei mit nicht verwendeten Blöcken (z.B. nur Block 1 und 100 geschrieben) führt zu falschen Ergebnissen.

  datei nicht angegeben:
  Der vom aktuellen Dateiverzeichnis und allen Unterverzeichnissen belegte Speicherplatz wird ausgegeben.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *du*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Auflisten der Anzahl Speicherblöcke von 512 byte, die von den Unter-Dateiverzeichnissen des aktuellen Dateiverzeichnisses belegt werden und deren Namen mit *DIR* beginnen. Der von einfachen Dateien belegte Speicherplatz ist enthalten, wird aber nicht einzeln aufgelistet.

```
$ du DIR*
6       DIR-1
136     DIR-2/DVZ-1
140     DIR-2
5       DIR-3
54      DIR-4
52      DIR-5
```

Beispiel 2 Auflisten der Anzahl Speicherblöcke von 512 byte, die von den Unter-Dateiverzeichnissen des aktuellen Dateiverzeichnisses belegt werden und deren Namen mit *DIR* beginnen. Der von einfachen Dateien belegte Speicherplatz wird mit Option *-a* einzeln aufgelistet.

```
$ du -a DIR*
1       DIR-1/datei1
1       DIR-1/datei2
1       DIR-1/datei3
6       DIR-1
0       DIR-2/datei4
1       DIR-2/datei5
34      DIR-2/DVZ-1/datei6
99      DIR-2/DVZ-1/datei7
136     DIR-2/DVZ-1
140     DIR-2
1       DIR-3/datei8
1       DIR-3/datei9
5       DIR-3
50      DIR-4/datei10
1       DIR-4/datei10.bak
54      DIR-4
50      DIR-5/datei11
52      DIR-5
```

Siehe auch *df*

dumpfs Interne Dateisystem-Information ausgeben (dump filesystem)

Das Kommando *dumpfs* gibt die Superblock- und Zylindergruppeninformation für das angegebene Dateisystem aus. Mit diesem Kommando kann z.B. festgestellt werden, welche Block- und Fragmentgrößen für das Dateisystem festgelegt sind und wie groß mindestens der freie Speicherplatz in Prozenten sein muss.

Syntax **dumpfs**␣dateisystem

dateisystem
Name des Dateisystems, über das Informationen ausgegeben werden sollen.

Dieses Kommando darf nur der POSIX-Verwalter eingeben. Es ist nur zu Diagnosezwecken durch den Systemdienst sinnvoll einsetzbar.

Hinweis Das Kommando *dumpfs* wird für bs2fs-Dateisysteme nicht unterstützt.

Beispiel Informationen über das /-Verzeichnis ausgeben

```
# dumpfs /
magic   11954   time    Thu Jun  5 10:04:06 2005
sblkno  4       cblkno  6       iblkno  8       dblkno  72
sbsize  4096    cgsize  4096    cgoffset 8      cgmask  0xfffffff8
ncg     2       size    2048    blocks  1907
bsize   8192    shift   13      mask    0xffffe000
    fsize   4096    shift   12      mask    0xfffff000
frag    2       shift   1       fsbtodb 1
minfree 10%     maxbpg  1024
maxcontig 1     rotdelay 0ms    rps     60
csaddr  72      cssize  4096    shift   9       mask    0xfffffe00
ntrak   8       nsect   16      spc     128     ncyl    32
cpg     16      bpg     512     fpg     1024    ipg     2048
nindir  2048    inopb   64      nspf    2
nbfree  400     ndir    87      nifree  2157    nffree  5
cgrotor 0       fmod    0       ronly   0

cs[].cs_(nbfree,ndir,nifree,nffree):
        (66,29,918,3) (334,58,1239,2)


cg 0:
magic   90255   tell    6000    time    Thu Jun  8 08:35:05 2005
cgx     0       ncyl    16      niblk   2048    ndblk   1024
nbfree  66      ndir    29      nifree  918     nffree  3
rotor   870     irotor  833     frotor  864
frsum   3
```

```
sum of frsum: 3
iused:  0-6, 8-832, 835-846, 848-1126, 1129-1134, 1136
free:   575, 844-845, 864, 871, 876-881, 894-901, 904-913, 916-917,
        920-1023
b:
   c0:  (0)      0 0 0 0 0 0 0 0
   c1:  (0)      0 0 0 0 0 0 0 0
   c2:  (0)      0 0 0 0 0 0 0 0
   c3:  (0)      0 0 0 0 0 0 0 0
   c4:  (0)      0 0 0 0 0 0 0 0
   c5:  (0)      0 0 0 0 0 0 0 0
   c6:  (0)      0 0 0 0 0 0 0 0
   c7:  (0)      0 0 0 0 0 0 0 0
   c8:  (0)      0 0 0 0 0 0 0 0
   c9:  (0)      0 0 0 0 0 0 0 0
   c10: (0)      0 0 0 0 0 0 0 0
   c11: (0)      0 0 0 0 0 0 0 0
   c12: (0)      0 0 0 0 0 0 0 0
   c13: (5)      1 0 0 0 2 0 2 0
   c14: (29)     8 0 7 0 8 0 6 0
   c15: (32)     8 0 8 0 8 0 8 0

cg 1:
magic   90255   tell    40e000  time    Thu Jun  8 10:04:51 2005
cgx     1       ncyl    0       niblk   2048    ndblk   1024
nbfree  334     ndir    58      nifree  1239    nffree  2
rotor   234     irotor  716     frotor  256
frsum   2
sum of frsum: 2
iused:  0-805, 807, 809, 811
free:   353, 355-1023
b:
   c0:  (0)      0 0 0 0 0 0 0 0
   c1:  (0)      0 0 0 0 0 0 0 0
   c2:  (0)      0 0 0 0 0 0 0 0
   c3:  (0)      0 0 0 0 0 0 0 0
   c4:  (0)      0 0 0 0 0 0 0 0
   c5:  (14)     3 0 3 0 4 0 4 0
   c6:  (32)     8 0 8 0 8 0 8 0
   c7:  (32)     8 0 8 0 8 0 8 0
   c8:  (32)     8 0 8 0 8 0 8 0
   c9:  (32)     8 0 8 0 8 0 8 0
   c10: (32)     8 0 8 0 8 0 8 0
   c11: (32)     8 0 8 0 8 0 8 0
   c12: (32)     8 0 8 0 8 0 8 0
   c13: (32)     8 0 8 0 8 0 8 0
   c14: (32)     8 0 8 0 8 0 8 0
   c15: (32)     8 0 8 0 8 0 8 0
```

echo Aufruf-Argumente ausgeben (write arguments to standard output)

Das in die POSIX-Shell *sh* eingebaute Kommando *echo* schreibt seine Aufruf-Argumente auf die Standard-Ausgabe und beendet sich. Vor der Ausgabe passiert Folgendes:

1. Wie bei jedem anderen Kommando, interpretiert die Shell die Kommando-Zeile und übergibt an *echo* die aufbereiteten Aufruf-Argumente. Argument-Trenner sind für die Shell Leer- und Tabulatorzeichen.
2. *echo* interpretiert noch verbliebene Sonderzeichen, die die Ausgabe steuern (siehe Beschreibung zu *argument*).
3. *echo* gibt die bearbeiteten Argumente wie folgt aus:
   – Die einzelnen Argumente werden durch ein Leerzeichen voneinander getrennt, auch wenn Sie beim Aufruf mehrere Argument-Trennzeichen zwischen den Argumenten eingegeben haben.
   – Nach dem letzten Argument wird ein Neue-Zeile-Zeichen ausgegeben.

Sie können mit *echo*
– den Inhalt von Shell-Variablen abfragen,
– Meldungen in Shell-Prozeduren erzeugen und so die Funktionsweise testen,
– testen, wie die Shell einen Kommando-Aufruf interpretiert, ohne dass das Kommando ausgeführt wird,
– Daten in eine Pipe schicken und testen, wie die Pipe diese Eingabe verarbeitet.

Neben dem eingebauten Kommando *echo* gibt es auch */usr/bin/echo*. Für die Ausführung von */usr/bin/echo* erzeugt die Shell einen neuen Prozess. Sonst verhält sich */usr/bin/echo* wie *echo*.

Syntax **echo**[␣argument...]

argument
beliebige Zeichenkette, die jeweils durch Leer- oder Tabulatorzeichen begrenzt wird. Das letzte Argument wird durch ein Kommando-Trennzeichen abgeschlossen.

Sie können beliebig viele Argumente angeben, jeweils getrennt durch mindestens ein Tabulator- bzw. ein Leerzeichen.

Wie bei jedem anderen Kommando auch wird die Zeichenkette zunächst von der Shell interpretiert:

– Enthält die Zeichenkette die Zeichen Stern * bzw. Fragezeichen ? bzw. [...], so ersetzt die Shell diese Zeichenkette durch alle passenden Dateinamen im aktuellen Dateiverzeichnis. Passt kein Dateiname, übergibt die Shell diese Zeichenkette unverändert an *echo*.

– ’zeichenkette’
  Die Hochkommas entwerten alle Sonderzeichen der Shell. Die Shell übergibt die Zeichenkette ohne Hochkommas als ein Argument an *echo*. Alle zwischen den Hochkommas eingegebenen Leer-, Tabulator- und Neue-Zeile-Zeichen bleiben erhalten.

– ` zeichenkette`
  Die Shell führt *zeichenkette* als POSIX-Kommando aus und übergibt an *echo* die Ausgabe dieses Kommandos. Dabei interpretiert die Shell in der Ausgabe die Zeichen als Argument-Trennzeichen, die der Umgebungsvariablen IFS zugewiesen sind. Standardmäßig sind IFS das Leer-, das Tabulator- und das Neue-Zeile-Zeichen zugewiesen.

– "zeichenkette"
  Die Anführungszeichen entwerten alle Sonderzeichen der Shell bis auf Gegenschrägstrich \, Gegenhochkommas ` ...` und Dollarzeichen $. Die Shell übergibt die bearbeitete Zeichenkette als ein Argument an *echo*. Alle zwischen den Anführungszeichen eingegebenen Leer-, Tabulator- und Neue-Zeile-Zeichen bleiben erhalten.

Das eingebaute Kommando *echo* interpretiert zusätzlich die nachfolgend beschriebenen Steuerzeichen. Da der Gegenschrägstrich für die Shell eine Sonderbedeutung hat, muss er entwertet werden:

– Durch einen vorangestellten Gegenschrägstrich \
  Das gilt auch, wenn das Argument, das dieses Steuerzeichen enthält, in Anführungszeichen eingeschlossen ist.

– Durch Hochkommas ’...’
  Ist das Argument, das dieses Steuerzeichen enthält, in Hochkommas eingeschlossen, so ist der Gegenschrägstrich bereits entwertet.

Beispiel

```
$ echo hello\\tuser
hello   user
$ echo hello’\’tuser
hello   user
$ echo ’hello\tuser’
hello   user
$ echo "hello\tuser"
hello   user
```

Steuerzeichen

Die folgenden Steuerzeichen beeinflussen die Ausgabe von *echo*, falls der Gegenschrägstrich entsprechend entwertet ist.

Beachten Sie bitte die terminalspezifischen Abhängigkeiten.

\c
echo gibt nur aus, was bis zu diesem Steuerzeichen eingegeben wurde, und schließt die Ausgabe nicht mit einem Neue-Zeile-Zeichen ab.

\f
(f - form feed) Seitenvorschub; bei Ausgabe auf den Bildschirm wird dieses Steuerzeichen nicht umgesetzt.
echo schreibt das, was nach dem Steuerzeichen eingegeben wurde, auf die nächste Seite, z.B. wenn die Ausgabe von *echo* an den Drucker geschickt wird.
\ *f* entspricht CTRL L

\n
(n - new line) Zeilenvorschub; *echo* schreibt das, was nach dem Steuerzeichen eingegeben wurde, in die nächste Zeile.
\ *n* entspricht ↵ bzw. CTRL J

\t
Steuerzeichen für Tabulator; *echo* setzt die Schreibmarke auf den nächsten Tabulator. Die Zeichen, die nach \t eingegeben wurden, schreibt *echo* ab dieser Spalte weiter. Die Tabulator-Positionen sind abhängig von der verwendeten Datensichtstation. Für eine Siemens-Datensichtstation 97801 sind Tabulatoren auf folgende Spalten gesetzt: 1, 9, 17, 25, 33, 41, 49, 57, 65, 73, 79
\ *t* entspricht →| bzw. CTRL I

\0n
n muss eine ein-, zwei- oder dreistellige Oktalzahl sein. Das Kommando *echo* gibt das entsprechende Zeichen aus (siehe *Tabellen und Verzeichnisse, Zeichensatz EDF03*). Auf diese Weise können Sie beispielsweise Zeichen erzeugen, deren interner Code einen Wert größer FF (EDF03) hat, auch wenn Sie keine 8-Bit-Datensichtstaion haben.
Für nicht darstellbare Zeichen wird ein Schmierzeichen ausgegeben (geräteabhängig).

Wenn Sie nach einem oktal beschriebenen Zeichen eine Ziffer ausgeben wollen, dann müssen Sie die Oktalzahl dreistellig angeben.

Beispiel

```
$ echo '\0337'|od -xc
0000000    df15
         337  \n
0000002
$ echo '\00337'|od -xc
0000000    1bf7    1500
         033   7  \n
0000003
```

Sollen diese Steuerzeichen weder von der Shell noch von *echo* interpretiert werden, müssen Sie sie wie folgt eingeben:

Beispiel

```
$ echo \\
\
$ echo \\\\t
\t
$ echo '\\t'
\t
$ echo "\\\t"
\t
```

Für die übrigen Zeichenketten gilt dies entsprechend.

argument nicht angegeben:
echo gibt nur eine Leerzeile aus.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *echo*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
: Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH
: Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Der folgende Bildschirm-Dialog soll zeigen, wann Leerzeichen, Tabulatorzeichen und Neue-Zeile-Zeichen erhalten bleiben:

```
$ echo Heute ist        Montag.
Heute ist Montag.
```

Hier erhält *echo* drei Argumente und gibt sie aus, jeweils getrennt durch ein Leerzeichen.

```
$ echo "Heute ist        Montag."
Heute ist        Montag.
```

Zwischen den Anführungszeichen bleiben alle eingegebenen Leerzeichen erhalten.

Beispiel 2 Die Benutzerin *anna* will wissen, welcher Wert der Variablen HOME zugewiesen ist:

```
$ echo $HOME
/home/anna
```

Beispiel 3 Die folgende Zeile in einer Shell-Prozedur bewirkt, dass auf die Standard-Fehlerausgabe eine Fehlermeldung ausgegeben wird:

```
echo Die Datei $1 ist nicht vorhanden >&2
```

Beispiel 4 Vor die Ausgabe des Kommandos *date* soll eine konstante Zeichenkette gestellt werden:

```
$ echo "Heutiges Datum mit Uhrzeit:  \\c"; date
Heutiges Datum mit Uhrzeit:  Mon Mar  9 17:22:05 MEZ 2009
```

Das Steuerzeichen \c bewirkt, dass *echo* kein Neue-Zeile-Zeichen ausgibt. Trotz der Anführungszeichen muss der Gegenschrägstrich entwertet werden.

Siehe auch *print*, *printf*

ed Zeilenorientierter Editor im Dialogbetrieb (interactive line editor)

ed ist ein interaktiver zeilenorientierter Editor. Mit Hilfe von *ed-Skripts* (siehe *Arbeiten mit ed-Skripts*) können Sie bequem mehrere Dateien mit derselben Kommandofolge bearbeiten. *ed* verarbeitet die Ausgabe des Kommandos *diff -e* (siehe *diff*).

Syntax **ed**[␣**-**|␣**-s**][␣**-p**␣zeichenkette][␣datei]

␣**-**|␣**-s**
Die Option *-s* entspricht der alten Option -, die weiterhin unterstützt wird.

Die Option *-s* unterdrückt die folgenden standardmäßigen Ausgaben:

– Anzahl der verarbeiteten Zeichen bei den *ed*-Kommandos
e (edit - editieren)
r (read - lesen)
w (write - schreiben)

– Fragezeichen ?, das vor versehentlichem Löschen des Pufferinhalts warnt, bei den *ed*-Kommandos
e (edit - editieren)
q (quit - verlassen)

– Ausrufezeichen ! als Bereitzeichen von *ed* nach einem !-Kommando.

-p␣zeichenkette
Mit *zeichenkette* können Sie die Bereitzeichenkette definieren, die *ed* im Kommandomodus ausgibt. Für *zeichenkette* können Sie ein oder mehrere Zeichen angeben.

-p␣zeichenkette nicht angegeben:
ed gibt keine Bereitzeichenkette aus.

datei
Name der Datei, die Sie bearbeiten möchten. *ed* kopiert die Datei in den internen Puffer und speichert *datei* als aktuellen Dateinamen.

datei nicht angegeben:
Sie beginnen in einem leeren Puffer zu arbeiten und bestimmen erst beim Schreiben des Pufferinhalts mit dem Kommando *w␣datei* (write - schreiben) in eine Datei deren Namen.

ed-Puffer

Beim Aufruf von *ed* wird ein Puffer eröffnet.

Wenn Sie keine Datei angegeben haben, ist der Puffer leer. Sie füllen ihn während Ihrer Editorsitzung mit Text.

Wenn Sie eine Datei angegeben haben, wird eine Kopie dieser Datei in den Puffer eingelesen. Den Text im Puffer bearbeiten Sie während Ihrer Editorsitzung.

Bevor Sie den Editor wieder verlassen, müssen Sie entscheiden, ob Sie den neu erstellten oder geänderten Pufferinhalt sichern, d.h. in eine Datei schreiben möchten.

Möchten Sie den Pufferinhalt sichern, schreiben Sie den Pufferinhalt mit dem Kommando *w*[␣*datei*] (write - schreiben) in die angegebene Datei (Standard: die beim Aufruf von *ed* angegebene) zurück und verlassen dann den Editor mit einem der Kommandos *q* (quit - verlassen), *Q* (Quit - verlassen) oder mit der Taste END.

Möchten Sie den Pufferinhalt nicht sichern, können Sie mit *Q* oder zweimal *q* den Editor verlassen, ohne vorher den Pufferinhalt mit *w* zurückzuschreiben. Nach der ersten Eingabe von *q* gibt *ed* als Warnung vor versehentlichem Löschen des Pufferinhalts ein ? aus. Gelöscht wird der Pufferinhalt erst, wenn Sie dann ein zweites Mal *q* eingeben. Statt des zweiten *q* können Sie auch *Q* oder END eingeben.

Arbeitsmodi

Der *ed* bietet Ihnen zwei Arbeitsmodi: den Kommandomodus und den Eingabemodus. Nach dem Aufruf mit *ed[␣datei]* ↵ befindet sich *ed* im Kommandomodus. Im Kommandomodus geben Sie i.a. ein Kommando in einer Zeile an und schließen es mit der Taste ↵ ab.

Den Eingabemodus schalten Sie ein mit einem der Kommandos

a (append - anfügen)
i (insert - einfügen)
c (change - verändern)
(siehe unten, *ed-Kommandos*).

Im Eingabemodus werden alle folgenden Eingabezeichen, auch nicht-druckbare Zeichen (wie z.B. die Tastencodes der Schreibmarkentasten), in die Arbeitskopie im Puffer geschrieben. Im Eingabemodus erkennt *ed* keine Kommandos. Sie können für den Kommandomodus ein Bereitzeichen definieren (siehe Option *-p* und *ed*-Kommando *P*) und erkennen dann sofort, in welchem Modus Sie gerade arbeiten. Verlassen können Sie den Eingabemodus mit einem Punkt in der ersten Spalte .↵ oder mit der Taste DEL. Die Taste DEL bewirkt generell, dass *ed* alle Eingaben seit dem letzten ↵ ignoriert und ein ? als Warnung ausgibt.

Kommandostruktur

Für *ed* existiert zu jedem Zeitpunkt eine *aktuelle Zeile*. Die aktuelle Zeile ist in der Regel die zuletzt durch ein Kommando bearbeitete Zeile. Auf diese aktuelle Zeile beziehen sich die Kommandos immer dann, wenn Sie vor den Kommandos keine anderen Adressen angeben.

Die meisten *ed*-Kommandos haben folgende Struktur:

```
[bereich]ed-kommando[parameter...] ↵
```

bereich
: Mit *bereich* wählen Sie die Zeilen im Puffer aus, auf die das *ed*-Kommando angewendet werden soll. Für *bereich* können Sie eine oder zwei Adressen angeben:

  bereich = adresse
  : Die durch *adresse* bezeichnete Zeile gilt als ausgewählt.

  bereich = adresse1**,**adresse2
  : *adresse1*,*adresse2* kennzeichnen den Bereich zwischen den angegebenen Intervallgrenzen einschließlich. Die Suche nach beiden Adressen beginnt in der aktuellen Zeile. Verändert wird die aktuelle Zeile erst bei der Ausführung von Kommandos.
  adresse2 muss sich auf eine Zeile im Puffer beziehen, die hinter der mit *adresse1* bezeichneten Zeile liegt, sonst meldet *ed* einen Fehler.

  bereich = adresse1**;**adresse2
  : *adresse1*;*adresse2* kennzeichnen den Bereich zwischen den angegebenen Intervallgrenzen einschließlich. Die Suche nach *adresse1* beginnt in der aktuellen Zeile. Die mit *adresse1* bezeichnete Zeile wird dann zur neuen aktuellen Zeile, dann erst wird *adresse2* ermittelt. Sie können so die Zeile festlegen, bei der ein Suchvorgang in Vor- und Rückwärtsrichtung beginnen soll, siehe „Adressen“ auf Seite 306, *//* und */rA/* auf der folgenden Seite.
  adresse2 muss sich auf eine Zeile im Puffer beziehen, die hinter der mit *adresse1* bezeichneten Zeile liegt, sonst meldet *ed* einen Fehler.

  bereich nicht angegeben:
  ed nimmt die zum jeweiligen Kommando gehörige Standard-Adresse an; diese ist bei jedem *ed*-Kommando beschrieben.

Benötigt *ed* keine Adresse und haben Sie trotzdem eine angegeben, meldet *ed* einen Fehler.

Haben Sie mehr Adressen angegeben als das Kommando erlaubt, nimmt *ed* die letzten.

Adressen

Adressen konstruieren Sie folgendermaßen :

Adresse	*Bedeutung*
.	aktuelle Zeile
$	letzte Zeile
n	n-te Zeile
'x	die mit dem Buchstaben *x* markierte Zeile. *x* muss ein Kleinbuchstabe sein (siehe *k*).
/rA**/**	Ein einfacher regulärer Ausdruck *rA* in /.../ eingeschlossen (siehe Seite 939), adressiert die erste Zeile, die eine zu dem regulären Ausdruck passende Zeichenkette enthält. Es wird dabei von der aktuellen Zeile aus vorwärts gesucht. Findet *ed* keine passende Zeile, setzt *ed* die Suche am Dateianfang fort und sucht wieder bis zur aktuellen Zeile. Enthält der reguläre Ausdruck *rA* selbst eines der Begrenzungszeichen / oder ?, muss es mit \ entwertet werden. Die Zeichen \n in einem regulären Ausdruck erzielen bei Neue-Zeile-Zeichen im durchsuchten Text *keinen* Treffer! Ein regulärer Ausdruck kann in *bereich* nur einmal vorkommen, z.B. adressiert /rA1/,/rA2/ nur die erste zu /rA2/ passende Zeile.
//	Ein leerer regulärer Ausdruck // adressiert die Zeile, die zu dem zuletzt angegebenen regulären Ausdruck passt.
?rA**?**	Wie /rA/, aber es wird von der aktuellen Zeile aus rückwärts in Richtung Dateianfang gesucht. Enthält der reguläre Ausdruck *rA* selbst eines der Begrenzungszeichen / oder ?, muss es mit \ entwertet werden.
adr[**+**]n	*n*-te Zeile nach der durch *adr* bezeichneten Zeile.
adr[**-**]n	*n*-te Zeile vor der durch *adr* bezeichneten Zeile.
+n	*n* Zeilen nach der aktuellen Zeile.
-n	*n* Zeilen vor der aktuellen Zeile.
[adr]**+**... [adr]**–**...	Eine Zeile nach (+) bzw. vor (-) der durch *adr* bezeichneten Zeile. Jedes Vorkommen von + erhöht die Adressangabe um 1, - erniedrigt die Adressangabe um 1. Die Adressangabe ++ adressiert also die zweite Zeile nach der aktuellen Zeile.
,	Ein Komma , ist gleichbedeutend mit dem Adressenpaar 1,$, wenn ein Kommando folgt, sonst wird die letzte Zeile ausgegeben.

; Ein Strichpunkt ; ist gleichbedeutend mit dem Adressenpaar .,$, wenn ein Kommando folgt, sonst wird die letzte Zeile ausgegeben.

Kommandos

Die folgende Liste enthält eine systematische Übersicht aller *ed*-Kommandos, die Sie im Kommandomodus eingeben können. Die daran anschließende ausführliche Kommandobeschreibung ist alphabetisch geordnet.

- **Eingabemodus einschalten**
  a (append)
  anfügen
  c (change)
  löschen und ersetzen
  i (insert)
  einfügen
- **Bereitzeichen im Kommandomodus ausgeben**
  P (prompt)
  Bereitzeichen * ausgeben
- **Kommandos rückgängig machen**
  u (undo)
  rückgängig machen
- **Kommandos abbrechen**
  DEL
  abbrechen
- **Fehler erläutern**
  h (help)
  letzte Fehlermeldung erläutern
  H (Help)
  Hilfemodus ein-/ausschalten. Bei jeder Fehleranzeige in Form eines Fragezeichens ? wird bei eingeschaltetem Hilfemodus eine Fehlermeldung ausgegeben (siehe *Fehler*).

- **Text ändern**
  a (append)
  anfügen
  c (change)
  löschen und ersetzen
  d (delete)
  löschen
  i (insert)
  einfügen

- **Zeilen ausgeben**
  p (print)
  ausgeben
  l (list)
  ausgeben mit nicht-druckbaren Zeichen in Ersatzdarstellung oder als Oktalzahlen
  adresse
  adressierte Zeile ausgeben
  adresse +zahl
  adressierte Zeile ausgeben
  n (number)
  Zeilen nummerieren und ausgeben
  ↵
  Zeile hinter aktueller Zeile ausgeben

- **Zeilennummern ausgeben**
  adresse=
  Nummer der adressierten Zeile ausgeben
  n (number)
  Zeilen nummerieren und ausgeben

- **Zeilenbereiche verschieben**
  t (transfer)
  kopieren von Zeilen
  m (move)
  verschieben von Zeilen

- **Textmuster suchen und ersetzen**
  s (substitute)
  suchen und ersetzen

- **Zeilen verbinden**
  j (join)
  aufeinanderfolgende Zeilen verbinden

- **Zeilen markieren**
  k (mark)
  Zeilen markieren

- **Ausgewählte Zeilen mit Kommandos bearbeiten**

  g (global)
  Kommandoliste global für alle Zeilenausführen, die zu */rA/* passen

  G (Global)
  interaktiv Kommandoliste global für alle Zeilen ausführen, die zu */rA/* passen

  v (vice-versa)
  wie g, aber für alle Zeilen, die nicht zu */rA/* passen

  V (Vice-Versa)
  wie G, aber für alle Zeilen, die nicht zu */rA/* passen

- **Aktuellen Dateinamen ändern**

  f (file-name)
  aktuellen Dateinamen ändern/ausgeben

- **Shell-Kommandos ausführen**

  !
  Shell-Kommando aufrufen

- **Dateien in Puffer einlesen**

  e (edit)
  Pufferinhalt löschen und neu einlesen

  E (Edit)
  Pufferinhalt ohne Warnung löschen und neu einlesen

  r (read)
  Datei in Puffer einlesen

- **Pufferinhalt sichern**

  w (write)
  Pufferinhalt in Datei schreiben

  W (write)
  Pufferinhalt an Datei anfügen

- **Editor verlassen**

  q (quit)
  ed verlassen

  Q (Quit
  ed verlassen ohne Warnung

  END
  ed verlassen

Beschreibung der ed-Kommandos

Die eckigen Klammern [] sind nicht einzugeben! Sie zeigen an, dass die dazwischen eingeschlossene Adressangabe fakultativ ist.
In der Regel darf in einer Zeile nur ein Kommando stehen. Jedoch können Sie an die Kommandos (mit Ausnahme von *e, f, r* und *w*) als Suffix *l, n* oder *p* anhängen, wenn die im folgenden unter den Kommandos *l, n* und *p* beschriebenen Funktionen ausgeführt werden sollen.

[adresse]**a**
text

. (a - append) liest den eingegebenen *text* und fügt ihn hinter der mit *adresse* adressierten Zeile an. Die aktuelle Zeile ist nun entweder die letzte Zeile des eingefügten Textes oder, falls Sie keinen Text eingegeben haben, die adressierte Zeile. Die Adresse 0 ist bei diesem Kommando erlaubt; der Text wird dann vor die erste Zeile des Puffers eingefügt. Über die Datensichtstation können Sie maximal 2048 Zeichen je Zeile einschließlich Neue-Zeile-Zeichen eingeben.

adresse nicht angegeben:
adresse = .

[bereich]**c**
text

. (c - change) löscht den angegebenen *bereich* und ersetzt ihn durch den eingegebenen *text*. Die aktuelle Zeile ist nun entweder die letzte Zeile des eingegebenen Textes oder, falls Sie keinen Text eingegeben haben, die Zeile vor den gelöschten Zeilen.

bereich nicht angegeben:
bereich = .,.

[bereich]**d**
(d - delete) löscht den angegebenen *bereich*. Die Zeile hinter der letzten gelöschten Zeile wird zur aktuellen Zeile. Standen die gelöschten Zeilen am Ende des Puffers, wird die neue letzte Zeile die aktuelle Zeile.

bereich nicht angegeben:
bereich = .,.

e[␣datei]
(e - edit) löscht den gesamten Puffer und liest eine Kopie des Inhalts von *datei* ein. Ist der Pufferinhalt seit dem letzten Sichern oder Überschreiben verändert und nicht mit *w* gesichert worden, löscht *ed* den Pufferinhalt nicht sofort, sondern warnt Sie mit einem ? vor versehentlichem Löschen. Geben Sie daraufhin ein weiteres Mal *e* ein, wird der Pufferinhalt ohne weitere Warnung überschrieben. Die Anzahl der eingelesenen Bytes wird ausgegeben, wenn Sie *ed* nicht mit der Option *-s* aufgerufen haben. Die aktuelle Zeile ist die letzte Zeile des Puffers. Der Name *datei* wird gespeichert, so dass er später

bei den Kommandos *e, r* und *w* als aktueller Dateiname verwendet werden kann. Wenn Sie für *datei* ein Ausrufezeichen ! angeben, wird der Rest der Zeile als Shell-Kommando interpretiert. Das Ergebnis dieses Shell-Kommandos wird in den Puffer eingelesen. Ein mit ! eingeleitetes Shell-Kommando wird nicht als Dateiname gespeichert.

datei nicht angegeben:
datei = aktueller Dateiname

E[␣datei]

(E - edit) verhält sich wie *edit*, außer, dass es den Puffer ohne warnendes ? überschreibt, auch wenn der Inhalt des Puffers verändert und nicht gerettet wurde.

datei nicht angegeben:
datei = aktueller Dateiname

f[␣datei]

(f - file) setzt den aktuellen Dateinamen auf *datei*. Den aktuellen Dateinamen verwenden die Kommandos *e, E, r* und *w*.

datei nicht angegeben:
ed gibt den aktuellen Dateinamen aus.

[bereich]␣**g/**rA**/**kommandoliste

(g - global) markiert im ersten Schritt alle Zeilen, die eine Zeichenkette enthalten, die zu *rA* passt. *rA* ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Dann wird die nächste markierte Zeile zur aktuellen Zeile und *kommandoliste* darauf angewandt.

Ein einzelnes Kommando oder das erste einer *kommandoliste* müssen Sie in dieselbe Zeile schreiben wie das Kommmando *g*. Alle Zeilen einer *kommandoliste* außer der letzten müssen Sie mit \↵, die letzte mit ↵ abschließen.

Die Kommandos *a*, *i* und *c* mit der dazugehörigen Eingabe *text* sind zugelassen. Auch Zeilen in *text* müssen Sie mit \↵ abschließen. Den die Eingabe normalerweise abschließenden Punkt . können Sie in der letzten Zeile der *kommandoliste* weglassen.

Eine leere *kommandoliste* entspricht dem Kommando *p*.

Die Kommandos *g,G,v* und *V* sind in der *kommandoliste* nicht zugelassen.

Nicht mit dem Kommando ! zu kombinieren!

bereich nicht angegeben:
bereich = 1,$

[bereich]**G/**rA**/**

(G - global) *G* ist die interaktive Variante des Kommandos *g*. Zuerst wird jede Zeile markiert, die zu *rA* passt. *rA* ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Die erste der markierten Zeilen wird ausgegeben. Sie wird gleichzeitig zur aktuellen Zeile.

Sie haben nun die Möglichkeit, ein Kommando anzugeben, das ausgeführt werden soll. Hierbei darf es sich nicht um *a, c, i, g, G, v* oder *V* handeln. Nach der Ausführung dieses Kommandos wird die nächste markierte Zeile ausgegeben, usw.

Ein Neue-Zeile-Zeichen hat dieselbe Wirkung wie ein leeres Kommando. Wenn Sie das kommerzielle Und & eingeben, wird das Kommando wiederholt, das seit dem letzten Aufrufen von *G* als letztes ausgeführt wurde.

Mit Kommandos, die unter *G* aufgerufen werden, können beliebige Zeilen im Puffer adressiert und bearbeitet werden. *G* können Sie mit der Taste DEL abbrechen.

bereich nicht angegeben:
bereich = 1,$

h (h - help) gibt eine kurze Fehlermeldung aus, die den Grund für das letzte Fragezeichen ? auf dem Bildschirm erklärt. Mögliche Fehlermeldungen siehe *Fehler*.

H (H - help) bewirkt, dass *ed* in einen Modus übergeht, in dem bei auftretenden Fehlern anstelle von ? erläuternde Fehlermeldungen ausgegeben werden. Auch das zuletzt ausgegebene ? wird, falls vorhanden, erläutert. Diesen Modus können Sie durch ein zweites Aufrufen von *H* ausschalten.
Standardmäßig ist der Modus *H* ausgeschaltet.
Mögliche Fehlermeldungen siehe *Fehler*.

[adresse]**i**
text

. (i - insert) fügt *text* vor der mit *adresse* adressierten Zeile ein. Die aktuelle Zeile ist nun entweder die letzte Zeile des eingefügten Textes oder, falls Sie keinen Text eingegeben haben, die adressierte Zeile. Dieses Kommando unterscheidet sich vom Kommando *append* nur durch die Positionierung des eingegebenen Textes. Adresse 0 ist nicht zulässig. Über eine Datensichtstation dürfen Sie maximal 2048 Zeichen einschließlich des Neue-Zeile-Zeichens je Zeile eingeben.

adresse nicht angegeben:
adresse = .

[bereich]**j**
(j - join) verbindet alle in *bereich* liegenden Zeilen zu einer einzigen Zeile und löscht dabei die entsprechenden Neue-Zeile-Zeichen. Wenn Sie als Adresse nur eine Zeile angeben, geschieht nichts. Die neue Zeile wird zur aktuellen Zeile.

bereich nicht angegeben:
bereich = .,.

[adresse]**k**x

(k - mark) markiert die mit *adresse* adressierte Zeile mit dem für *x* angegebenen Buchstaben, wobei *x* ein Kleinbuchstabe sein muss. Die Markierung wird aber nicht ausgegeben. Adressieren können Sie die markierte Zeile mit Hochkomma x ('x). Die aktuelle Zeile bleibt unverändert.

adresse nicht angegeben:
adresse = .

[bereich]**l**

(l - list) gibt im Gegensatz zu *p* den angegebenen *bereich* wie folgt aus: Einige nicht-druckbare Zeichen werden Ersatzdarstellung (z.B. Tabulatorzeichen), die übrigen nicht-druckbaren Zeichen werden als Oktalzahlen ausgegeben; überlange Zeilen werden mehrzeilig ausgegeben, am Ende mit dem Zeilenfortsetzungszeichen \ versehen; jedes Zeilenende wird mit $ gekennzeichnet. Das Kommando *l* können Sie an jedes Kommando anhängen, mit Ausnahme von *e, f, r* und *w*.

Folgende Ersatzdarstellungen werden verwendet:

\\ Backslash (zur Unterscheidung von Oktalzahlen)
\a Warnung, Klingel *)
\b Backspace, Rücksetzzeichen *)
\f Form Feed, Seitenvorschub
\n Newline, Neue-Zeile-Zeichen
\r Carriage Return, Wagenrücklauf
\t Tabulator
\v Vertikal-Tabulator *)

*) Die so markierten Sonderzeichen werden nur auf Zeichenterminals unterstützt (also bei Zugang zur POSIX-Shell über rlogin)

bereich nicht angegeben:
bereich = .,.

[bereich]**m**adresse

(m - move) verschiebt den *bereich* hinter die mit *adresse* adressierte Zeile. Die letzte der verschobenen Zeilen wird die aktuelle Zeile. Wenn Sie für *adresse* den Wert 0 angeben, wird *bereich* ganz an den Anfang der Datei gesetzt. Liegt *adresse* innerhalb von *bereich*, gibt *ed* eine Fehlermeldung aus.

bereich nicht angegeben:
bereich = .,.

[bereich]**n**

(n - number) gibt die mit *bereich* adressierten Zeilen aus, wobei an den Anfang jeder Zeile die Zeilennummer und ein Tabulatorzeichen gesetzt werden. Die zuletzt ausgegebene Zeile wird zur neuen Zeile. Das Kommando *n* können Sie an jedes Kommando anhängen, mit Ausnahme von *e, f, r* und *w*.

bereich nicht angegeben:
bereich = .,.

[bereich]**p**

(p - print) gibt die mit *bereich* adressierten Zeilen aus. Nicht-druckbare Zeichen werden unverändert ausgegeben. Überlange Zeilen laufen ohne besondere Kennzeichnung auf der nächsten Zeile weiter, d.h. man kann sie nicht als solche erkennen. Die zuletzt ausgegebene Zeile wird zur aktuellen Zeile. Das Kommando *p* können Sie an jedes Kommando anhängen, mit Ausnahme von *e, f, r* und *w*. Mit *dp* wird also die aktuelle Zeile gelöscht und die neue aktuelle Zeile ausgegeben.

bereich nicht angegeben:
bereich = .,.

P (P - prompt) bewirkt, dass *ed* im Kommandomodus als Bereitzeichen den Stern * oder die Bereitzeichenkette (siehe Option *-p*) ausgibt. Diesen Modus können Sie durch einen zweiten Aufruf von *P* wieder ausschalten. Standardmäßig ist dieser Modus ausgeschaltet.

q (q - quit) beendet den *ed*. Ist der Pufferinhalt seit dem letzten Sichern oder Überschreiben verändert und nicht mit *w* gesichert worden, gibt *ed* als Warnung vor versehentlichem Löschen ein ? aus und wartet auf weitere Eingabe. Wenn Sie dann END, *Q* oder zum zweitenmal *q* eingeben, beenden Sie den *ed* ohne weitere Warnung und ohne den Pufferinhalt gesichert zu haben.

Achtung!
Wenn Sie nach dem ersten *q* noch Aktionen durchführen, die den Pufferinhalt nicht verändern, beenden Sie mit dem zweiten *q* auch den *ed* ohne weitere Warnung und ohne Sicherung des Pufferinhalts.

Q (Q - quit) beendet den *ed* sofort ohne Warnung, auch wenn Sie den Pufferinhalt seit dem letzten Sichern oder Überschreiben verändert und nicht mit *w* gesichert haben.

[adresse]**r**[␣datei]

(r - read) liest *datei* und fügt ihren Inhalt hinter der mit *adresse* adressierten Zeile ein. Die Adresse 0 ist für dieses Kommando erlaubt.
Sie bewirkt, dass *datei* an den Anfang des Puffers geschrieben wird. Nach erfolgreichem Lesen wird die Anzahl der gelesenen Bytes ausgegeben, wenn Sie *ed* nicht mit der Option *-s* aufgerufen haben. Die aktuelle Zeile ist die letzte eingelesene Zeile. Der aktuelle Dateiname wird nicht auf *datei* gesetzt, es sei denn, Sie haben *ed* ohne Dateinamen aufgerufen und *datei* ist der erste seit dem Aufruf angesprochene Dateiname.

Wenn Sie für *datei* ein Ausrufezeichen ! angeben, dann wird der Rest der Zeile als Shell-Kommando interpretiert, ausgeführt und die Ausgabe davon eingelesen. Ein solches Kommando wird nicht als aktueller Dateiname gespeichert.

adresse nicht angegeben:
adresse = $

datei nicht angegeben:
datei = aktueller Dateiname

[bereich]**s/**rA**/**Ersetzungszeichenkette**/**[**g**][**l**][**n**][**p**][count]

(s - substitute) durchsucht jede Zeile in *bereich* nach Zeichenketten, die zu *rA* passen. *rA* ist ein einfacher internationalisierter regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). In *jeder* so gefundenen Zeile wird die erste zu *rA* passende Zeichenkette durch *Ersetzungszeichenkette* ersetzt. Ist der Schalter *g* gesetzt, werden alle zu *rA* passenden Zeichenketten durch *Ersetzungszeichenkette* ersetzt.
Falls keine passende Zeichenkette gefunden wurde, meldet *ed* ein ? als Fehler.

Als Trennzeichen zwischen dem Kommando *s*, dem regulären Ausdruck *rA* und *Ersetzungszeichenkette* können Sie nicht nur den Schrägstrich / verwenden, sondern auch jedes andere beliebige Zeichen außer: Leerzeichen und Neue-Zeile-Zeichen. Das Zeichen wird dadurch als Trennzeichen definiert, dass es unmittelbar auf *s* folgt. Zur aktuellen Zeile wird die Zeile, auf der die letzte Ersetzung stattgefunden hat.

count — Ersetzt das *count*e Auftreten des regulären Ausdrucks in einer Zeile.

g — Es wird nicht nur das erste Auftreten des *rA*, sondern es werden global alle alle nicht-überlappenden Vorkommen des *rA* ersetzt. Werden sowohl *g* als auch *count* angegeben, ist das Ergebnis undefiniert.

l — Die letzte Zeile, in der eine Ersetzung stattgefunden hat, wird auf die Standardausgabe im Format des *ed*-Kommandos *l* (list) geschrieben.

n — Die letzte Zeile, in der eine Ersetzung stattgefunden hat, wird auf die Standardausgabe im Format des *ed*-Kommandos *n* (number) geschrieben.

p — Die letzte Zeile, in der eine Ersetzung stattgefunden hat, wird auf die Standardausgabe im Format des *ed*-Kommandos *p* (print) geschrieben.

Sonderzeichen in der Ersetzungszeichenkette

Zeichen und Bedeutung

& — wird durch die Zeichenkette ersetzt, die in der aktuellen Zeile zu dem regulären Ausdruck *rA* passt.

\m — wird durch die Zeichenkette ersetzt, die zum *m*-ten in \(...\) eingeschlossenen regulären Unterausdruck von *rA* passt, wobei *m* eine einstellige Dezimalzahl ist. Bei durch Klammern ineinander geschachtelten Unterausdrücken wird *m* durch Zählen des Auftretens von \ von links nach rechts bestimmt.

% wird durch die Zeichenkette ersetzt, mit der beim zuletzt abgelaufenen *s*-Kommando ersetzt wurde, wenn *Ersetzungszeichenkette* nur aus dem Prozent-Zeichen % besteht.

Die Sonderbedeutung dieser Zeichen können Sie aufheben, indem Sie Ihnen jeweils einen Gegenschrägstrich \ voranstellen.

Zeile trennen in Ersetzungszeichenkette

Wenn Sie in *Ersetzungszeichenkette* ein Neue-Zeile-Zeichen haben möchten, müssen Sie es mit \ davor entwerten. Sie geben also \↵ ein. Ein solches Ersetzungskommando darf nicht Teil von *kommandoliste* der Kommandos *g* und *v* sein.

bereich nicht angegeben:
bereich = .,.

[bereich]**t**a
kopiert *bereich* hinter die angegebene Zeile *a*. Als Adresse für *a* ist 0 zugelassen. Zur aktuellen Zeile wird die letzte der kopierten Zeilen.

bereich nicht angegeben:
bereich = .,.

u (u - undo) macht das letzte Kommando rückgängig, mit dem der Inhalt des Puffers geändert wurde. Rückgängig machen können Sie die Kommandos *a, c, d, g, i, j, m, r, s, t, v, G* und *V.*

[bereich]**v/**rA**/**kommandoliste
(v - vice-versa) bearbeitet alle Zeilen mit *kommandoliste*, die *keine* Zeichenkette enthalten, die zu *rA* passt. *rA* ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Bis auf diese Zeilenauswahl funktioniert *v* wie das Kommando *g*.

Nicht mit dem Kommando ! zu kombinieren!

bereich nicht angegeben:
bereich = 1,$

[bereich]**V/**rA**/**
(V - vice-versa) ist die interaktive Variante des Kommandos *v*. Sie können mit *V* alle Zeilen bearbeiten, die *keine* Zeichenkette enthalten, die zu *rA* passt. *rA* ist ein einfacher internationalisierter regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Bis auf diese Zeilenauswahl funktioniert *V* wie das Kommando *G*.

bereich nicht angegeben:
bereich = 1,$

[bereich]**w**[␣datei]

(w - write) schreibt *bereich* in die *datei*. Der aktuelle Dateiname ändert sich nicht, falls er bereits gesetzt ist. Ist er nicht gesetzt, wird *datei* zum neuen aktuellen Dateinamen. Der alte Inhalt von *datei* wird dabei überschrieben. Existiert *datei* noch nicht, so wird sie angelegt. Die aktuelle Zeile bleibt unverändert. Nach erfolgreichem Schreiben wird die Anzahl der geschriebenen Bytes ausgegeben, vorausgesetzt, Sie haben den *ed* nicht mit der Option *-s* aufgerufen.

Wenn Sie für *datei* statt eines Dateinamens das Ausrufezeichen ! angeben, wird der Rest der Zeile als Shell-Kommando interpretiert und ausgeführt. *bereich* ist dann die Standard-Eingabe für das Shell-Kommando. Ein solches Kommando wird nicht als aktueller Dateiname gespeichert.

bereich nicht angegeben:
bereich = 1,$

datei = aktueller Dateiname

[bereich]**W**[␣datei]

(W - write) fügt *bereich* an die Datei *datei* an, sonst verhält es sich wie das Kommando *w*, das *datei* aber überschreibt. Existiert *datei* noch nicht, wird sie angelegt.

adresse

Die mit *adresse* adressierte Zeile wird ausgegeben.

[adresse]**+**zahl

Dieses Kommando gibt die Zeile aus, die *zahl* Zeilen hinter der mit *adresse* adressierten Zeile liegt.

adresse nicht angegeben:
adresse = .

[adresse]**=**

Die Nummer von *adresse* wird ausgegeben. Die aktuelle Zeile wird dadurch nicht verändert.

adresse nicht angegeben:
adresse = $

!kommando

Der Rest der Zeile hinter dem ! wird als Shell-Kommando interpretiert und ausgeführt. Ist in *kommando* ein nicht entwertetes Prozent-Zeichen % enthalten, wird es durch den gespeicherten Dateinamen ersetzt. !! wiederholt das letzte *kommando*. In beiden Fällen wird die expandierte Kommandozeile ausgegeben. Nach Beendigung des *kommandos* ist *ed* wieder aktiv. Die aktuelle Zeile wird nicht verändert. Nicht zu kombinieren mit den Kommandos *g* und *v*!

↵ Geben Sie im Kommandomodus die Taste ↵ alleine ein, wird die Zeile hinter der aktuellen Zeile ausgegeben. Dies ist mit der Eingabe .+1p gleichbedeutend. Sie können so im Puffer von einer Zeile zur nächsten springen.

Die Taste ↵ müssen Sie drücken, um im Kommandomodus die Eingabe eines Kommandos oder um im Eingabemodus die Eingabe einer Textzeile abzuschließen.

DEL
Mit der Taste DEL können Sie laufende *ed*-Kommados unterbrechen oder die Eingabe einer Zeile abbrechen. *ed* meldet sich anschließend mit einem ? zurück.

END oder CTRL D oder @@d
Die Wirkung der Taste END (oder der Tastenkombinationen) ist dieselbe wie beim Kommando *q*.

Steht das Zeichen, das einen regulären Ausdruck oder eine Ersetzungszeichenkette abschließt (z.B. ein /), unmittelbar vor einem Neue-Zeile-Zeichen, dann können Sie dieses Zeichen weglassen. Die adressierte Zeile wird dann ausgegeben. Die folgenden Beispielpaare haben die gleiche Funktion:

```
s/s1/s2       s/s1/s2/p
g/s1          g/s1/p
?s1           ?s1?
```

Arbeiten mit ed-Skripts

ed liest Kommandos und einzufügenden Text von der Standard-Eingabe. Deshalb können Sie die Eingabe auch umlenken, so dass *ed* aus einer Datei liest. Mit

```
$ ed –s datei < ed_skriptdatei > ausgabe
```

wird die Datei *datei* editiert und mit den *ed*-Kommandos bearbeitet, die in der Datei *ed_skriptdatei* stehen. Die Option *-s* unterdrückt die standardmäßige Ausgabe der Meldungstexte auf den Bildschirm.

Das Arbeiten mit *ed-Skripts* hat den Vorteil, dass Sie bestimmte Kommandofolgen jederzeit reproduzieren und beliebig oft verwenden können. Außerdem können Sie so diesen Vorgang auch als Hintergrundprozess ablaufen lassen und selbst ungestört am Bildschirm weiterarbeiten:

```
$ ed datei < ed_skriptdatei&
```

Arbeiten Sie mit einem fehlerhaften *ed-Skript*, dann beendet *ed* beim ersten Fehler.

Endestatus

0 bei Erfolg

>0 bei fehlerhaftem Aufruf des *ed* oder bei Abbruch einer Prozedur aufgrund fehlerhafter *ed*-Kommandos.

Fehler

Wenn Sie beim Aufruf von *ed* zwischen Option *-p* und *zeichenkette* kein Leerzeichen eingeben oder *zeichenkette* vergessen:

```
ed: -p arg missing
ed: -p argument fehlt
```

Wenn Ihnen bei *ed*-Kommandos Fehler unterlaufen:

`?`
Syntaxfehler im Kommando

`?datei`
Datei *datei* ist nicht vorhanden oder kann nicht gelesen werden.

Nähere Informationen bekommen Sie mit den Kommandos *h* und *H*. Die häufigsten Fehlermeldungen sind:

`line out of range`
Zeile außerhalb des gültigen Zeilenbereichs

`warning: expecting 'w'`
Warnung: 'w' wird erwartet

`no space after command`
kein Leerzeichen hinter dem Kommando

`unknown command`
unbekanntes Kommando

`bad range`
ungültige Bereichsangabe

`cannot open input file`
Eingabedatei kann nicht geöffnet werden

`illegal or missing delimiter`
unzulässiges oder fehlendes Trennzeichen

`illegal suffix`
unzulässiges Suffix

`illegal or missing filename`
unzulässiger oder fehlender Dateiname

`no match`
keine passende Zeichenkette gefunden

Datei *ed.hup*
In dieser Datei werden die Daten gesichert, wenn *ed* das Signal SIGHUP empfängt (siehe *kill()*).

/var/tmp
Wenn dieses Dateiverzeichnis existiert, dann wird es als Dateiverzeichnis für die Speicherung der Temporärdatei verwendet.

/tmp
Wenn die Variable *TMPDIR* nicht auf ein existierendes Dateiverzeichnis gesetzt ist und das Verzeichnis */var/tmp* nicht existiert, dann wird */tmp* zur Speicherung der Temporärdatei verwendet.

Variable *TMPDIR*
Wenn diese Variable gesetzt und nicht leer ist, dann wird ihr Wert anstelle von */var/tmp* als Dateiverzeichnis für die Speicherung der Temporärdatei verwendet.

HOME
Legt den Pfadname für das HOME-Dateiverzeichnis des Benutzers fest.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *ed*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt in geklammerten regulären Ausdrücken die Bedeutung von Zeichenbereichen, Äquivalenzklassen und Zeicheneinheiten fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Beispiel für ein *ed-Skript*:

In einer Datei sollen die ersten drei Zeilen durch eine Zeile mit dem Text „Adressen" und überall soll das Wort „Stachus“ durch „Karlsplatz“ ersetzt werden.

Inhalt der Datei *edskript*:

```
1,3c
Adressen
.
1,$s/Stachus/Karlsplatz/g
w
```

Bearbeitung einer Datei mit den Kommandos aus *edskript*:

```
$ ed datei < edskript
```

Wenn *ed* seine Kommandos nicht von der Tastatur, sondern aus einer Datei liest, wird der Editor nach dem ersten für *ed* unverständlichen Kommando verlassen.

Beispiel 2 Beispiel für ein *here-skript*:

In beliebigen Dateien, die beim Aufruf der Prozedur *xy* als Argumente übergeben werden, sollen die ersten drei Zeilen durch eine Zeile mit dem Text „Adressen" und überall soll das Wort „Stachus" durch „Karlsplatz" ersetzt werden.

Inhalt der Prozedurdatei *xy*:

```
for i in $*
do
ed $i << scrend
1,3c
Adressen
.
1,\$s/Stachus/Karlsplatz/g
w
scrend
done
```

Bearbeitung der Dateien *text1, text2* und *text3* mit der Prozedur *xy*:

```
$ sh xy text1 text2 text3
```

Die Zeichenkette << *scrend* hinter dem *ed*-Aufruf bewirkt, dass die Shell den Text bis zur Zeichenkette *scrend* an *ed* als Eingabe übergibt. Die zweite Zeichenkette *scrend* muss ohne führende Leerzeichen als einziges Wort in einer Zeile stehen. Bei der Adressangabe 1,$ muss die Sonderbedeutung von $ mit \ aufgehoben werden, da die Shell sonst das nachfolgende *s* als Name einer Shell-Variablen interpretieren würde.

Beispiel 3 Im folgenden Beispiel werden einige *ed*-Kommandos vorgeführt und erläutert:

```
$ ed                  - Aufruf
P                     - Bereitzeichen * im Kommandomodus ausgeben
*a                    - Anfügen an aktuelle Zeile, hier Dateianfang
zeile1                - Eingabetext
zeile2
zeile3
.                     - Eingabemodus beenden
*1,$p                 - Zeile 1 bis letzte Zeile ausgeben
zeile1
zeile2
zeile3
*p                    - Aktuelle (= zuletzt bearbeitete) Zeile ausgeben
zeile3
*n                    - Aktuelle Zeile mit Zeilennummer ausgeben
3     zeile3
*1,$n                 - Zeile 1 bis letzte Zeile mit Zeilennummern ausgeben
1     zeile1
2     zeile2
3     zeile3
*2c                   - Zeile 2 ersetzen durch nachfolgende Eingabe
                        mit Tabulatorzeichen
zeile2 ⇥ zeile2 ⇥ zeile2
noch eine zeile2
.                     - Eingabemodus beenden
*p                    - Aktuelle Zeile ausgeben
noch eine zeile2
*1,$n                 - Zeile 1 bis letzte Zeile mit Zeilennummern ausgeben
1     zeile1
2     zeile2  zeile2  zeile2
3     noch eine zeile2
4     zeile3
*4s/3/4               - In Zeile 4 Zeichen 3 suchen und ersetzen durch 4
zeile4
*n                    - Aktuelle Zeile mit Zeilennummer ausgeben
4     zeile4

*1,$s/z/Z/g           - Von Zeile 1 bis letzte Zeile alle Zeichen z
                        suchen und durch Z ersetzen
*1,$p                 - Zeile 1 bis letzte Zeile ausgeben
Zeile1
Zeile2  Zeile2  Zeile2
noch eine Zeile2
Zeile4
*2l                   - Zeile 2 mit nicht druckbaren Zeichen als
                        Mnemoniks ausgeben.
Zeile2>Zeile2>Zeile2
```

```
*
*2r datei          - Hinter Zeile 2 Inhalt von datei in Puffer einlesen
57                   und Anzahl der eingelesenen Zeichen ausgeben.
                     datei muss hierfür existieren!
*1,$n              - Zeile 1 bis letzte Zeile mit Zeilennummern ausgeben
1      Zeile1
2      Zeile2  Zeile2  Zeile2  - Diese und die zwei folgenden Zeilen
3      zeile1 aus datei          kommen aus datei
4      zeile2 aus datei
5      letzte zeile aus datei
6      noch eine Zeile2
7      Zeile4

*6,7c              - Zeile 6 bis 7 ersetzen durch nachfolgende Eingabe
allerletzte Zeile
.                  - Eingabemodus beenden
*1,$n              - Zeile 1 bis letzte Zeile mit Zeilennummern ausgeben
1      Zeile1
2      Zeile2  Zeile2  Zeile2
3      zeile1 aus datei
4      zeile2 aus datei
5      letzte zeile aus datei
6      allerletzte Zeile
*q                 - Versuch, Editor mit q zu verlassen
?                  - Warnung
*h                 - Fragezeichen erklären
                   - Erklärung: erwartetes Kommando w
warning:expecting 'w'
                     zum Sichern des Pufferinhalts wurde nicht
                     eingegeben
*w bsp             - Sichern des Pufferinhalts in der Datei bsp
103                - Ausgabe der Anzahl der gesicherten Zeichen
*q                 - Editor verlassen
$                  - Bereitzeichen der Shell
```

Siehe auch *ex*, *grep*, *sed*, *sh*, *stty*, *umask*, *vi*
Tabellen und Verzeichnisse, Reguläre Ausdrücke

edt BS2000-Dateibearbeiter EDT aufrufen *(BS2000)* (screen oriented editor)

edt ruft den BS2000-Dateibearbeiter EDT auf, siehe „EDT (BS2000/OSD) Anweisungen“ [7]. EDT ab V17.0 wird im Kompatibilitätsmodus aufgerufen. *edt* ist der Dateibearbeiter für Blockterminals. Für den Aufruf des EDT im Unicode-Modus steht das Kommando *edtu* zur Verfügung. Für Zeichenterminals stehen z.B. der *ex* oder *vi* zur Verfügung.
Mit diesem Kommando und der EDT-Funktionalität können Sie auch Dateien vom POSIX-Dateisystem in das BS2000 kopieren und umgekehrt.

Syntax

edt[␣**-k**][␣**-i**index]␣datei

-k Vor dem Editieren wird der Inhalt der Datei von ASCII nach EBCDIC umcodiert. Vor dem Zurückschreiben wird der Inhalt der Datei wieder nach ASCII zurückkonvertiert. Mit dieser Option kann eine in ASCII codierte Datei mit *edt* bearbeitet werden.

-iindex
Diese Option wird benötigt, wenn Dateien mit mehr als 9999 Zeilen eingelesen werden sollen.
Das Produkt <index> * 0,0001 ergibt die Nummer der ersten Zeile und die Schrittweite der Zeilennummerierung. Fehlt die Angabe zu <index>, so ist <index>=10000, d.h. die Nummer der ersten Zeile und die Schrittweite sind 1. Im EDT selbst kann mit der EDT-Anweisung *renumber* renummerirt werden.

datei
Dateiname der POSIX-Datei. Es darf nur eine Datei angegeben werden. Existiert die angegebene Datei noch nicht, wird die leere Arbeitsdatei 0 eröffnet.

Hinweis

Der EDT kann POSIX-Dateien erzeugen, einlesen, kopieren, zurückschreiben und schließen. Dazu stehen die Anweisungen *@XOPEN*, *@XCOPY*, *@XWRITE* und *@CLOSE* zur verfügung (siehe Handbuch „EDT (BS2000/OSD) Anweisungen“ [7]).

Das Rückschreiben der geöffneten POSIX-Datei in das BS2000-Dateisystem erfolgt von der EDT-Ebene 0 aus mit der EDT-Anweisung *@WRITE'dateiname'*.

EDT wird mit *@HALT* (Rückschreiben der geöffneten POSIX-Datei in das POSIX-Dateisystem mit Dialog) oder *@RETURN* (unbedingtes Rückschreiben der geöffneten POSIX-Datei in das POSIX-Dateisystem) beendet. In der EDT-Ebene 0 wird mit *@WRITE* eine geöffnete UFS-Datei (in das UFS) zurückgeschrieben.

Leere Zeilen in der Datei werden als Zeilen mit der Länge 1 (Zeichen X'0D') dargestellt. Beim Rückschreiben der Datei werden diese Zeilen wieder in Leerzeilen umgewandelt.

Das Tabulatorzeichen ('\t') wird nicht in die entsprechende Anzahl von Leerzeichen umgewandelt.

Die Verwendung des Kommandos *edt* ist nicht in Verbindung mit einer Pipe erlaubt.

Fehler

`edt cannot be used within a pipe`
Das *edt*-Kommando wurde fälschlicherweise in Verbindung mit einer Pipe verwendet.

`edt cannot be used within a forked process`
Das *edt*-Kommando wurde in einer Subshell eingegeben. Dies ist nicht erlaubt.

`edt: file` *datei* `open for read failed`
Sie haben kein Leserecht für *datei*.

`*** read only ***` in der edt-Meldungszeile
Die zu bearbeitende Datei ist schreibgeschützt.

`edt: file` *datei* `open for write failed, permission denied`
Fehlendes Schreibrecht.
Anschließend folgen die Meldungen:
```
edt: edited file(s) not saved!
edt: terminate edt? reply (y=yes; n=no)?
```

`edt: file` *datei* `open for write failed, no such file or directory`
Nicht existierender Pfad.
Anschließend folgen die Meldungen:
```
edt: edited file(s) not saved!
edt: terminate edt? reply (y=yes; n=no)?
```

`edt: no such file or directory`
nicht existierende Datei oder Dateiverzeichnis.

`edt: file` *datei* `write: UFS file system failed, no space`
Fehlender Speicherplatz im POSIX-Dateisystem für das Schreiben der Datei.
Anschließend folgen die Meldungen:
```
edt: edited file(s) not saved!
edt: terminate edt? reply (y=yes; n=no)?
```

Wenn die Datei aus oben genannten Gründen nicht zurückgeschrieben werden kann, kann der Anwender im EDT bleiben und Aktionen zum Retten der Datei setzen, wie z.B. Schreibrecht ändern oder Retten der Datei in das BS2000 durch *@WRITE'datei'*.

Beispiel

Die POSIX-Datei */usr/home/file.pos* soll als BS2000-Datei *file.bs2* im BS2000-Dateisystem abgelegt werden.

```
$ edt /usr/home/file.pos

.....
@WRITE 'file.bs2'   (EDT-Anweisung)
.....
```

edtu BS2000-Dateibearbeiter EDT im Unicode-Modus aufrufen *(BS2000)* (screen oriented editor)

edtu ruft den BS2000-Dateibearbeiter EDT im Unicode-Modus auf, siehe „EDT (BS2000/OSD) Unicode-Modus Anweisungen“ [8]. Damit steht im Gegensatz zum Kommando *edt*, das den EDT im Kompatibilitätsmodus aufruft, der vollständige Funktionsumfang des EDT ab V17.0 zur Verfügung.

Wichtige Funktionserweiterungen gegenüber dem Kompatibilitätsmodus sind:

- Unterstützung von XHCS-Zeichensätzen inklusive Unicode
- Bearbeitung von Dateien mit Satzlängen bis 32768 Bytes
- Verfügbarkeit aller 22 Arbeitsdateien auch im F-Modus
- neue Bildschirmformate (@VDT F3..F4)
- Unterstützung von POSIX-Dateien in den EDT-Anweisungen @COPY, @OPEN und @WRITE (anstelle der EDT-V16-Anweisungen @XCOPY, @XOPEN und @XWRITE)

So wie *edt* steht auch *edtu* nur an BS2000-Block-Terminals und nicht an Zeichenterminals (*rlogin*, *telnet*, *ssh*) zur Verfügung.

Im Gegensatz zu *edt* kann *edtu* jedoch auch in Sub-Shells aufgerufen werden. Das gilt allerdings nicht für Shells, in denen die notwendige SYSFILE-Umgebung nicht initialisiert ist, weil sie z.B. in einer *su*-Sitzung erzeugt wurden.

Syntax

edtu[␣**-hrl**][**-i** n] [**-c** ccsn | **-k**] [**-v** vdt] [pfadname] ...

-h Hilfe. Die Syntax des Kommando *edtu* wird in englischer Sprache ausgegeben.

-r Alle Dateien werden nur zum Lesen geöffnet. Beim Beenden von edtu werden alle geöffneten Dateien ohne Rückfrage verworfen.

-l Informationszeilen werden angezeigt (entspricht @PAR GLOBAL,INFO=ON)..

-i n
Die Zeilen werden mit der Schrittweite <n> * 0,0001 nummeriert (wird nur noch aus Kompatibilitätsgründen unterstützt). Standardmäßig erfolgt die Zeilennummerierung automatisch.

-c ccsn
Die mit *pfadname* spezifizierte(n) Datei(en) liegt/liegen im Zeichensatz *ccsn* vor (Standard: EDF041).

-k Die mit *pfadname* spezifizierte(n) Datei(en) liegt/liegen im ISO-Zeichensatz vor (entspricht: `-c ISO88591`).

-v vdt
EDT wird mit Bildschirmformat *vdt* (F1..F4, Standard: F1) gestartet (entspricht @VDT *vdt)*.

pfadname
Die UFS-Datei *pfadname* wird zum Bearbeiten geöffnet. Bis zu 22 Dateien können in den Arbeitsdateien 0 bis 21 geöffnet werden.

Pipe-Ausgaben (z.B. `ls | edtu`) werden immer in Arbeitsdatei 0 eingelesen und beim Beenden von *edtu* immer verworfen

Hinweis *edtu* schreibt ein Inhaltsverzeichnis aller geöffneten Dateien in die Arbeitsdatei 22. Der EDT-Dialog beginnt in dieser Arbeitsdatei, falls mehrere Dateien geöffnet wurden. Wenn nur eine Datei geöffnet wurde, beginnt der Dialog in Arbeitsdatei 0.

EDT kann auf folgende Arten beendet werden:

– *@HALT* oder *@END*
Speichern geänderter Dateien nur nach Rückfrage

– *@HALT FORGET*
Verwerfen geänderter Dateien ohne Rückfrage

– *@RETURN*
Speichern geänderter Dateien ohne Rückfrage

Beim Beenden des EDT werden folgende Meldungen ausgegeben, falls die Arbeitsdatei *nummer* geändert wurde und der EDT nicht mit *@RETURN* beendet wurde:

`workfile` *nummer datei* `not saved (readonly file); return to EDT dialog? (y=yes, n=no):`
Die in der Arbeitsdatei *nummer* geöffnete Datei *datei* konnte nicht zurück geschrieben werden, da sie schreibgeschützt ist

`workfile` *nummer*`: ufs file` *datei* `not saved; ...`
Die in der Arbeitsdatei *nummer* geöffnete POSIX-Datei *datei* wurde noch nicht zurück geschrieben

`workfile` *nummer*`: element` *element* `in lib` *bibliothek* `not saved; ...`
Das in der Arbeitsdatei *nummer* geöffnete Bibliothekselement *element* aus *bibliothek* wurde noch nicht zurück geschrieben

`workfile` *nummer*`: BS2000 file '`*datei*`' not saved; ...`
Die in der Arbeitsdatei *nummer* geöffnete BS2000-Datei *datei* wurde noch nicht zurück geschrieben

Abhängig von der Art der Beendigung wird diesen Meldungen noch einer der folgenden Texte angefügt:

– bei *@HALT FORGET*

`(discarded because of @HALT FORGET)`
Die Änderungen wurden wegen der Beendigung mit *@HALT FORGET* verworfen

– bei *@HALT oder @END*

`save it? (y=yes, n=no, r=return):`
Abfrage, ob die Datei zurück geschrieben oder zum EDT zurückgekehrt werden soll.

- Wenn die geänderte Arbeitsdatei *nummer* zurück geschrieben wurde, wird eine der folgenden Meldungen ausgegeben:

`workfile` *nummer*`: ` *datei* `saved`
Die in der Arbeitsdatei *nummer* geöffnete Datei *datei* wurde zurück geschrieben

`workfile` *nummer*`: ufs file` *datei* `saved`
Die in der Arbeitsdatei *nummer* geöffnete POSIX-Datei *datei* wurde zurück geschrieben

`workfile` *nummer*`: element %s in lib %s saved`
Das in der Arbeitsdatei *nummer* geöffnete Bibliothekselement *element* aus *bibliothek* wurde zurück geschrieben

`workfile` *nummer*`: BS2000 file` *datei* `saved`
Die in der Arbeitsdatei *nummer* geöffnete BS2000-Datei *datei* wurde zurück geschrieben

Die Optionen *-k* und *-c* haben nur Einfluss auf geöffnete POSIX-Dateien und nicht auf nachträglich im EDT-Dialog geöffnete BS2000-Dateien. Falls keine der Optionen *-k* und *-c* angegeben ist, wird auch die automatische Konvertierung von POSIX-Dateien auf ASCII-Dateisystemen mit Hilfe der Umgebungsvariablen *IO_CONVERSION* unterstützt.

Das Tabulatorzeichen ('\t') wird nicht in die entsprechende Anzahl von Leerzeichen umgewandelt.

Im Gegensatz zu *edt* kann *edtu* auch von einer Pipe lesen:

```
kommando | edtu [optionen] [datei ...]
```

Dabei wird die Ausgabe der Pipe in die Arbeitsdatei 0 eingelesen und diese mit dem Read-Only-Attribut gekennzeichnet, d.h. der Anwender muss ggf. selbst für das Speichern der Arbeitsdatei 0 sorgen. Eventuell angegebene Dateien werden in die Arbeitsdateien 1 und weitere eingelesen.

Fehler

`edtu can run on BLOCK terminals (/dev/term/*) only`
Das *edtu*-Kommando wurde an einem Zeichenterminal (*rlogin*, *telnet*, *ssh*) eingegeben.

`edtu cannot be used within this forked process`
Das *edtu*-Kommando wurde in einer Subshell eingegeben, in der die notwendige SYSFILE-Umgebung nicht initialisiert ist. Dies ist nicht erlaubt.

`file` *datei* `does not exist`
Die Datei *datei* ist nicht vorhanden.

`new file` *datei* `may not be created`
Die Datei *datei* kann nicht erzeugt werden.

`open failed; file` *datei* `is a directory`
Bei *datei* handelt es sich um ein Verzeichnis.

`open failed; no read access for file` *datei*
Sie haben kein Leserecht für *datei*.

`EDT reports error for cmd '`*kommando*`'`
EDT hat für das Kommando *kommando* einen Fehler gemeldet.

egrep Muster suchen (search a file with an ERE pattern)

egrep liest Zeilen aus einer oder mehreren Dateien oder von der Standard-Eingabe und vergleicht die Zeilen mit den angegebenen Mustern. Ist mittels Optionen nichts anderes angegeben, schreibt *egrep* alle Zeilen, die zu einem der Muster passen, auf die Standard-Ausgabe. Als Muster können Sie erweiterte reguläre Ausdrücke angeben (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Geben Sie mehrere Eingabedateien an, dann wird jeder Ausgabezeile der Name der betreffenden Datei vorangestellt.

Syntax

Format 1: egrep[␣**-bchilnvy**]␣**-e**␣musterliste[␣datei...]

Format 2: egrep[␣**-bchilnvy**]␣**-f**␣musterdatei[␣datei...]

Format 3: egrep[␣**-bchilnvy**]␣musterliste[␣datei...]

Die Formate werden gemeinsam beschrieben, da die Muster, nach denen *egrep* die Eingabezeilen vergleichen soll, entweder über *musterliste* oder über die Option *-e musterliste* oder *-f musterliste* angegeben werden. Eines dieser Argumente müssen Sie angeben, zwei zusammen oder alle drei sind nicht erlaubt.

Keine Option angegeben
: *egrep* gibt alle Zeilen aus, die zu mindestens einem der in *musterliste* angegebenen Muster passen. Geben Sie mehrere Eingabedateien an, dann wird jeder Ausgabezeile der Name der Datei vorangestellt, aus der die Zeile gelesen wurde.

option

-b (b - block) Jeder Ausgabezeile wird die Nummer des Blocks vorangestellt, in dem sie enthalten ist. Die Blöcke, aus denen eine Datei besteht, sind je 512 byte groß und werden, mit 0 beginnend, durchnummerirt.
Option *-b* kann hilfreich sein, wenn die Nummern von Blöcken nach dem Kontext ermittelt werden sollen (siehe *od*, Argument *offset*).

-c (c - count) *egrep* gibt nur die Anzahl der gefundenen Zeilen aus (das sind die Zeilen, die *egrep* ohne die Option *-c* ausgeben würde, siehe *Beispiel 4*); die Zeilen selbst werden nicht ausgegeben.

-h (h - hidden) Beim Durchsuchen mehrerer Eingabedateien unterlässt *egrep* die Voranstellung der Dateinamen vor jeder Ausgabezeile.

-i oder **-y**
: (i - ignore) *egrep* unterscheidet beim Vergleich nicht zwischen Groß- und Kleinbuchstaben.

-l (l - list) *egrep* gibt nur die Namen der Dateien aus, die mindestens eine der gefundenen Zeilen enthalten. (Das sind die Zeilen, die *grep* ohne die Option *-l* ausgeben würde, siehe *Beispiel 5*). Jeder Dateiname wird nur einmal ausgegeben. Die Zeilen selbst gibt *egrep* nicht aus.

-n (n - number lines) Jeder Ausgabezeile wird die Zeilennummer aus der betreffenden Eingabedatei vorangestellt, wobei von 1 an nummerirt wird. Liest *egrep* von der Standard-Eingabe, bezieht sich die Zeilennummer auf die Standard-Eingabe.

-v (v - vice versa) *egrep* gibt alle Zeilen aus, die zu *keinem* der angegebenen Muster passen.

Zusammen mit Option *-c*:
egrep gibt nur die Anzahl solcher Zeilen aus.

Zusammen mit Option *-l*:
egrep gibt nur die Namen der Dateien aus, die solche Zeilen enthalten.

-e␣musterliste
(e - expression) Diese Option brauchen Sie, wenn der erste Ausdruck in *musterliste* mit einem Bindestrich - beginnt. Zusammen mit *-e* wird eine solche Musterliste nicht als Option interpretiert, sondern als Liste von Mustern, mit denen *egrep* die Eingabezeilen vergleichen soll.

-f␣musterdatei
(f - file) *egrep* liest die Musterliste aus der Datei *musterdatei*. Jede Zeile von *musterdatei* wird als ein erweiterter regulärer Ausdruck interpretiert.

musterliste
Liste von erweiterten regulären Ausdrücken, mit denen *egrep* die Eingabezeilen vergleichen soll (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Die regulären Ausdrücke trennen Sie durch Neue-Zeile-Zeichen. Ein Neue-Zeile-Zeichen in *musterliste* wird wie ein senkrechter Strich | (Zeichen für die Alternative) in einem erweiterten regulären Ausdruck interpretiert.
Reguläre Ausdrücke der Form (r|s) können Sie auch ohne Klammern angeben: r|s (siehe *Beispiel 1*).
Enthält *musterliste* Neue-Zeile-Zeichen oder sonstige Zeichen, die für die Shell eine Sonderbedeutung haben, dann schließen Sie *musterliste* in Hochkommas ein: '*musterliste*'.

Beginnt der erste Ausdruck in *musterliste* mit einem Bindestrich -, dann müssen Sie *musterliste* mit Option *-e* angeben oder die Optionsliste mit -- beenden, da sonst *musterliste* selbst als Option interpretiert wird.

datei
Name der Datei, die *egrep* durchsuchen soll. Pro Aufruf können Sie mehrere Dateinamen angeben.

datei nicht angegeben:
egrep liest die Eingabezeilen von der Standard-Eingabe.

grep, fgrep und egrep

Die Kommandos *grep*, *fgrep* und *egrep* sind von der Oberfläche her weitgehend identisch. Im Folgenden sind die wichtigsten Unterschiede zwischen diesen Kommandos aufgeführt.

grep verarbeitet einfache reguläre Ausdrücke. Pro Aufruf können Sie nur einen einzigen regulären Ausdruck angeben.

fgrep verarbeitet nur Zeichenketten. Pro Aufruf können Sie jedoch mehrere Zeichenketten angeben: Die Zeichenketten geben Sie entweder direkt in der Aufrufzeile, getrennt durch Neue-Zeile-Zeichen, an oder Sie übergeben sie in einer Datei.
fgrep kann effizient sehr viele Zeichenketten suchen: *fgrep* sucht jede einzelne Zeile nach allen Zeichenketten ab.

egrep verarbeitet erweiterte reguläre Ausdrücke. Diese umfassen u.a. die einfachen regulären Ausdrücke bis auf eine Ausnahme: Der einfache reguläre Ausdruck \(*regausdruck*\) hat bei erweiterten regulären Ausdrücken keine Sonderbedeutung und wird deshalb auch nicht von *egrep* verarbeitet.
Pro Aufruf können Sie mehrere reguläre Ausdrücke, durch Neue-Zeile-Zeichen getrennt, angeben. *egrep* interpretiert diese Neue-Zeile-Zeichen wie einen senkrechten Strich (Zeichen für die Alternative, siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*) Die regulären Ausdrücke geben Sie entweder direkt in der Aufrufzeile an, oder Sie übergeben sie in einer Datei.

Endestatus

0 Zeilen gefunden

1 keine Zeile gefunden

>1 Syntaxfehler oder „Datei kann nicht geöffnet werden“. Dieser Endestatus gilt auch dann, wenn in anderen Eingabedateien Zeilen gefunden wurden

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *egrep*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_COLLATE beeinflusst die Sortierreihenfolge.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Grundlage für alle Beispiele sind die Dateien *kunden1* und *kunden2*. Sie haben folgenden Inhalt:

kunden1:

```
080685    999.98  20 LE  Art.  038   Fa. Holzinger
120387   1240.25   3 LE  Art.  023   Fa. Wanninger
180588    330.87   1 LE  Art.  332   Fa. Wanninger
```

kunden2:

```
hinterhuber berta, rosenheim, zugspitzstr.1
wanninger herbert, muenchen, kirschstr.3
```

Beispiel 1 Zeilen ausgeben, die zu einem Muster passen (keine Option und Option *-i*):

```
$ egrep Wanninger kunden1 kunden2
kunden1:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180588    330.87   1 LE  Art.  332   Fa. Wanninger
```

Wenn Sie auch Zeilen mit kleingeschriebenem *wanninger* ausgeben lassen möchten, geben Sie ein:

```
$ egrep –i wanninger kunden1 kunden2
kunden1:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180588    330.87   1 LE  Art.  332   Fa. Wanninger
kunden2:wanninger herbert, muenchen, kirschstr.3
```

Kompliziertere Muster stellen Sie mit Hilfe von regulären Ausdrücken dar, z.B.:
Zeilen ausgeben, die die Zeichenkette *Holzinger* oder *Wanninger* enthalten:

```
$ egrep '(Holz|Wann)inger' kunden1 kunden2
kunden1:080685    999.98  20 LE  Art.  038   Fa. Holzinger
kunden1:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180588    330.87   1 LE  Art.  332   Fa. Wanninger
```

Statt des regulären Ausdrucks (Holz|Wann)inger können Sie auch den folgenden regulären Ausdruck angeben:

```
(Holzinger|Wanninger)
```

Hier können Sie die Klammern weglassen:

```
Holzinger|Wanninger
```

Statt des senkrechten Strichs im letzten Ausdruck Holzinger|Wanninger können Sie auch ein Neue-Zeile-Zeichen angeben (siehe *Beispiel 2*).

Beispiel 2 Mehrere Muster angeben (keine Option und Option *-f*):

```
$ egrep '^1
> 1$' kunden1 kunden2
kunden1:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180588    330.87   1 LE  Art.  332   Fa. Wanninger
kunden2:hinterhuber berta, rosenheim, zugspitzstr.1
```

Sie können die beiden Muster auch in eine Datei *namen* schreiben (je Muster eine Zeile) und *egrep* folgendermaßen aufrufen:

```
$ egrep -f namen kunden1 kunden2
```

Dasselbe Ergebnis erhalten Sie, wenn Sie das Neue-Zeile-Zeichen, das die Muster ^1 und 1$ trennt, durch einen senkrechten Strich ersetzen:

```
$ egrep '^1 | 1$' kunden1 kunden2
```

Beispiel 3 Zeilen ausgeben, die zu *keinem* der angegebenen Muster passen (Option *-v*):

```
$ egrep -v '^1
> 1$' kunden1 kunden2
kunden1:080685    999.98  20 LE  Art.  038   Fa. Holzinger
kunden2:wanninger herbert, muenchen, kirschstr.3
```

Mit obigem Aufruf erhalten Sie also alle Zeilen, die weder mit 1 beginnen noch mit 1 enden. Dasselbe Ergebnis erhalten Sie mit folgendem Aufruf (siehe *Beispiel 2*):

```
$ egrep -v '^1 | 1$' kunden1 kunden2
```

Beispiel 4 Anzahl der gefundenen Zeilen ausgeben (Option *-c*):

Zuerst soll für jede Eingabedatei die Anzahl der Zeilen, die mit einer 1 beginnen, ausgegeben werden.

```
$ egrep -c '^1' kunden1 kunden2
kunden1:2
kunden2:0
```

Nun soll die Anzahl der Zeilen, die *nicht* mit einer 1 beginnen, ausgegeben werden.

```
$ egrep -c -v '^1' kunden1 kunden2
kunden1:1
kunden2:2
```

Beispiel 5 Nur Dateinamen ausgeben (Option *-l*):

Zuerst sollen die Namen der Dateien, die Zeilen mit einer 1 am Anfang enthalten, ausgegeben werden.

```
$ egrep -l '^1' kunden1 kunden2
kunden1
```

Nun sollen die Namen der Dateien, die Zeilen ohne 1 am Anfang enthalten, ausgegeben werden.

```
$ egrep -l -v '^1' kunden1 kunden2
kunden1
kunden2
```

Beispiel 6 Gefundene Zeilen mit Zeilennummer ausgeben (Option *-n*):

```
$ egrep -n -i wanninger kunden1 kunden2
kunden1:2:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:3:180588    330.87   1 LE  Art.  332   Fa. Wanninger
kunden2:2:wanninger herbert, muenchen, kirschstr.3
```

Siehe auch *ed*, *fgrep*, *grep, sed*

env Umgebung bei Ausführung von Kommandos ändern (set environment for command execution)

Mit *env* können Sie sich die aktuellen Umgebungsvariablen und ihre Werte ausgeben lassen oder sie für ein Kommando verändern. *env* liest die aktuelle Umgebung ein, ändert sie entsprechend der Angabe *name=wert* und führt das Kommando dann in der veränderten Umgebung aus. Die schon vorhandenen Angaben für *name* und *wert* werden durch die neuen Angaben überschrieben und vor Ausführung des Kommandos der usprünglichen Umgebung hinzugefügt. Die neuen Angaben bilden zusammen mit den unveränderten Umgebungsvariablen die für die Ausführung von *kommando* gültige Umgebung.

Wenn kein Kommando angegeben ist, wird die durch *env* veränderte Umgebung ausgegeben.

Syntax

env[␣**-i**|␣**-**][␣name=wert]...[␣kommando[␣arg...]]

–i|␣**-**
: die ursprüngliche Umgebung wird ignoriert; *kommando* wird dann exakt in der angegebenen Umgebung ausgeführt.

  Die Option *-i* entspricht der alten Option -, die weiterhin unterstützt wird.

name**=**wert
: *name* ist der Name einer Variablen, die für *kommando* Gültigkeit haben soll.

  wert ist der Wert von *name*, der für *kommando* Gültigkeit haben soll.

kommando
: Name des Kommandos oder der Shell-Prozedur, die Sie unter der definierten Umgebung ausführen lassen möchten.

arg
: Argument, z.B. Stellungs- oder Kennwortparameter, das Sie an *kommando* übergeben können.

Endestatus

0	*env* wurde erfolgreich ausgeführt
1-125	Fehler
126	Das angegebene *kommando* existiert, kann aber nicht aufgerufen werden.
127	Das angegebene *kommando* ist nicht auffindbar.

Variable *PATH*

Bestimmt die Lage von *kommando*. Wird *PATH* als *name=value* Operand *env* zugewiesen, wird *value* für die Suche nach *kommando* verwendet

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *env*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Ausgabe der aktuellen Werte der Umgebungsvariablen:

```
$ env
_=/usr/bin/env
TTY=/dev/term/004
IO_CONVERSION=NO
PATH=/usr/bin:/opt/bin::/etc:/usr/sbin
TERM=BLOCK
HZ=100
LOGNAME=TESTUSER
PROGRAM_ENVIRONMENT=SHELL
HOME=/TESTUSER
TZ=MEZ-1MSZ-2,M3.5.0/02:00:00,M9.5.0/03:00:00
MAIL=/var/mail/TESTUSER
SHELL=/sbin/sh
USER=TESTUSER
LANG=
```

Beispiel 2 Ausgabe der geänderten Werte der Umgebungsvariablen:

```
$ env PATH=$HOME/proz
_=/usr/bin/env
TTY=/dev/term/004
IO_CONVERSION=NO
PATH=/TESTUSER/proz
TERM=BLOCK
HZ=100
LOGNAME=TESTUSER
PROGRAM_ENVIRONMENT=SHELL
HOME=/TESTUSER
TZ=MEZ-1MSZ-2,M3.5.0/02:00:00,M9.5.0/03:00:00
MAIL=/var/mail/TESTUSER
SHELL=/sbin/sh
USER=TESTUSER
LANG=
```

Die Umgebungsvariable *PATH* wurde geändert.

Beispiel 3 Ausgabe der geänderten Umgebungsvariablen mit der Option -:

```
$ env - PATH=$HOME/proz
PATH=/TESTUSER/proz
```

Die ursprüngliche Umgebung wird ignoriert.

Beispiel 4 Aufruf der Datei *fly*, die sich in */TESTUSER/sprueche*, also in einem Unterdateiverzeichnis des HOME-Dateiverzeichnisses befindet.

Inhalt der Datei *fly*:

```
echo "Wenn hinter $1 $1 $2, $2 $1 $1 nach !"
```

fly wird nun von einer beliebigen Stelle in Ihrem Dateibaum aus aufgerufen, hier mit den Argumenten *Fliegen* und *fliegen*.

```
$ env PATH=$HOME/sprueche fly Fliegen fliegen
Wenn hinter Fliegen Fliegen fliegen, fliegen Fliegen Fliegen nach!
```

Mit der neuen Variablendefinition für *PATH* legen Sie fest, wo das eingegebene Kommando, in diesem Fall die Datei *fly*, gesucht werden soll: in einem Unterdateiverzeichnis des *HOME*-Dateiverzeichnisses, das Sie zusammen mit dem Wert der Variablen *HOME ($HOME)* angeben.
Als Argumente übergeben Sie an die Stellungsparameter $1 und $2 die Zeichenketten *Fliegen* und *fliegen*.

Der Inhalt der Datei *fly* wird nur deshalb korrekt ausgeführt, weil das Kommando *echo* ein eingebautes *sh*-Kommando ist. Alle POSIX-Kommandos, die in */usr/bin*, */usr/sbin* oder */opt/bin* stehen, können durch die Veränderung der Variablen *PATH* nicht mehr gefunden werden. Damit die POSIX-Kommandos weiterhin ausgeführt werden, muss die Variable *PATH* wie im folgenden Beispiel geändert werden.

Beispiel 5 Aufruf der Datei *loesch*, die sich im Dateiverzeichnis */TESTUSER/proz* befindet. In dieser Datei steht eine Prozedur, die zwei Dateien vergleicht und die eine löscht, wenn die Dateien gleich sind.

Inhalt der Datei *loesch*:

```
if cmp -s $1 $2
then
rm $2
fi
```

Aufruf von *loesch* von einer beliebigen Stelle in Ihrem Dateibaum aus mit den Argumenten *dat1* und *dat2*:

```
$ env PATH=$PATH:$HOME/proz loesch dat1 dat2
```

Hier wurde an den ursprünglichen Suchpfad der neue angefügt, damit sowohl die Prozedur *loesch* als auch die in der Prozedur enthaltenen POSIX-Kommandos ausgeführt werden können. Wenn nur der Suchpfad für *loesch* angegeben wird, wird folgender Fehler gemeldet:

```
/TESTUSER/proz/loesch: cmp: not found
```

Siehe auch *sh, set, exec*
profile, environ [13]

eval Aufruf-Argumente bearbeiten und als Kommando ausführen (construct command by concatenating arguments)

Mit dem in die POSIX-Shell *sh* eingebauten Kommando *eval* können Sie der Shell *sh* die angegebenen Aufruf-Argumente als Kommando übergeben. Die Shell führt dieses Kommando aus.

Auf diese Weise werden die Aufruf-Argumente zweimal von der Shell bearbeitet:
– das erste Mal, wenn die Shell die *eval*-Kommandozeile bearbeitet.
– das zweite Mal, wenn die Shell die bearbeiteten Aufruf-Argumente als Kommando ausführt. Die Shell bearbeitet jede Kommando-Zeile vor der Ausführung.

Wann brauchen Sie eval?

Die Shell bearbeitet jede Kommando-Zeile in mehreren Schritten (siehe *Ausführung* in Kapitel 2). Bei jedem Bearbeitungsschritt interpretiert die Shell bestimmte Sonderzeichen. Die ursprüngliche Kommando-Zeile kann sich durch die Bearbeitungsschritte verändern.

Wenn die Shell beispielsweise eine Variable durch ihren Wert oder ein Kommando durch seine Ausgabe ersetzt, können in der Kommando-Zeile Sonderzeichen auftreten, die die Shell in den restlichen Bearbeitungsschritten nicht mehr interpretiert. Das eingebaute *sh*-Kommando *eval* sorgt dafür, dass die Shell verbliebene Sonderzeichen im zweiten Durchgang interpretiert (siehe Beispiele).

Syntax

eval ␣[argument␣...]

argument
: beliebige Zeichenkette, die durch Leer- oder Tabulatorzeichen begrenzt wird. Das letzte Argument wird durch ein Kommando-Trennzeichen abgeschlossen.
Sie können beliebig viele Argumente angeben, jeweils getrennt durch mindestens ein Leer- oder Tabulatorzeichen.
Die Shell führt diese Argumente als Kommando aus.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *eval*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die folgende Kommando-Zeile wird nur mit *eval* wie gewünscht ausgeführt:

```
$ kom='date;echo name'
$ $kom
sh:date;echo: not found
$ eval $kom
Mon Mar 09 15:46:02 MEZ 2009
name
```

Die Shell ersetzt die Variable *kom* durch ihren Wert. Nach der Variablen-Ersetzung erkennt die Shell den Strichpunkt ; nicht mehr als Kommando-Trennzeichen. Bei Aufruf mit *eval* interpretiert die Shell den Strichpunkt ; wie gewünscht, weil sie das Aufruf-Argument $kom zweimal bearbeitet.

Beispiel 2 Die Shell-Prozedur *evaltest* soll zeigen, wie das Kommando *eval* von der Shell abgearbeitet wird. Sie hat folgenden Inhalt:

```
set -x
kennung=max
for i in 1 2 3 4
do
  eval gruppe$i=$kennung$i
  eval echo \$gruppe$i
done
```

set -x bewirkt, dass die Shell, die die Shell-Prozedur ausführt, die bearbeiteten Kommandos vor der Ausführung auf die Standard-Fehlerausgabe schreibt.
Wenn Sie diese Shell-Prozedur ausführen, erhalten Sie am Bildschirm folgende Ausgabe, allerdings ohne Zeilennummern:

```
   $ sh evaltest
1  + kennung=max
2  + eval gruppe1=max1
3  + gruppe1=max1
4  + eval echo $gruppe1
5  + echo max1
6  max1
:
```

Hier ist nur die Ausgabe nach dem ersten Schleifendurchlauf gezeigt.

Zeile 2 zeigt die Bearbeitung der ersten *eval*-Kommandozeile:
Die Shell hat $i durch den Wert 1 und $kennung durch den Wert max ersetzt. Der Aufruf mit *eval* ist nötig, weil die Shell im ersten Durchgang das Gleichheitszeichen nicht als Zeichen für Zuweisung erkennt. Die Shell führt eine Zuweisung nur aus, wenn der Variablen-Name, also die Zeichenkette vor dem Gleichheitszeichen = mit einem Buchstaben oder Unterstrich _ beginnt und nur Buchstaben, Ziffern und Unterstriche enthält. *gruppe$i* ist wegen des Dollarzeichens $ kein erlaubter Variablen-Name.

Zeile 3 zeigt die Ausführung der Zuweisung:
Bei der zweiten Bearbeitung der Aufruf-Argumente interpretiert die Shell das Gleichheitszeichen wie gewünscht, weil das Argument vor = ein erlaubter Name für eine Shell-Variable ist.

Zeile 4 zeigt die Bearbeitung der zweiten *eval*-Kommandozeile:
Die Shell hat den Stellungsparameter $i durch den Wert 1 ersetzt und das Entwertungszeichen \ vor $ entfernt. Hier ist *eval* nötig, damit die Shell die Variable *gruppe1* durch ihren Wert ersetzt.
Zeile 5 zeigt die Bearbeitung des Kommandos *echo*. Hier hat die Shell *$gruppe1* durch den Wert max1 ersetzt. Zeile 6 enthält die Ausgabe des Kommandos *echo*.

Siehe auch *set*

ex Zeilenorientierter Editor (command and display editor)

ex ist ein zeilenorientierter Texteditor.

Den *ex* (außer im Line-Mode) können nur die Benutzer verwenden, die sich über *rlogin* Zugang zur POSIX-Shell verschafft haben.

ex bietet Ihnen verschiedene Bearbeitungsmodi.

- Im *ex*-Eingabe-Modus können Sie direkt Text eingeben.
- Im *ex*-Kommando-Modus können Sie Kommandos eingeben, um zum Beispiel
  - die Schreibmarke zu positionieren
  - Textmuster mit regulären Ausdrücken zu suchen (und zu ersetzen)
  - in eine andere Datei zu wechseln
  - eine Shell aufzurufen.

Außerdem können Sie vom zeilenorientierten Editor *ex* in den bildschirmorientierten Editor *vi* wechseln.

Aufbau dieser Beschreibung

Nach der Beschreibung des Aufrufs von *ex* auf POSIX-Ebene finden Sie folgende Abschnitte:

Arbeitsweise des *ex*
- Modi des *ex*
- Editor-Puffer sichern und *ex* verlassen
- Voreinstellung
- Aktuelle und sekundäre Datei
- Reguläre Ausdrücke
- Ersetzungszeichenketten
- Puffer
- Fehler- und Signalbehandlung

ex-Kommandos
- Adressen
- Parameter
- Kommandos

ex-Optionen

Syntax **ex**[␣option]␣datei...

option

-s (s - silent) Alle interaktiven Ausgaben des Editors werden unterdrückt. Die Option *-s* setzen Sie, wenn *ex* seine Kommandos aus einem Kommandoskript lesen soll.

Die Option *-s* ersetzt die alte Option -.

-t␣markierung
(t - tag) Die Datei mit *markierung* wird von *ex* zum Editieren aufgerufen. *ex* positioniert dann die Zeile mit der Definition der Markierung in der Mitte des Bildschirms. Die Suchzeichenketten für die Definitionen müssen in der Datei *tags* im gleichen Verzeichnis enthalten sein. Diese Option wird z.B. von *C*-Programmierern dazu verwendet, den Editor beim Aufruf auf die Definition einer Funktion oder eines Makros zu positionieren. Die dafür benötigte *tags*-Datei muss dazu vorher mit dem Kommando *ctags* erzeugt worden sein.

-r␣datei
(r - recover) Stellt Ihre *ex*-Sitzung von *datei* wieder her, falls das System oder *ex* während der Sitzung abgestürzt ist.
Bei einem Absturz des Systems oder des *ex*-Editors werden die Änderungen, die nur im Editor-Puffer stehen, nicht in die Datei geschrieben. POSIX versucht jedoch den Inhalt des Puffers zu retten, indem eine Kopie des Pufferinhalts angelegt wird, sofern dies möglich ist. *datei* wird mit den Änderungen, die Sie vor dem Absturz gemacht haben, in den *vi*-Puffer geholt. Sie können nun die Datei weiter editieren oder die Änderungen in eine Datei schreiben.

-L Nach einem Absturz des Systems oder des *ex*-Editors wird eine Liste aller geretteten Dateien ausgegeben.

-w␣n
Setzt den Wert der ex-Option *window* (siehe Seite 371) auf *n* fest.

-R (R - read-only) *datei* wird nur zum Lesen geöffnet. Damit können Sie ein versehentliches Überschreiben von *datei* verhindern. Jedoch können Sie den Pufferinhalt im Readonly-Modus in eine Datei mit anderem Namen schreiben.

Achtung!
Die Datei kann mit dem *ex*-Kommando-Modus *w!* dennoch überschrieben werden.

-v (v - vi) Der Editor *vi* wird aufgerufen. Eine ausführliche Beschreibung des *vi* finden Sie in diesem Handbuch (siehe *vi*).

-c␣kommando

Positioniert beim Aufruf des *ex* auf eine bestimmte Zeile der zu editierenden Datei oder führt ein *ex*-Kommando aus. Wenn Sie auf eine Zeile positionieren, steht die gewünschte Zeile danach in der Mitte des Bildschirms. Wenn Sie ein *ex*-Kommando ausführen lassen, wird nach Ausführung des *ex*-Kommandos auf die letzte Zeile der Datei positioniert - falls das *ex*-Kommando keine Positionierung bewirkt (z.B. Suchen).

kommando nicht angegeben:
ex positioniert auf das Ende der Datei.

kommando angegeben:
kommando kann entweder eine Zeilenangabe (*n*) sein, ein Suchkommando (/*muster*) oder ein sonstiges Kommando des *ex* ("*ex-kommando*"). Es wird beim Aufruf des *ex* ausgeführt:

n — *n* ist eine ganze Zahl. *ex* positioniert auf die *n*-te Zeile der Datei.

/muster — *ex* positioniert auf die Zeile, die *muster* enthält. Falls *muster* Sonderzeichen enthält, müssen Sie die Sonderzeichen entwerten, damit die Shell diese nicht interpretiert.

’ex-kommando’ oder "ex-kommando"
ex-kommando kann ein beliebiges *ex*-Kommando sein. Das *ex*-Kommando muss in einfachen Hochkommas oder Anführungszeichen eingeschlossen werden, damit es von der Shell nicht interpretiert wird. Falls nicht schon durch das *ex*-Kommando positioniert wird, positioniert der *ex* auf die letzte Zeile der Datei.

Beispiel

Nach dem Aufruf des *ex* mit

```
ex –c /Dienstag termine
```

wird die Datei *termine* geöffnet und die Schreibmarke auf die erste Zeile der Datei positioniert, die das Wort *Dienstag* enthält (vgl. das Beispiel bei *vi*).

datei

Name der Datei, die Sie editieren möchten. Wenn Sie mehrere Dateien angeben, werden sie in der Reihenfolge bearbeitet, in der Sie diese angegeben haben. Mit dem *ex*-Kommando *n* wechseln Sie in die nächste Datei.

datei nicht angegeben:
ex öffnet einen leeren Editor-Puffer, in den Sie Text schreiben können. Erst beim Zurückschreiben des Pufferinhalts mit *w* (write) in eine Datei oder durch ein *f*-Kommando bestimmen Sie deren Namen.

Achtung!
Enthält Ihre Datei Nullbytes, dann werden diese von *ex* beim Lesen weggeworfen. Eine von *ex* geschriebene Datei enthält keine Nullbytes.

Arbeitsweise von ex

ex arbeitet immer mit einem Editor-Puffer. Bei Beginn einer *ex*-Sitzung wird eine Arbeitskopie der Datei, die Sie bearbeiten, im Editor-Puffer angelegt. Alle Ihre Änderungen werden zunächst im Editor-Puffer vorgenommen. Sie werden erst gesichert, wenn Sie den Pufferinhalt mit *w* (write) in die Originaldatei zurückschreiben. Danach können Sie *ex* mit *q* (quit) verlassen.
Wenn Sie *ex* verlassen wollen, ohne die Änderungen in die Orginaldatei zurückzuschreiben, müssen Sie *q!* eingeben, ohne ein vorheriges *w*!

Ist in den Eingabedateien das ASCII-Zeichen NUL (Nullbyte) (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*) enthalten, so wird es gelöscht; in den Ausgabedateien kann es daher nicht enthalten sein.

Modi des ex

ex bietet Ihnen zwei Modi zur Bearbeitung einer Datei:

– den *ex*-Kommando-Modus und

– den *ex*-Eingabe-Modus.

Zusätzlich können Sie von *ex* in den *vi* wechseln, in dem wieder verschiedene Arbeitsmodi zur Verfügung stehen (ausführlicher siehe *vi*, *Modi des vi*).

Nach dem Aufruf befindet *ex* sich im Kommando-Modus, den Sie am Bereitzeichen Doppelpunkt : am Bildschirm erkennen. Mit den Kommandos *a* (append), *i* (insert) und *c* (change) wechseln Sie in den Eingabe-Modus, in dem Sie Text im Puffer ergänzen und ändern können (siehe *ex*-Kommandos).

Im Eingabe-Modus werden alle folgenden Eingabezeichen, auch verschiedene nichtdruckbare Zeichen, in den Puffer geschrieben. Kommandos werden im Eingabe-Modus nicht als solche interpretiert. Verlassen können Sie den Eingabe-Modus, indem Sie in der ersten Spalte einer neuen Zeile einen Punkt . eingeben und danach ↵ drücken.

Editor-Puffer sichern und ex verlassen

Um den Editor-Puffer zu sichern oder den *ex* zu verlassen, muss sich der *ex* im *ex*-Kommandomodus befinden.

Mit folgendem Kommando sichern Sie den Inhalt Ihres Editor-Puffers in die editierte oder eine angegebene Datei:

w[␣datei]
(w - write) Der Inhalt des Editor-Puffers wird in die angegebene Datei gesichert.

datei nicht angegeben:
Der Inhalt des Editor-Puffers wird in die editierte Datei geschrieben.

Um den *ex* zu verlassen, haben Sie folgende Möglichkeiten:

q (q - quit) *ex* verlassen. Funktioniert nur, falls noch keine Änderungen am Editor-Puffer vorgenommen wurden oder der geänderte Editor-Puffer in eine Datei gesichert wurde.

q! (q - quit) *ex* verlassen; am Editor-Puffer vorgenommene Änderungen gehen verloren.

x *ex* verlassen und den geänderten Editor-Puffer in die editierte Datei schreiben.

wq[␣datei]
(wq - write and quit) *ex* verlassen und den geänderten Editor-Puffer in die angegebene Datei *datei* schreiben.

datei nicht angegeben:
Der Inhalt des Editor-Puffers wird in die editierte Datei geschrieben.

Achtung!
ex gibt keine Warnung aus, wenn Sie beim Verlassen des Editors noch Text in benannten Puffern haben, die Sie nicht verwendet haben. Der Text ist unwiederbringlich verloren.

Voreinstellung

Sie können den *ex* in begrenztem Rahmen an Ihre Bedürfnisse und Gewohnheiten anpassen. Dazu müssen Sie das *ex*-Kommando *se* (set) verwenden, mit dem Sie bestimmte Optionen setzen oder verändern können (siehe *ex*-Optionen). Diese Änderungen gelten in der aktuellen Sitzung. Wie Sie diese Änderungen dauerhaft machen können, ist beschrieben bei *vi, Voreinstellung des vi*.

Aktuelle und sekundäre Datei

Die Datei, bzw. deren Kopie, die im Augenblick editiert wird, wird als aktuelle Datei bezeichnet. Die sekundäre Datei ist die Datei, die zuletzt bei einem Editier-Kommando angegeben wurde. Falls die sekundäre Datei zur aktuellen Datei wurde, wird die vorhergehende aktuelle Datei zur sekundären Datei. Die aktuelle Datei können Sie mit dem Prozentzeichen %, die sekundäre Datei mit dem Nummernzeichen # angeben; in Dateinamen wird % durch den Namen der aktuellen Datei und # durch den Namen der sekundären Datei ersetzt.

Beispiel

```
w %.bak
```

sichert die aktuelle Datei in eine Datei mit dem gleichen Namen, aber der Endung *.bak*.

Reguläre Ausdrücke

Die Bedeutung von Sonderzeichen in regulären Ausdrücken hängt bei *ex* davon ab, ob die Option *magic* gesetzt ist (vgl. *ex*-Optionen).

magic gesetzt:

zeichen
: Ein einfaches Zeichen steht für sich selbst. Die folgenden Zeichen sind keine einfachen Zeichen, sondern Metazeichen:

  ^ (Dach) am Anfang eines Musters

  $ (Dollar-Zeichen) am Ende eines Musters

  * (Stern) überall außer am Anfang eines Musters

  . (Punkt) an beliebiger Stelle in einem Muster

  [(öffnende eckige Klammer) an beliebiger Stelle in einem Muster

  ~ (Tilde) an beliebiger Stelle in einem Muster

  Metazeichen haben eine besondere Bedeutung und müssen durch einen Gegenschrägstrich \ entwertet werden, wenn Sie ihre Sonderbedeutung verlieren sollen.

^ Am Anfang eines Musters steht ^ für den Zeilenanfang.

$ Am Ende eines Musters steht $ für das Zeilenende.

. Ein beliebiges Zeichen.

\<
\>

: Die Zeichen \< passen zum Beginn eines Wortes. Das Wort muss dabei mit einem Buchstaben, einer Ziffer oder einem Unterstreichungszeichen _ beginnen; vor dem Wort muss entweder der Zeilenanfang stehen oder ein Zeichen, das nicht zu den oben aufgeführten gehört. Die Zeichen \> passen zum Ende eines Wortes.

[zeichenkette]
: Ein beliebiges Zeichen aus *zeichenkette*, wobei *zeichenkette* eine nicht leere Folge von Zeichen ist. Innerhalb von *zeichenkette* gibt es folgende Sonderbedeutungen:
  – Ein Bindestrich – zwischen zwei Zeichen definiert einen Bereich, zum Beispiel *[a-z]* passt zu *a*, *z* und allen Zeichen, die in der ASCII-Sortierreihenfolge dazwischen liegen.
  – Ein Dach ^ als erstes Zeichen in *zeichenkette* kehrt die Bedeutung von *[zeichenkette]* um: Ein beliebiges Zeichen, das nicht in *zeichenkette* enthalten ist.

  Diese Sonderbedeutungen können durch einen Gegenschrägstrich aufghoben werden.

* Null-, ein- oder mehrmaliges Auftreten des vorausgehenden regulären Ausdrucks.

~ Passt zu der Ersetzungszeichenkette, die beim letzten *s*-Kommando (substitute, siehe *ex*-Kommandos) verwendet wurde.

\(muster\)
Ein regulärer Ausdruck *muster* kann in durch Gegenschrägstrich \ entwertete runde Klammern eingeschlossen werden. Dies dient nur dazu, den regulären Ausdruck zu identifizieren, wenn er in Ersetzungszeichenketten verwendet werden soll (siehe Abschnitt „Ersetzungszeichenketten“ auf Seite 349).

rs Eine Folge *rs* von zwei regulären Ausdrücken *r* und *s* ist wieder ein regulärer Ausdruck. *rs* passt zu allen Zeichenketten, die aus einer zu *r* passenden Zeichenkette, gefolgt von einer zu *s* passenden Zeichenkette, bestehen.

nomagic gesetzt:

Ist *nomagic* gesetzt, so haben nur folgende Zeichen eine Sonderbedeutung:
– Dach ^ am Anfang eines Musters
– Dollar $ am Ende eines Musters sowie
– Gegenschrägstrich \

Folgende Zeichen haben nur dann eine Sonderbedeutung, wenn sie mit einem führenden Schrägstrich \ entwertet worden sind:
– Punkt .
– Stern *
– öffnende eckige Klammer [
– und Tilde ~

Ersetzungszeichenketten

Das kommerzielle Und & (Gegenschrägstrich kommerzielles Und \& bei *nomagic*) steht für die Zeichenkette, zu der das Muster passt und die daher ersetzt werden soll.

Das Zeichen Tilde ~ (\~ bei *nomagic*) wird durch die Ersetzungszeichenkette des letzten *s*-Kommandos (substitute) ersetzt.

Die Zeichenkette *n* (*n* ist eine ganze Zahl) wird durch die Zeichenkette ersetzt, zu der das im *n*-ten Klammernpaar \(...\) enthaltene Muster passt.

Ist in einer Ersetzungszeichenkette die Zeichenkette \ *u* oder \ *l* enthalten, so wird das erste Zeichen in der Ersetzungszeichenkette, das unmittelbar auf \ *u* (upper) oder \ *l* (lower) folgt, in einen Großbuchstaben (bei *u*) bzw. Kleinbuchstaben (bei *l*) umgewandelt, falls es sich bei diesem Zeichen um einen Buchstaben handelt. Mit den Zeichen \ *U* oder \ *L* werden alle Buchstaben bis zum Ende der Ersetzungszeichenkette bzw. bis zu den Zeichen \ *e* oder \ *E* in Groß- bzw. Kleinbuchstaben umgewandelt.

Puffer

Es gibt einen unbenannten und 26 alphabetische Puffer.

In diese Puffer kann Text kopiert werden (*ya* - yank) oder wird gelöschter Text gesichert (*d* - delete), der dann mit *pu* (put) zurückgeholt werden kann.

Die Kommandos *d*, *pu* und *ya* arbeiten mit dem unbenannten Puffer, wenn Sie keinen alphabetischen Puffer angeben. D.h. auch wenn Sie z.B. bei einer Löschoperation keinen Puffer angeben, wird das Ergebnis Ihrer Löschoperation im unbenannten Puffer gesichert.

Sie können in 26 alphabetischen Puffern Textblöcke sichern. Die Puffer werden mit den Kleinbuchstaben a-z oder mit den Großbuchstaben A-Z bezeichnet. Verwenden Sie Kleinbuchstaben, wird der alte Pufferinhalt mit dem neuen überschrieben, der alte somit gelöscht. Verwenden Sie Großbuchstaben, wird der alte Pufferinhalt nicht überschrieben, sondern der neue Text an den alten angehängt.

Fehler- und Signalbehandlung

Tritt während einer *ex*-Sitzung ein Fehler auf, so sendet der *ex* an die Datensichtstation das Zeichen BEL (akustisches Signal; siehe *Tabellen und Verzeichnisse*, *Zeichensatz ISO 646*) und gibt eine Fehlermeldung aus.

Bei einem Unterbrechungssignal kehrt *ex* zusätzlich in den *ex*-Kommando-Modus zurück. Sie können nun ein *ex*-Kommando eingeben.
Liest *ex* die Eingabe aus einer Datei, so wird *ex* verlassen.

Eingeben von Kommandos

Im Unterschied zu den meisten *vi*-Kommandos, die sofort vom *vi* interpretiert und ausgeführt werden, müssen *ex*-Kommandos mit ↵ abgeschlossen werden.

Kommandozeilen, die mit Anführungszeichen " beginnen, werden ignoriert. Auf diese Weise können Sie in ein Kommando-Skript Kommentarzeilen einfügen.

Adressen

Mit einer Adresse geben Sie eine bestimmte Zeile an. Einige *ex*-Kommandos erwarten eine oder mehrere Adressen, um dann zum Beispiel die angegebene Zeile oder den Bereich zwischen zwei angegeben Zeilen einschließlich zu bearbeiten.

Für *ex* existiert zu jedem Zeitpunkt eine aktuelle Zeile. Sie wird durch einen Punkt . adressiert. Die aktuelle Zeile ist in der Regel die Zeile, die zuletzt durch ein Kommando bearbeitet wurde oder auf die gezielt (z.B. durch Suchen) positioniert wurde.

Die Adressen trennen Sie voneinander durch ein Komma , oder einen Strichpunkt ; . Diese Adressenliste arbeitet *ex* von links nach rechts ab.

– Durch das Trennzeichen Strichpunkt ; wird die zuerst adressierte Zeile zur aktuellen Zeile, erst dann wird die nächste Adresse ausgewertet.

– Beim Trennzeichen Komma , werden alle Adressen von der aktuellen Zeile aus ermittelt. Die aktuelle Zeile ändert sich erst bei der Durchführung eines Kommandos.

Wenn Sie bei einem Kommando mehr Adressen angegeben haben, als das Kommando erwartet, werden alle Adressen außer der letzten (wenn das Kommando eine Zeile als Adresse erwartet) oder den letzten beiden (wenn das Kommando einen Zeilenbereich erwartet) ignoriert. Sind bei einem Kommando zwei Adressen erforderlich, so muss sich die zuerst adressierte Zeile im Puffer vor der danach adressierten Zeile befinden. Wenn Sie keine Adresse angeben, verwendet *ex* standardmäßig die aktuelle Zeile.

adresse

. aktuelle Zeile des Puffers.

$ letzte Zeile des Puffers.

n *n*-te Zeile im Puffer. Die Zeilen sind sequentiell von 1 ab durchnummerirt.

'x die mit dem Buchstaben *x* markierte Zeile. *x* muss ein Kleinbuchstabe sein (siehe *ex*-Kommandos *ma* bzw. *k*). '*x* ist die Adresse der markierten Zeile. Bevor *ex* ein Positionierkommando durchführt, wird die momentan aktuelle Zeile markiert. Mit 2 Hochkommata '' können Sie zu ihr zurückkehren.

/rA/ Ein einfacher regulärer Ausdruck in /.../ eingeschlossen (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*) adressiert die erste Zeile, beginnend mit der aktuellen Zeile, die eine Zeichenkette enthält, die zu dem regulären Ausdruck passt. Falls notwendig, springt *ex* vom Ende des Puffers an seinen Anfang, um die Suche fortzusetzen (siehe *ex*-Option, *wrapscan*). Soll die zu *rA* passende Zeile nur ausgegeben werden, können Sie den zweiten Schrägstrich / weglassen.

rA nicht angegeben:
das zuletzt angegebene Suchmuster wird verwendet.

?rA? Wie */rA/*, nur wird rückwärts gesucht.

+n
-n

Beginnt eine Adresse mit Plus- oder Minuszeichen, gefolgt von einer Dezimalzahl *n* (optional), ist die Zeile adressiert, die *n* Zeilen hinter (+) bzw. vor (-) in der aktuellen Zeile liegt. .+3, +3 und +++ haben die gleiche Wirkung.

adr+
adr-

Endet eine Adresse mit einem Plus oder Minus, ist die Zeile adressiert, die eine Zeile hinter (+) bzw. vor (-) der durch die Adresse bezeichneten Zeile liegt. Mit - wird die Zeile vor der aktuellen Zeile adressiert. Auch am Ende einer Adresse haben + und - kumulativen Effekt. Die Adresse ++ adressiert somit die zweite Zeile, d.h. die Zeile, die auf die aktuelle Zeile folgt; z.B. adressiert 3++ Zeile 5 oder ++ ist gleichbedeutend mit .+2.

% Das Prozent-Zeichen % ist gleichbedeutend mit dem Adressenpaar 1,$, d.h. dem gesamten Puffer.

Parameter

Bei den *ex*-Kommandos werden folgende Parameter benutzt:

zeile = *adresse*
eine bestimmte einzelne Zeile, die Sie in einem der im Abschnitt Adressen angegebenen Formate angeben können.

zeile nicht angegeben:
der Standard-Wert Punkt . gilt. Punkt . ist die aktuelle Zeile.

bereich = *adresse,adresse* bereich = *adresse;adresse*
Angabe eines Bereiches zwischen zwei Zeilen einschließlich. Die Adressen können durch ein Komma , oder einen Strichpunkt ; (siehe oben) voneinander getrennt sein. *bereich* sollten Sie nicht zusammen mit einer Zahl *n* angeben, da sonst die *letzte* Adresse des angegebenen Bereiches zur *Anfangs*adresse eines Bereiches wird, der einschließlich dieser Zeile *n* Zeilen umfasst. Damit ist auf jeden Fall ein Bereich adressiert, der hinter dem gemeinten *bereich* liegt. In der Syntax der *ex*-Kommandos ist dies berücksichtigt.

bereich nicht angegeben:
Der Standard-Wert .,. gilt.

.,. (nur die aktuelle Zeile)

n eine positive ganze Zahl. Mit *n* geben Sie die Anzahl der zu bearbeitenden Zeilen an.

n nicht angegeben:
der Standard-Wert 1 gilt.

zusatz
eines oder eine Kombination der *ex*-Kommandos # (siehe *nu*), *p* und *l*. Diese Kommandos geben eine Zeile in einem bestimmten Format aus und werden im Anschluss an das vorangestellte *ex*-Kommandos ausgeführt.

␣ Die Leerzeichen müssen nicht in jedem Fall eingegeben werden. Hier sind sie aus Gründen der Übersichtlichkeit nicht als optional angegeben.

Diese Parameter können mit einer beliebigen Zahl von Plus- oder Minuszeichen kombiniert werden.

Kommandos

Übersicht der ex-Kommandos und ihrer Abkürzungen

CTRL **D**	(scroll)	n Zeilen ausgeben, wobei n=$scroll
ab	abbrev	Abkürzung definieren
a	append	Text-Zeile anfügen
ar	args	Argumentliste der Kommandozeile ausgeben
c	change	Text-Zeile ändern
co	copy	Text-Bereich kopieren
d	delete	Text-Bereich löschen
e	edit	editierte Datei wechseln
f	file	aktuellen Dateinamen ausgeben
g	global	für alle Zeilen, die zu /rA/ passen, Kommando ausführen
i	insert	Text-Zeile einfügen
j	join	Zeilen verbinden
l	list	Text mit nicht-druckbaren Zeilen ausgeben
map	---	Makros definieren
k	mark	Zeilen markieren
m	move	Text-Bereich verschieben
ma	mark	Zeilen markieren
n	next	nächste Datei editieren
nu	number	Zeilen nummeriren und ausgeben
#	number	Zeilen nummerieren und ausgeben
pre	preserve	Editor-Puffer retten
p	print	Text-Bereich ausgeben
pu	put	Puffer ausgeben
q	quit	Editor verlassen
r	read	Kopie einer Datei in Puffer einlesen
rec	recover	Datei nach Systemabsturz wiederherstellen
rew	rewind	Argumentliste rücksetzen (vgl. *ar*)
se	set	Optionen ausgeben und setzen
sh	shell	Shell aufrufen
so	source	Kommandos aus Datei lesen
s	substitute	suchen und ersetzen
unab	unabbrev	Definition einer Abkürzung mit *ab* aufheben
u	undo	Kommando rückgängig machen
unm	unmap	Makrodefinition mit *map* aufheben
v	---	wie g, aber für alle Zeilen, die nicht zu /rA/ passen
ve	version	Versionsnummer des Editors ausgeben
vi	visual	in *vi* wechseln
w	write	Editor-Puffer in Datei schreiben
x	exit	Editor verlassen mit Sichern des Editor-Puffers
ya	yank	Zeilen in Puffer kopieren
z	(window)	Text-Bereich ausgeben

!	(escape)	Shell-Kommando ausführen
& s	(resubst)	wiederholen des letzten substitute-Kommandos
<	(lshift)	Text-Bereich nach links verschieben
>	(rshift)	Text-Bereich nach rechts verschieben
=	(line no)	Zeilennummer ausgeben

Kein Kommando angegeben
Wenn Sie nur *zeile* oder *bereich* angeben, so wird automatisch *p* (print) ausgeführt. Eine leere Eingabe bewirkt, dass die darauffolgende Zeile ausgegeben wird (wie bei .+1p).

CTRL **D**
(ASCII EOT) gibt die nächsten *n* Zeilen aus, wobei *n* den Wert der *ex*-Option *scroll* hat.

ab[**brev**]␣wort␣text
(ab - abbreviate) Nur im *vi* wirksam!

Wenn Sie im *vi*-Eingabe-Modus *wort* als komplettes Wort (d.h. nicht nur als Teil eines Wortes) eingeben, wird es durch die Zeichenkette *text* ersetzt. Falls *text* Sonderzeichen, z.B. ↵, enthalten soll, müssen Sie dem Sonderzeichen CTRL V voranstellen.

[zeile]␣**a**[**ppend**][**!**]
(a - append) *ex* wechselt in den Eingabe-Modus; der Text wird hinter der angegebenen Zeile eingefügt. Bei *zeile* gleich 0 wird der Text ganz an den Anfang des Puffers gesetzt. Die aktuelle Zeile ist die zuletzt eingegebene bzw., falls keine Eingabe erfolgte, die adressierte Zeile.
Die Eingabe wird mit einem Punkt . in der ersten Spalte beendet.

ar[**gs**]
(ar - arguments) Die Argumentliste aus der Kommandozeile des *ex*-Aufrufes wird, eingeschlossen in eckigen Klammern [...], ausgegeben.

zeile␣**c**[**hange**][**!**]␣n
bereich␣**c**
(c - change) *ex* wechselt in den Eingabe-Modus; es werden *n* Zeilen bzw. die Zeilen im angegebenen Bereich *bereich* durch die eingegebenen Textzeilen ersetzt.
Die letzte der eingegebenen Zeilen wird zur aktuellen Zeile; werden keine Textzeilen eingegeben, so wird der angegebene Bereich gelöscht, d.h. *c* wirkt wie *d* (delete). Die Eingabe wird mit einem Punkt . in der ersten Spalte beendet.

chd[**ir**][**!**]␣[directory]
cd[**!**]␣[directory]
(chd/cd - change directory) *ex* wechselt vom aktuellen Arbeitszeichnis nach *directory*.

directory nicht angegeben:
Es wird zu dem Verzeichnis, das in der *HOME*-Variablen hinterlegt ist, gewechselt.

[bereich]␣**co**[**py**]␣zeile[␣zusatz]
[bereich]␣**t**␣zeile[␣zusatz]

(co/t - copy) Eine Kopie des angegebenen Bereiches *bereich* wird hinter die Zeile *zeile* kopiert; hat *zeile* die Zeilennummer 0, wird *bereich* an den Anfang des Puffers kopiert.

[zeile]␣**d**[**elete**][␣puffer][␣n]
[bereich]␣**d**[**elete**][␣puffer]

(d - delete) Die Zeilen im Bereich *bereich* bzw. *n* Zeilen von der angegebenen Zeile *zeile* ab werden gelöscht. Wird der Name eines Puffers *puffer* angegeben, so werden die gelöschten Zeilen in diesem Puffer gesichert. Mit dem *ex*-Kommando *pu* können Sie auf diesen Puffer zugreifen. Die erste auf die gelöschten Zeilen folgende Zeile wird zur aktuellen Zeile; befanden sich die gelöschten Zeilen am Ende des Puffers, so wird die letzte Zeile im Puffer zur aktuellen Zeile.

e[**dit**][**!**][␣+zeile][␣datei]
ex[**!**][␣+zeile][␣datei]

(e - edit) Die Datei *datei* wird editiert. Wurden am aktuellen Editor-Puffer seit dem letzten *w*-Kommando (write) Änderungen vorgenommen, so wird eine Warnung ausgegeben und das Kommando nicht ausgeführt. Sie können jedoch die Ausführung des *e*-Kommandos erzwingen, indem Sie dem *e* ein Ausrufezeichen nachstellen: *e! datei*. Die letzte Zeile des Puffers wird zur aktuellen Zeile. Wird dieses Kommando jedoch von *vi* aus aufgerufen, so wird die erste Zeile im Puffer zur aktuellen Zeile.

+zeile

Die angegebene *zeile* wird zur aktuellen Zeile. *zeile* kann sein:

- eine Zeilennummer *n*
- das Dollarzeichen $ (Dateiende)
- ein regulärer Ausdruck in der Form /*rA* oder ?*rA*.

f[**ile**][␣name]

(f - file) Ausgabe des aktuellen Dateinamens sowie Informationen, wie z.B. die Zahl der Zeilen und die Position der aktuellen Zeile,

name angegeben:
name wird zum aktuellen Dateinamen

[bereich]␣**g**[**lobal**]/rA/[kommandoliste]

(g - global) Zunächst werden alle Zeilen innerhalb des Bereiches *bereich* markiert, zu denen das durch *rA* angegebene Muster passt. Dann wird das angegebene Kommando *kommando* bzw. werden die in *kommandoliste* enthaltenen Kommandos ausgeführt, wobei die jeweils markierte Zeile zur aktuellen Zeile wird.

Die Kommandos in *kommandoliste* können in mehreren Zeilen stehen, vorausgesetzt, die Neue-Zeile-Zeichen sind mit einem Gegenschrägstrich \ entwertet. Ist *kommandoliste* leer, so werden alle Zeilen ausgegeben. Die Kommandos *a(ppend)*, *c(hange)*, und *i(nsert)* sind erlaubt; am Ende von *kommandoliste* können Sie den abschließenden Punkt weglassen. Das *visual*-Kommando ist ebenfalls erlaubt und liest die Eingabe von der Datensichtstation.

Folgende Kommandos dürfen in *kommandoliste* nicht enthalten sein: *global* und *undo*.

Folgende Optionen dürfen in *kommandoliste* nicht enthalten sein: *autoprint*, *autoindent* und *report*.

Beispiel

```
1,$g/^\*/s//+/
```

Von Zeile 1 bis Dateiende ($) werden in der ersten Spalte ^ alle * durch das Substitute-Kommando (*s*) durch ein Plus-Zeichen (+) ersetzt.

[zeile]␣**i**[**nsert**][**!**]

(i - insert) *ex* wechselt in den Eingabe-Modus; der eingegebene Text wird vor der angegebenen Zeile eingefügt. Die letzte der eingegebenen Zeilen wird zur aktuellen Zeile. Erfolgt keine Eingabe, so wird die Zeile vor der adressierten Zeile *zeile* zur aktuellen Zeile. Die Eingabe wird mit einem Punkt . in der ersten Spalte beendet.

[zeile]␣**j**[**oin**][**!**][␣n][␣zusatz]
[bereich]␣**j**[**oin**][**!**][␣zusatz]

(j - join) Fasst die mit *bereich* angegebenen Zeilen bzw. *n* Zeilen von *zeile* ab zu einer Zeile zusammen. Zwischen den ehemaligen Zeilen steht mindestens ein Leerzeichen. Wird eine Zeile durch einen Punkt abgeschlossen, so werden zwei Leerzeichen eingesetzt. Beginnt die anzufügende Zeile mit einer schließenden runden Klammer), wird kein Leerzeichen eingefügt. Zusätzliche Leer- und Tabulatorzeichen am Anfang einer Zeile entfallen.

Wird an das Kommando *j* ein Ausrufezeichen ! angehängt, werden an die Nahtstellen zwischen zwei verbundenen Zeilen keine Leerzeichen gesetzt.

[zeile]␣**l**[**ist**][␣n][␣zusatz]
[bereich]␣**l**[**ist**][␣zusatz]

(l - list) Die angegebenen Zeilen werden ausgegeben; Tabulatorzeichen werden durch `^I` ersetzt; das Ende jeder Zeile wird durch ein Dollarzeichen $ markiert. Der einzige sinnvolle *zusatz* ist hier #, wenn den Zeilen ihre Zeilennummern vorangestellt werden sollen. Die letzte der ausgegebenen Zeilen wird zur aktuellen Zeile.

map[␣x␣kommandos] Format 1
map![␣x␣text] Format 2

Nur im *vi* wirksam!

Format 1: Makrodefinition für *vi*-Kommando-Modus

Definieren von Makros, die im *vi*-Kommando-Modus verwendet werden können. Beim ersten Argument *x* handelt es sich um ein einzelnes Zeichen oder die Zeichenkette #*n*, wobei *n* eine Ziffer ist, mit der die Funktionstaste *n* angegeben wird. Wird im *vi*-Kommando-Modus dieses Zeichen bzw. diese Funktionstaste eingegeben, so verhält *vi* sich so, als ob *kommandos* eingegeben wird. *kommandos* wird dabei als Folge von *vi*-Kommandos interpretiert. Sind in *kommandos* Sonderzeichen, Leerzeichen oder Neue Zeile-Zeichen enthalten, so müssen sie mit einem CTRL V entwertet werden.

Beispiel 1

Sie wollen ein Makro definieren, das in der gesamten Datei das Zeichen Stern * am Zeilenanfang (Spalte 1) durch ein Leerzeichen ␣ ersetzt. Der Name dieses Makros soll "Stern" * sein. Damit Sie dieses Makro stets zur Verfügung haben, machen Sie in Ihrer *.exrc*-Datei (siehe *Voreinstellung des ex*) folgenden Eintrag:

```
:map * :1,$s/^\*/  /
```

Beispiel 2

Sie möchten die Funktionstaste F1 belegen. Es soll die nächste Zeile gesucht werden, an deren Anfang ein bestimmter regulärer Ausdruck *rA* steht; diese Zeile soll dann automatisch gelöscht werden. Dazu geben Sie (aus dem *vi*-Kommando-Modus) ein:

```
:map CTRL V F1 /^rA  CTRL V ↵ dd ↵
```

Format 2: Makrodefinition für *vi*-Eingabe-Modus

Definieren von Makros, die im *vi*-Eingabe-Modus verwendet werden können (vgl. Format 1). *Jedes* im *vi*-Eingabe-Modus eingegebene *x* wird durch *text* ersetzt. (Bei dem Kommando *ab* wird *x* nur dann ersetzt, wenn es allein steht.)

Wenn *x* nicht durch *text* ersetzt werden soll, muss *x* ein CTRL V vorangestellt werden.

Format 1 und Format 2:

Ein definiertes Makro können Sie wieder aufheben mit:

```
unmap x bzw. unmap! x
```

[zeile]␣**ma**[**rk**]␣x
[zeile]␣**k**␣x
(ma/k - mark) *zeile* wird mit *x* markiert; *x* muss ein einzelner Kleinbuchstabe sein, dem ein Leer- oder Tabulatorzeichen vorangestellt ist. An der Adresse der aktuellen Zeile ändert sich nichts.

[bereich]␣**m**[**ove**]␣zeile
(m - move) *bereich* wird hinter *zeile* versetzt. Die erste der versetzten Zeilen wird zur aktuellen Zeile.

n[**ext**][**!**][␣datei]...
datei nicht angegeben:
(n - next) Die nächste der in der Kommandozeile aufgeführten Dateien (Argumentliste) wird editiert (vgl. *f* und *ar*). Wurden am aktuellen Editor-Puffer seit dem letzten *w*-Kommando (write) Änderungen vorgenommen, so wird eine Warnung ausgegeben und das Kommando nicht ausgeführt. Sie können jedoch die Ausführung des *n*-Kommandos erzwingen, indem Sie dem *n* ein Ausrufezeichen nachstellen: *n!*.

datei angegeben:
Die bisherige Argumentenliste wird durch die angegebene(n) Datei(en) ersetzt; die erste angegebene Datei wird editiert.

[zeile]␣**nu**[**mber**][␣n][␣zusatz]
[bereich]␣**nu**[**mber**][␣zusatz]
[zeile]␣**#**[␣n][␣zusatz]
[bereich]␣**#**[␣zusatz]
(nu - number) Die Zeilen werden, versehen mit den entsprechenden Zeilennummern, ausgegeben. Der einzige hier sinnvolle Zusatz ist *l*. Die letzte der ausgegebenen Zeilen wird zur aktuellen Zeile.

pre[**serve**]
(pre - preserve) Der Inhalt des aktuellen Puffers wird gesichert, als ob es zu einem Absturz des Systems gekommen wäre. *pre* wird in Notfällen verwendet, wenn z.B. *w* (write) nicht ausgeführt werden kann, weil die Platte voll ist, und keine andere Möglichkeit zur Rettung des Pufferinhaltes vorhanden ist.

[zeile]␣**p**[**rint**][␣n]
[bereich]␣**p**[**rint**]
(p - print) Die angegebenen Zeilen werden ausgegeben, wobei nicht-druckbare Zeichen als Steuerzeichen in der Form ^x ausgegeben werden; DEL wird in Form von ^? ausgegeben. Die zuletzt ausgegebene Zeile wird zur aktuellen Zeile.

[zeile]␣**pu**[**t**][␣puffer]
(pu - put) Die mit *d* (delete) gelöschten oder mit *y* (yank) in einem Puffer gesicherten Zeilen werden hinter der angegebenen Zeile *zeile* eingefügt. *puffer* ist der Name eines alphabetischen Puffers.

puffer nicht angegeben:
Der Text wird aus dem unbenannten Puffer geholt.

q[**uit**][**!**]
(q - quit) Der Editor wird verlassen. Wurde der Puffer seit dem letzten *w*-Kommando (write) verändert, so wird eine Warnung ausgegeben und *q* wird nicht ausgeführt. Sie können jedoch die Ausführung des *q*-Kommandos erzwingen, indem Sie dem *q* ein Ausrufezeichen nachstellen: *q!*. Alle noch nicht (mit *w*) gesicherten Änderungen am Editor-Puffer gehen dann verloren.

[zeile]␣**r**[**ead**][**!**][␣datei]
(r - read) Eine Kopie von *datei* wird im Editor-Puffer hinter *zeile* eingelesen. Die Zeilennummer von *zeile* kann dabei 0 sein; die Textzeilen werden dann an den Anfang des Puffers gesetzt. Gibt es keine aktuelle Datei (Aufruf von *ex* ohne Dateinamen) wird *datei* zur aktuellen Datei. Die letzte der eingelesenen Zeilen wird zur aktuellen Zeile; im *vi* wird die erste der eingelesenen Zeilen zur aktuellen Zeile.

datei nicht angegeben:
die aktuelle Datei wird eingelesen.

Wird statt *datei* !*kmdo* angegeben, so wird *kmdo* als POSIX-Kommando interpretiert und an die Shell übergeben. Die Ausgabe des Kommandos wird in den Puffer eingelesen.

rec[**over**]␣datei
datei wird nach einem Absturz des Systems oder des Editors wiederhergestellt.

rew[**ind**][**!**]
(rew - rewind) Die Argumentliste (siehe *ar*) wird zurückgesetzt, und die erste Datei in der Liste wird editiert. Wurde der Editor-Puffer seit dem letzten *w*-Kommando (write) verändert, so wird eine Warnung ausgegeben und *rew* wird nicht ausgeführt. Sie können jedoch die Ausführung des *rew*-Kommandos erzwingen, indem Sie dem *rew* ein Ausrufezeichen nachstellen: *rew!*. Alle noch nicht (mit *w*) gesicherten Änderungen am Editor-Puffer gehen dann verloren.

se[**t**][␣parameter]
(se - set) Mit dem *set*-Kommando können Optionen aufgelistet bzw. gesetzt werden. Es gibt zwei Typen von Optionen: Optionen mit Boole'schen Werten und Optionen mit nicht-Boole'schen Werten. Ein Beispiel für den Boole'schen Typ ist z.B. die Option *number*. Ist sie gesetzt, werden Zeilennummern ausgegeben, ist *nonumber* gesetzt, werden keine Zeilennummern ausgegeben. Ein Beispiel für den nicht-Boole'schen Typ ist *report*. Der Wert dieser Option (Standard = 5) bestimmt die Anzahl der Zeilen, die durch ein Kommando verändert werden muss, damit *ex* und *vi* eine Meldung geben.

parameter

all die Werte aller Optionen werden ausgegeben.

option**?** der gegenwärtige Wert der angegebenen Option wird ausgegeben.

option Nur für Optionen mit Boole'schen Werten: die Option wird gesetzt.

nooption Nur für Optionen mit Boole'schen Werten: die Option wird nicht gesetzt.

option**=**wert
Nur für Optionen mit nicht-Boole'schen Werten: der Option wird der Wert *wert* zugewiesen.

parameter nicht angegeben:
set gibt diejenigen *ex*-Optionen aus, deren Werte vom Benutzer festgelegt wurden und daher von den Standard-Werten abweichen.

Nähere Informationen über die *ex*-Optionen finden Sie in *ex-Optionen*.

sh[**ell**]
Ist die Variable *SHELL* gesetzt, wird die dort angegebene Shell gestartet, sonst *sh*. Nach Beendigung der Shell wird die Editorsitzung an derselben Stelle fortgesetzt.

so[**urce**]␣datei
Kommandos werden aus *datei* gelesen und ausgeführt. *so*-Kommandos können ineinander verschachtelt werden.

[zeile]␣**s**[**ubstitute**][/rA/ersatz/[schalter][␣n][␣zusatz]]
[bereich]␣**s**[**ubstitute**][/rA/ersatz/[schalter][␣zusatz]]
(s - substitute) Das erste Auftreten des mit *rA* (regulärer Ausdruck) getroffenen Musters in jeder Zeile des angegebenen Bereiches wird durch die Zeichenkette *ersatz* ersetzt (siehe *Reguläre Ausdrücke* und *Ersetzungszeichenketten*). Der angegebene Bereich ist entweder *bereich* oder umfasst *n* Zeilen von *zeile* ab.

/rA/ersatz nicht angegeben:
es wird */rA/ersatz/* vom vorhergehenden substituierten Kommando verwendet.

schalter

g (g - global) *alle* Zeichenketten in der Zeile, zu denen *rA* passt werden ersetzt.

c (c - confirm) zunächst wird die Zeile ausgegeben, wobei die zu ersetzende Zeichenkette in der darunterliegenden Zeile durch ein ^ markiert wird. Die Eingabe des Buchstabens *y* bewirkt dann, dass die Substitution durchgeführt wird; bei jeder anderen Eingabe wird das Kommando abgebrochen. Die Zeile, in der zuletzt eine Substitution durchgeführt wurde, wird zur aktuellen Zeile.

una[**bbrev**]␣wort
(una - unabbreviate) *wort* wird aus der Liste der Abkürzungen gestrichen (vgl. *ab*).

u[**ndo**]
(u - undo) Die mit dem letzten Editierkommando vorgenommenen Änderungen werden rückgängig gemacht. *g* (global) und *vi* (visual) werden in diesem Fall als eigenständige Kommandos betrachtet. Kommandos, die die äußere Umgebung verändern (z.B. *w* (write), *e* (edit) und *n* (next)) können nicht rückgängig gemacht werden.

Ein *undo*-Kommando selbst kann mit *u* wieder rückgängig gemacht werden.

Bei *u* gehen alle Markierungen für Zeilen, die geändert und dann wieder zurückgeholt wurden, verloren.

unm[**ap**][**!**]␣x
(unm - unmap) Die mit *map* vorgenommene Definition des Makros *x* wird aufgehoben.

[bereich]␣**v**[**ice**]/rA/kommandoliste
(v - vice versa) Hat dieselbe Funktion wie *global*, nur wird *kommandoliste* auf alle Zeilen angewandt, zu denen das Muster *rA* nicht passt.

ve[**rsion**]
Die Versionsnummer des Editors wird ausgegeben.

[zeile]␣**vi**[**sual**][␣stelle][␣n]
Bei der angegebenen *zeile* wird in den *vi* gewechselt. Der optionale Parameter *stelle* kann eines der Zeichen Bindestrich - oder Punkt . sein; wie beim Kommando *z* wird damit die Position von *zeile* auf dem Bildschirm angegeben. Standard: oberer Rand des Bildschirms. Mit *n* wird die Größe des Bildschirms angegeben; standardmäßig hat er die Größe, die über die *ex*-Option *window* definiert ist. Mit *Q* wird der *vi* wieder verlassen und wieder in den *ex* gewechselt. Nähere Information siehe *vi*.

[bereich␣]**w**[**rite**][**!**][␣datei] (Format 1)
[bereich␣]**wq**[**!**][␣datei] (Format 2)

Format 1: In Datei schreiben
(w - write) Der angegebene *bereich* wird in die *datei* geschrieben, wobei die Anzahl der geschriebenen Zeilen und Zeichen ausgegeben wird.

Wird eine (vorhandene) *sekundäre* Datei angegeben, so wird das Kommando nicht ausgeführt, damit die sekundäre Datei nicht versehentlich überschrieben wird. Die Ausführung von *w* kann jedoch erzwungen werden, indem ein Ausrufezeichen ! angefügt wird: *w! #*.

Soll am Ende von *datei* angefügt werden, so muss das Kommando in der Form *w>>* angegeben werden; existiert *datei* nicht, so wird eine Fehlermeldung ausgegeben.

Wird statt *datei* !*kmdo* angegeben, so wird *kmdo* als POSIX-Kommando interpretiert und an die Shell übergeben; der angegebene *bereich* ist dann für das Kommando die Standard-Eingabe.

bereich nicht angegeben:
die ganze aktuelle Datei wird nach *datei* geschrieben.

datei nicht angegeben:
standardmäßig wird die aktuelle Datei verwendet. Wenn weder eine aktuelle Datei vorhanden ist noch eine *datei* angegeben wird, so kann *w* nicht ausgeführt werden.

Format 2: In Datei schreiben und *ex* verlassen
(wq - write and quit) *wq* hat dieselbe Funktion wie *w*, auf das *q* folgt; *wq!* hat dieselbe Funktion wie *w!*, auf das *q* folgt.

x[**it**][**!**]
(exit) Der Editor wird verlassen; wurden seit dem letzten Sichern mit *w* (write) Änderungen vorgenommen, so werden diese zuvor in die Datei geschrieben.

[zeile]␣**ya**[**nk**][␣puffer][␣n]
[bereich]␣**ya**[**nk**][␣puffer]
Der angegebene *bereich* bzw. *n* Zeilen von *zeile* ab werden in den angegebenen *puffer* gesichert.

puffer nicht angegeben:
Wird kein Puffer angegeben, so wird der unbenannte Puffer verwendet.

[zeile]␣**z**[␣stelle][␣n]
(window) *n* Zeilen aus der Umgebung von *zeile* werden ausgegeben.

n nicht angegeben:
n hat den Wert der Variablen *window*.

stelle

Für den optionalen Parameter *stelle* können folgende Zeichen angegeben werden:

- Der Bindestrich - bewirkt, dass die *zeile* an den unteren Rand des ausgegebenen Bereiches gesetzt wird.

+ Das Pluszeichen + bewirkt, dass die *zeile* an den oberen Rand des ausgegebenen Bereiches gesetzt wird.

. Der Punkt . bewirkt, dass sich die Zeile in der Mitte des ausgegebenen Bereiches befindet.

^ Das Zeichen ^ bewirkt, dass *n* Zeilen ausgegeben werden, wobei *n*2* Zeilen vor der angegebenen *zeile* begonnen wird. Dies hat den Effekt, dass ein *z*^-Kommando, das nach einem anderen *z*-Kommando angegeben wird, die vorherige Seite ausgibt.

= Das Gleichheitszeichen = bewirkt, dass die angegebene *zeile* zentriert am Bildschirm ausgegeben werden, mit je einer Zeile von Bindestrichen oberhalb und unterhalb. Die Anzahl der vorhergehenden bzw. nachfolgenden Textzeilen wird um die Zahl der Bindestrichzeilen verringert.

Die letzte der ausgegebenen Zeilen wird zur aktuellen Zeile.

stelle nicht angegeben:
zeile befindet sich am oberen Rand des ausgegebenen Bereiches.

Vorsicht: *ex* gibt die gewünschte Anzahl realer Zeilen aus. Sind Zeilen länger, wie der Bildschirm breit ist, dann kann unter Umständen mehr als ein Bildschirm ausgegeben werden.

!␣kommando
(escape) Die auf das Ausrufezeichen ! folgenden Zeichen werden als Shell-Kommando interpretiert und an den Kommandointerpreter übergeben. Wurden am Puffer seit dem letzten *w*-Kommando (write) Änderungen vorgenommen, so wird eine Warnung ausgegeben. Dann wird das Kommando ausgeführt. Wenn die *Ausgabe des Kommandos* nicht umgelenkt wird, erscheint sie auf dem Bildschirm, ändert aber an der Datei nichts. Allerdings kann das *Kommando selbst* die Datei ändern, z.B. *!cat /etc/profile >> %*. Nach erfolgreicher Ausführung des Kommandos wird ein ! ausgegeben. An der Adresse der aktuellen Zeile ändert sich nichts.

Kommen in *kommando* das Prozentzeichen % und das Nummernzeichen # vor, so werden sie als Dateinamen expandiert (siehe *Aktuelle und sekundäre Datei*).

Ein Ausrufezeichen ! in *kommando* wird durch das vorherige !-Kommando ersetzt. Mit !! wird also das letzte !-Kommando wiederholt. Die so erweiterten Kommandozeilen werden auf dem Bildschirm ausgegeben.

bereich␣**!**␣kommando
(escape) Bei einem !-Kommando in dieser Form werden die mit *bereich* angegebenen Zeilen an *kommando* als Standard-Eingabe übergeben und durch die Ausgabe von *kommando* ersetzt.

bereich nicht angegeben:
bereich wird nicht durch .,. ersetzt, sondern es gilt die andere Form des !-Kommandos.

" Kommandozeilen, die mit Anführungszeichen " beginnen, werden ignoriert. Auf diese Weise können Sie in ein Kommando-Skript Kommentarzeilen einfügen.

zeile␣**&**schalter␣n␣zusatz
bereich**&**schalter␣zusatz
zeile␣**s**[**ubstitute**]schalter␣n␣zusatz
bereich**s**[**ubstitute**]schalter␣zusatz
(resubst) Das letzte *s*-Kommando (substitute) wird wiederholt, als ob *&* durch das letzte *s/rA/ersatz/* ersetzt würde. Dieselbe Funktion hat ein *s*-Kommando, in dem */rA/ersatz/* weggelassen wird.

zeile␣**<**␣n
bereich␣**<**

(lshift) Die Zeilen innerhalb des angegebenen Bereichs bzw. *n* Zeilen, die auf *zeile* folgen, werden um *shiftwidth* Positionen nach links verschoben. Leer- und Tabulatorzeichen gehen dabei verloren; die übrigen Zeichen bleiben unverändert erhalten. Die letzte der geänderten Zeilen wird zur aktuellen Zeile.

zeile␣**>**␣n
bereich␣**>**

(rshift) Die Zeilen innerhalb des angegebenen Bereichs bzw. *n* Zeilen, die auf *zeile* folgen, werden durch Einfügen von Leer- und Tabulatorzeichen um *shiftwidth* Positionen nach rechts verschoben.

zeile**=**

(line number) Die Zeilen-Nummer der angegebenen *zeile* wird ausgegeben. Die Position der aktuellen Zeile wird nicht verändert.

zeile nicht angegebnen:
Die Zeilen-Nummer der letzten Zeile wird ausgegeben.

ex-Optionen

Mit den Optionen des *ex* kann das Verhalten der Editoren *ex* und (vor allem) *vi* beeinflusst werden (siehe *vi*, *Voreinstellung des vi*).

Für alle Optionen gibt es Standard-Einstellungen, die beim Aufruf von *ex* oder *vi* wirksam werden. Mit dem *ex*-Kommando *se* (set) können Sie sich alle Optionen ausgeben lassen:

```
set all↵
```

Sie können aber auch mit diesem Kommando eine oder mehrere Optionen nach Ihren Vorstellungen verändern:

```
set showmode scroll=15↵
```

In diesem Fall wird im *vi* der Eingabe-Modus in der Status-Zeile angezeigt und die Anzahl der Zeilen, um die CTRL *D* den Bildschirm rollt, auf *15* gesetzt.

Mit

```
set↵
```

werden Ihnen alle Optionen ausgegeben, die von den Standard-Werten abweichen.

Es gibt zwei Typen von Optionen: Optionen mit Boole'schen Werten und Optionen mit nicht-Boole'schen Werten. Ein Beispiel für den Boole'schen Typ ist z.B. die Option *showmode*. Ist diese Option nicht gesetzt, hat sie den Namen *noshowmode*. Ein Beispiel für den nicht-Boole'schen Typ ist *scroll*.

autoindent, ai
noautoindent, noai (Standard)

Ist die Option *autoindent* gesetzt, so wird im Eingabe-Modus jede Zeile entsprechend der vorhergehenden Zeile eingerückt (es werden Leer- und Tabulatorzeichen eingefügt). Der Beginn der Einrückung wird bestimmt von der Zeile, hinter die angefügt (*a*,) vor die eingefügt (*i*) oder die als erste geändert (*c*) wird. Zusätzliche Einrückungen können wie gewöhnlich durchgeführt werden, die folgenden Zeilen werden dann automatisch auf die neue Grenze eingerückt.

Soll der Text weniger eingerückt werden, so können Sie durch ein- oder mehrmalige Eingabe von CTRL D den Zeilenanfang auf die *shiftwidth*-te Position nach links verschieben. Ist die aktuelle Cursorposition z.B. 25 und *shiftwidth* auf 10 eingestellt, dann bewirkt die einmalige Eingabe von CTRL D, dass der Cursor nach links auf die Position 20 rückt und die zweimalige Eingabe von CTRL D, dass der Cursor zweimal nach links auf die Position 10 rückt.
Mit einem Dach ^, gefolgt von einem CTRL D, wird die Einrückung für die aktuelle Zeile aufgehoben; mit einer Null *0*, gefolgt von einem CTRL D werden alle Einrückungen aufgehoben.

autoprint, ap
noautoprint, noap (Standard)
Nach jedem Kommando, mit dem der Inhalt der Editor-Puffers geändert wurde, wird die aktuelle Zeile ausgegeben. Bei globalen Suchen/Ersetzen-Kommandos (siehe *g*, *s* und *v*) wird *autoprint* unterdrückt.

autowrite, aw
noautowrite, noaw (Standard)
Der Inhalt des Editor-Puffers wird in die aktuelle Datei geschrieben, wenn Änderungen vorgenommen worden sind und eines der Kommandos *e* (edit), *n* (next), *rew* (rewind) oder *!* (escape) aufgerufen wird.

beautify, bf
nobeautify, nobf (Standard)
Alle Steuerzeichen außer dem Tabulatorzeichen, dem Neue Zeile- und dem Formularvorschubzeichen werden aus dem Eingabetext entfernt.

directory=dvz
dir=dvz
Mit *dvz* wird das Dateiverzeichnis angegeben, in welchem der Editor-Puffer angelegt werden soll. Hat der Benutzer für dieses Dateiverzeichnis keine Schreiberlaubnis, so wird der Editor verlassen.

edcompatible, ed
noedcompatible, noed
Ist das *s*-Kommando (substitute) mit den Schaltern *g* und *c* versehen, so werden diese Schalter gespeichert; die nochmalige Angabe der Schalter hebt ihre Wirkung auf.

errorbells, eb
noerrorbells, noeb
Vor Fehlermeldungen, die in der letzten Zeile (in der Regel invers) erscheinen, wird ein Klingelsignal gesendet.

exrc, ex
Diese Option ist aus Sicherheitsgründen nicht wirksam. Sie sollte bewirken, dass *.exrc*-Dateien zulässig sind, die im aktuellen Dateiverzeichnis stehen.

flash, fl (Standard)
noflash, nofl
Wenn in der Beschreibung des Terminals ein Eintrag darüber enthalten ist, wie am Bildschirm ein optisches Signal erzeugt werden kann, so wird der Benutzer auf die dort beschriebene Art auf eine fehlerhafte Eingabe aufmerksam gemacht.

hardtabs=zahl
ht=zahl
Mit dieser Option legen Sie fest, in welchen Schritten auf dem Terminal Tabulatoren definiert sind (zur Optimierung der Bildschirmausgabe).

ignorecase, ic
noignorecase, noic (Standard)
Alle Großbuchstaben im Text werden beim Suchen mit regulären Ausdrücken wie Kleinbuchstaben behandelt. Gleichfalls werden alle Großbuchstaben in regulären Ausdrücken wie Kleinbuchstaben behandelt, außer in Ausdrücken für Zeichenklassen.

lisp
nolisp (Standard)
Die Option *autoindent* wird gesetzt und die *vi*-Kommandos öffnende runde Klammer (, schließende runde Klammer), öffnende geschweifte Klammer {, schließende geschweifte Klammer }, doppelte öffnende eckige Klammern [[und doppelte schließende eckige Klammern]] werden angepasst.

list
nolist (Standard)
In den ausgegebenen Zeilen werden Tabulatorzeichen durch `^I` ersetzt und das Ende jeder Zeile durch ein $ markiert.

magic (Standard)
nomagic
Ändert die Interpretation der in regulären Ausdrücken und Ersetzungszeichenketten verwendeten Zeichen (siehe *Reguläre Ausdrücke und Ersetzungszeichenketten*).

mesg (Standard)
nomesg
Ein-/Ausschalten der Schreiberlaubnis für andere Benutzer auf das Terminal (z.B durch das Kommando *write*).

modelines (Standard)
nomodelines
Die ersten und die letzten fünf Zeilen einer eingelesenen Datei werden als Editorkommandos betrachtet und ausgeführt, wenn sie von der Form sind:
`ex:`*`kommando`*`:`

novice
nonovice (Standard)
Es sollen ausführlichere Fehlermeldungen erzeugt werden.

number, nu
nonumber, nonu (Standard)
Die Zeilen werden mit vorangestellten Zeilennummern ausgegeben.

optimize, opt
nooptimize, noopt (Standard)
Sie können das Terminal so umschalten, dass kein automatischer Wagenrücklauf erfolgt (zur Optimierung der Ausgabe).

paragraphs=zeichenkette
para=zeichenkette
Dieser Option wird eine Zeichenkette zugewiesen. Jeweils zwei in dieser Zeichenkette enthaltene Zeichen stehen für den Namen eines *nroff*-Makros, das einen Absatz (Paragraphen) einleitet. Ein Makro wird in einen Text im Format *.XX* eingetragen, wobei der Punkt . das erste Zeichen der Zeile ist.

prompt (Standard)
noprompt
Ist *prompt* gesetzt, so wird im Kommando-Modus das Bereitzeichen Doppelpunkt : angezeigt; andernfalls wird kein Bereitzeichen ausgegeben.

readonly
noreadonly (Standard)
Ist *readonly* gesetzt, so können die zu bearbeitenden Dateien nur gelesen werden. Veränderungen am Text sind nicht möglich. Ist *noreadonly* gesetzt, können Sie in den Dateien Veränderungen vornehmen.

redraw (Standard)
noredraw
Werden Änderungen vorgenommen, so sollen diese sofort auf dem Bildschirm ausgegeben werden. Wegen der großen Menge der ausgegebenen Daten ist dies jedoch nur sinnvoll, wenn die Daten mit einer hohen Geschwindigkeit übertragen werden.

remap (Standard)
noremap
Ist *remap* gesetzt, so werden bei der Makroumsetzung auch Makros berücksichtigt, die durch andere Makros definiert sind. Die Umsetzung wird fortgesetzt, bis das gewünschte Makro erstellt ist. Ist diese *noremap* gesetzt, so wird lediglich eine einstufige Umsetzung durchgeführt.

report=n
Wenn die Zahl der Zeilen, die mit dem letzten Kommando geändert, gelöscht oder kopiert wurde, größer als *n* ist, erscheint in der Status-Zeile eine Meldung.

scroll=n
Der bei *scroll* angegebene Wert *n* steht für die Anzahl von Zeilen, um die das Fenster bei einem CTRL D verschoben werden soll, und die bei einem *z*-Kommando (der zweifache Wert von *scroll*) ausgegeben werden sollen.

sections=zeichenkette
sect=zeichenkette
Dieser Option wird eine Zeichenkette zugewiesen. Jeweils zwei in dieser Zeichenkette enthaltene Zeichen stehen für den Namen eines *nroff*-Makros, das einen Abschnitt (Section) einleitet. Ein Makro wird in einen Text im Format *.XX* eingetragen, wobei der Punkt . das erste Zeichen der Zeile ist.

shiftwidth=n
sw=n
Der bei *sw* angegebene Wert *n* steht für die Schrittweite eines Software-Tabulators um die der Text bei gesetzter Option *autoindent* und den Kommandos Kleinerzeichen < und Größerzeichen > verschoben werden soll.

showmatch, sm
noshowmatch, nosm (Standard)
Wird im *vi* eine schließende runde Klammer) oder eine schließende geschweifte Klammer } eingegeben, so wird die zugehörige (oder { angezeigt, falls sie sich noch auf dem Bildschirm befindet.

showmode, smd
noshowmode, nosmd (Standard)
Wenn *smd* gesetzt ist, wird in der Status-Zeile angezeigt, wenn sich *vi* im *vi*-Eingabe-Modus befindet.

slowopen, slow
noslowopen, noslow (Standard)
Im *vi* soll die Korrektur des Bildschirminhalts während der Eingabe verzögert werden. Dadurch soll der Durchsatz bei nicht intelligenten Datensichtstationen erhöht werden.

tabstop=n
ts=n
n gibt die Software-Tabulatoren an, die vom Editor benutzt werden, um Tabulatoren in der editierten Datei zu expandieren.

taglength=zahl
tl=zahl
Im Zusammenhang mit dem *:tag*-Kommando wird hier bestimmt, wieviele Zeichen für die Suche signifikant sein sollen. Der Wert 0 besagt, dass alle Zeichen berücksichtigt werden.

tags=dateien
Namen der *tag*-Dateien, durch Leerzeichen getrennt.

term=zeichenkette
zeichenkette beschreibt für *vi* den zu verwendenden Terminaltyp. (Standardmäßig wird der Wert der Umgebungsvariable *TERM* verwendet.)

terse
noterse (Standard)
Ist *terse* gesetzt, so werden gekürzte Fehlermeldungen ausgegeben.

timeout
notimeout
Verzögerungszeit setzen/rücksetzen.

ttytype=zeichenkette
zeichenkette beschreibt für *vi* den zu verwendenden Terminaltyp. (Standardmäßig wird der Wert der Umgebungsvariable *TERM* verwendet.)

warn (Standard)
nowarn
Wenn *warn* gesetzt ist, erscheint bei einem *:!*-Kommando die Warnung *No write since last change*, wenn seit der letzten Sicherung mit *w* noch Änderungen am Editor-Puffer erfolgten.

window=n
Die Anzahl der Zeilen in einem Textfenster des *vi*.

wrapmargin=n
wm=n
Nur im *vi* wirksam. Wenn *n* > 0 ist, wird an das letzte Wort in einer Eingabezeile automatisch ein Neue-Zeile-Zeichen angefügt, so dass das Zeilenende um mindestens *n* Positionen vom rechten Rand des Bildschirms entfernt ist.

Mit *n*=0 (Standard) kann der automatische Zeilenumbruch ausgeschaltet werden.

wrapscan, ws (Standard)
nowrapscan, nows
Ist *wrapscan* gesetzt, so wird das Suchen mit /.../ oder ?...?, wenn das Ende bzw. der Anfang des Editor-Puffers erreicht ist, am Anfang bzw. Ende fortgesetzt. Falls *nows* gesetzt ist, wird das Suchen am Anfang bzw. am Ende der Datei beendet.

writeany, wa
nowriteany, nowa (Standard)
Wenn *nowa* gesetzt ist, wird bei einem Sicherungskommando *w* eine Prüfung auf die Existenz der Datei gemacht. Damit wird verhindert, dass Sie versehentlich eine schon existierende Datei überschreiben. Mit *w!* wird diese Prüfung umgangen. Wenn *wa* gesetzt ist, wird nicht geprüft, ob die Datei schon existiert.

Fehler

Tritt bei *ex* ein Fehler auf, so sendet der Editor an die Datensichtstation das akustische Signal BEL (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*) und gibt eine Fehlermeldung aus. Bei einem Unterbrechungssignal kehrt *ex* auf die Kommandoebene zurück, d.h. Sie können ein neues *ex*-Kommando eingeben. Liest der Editor die Eingabe aus einer Datei, so bricht *ex* bei einem Unterbrechungssignal ab.

Stellt *ex* einen internen Fehler fest, so wird versucht, den Puffer zu retten, wenn die zuletzt vorgenommenen Änderungen noch nicht abgespeichert worden sind. Durch einen erneuten Aufruf von *ex* mit der Option *-r* kann auf die gesicherten Daten zurückgegriffen werden.

Datei

$HOME/.exrc
Datei mit Voreinstellungen für *ex* und *vi*. Diese Voreinstellungen sind nicht wirksam, wenn durch *EXINIT* Widersprüchliches definiert wird.

Variable

TERM
In der Umgebungsvariablen *TERM* muss der Typ der benutzten Datensichtstation festgelegt sein.

EXINIT
Umgebungsvariable mit Voreinstellungen für *ex* und *vi*. Diese Voreinstellungen sind immer wirksam.

LINES
Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Berechnung der Spaltenzahl für die Ausgabe von *select*-Listen verwendet. *select*-Listen werden vertikal ausgegeben, bis ungefähr zwei drittel der Zeilenzahl *LINES* gefüllt sind.

COLUMNS
Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Definition der Breite des Editierfensters beim Editiermodus der POSIX-Shell und für die Ausgabe der *select*-Liste verwendet. Diese Variable ist sinnvoll, wenn der Zugang zur POSIX-Shell über *rlogin* erfolgt.

PATH
Der Suchpfad für Shell-Kommandos, die in den Editorkommandos *shell*, *read* und *write* angegeben werden können.

HOME
Legt das Dateiverzeichnis fest, in dem nach der Editor Startup-Datei mit dem Namen *.exrc* gesucht wird.

SHELL
Der Kommandozeilen-Interpreter, der bei *!*, *shell*, *read* oder anderen Kommandos mit einem Operanden der Form *!string* verwendet wird. Für das *shell* Kommando wird das Programm nur dem *-i* Argument aufgerufen, bei allen anderen mit zwei Argumenten *-c* und *string*. Ist *SHELL* nicht gesetzt oder hat den Wert 0, wird *sh* verwendet.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *ex*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für die Sortierreihenfolge in Bereichen, Äquivalenzklassen und Zeicheneinheiten in regulären Ausdrücken fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien), sowie das Verhalten von Zeichenklassen in regulären Ausdrücken, die Einteilung von Zeichen in Groß- und Kleinbuchstaben, deren Umwandlung sowie die Entdeckung von Wortgrenzen.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Es werden einige *ex*-Kommandos vorgeführt und erläutert:

```
$ ex zahlen                  -  ex aufrufen
"zahlen" 4 lines, 20 characters
:set number                  -  Zeilennummern ausgeben
:1,$p                        -  Zeile 1 bis letzte Zeile ausgeben
     1  eins                -  Ursprünglicher Inhalt von zahlen
     2  zwei
     3  drei
     4  vier
:1 c 1                       -  Zeile 1 ändern
     1  EINS                 -  Neue Zeile 1
     2  .                    -  Eingabemodus verlassen
:2,3 co 4                    -  Zeile 2 bis 3 hinter Zeile 4
     6  drei                    kopieren
:1,$ g/zwei/s//ZWEI/         -  In der gesamten Datei "zwei"
                                durch "ZWEI" ersetzen
:2 a                         -  Hinter Zeile 2 neue Zeile einfügen
     3  DREI                 -  Neue Zeile 3
     4  .                    -  Eingabemodus verlassen
:4 i                         -  Vor Zeile 4 neue Zeile einfügen
     4  VIER                 -  Neue Zeile 4
     5  .                    -  Eingabemodus verlassen
:5,$ d a                     -  Zeile 5 bis Dateiende löschen und
                                in Puffer a sichern
:     4  VIER
:1,$p                        -  Zeile 1 bis letzte Zeile ausgeben
     1  EINS
     2  ZWEI
     3  DREI
     4  VIER
:wq                          -  Sichern des Pufferinhalts und ex verlassen
"zahlen" 4 lines, 20 characters
```

Siehe auch *ed, vi*

exec Die aktuelle Shell überlagern (execute commands and open, close or copy file descriptors)

Das in die POSIX-Shell *sh* eingebaute Kommando *exec* hat zwei Funktionen:

- *exec* überlagert die aktuelle Shell durch ein anderes Programm (Format 1). Die aktuelle Shell ist beendet, wenn dieses Programm gestartet wird. Es entsteht kein neuer Prozess; das erkennen Sie daran, dass sich die Prozessnummer nicht ändert (siehe *Beispiel 2*).

  Wenn Sie *exec* im Dialog eingeben, ist nach der Ausführung des angegebenen Programms die übergeordnete Shell aktiv bzw. Ihre Sitzung beendet.
  Steht *exec* in einer Shell-Prozedur, so wird diese Prozedur beendet. Kommandos, die in der Prozedur nach dem Aufruf von *exec* stehen, werden nie ausgeführt.

- *exec* lenkt die Standard-Eingabe bzw. die Standard-Ausgabe der Shell um auf eine Datei (Format 2). Nach der Ausführung von *exec* lesen alle eingegebenen Kommandos aus dieser Datei bzw. schreiben in diese Datei, bis die aktuelle Shell beendet ist.

Syntax

Format 1: exec␣programm[␣umlenkung]

Format 2: exec␣umlenkung

Format 1 **Die Shell durch ein anderes Programm überlagern**

exec␣programm[␣umlenkung]

programm
: beliebiges Kommando, Programm oder Shell-Prozedur, aber kein eingebautes *sh*-Kommando. Für die dazugehörige Datei brauchen Sie Ausführrecht.

  Die aktuelle Shell beendet sich; stattdessen wird *programm* ausgeführt. Wenn *programm* ausgeführt ist, meldet sich die übergeordnete Shell zurück oder der Begrüßungsbildschirm wird ausgegeben.

umlenkung
: weist dem angegebenen Programm, falls es von der Standard-Eingabe liest bzw. auf die Standard-Ausgabe schreibt, eine Datei für die Eingabe bzw. Ausgabe zu.
  Folgende Angaben für *umlenkung* sind möglich:

  >datei — Die Standard-Ausgabe des angegebenen Programms wird auf *datei* umgelenkt. In *datei* enthaltene Daten werden vorher gelöscht.

  >>datei — Die Standard-Ausgabe des angegebenen Programms wird auf *datei* umgelenkt. In *datei* enthaltene Daten werden nicht gelöscht; die Ausgaben des Programms werden an den bisherigen Inhalt von *datei* angehängt.

2>datei Die Standard-Fehlerausgabe des angegebenen Programms wird auf *datei* umgelenkt.

<datei Die Standard-Eingabe des angegebenen Programms wird auf *datei* umgelenkt. Das Programm liest also seine Eingabe aus dieser Datei.

Format 2 **Standard-Eingabe bzw. -Ausgabe der Shell umlenken**

exec␣umlenkung

umlenkung

weist Kommandos, die von der Standard-Eingabe lesen bzw. auf die Standard-Ausgabe schreiben, eine Datei für die Eingabe bzw. Ausgabe zu. Diese Umlenkung gilt für alle Kommandos, die die aktuelle Shell nach *exec* ausführt.
Folgende Angaben für *umlenkung* sind möglich:

>datei Alle Kommandos, die die aktuelle Shell ausführt, schreiben ihre Ausgabe nacheinander in die angegebene Datei. In *datei* enthaltene Daten werden gelöscht.
So werden alle Ausgaben gesammelt.

Wenn Sie *exec* im Dialog eingeben, können Sie die Umlenkung nur rückgängig machen:

- terminalabhängig mit der Taste END oder @@d. Damit beenden Sie die aktuelle Shell.
- mit der Eingabe *exec >/dev/tty*. Damit lenken Sie die Standard-Ausgabe wieder auf den Bildschirm um.

Steht *exec* in einer Shell-Prozedur, so gilt die Umlenkung nur für die Kommandos, die in der Prozedur auf den Aufruf von *exec* folgen. Wollen Sie die Umlenkung innerhalb der Prozedur rückgängig machen, müssen Sie in der Prozedur an der betreffenden Stelle das Kommando *exec >/dev/tty* einfügen. Damit lenken Sie die Standard-Ausgabe der nachfolgenden Kommandos wieder auf den Bildschirm um.

>>datei Alle Kommandos, die die aktuelle Shell ausführt, schreiben ihre Ausgabe nacheinander in die angegebene Datei. In *datei* enthaltene Daten werden nicht gelöscht; die Ausgaben der Kommandos werden an den bisherigen Inhalt von *datei* angehängt.

2>datei Die Standard-Fehlerausgabe wird für alle Kommandos, die die aktuelle Shell ausführt, auf *datei* umgelenkt.

<datei Wenn Sie *exec* im Dialog eingeben, liest die aktuelle Shell die Kommandos, die sie ausführen soll, aus der angegebenen Datei. Nach dem letzten Kommando beendet sich die Shell.

Steht dieses Kommando in einer Shell-Prozedur, so lesen alle Kommandos, die in der Prozedur auf den Aufruf von *exec* folgen, ihre Eingabe aus der angegebenen Datei. Jeder Lesevorgang verändert die Position des Lesezeigers in dieser Datei. Wenn der Lesezeiger auf Dateiende (EOF) steht, erhalten alle noch folgenden Kommandos keine Eingabe.

Mit dem Kommando *exec </dev/tty* lenken Sie die Standard-Eingabe für die nachfolgenden Kommandos wieder auf die Tastatur um.

datei
Wenn Sie *exec* im Dialog eingeben, muss das der Name einer Shell-Prozedur sein. Für diese Shell-Prozedur brauchen Sie nur Leserecht.

Wenn *exec* in einer Shell-Prozedur steht, ist das der Name der Datei, aus der die nachfolgenden Kommandos ihre Eingabe lesen sollen.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *exec*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Was passiert bei folgender Eingabe:

```
$ exec date
```

Das Kommando *exec* beendet zuerst die aktuelle Shell und führt dann das Kommando *date* aus. Am Bildschirm wird das aktuelle Datum mit Uhrzeit ausgegeben.

Wenn Sie das Kommando in Ihrer POSIX-Shell eingegeben haben, wird anschließend der BS2000-Prompt ausgegeben. Um wieder mit der POSIX-Shell arbeiten zu können, müssen Sie erneut das Kommando *START-POSIX-SHELL* eingeben.

Wenn Sie das Kommando in einer Subshell eingegeben haben, meldet sich die übergeordnete Shell zurück.

Beispiel 2 Wenn Sie die aktuelle Shell durch ein anderes Programm überlagern, ändert sich die Prozessnummer nicht. Das soll mit den folgenden Dateien gezeigt werden:

– Die Datei *proz1* hat folgenden Inhalt:

```
echo proz1 hat Prozessnummer: $$
sh proz2
```

– Die Datei *proz2* hat folgenden Inhalt:

```
echo proz2 hat Prozessnummer: $$
exec proz3
```

– Die Datei *proz3* - sie muss ausführbar sein - hat folgenden Inhalt:

```
echo proz3 hat Prozessnummer: $$
```

Die Shell-Prozedur *proz1* wird gestartet:

```
$ sh proz1
proz1 hat Prozessnummer: 2755
proz2 hat Prozessnummer: 2760
proz3 hat Prozessnummer: 2760
```

Weil *proz3* von *proz2* mit *exec* aufgerufen wird, laufen beide Shell-Prozeduren unter derselben Prozessnummer. Genauer: Die Shell, die die Shell-Prozedur *proz2* ausführt, wird überlagert von der Shell, die *proz3* ausführt.

Beispiel 3 Mit dem Kommando *exec* wird in der Shell-Prozedur *exctest* die Standard-Eingabe für die nachfolgenden Kommandos umgelenkt auf die Datei */etc/group*. Die Datei *exctest* hat folgenden Inhalt:

```
: Aufruf mit sh exctest
exec </etc/group
read zeile
echo $zeile
echo
cat
```

Die Shell-Prozedur *exctest* wird gestartet:

```
$ sh exctest
root::0:root

daemon::1:daemon
sys::2:sys:
bin::3:bin,admin
uucp::4:
.
.
.
```

Das eingebaute *sh*-Kommando *read* weist der Variablen *zeile* die erste Zeile der Datei */etc/group* zu. Das Kommando *echo $zeile* gibt den Inhalt dieser Variablen aus.

Das Kommando *cat* liest seine Eingabe ebenfalls aus der Datei */etc/group*. Da der Lesezeiger jetzt auf der zweiten Zeile dieser Datei steht, gibt *cat* den Inhalt erst ab Zeile zwei aus.

Anschließend steht der Lesezeiger auf Dateiende. Weitere Kommandos nach dem Aufruf von *cat* würden deshalb als Eingabe die leere Zeichenkette erhalten.

Siehe auch *exec()* [4]

exit Shell-Prozedur beenden (cause the shell to exit)

Das in die POSIX-Shell *sh* eingebaute Kommando *exit* beendet Shell-Prozeduren. Beim Aufruf können Sie mit einer Zahl den Endestatus der entsprechenden Shell-Prozedur festlegen.

Auch ohne Aufruf von *exit* beenden sich Shell-Prozeduren:
- wenn das letzte Kommando ausgeführt ist, d.h. wenn die ausführende Shell Datei-Ende (EOF) erkennt. Endestatus ist der Endestatus des zuletzt ausgeführten Kommandos.
- wenn die ausführende Shell ein Kommando nicht findet oder einen Syntax-Fehler erkennt. Endestatus ist ein Wert ≠ 0.

Wenn Sie das Kommando *exit* im Dialog eingeben, wird entweder die aktuelle Shell (auch Subshell) oder die POSIX-Shell beendet.

Syntax

exit[␣n]

n Eine Zahl zwischen 0 und 255. Das ist der Endestatus, mit dem die Shell-Prozedur beendet wird.
Wurde *n* gesetzt, aber der Wert liegt nicht zwischen 0 und 255, so ist der Endestatus undefiniert.

Wie auch in anderen Abschnitten beschrieben wird, sind bestimmte Werte für den Endestatus für besondere Zwecke reserviert. Sie dürfen von Anwendungen auch nur für diese Zwecke verwendet werden:

126: Es wurde ein auszuführendes Kommando gefunden, aber die Datei war keine ausführbare Datei.

127: Es wurde kein auszuführendes Kommando gefunden.

>128: Ein Kommano wurde durch ein Signal unterbrochen.

trap für EXIT wird ausgeführt, bevor sich die Shell beendet. Dies ist nur dann nicht der Fall, wenn *exit* selbst in diesem *trap* aufgerufen wurde. In diesem zweiten Fall wird die Shell sofort beendet.

Mit dem Kommando *echo $?* können Sie den Endestatus abfragen.

n nicht angegeben:
Endestatus ist der Endestatus des Kommandos, das vor dem Aufruf von *exit* ausgeführt wurde.

Endestatus Wenn der Endestatus gesetzt wird, ist er *n*. Andernfalls ist der Endestatus der Endestatus des zuletzt ausgeführten Kommandos bzw. Null, wenn kein Kommando ausgeführt worden ist. Wenn *exit* im Rahmen eines *trap* ausgeführt wird, wird als zuletzt ausgeführtes Kommando das Kommando angenommen, das unmittelbar vor Ausführung des *trap* ausgeführt wurde.

Beispiel 1 Das Kommando *false* lässt sich durch folgende Shell-Prozedur realisieren:

```
: Shell-Version des Kommandos false
: Endestatus immer 1
exit 1
```

Beispiel 2 Die beiden Shell-Prozeduren *ende* und *abfrage* sollen zeigen, wie der Endestatus ausgewertet werden kann:

– Die Datei *ende* hat folgenden Inhalt:

```
: Aufruf mit sh ende datei
if [ -f "$1" ]
then exit 2
elif [ -d "$1" ]
then exit 3
else exit 4
fi
```

– Die Datei *abfrage* hat folgenden Inhalt:

```
: Aufruf mit sh abfrage datei
sh ende "$1"
case $? in
     2) echo Inhalt der Datei $1:; cat $1;;
     3) echo Inhalt von $1:; ls -l $1
     4) echo Die Eingabe $1 hat fuer diese Prozedur keinen Sinn!
esac
```

Die Shell-Prozedur *abfrage* wird gestartet:

```
$ sh abfrage $HOME/.profile
Inhalt der Datei .profile:
HOME=/home1/rosa/src
export HOME
...
```

Die Shell-Prozedur *abfrage* ruft die Prozedur *ende* auf. Diese Prozedur liefert als Endestatus den Wert 2 zurück, da *.profile* eine einfache Datei ist. Anschließend werden in der *case*-Anweisung die Kommandos ausgeführt, die bei dem entsprechenden Muster 2 angegeben sind.

Den Inhalt dieser beiden Shell-Prozeduren kann man nicht ohne weiteres in eine einzige Datei schreiben, denn das Kommando *exit* beendet die Prozedur.

Die Shell-Prozedur *endeaktion* zeigt, wie der Endestatus innerhalb einer Prozedur ausgewertet werden kann:

```
: Aufruf mit sh endeaktion datei
(if [ -f "$1" ]
   then exit 2
   elif [ -d "$1" ]
         then exit 3
         else exit 4
 fi)
 case $? in
      2) echo Inhalt der Datei $1:; cat $1;;
      3) echo Inhalt von $1:; ls -l $1;;
      4) echo Die Eingabe $1 hat fuer diese Prozedur keinen Sinn!
 esac
```

Die runden Klammern um die if-Anweisung bewirken, dass eine Subshell aufgerufen wird. Diese Subshell beendet sich mit dem Endestatus, der bei dem betreffenden Kommando *exit* angegeben ist. Anschließend meldet sich die übergeordnete Shell zurück; das ist die Shell, die die Prozedur *endeaktion* bearbeitet. Sie führt die restlichen Kommandos der Prozedur aus.

Siehe auch *false*
exit() [4]

expand Tabulatorzeichen in Leerzeichen umwandeln (convert tabs to spaces)

Das Kommando *expand* schreibt Dateien bzw. die Standard-Eingabe auf die Standard-Ausgabe. Tabulatorzeichen werden dabei durch ein oder mehrere Leerzeichen ersetzt, die zum Auffüllen der Zeile bis zum nächsten Tabulatorstopp erforderlich sind. Alle Backspace-Zeichen werden auf die Ausgabe kopiert und verringern dabei den Spaltenzähler zur Berechnung der Tabulatorstopposition jeweils um 1. Der Spaltenzähler kann dabei nicht kleiner als null werden.

Syntax

Format 1: expand[␣**-t**␣tabliste][␣datei...]

Format 2: expand[␣-tabstop |␣-tab1,tab2,...,tabn][␣datei...]

Format 1 **expand**[␣**-t**␣tabliste][␣datei...]

-t␣tabliste
: Gibt die Tabulatorstopps an. Das Argment *tabliste* muss aus einer oder mehreren Zahlen in aufsteigender Reihenfolge bestehen, die durch Leerzeichen oder Kommas getrennt werden. Eine durch Leerzeichen getrennte Liste muss dabei in Anführungszeichen gesetzt werden. Wenn nur eine Zahl angegeben ist, werden die Standard-Tabulatorstopps nicht alle 8 Spaltenpositionen gesetzt, sondern alle *tabliste* Spaltenpositionen. Sind mehrere Zahlen angegeben, werden die Tabulatorstopps an den angegebenen Spaltenpositionen gesetzt.

  Jede Tabulatorstopposition N muss ein ganzzahliger Wert größer null sein, und die Angaben müssen unbedingt in aufsteigender Reihenfolge erfolgen. Dies bedeutet, dass beim Springen mit der Tabulatortaste vom Anfang der Ausgabezeile zu Position N die nächste Zeichenausgabe in der (N+1)ten Spaltenposition in der Zeile erfolgt.

  Wenn das Kommando *expand* ein Tabulatorzeichen an einer Position hinter der letzten Position verarbeiten muss, die in einer Liste mit mehreren Tabulatorstopps definiert wurde, wird das Tabulatorzeichen in der Ausgabe durch ein Leerzeichen ersetzt.

datei
: Die Datei, deren Tabulatorzeichen durch Leerzeichen ersetzt werden sollen.

Format 2 **expand**[␣-tabstop |␣-tab1,tab2,...,tabn][␣datei...]

-tabstop |␣-tab1,tab2,...,tabn
Gibt die Tabulatorstopps an. Soll nur ein Tabulatorzeichen ersetzt werden, so wird *tabstop* als eine Zahl mit einem führenden Minuszeichen angegeben. Sollen mehrere Tabulatorzeichen ersetzt werden, so werden sie nacheinander, durch Komma getrennt, als *tab1, tab2, ..., tabn* angegeben. Das führende Minuszeichen muss nur einmal, vor *tab1*, angegeben werden.

datei
Die Datei, deren Tabulatorzeichen durch Leerzeichen ersetzt werden sollen.

Standard-Ausgabe (stdout)

Die Standard-Ausgabe entspricht den Eingabedateien, wobei jedoch Tabulatorzeichen in die entsprechende Anzahl von Leerzeichen umgewandelt wurden.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *expand*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien). Außerdem wird die internationale Umgebung für die Verarbeitung von Tabulator- und Leerzeichen sowie für die Angabe der Spalten festgelegt, die jedes Zeichen auf einem Ausgabegerät mit konstant breiter Schriftart einnimmt.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *tabs*, *unexpand*

export Shell-Variablen exportieren (set export attribute for variables)

Das in die POSIX-Shell *sh* eingebaute Kommando *export* exportiert die angegebene Shell-Variable. Jedes Kommando kennt dann den Namen dieser Variablen und kann auf ihren Wert zugreifen.

Wenn Sie die Shell beenden, in der Sie Variablen definiert und exportiert haben, sind diese Variablen wieder gelöscht. Deshalb sollten Sie häufig verwendete Variablen in Ihrer Datei *$HOME/.profile* definieren und exportieren.

Stellungsparameter und Shell-Funktionen können nicht exportiert werden. Einige Standard-Variablen der Shell sind auch in Subshells verfügbar, ohne exportiert worden zu sein. Dazu gehören z.B. *HOME*, *IFS*, *PATH*, *PS1*, *PS2*, usw. Wenn Sie einer dieser Variablen einen anderen Wert zuweisen und der geänderte Wert auch in jeder Subshell gelten soll, dann müssen Sie diese Variable mit *export* exportieren. Sonst gilt in jeder Subshell wieder der Standard-Wert.

Das eingebaute *sh*-Kommando *set* gibt alle Variablen mit ihrem jeweiligen Wert aus, die in der aktuellen Shell definiert sind, also auch solche, die Sie nicht exportiert haben.
Das Kommando *export* gibt die Namen und Werte aller Shell-Variablen aus, die an jedes aufgerufene Kommando und jede Subshell weitergereicht werden.

Syntax

Format 1: export[␣name[=wert]]...

Format 2: export␣**-p**

Format 1 **export**[␣name[=wert]]...

name[=wert]
: Name der Shell-Variablen, die Sie exportieren wollen. Sie können dieser Variablen auch erst nach dem Aufruf von *export* einen Wert *wert* zuweisen.
Sie können beliebig viele Shell-Variablen angeben, jeweils getrennt durch ein Leerzeichen.

  name nicht angegeben:
  export schreibt die Namen aller Shell-Variablen auf die Standard-Ausgabe, die in der aktuellen Shell exportiert wurden. Die Ausgabe hat folgende Form:

```
name=wert
...
```

Format 2 **export␣-p**

-p Wird *-p* gesetzt, so gibt *export* die Namen und die Werte aller Variablen, die exportiert wurden, im folgenden Format auf der Standard-Ausgabe aus:

„`export %s=%s\n“,` *name*`,` *wert*

Die Option *-p* ermöglicht einen übertragbaren Zugriff auf die Werte, die gesichert und später wiederhergestellt werden können (z.B. mit einer Punkt-Prozedur).

Die Shell formatiert die Ausgabe und berücksichtigt dabei die richtige Verwendung der Anführungszeichen. Daher ist die Ausgabe für eine Wiedereingabe an die Shell bei Kommandos, die die gleichen Export-Ergebnisse erzielen, geeignet.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *export*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Shell-Prozedur *ps1* soll den Wert der Variablen *PS1* ausgeben. Die Prozedur hat folgenden Inhalt:

```
: Aufruf mit sh ps1
echo PS1: $PS1
```

Der folgende Bildschirm-Dialog soll zeigen, warum die Shell-Variable *PS1* exportiert werden muss:

```
$ export
export HOME
export PATH
$ PS1=hello
$ sh ps1
PS1:
$ export PS1
$ export
export HOME
export PATH
export PS1
$ sh ps1
PS1: hello
```

Wenn Sie eine Shell-Variable definieren, ist sie nur der aktuellen Shell bekannt. Da jede Shell-Prozedur von einer Subshell ausgeführt wird, muss eine Variable für Shell-Prozeduren exportiert werden.

Siehe auch *env*, *set*

expr Ausdrücke auswerten (evaluate arguments as an expression)

expr interpretiert die in der Kommandozeile angegebenen Argumente als Ausdrücke und wertet sie nacheinander aus. Das Ergebnis der Auswertung wird auf die Standard-Ausgabe ausgegeben.
Mit *expr* können Sie z.B. ganzzahlige Berechnungen durchführen oder Zeichenketten miteinander vergleichen.

Syntax

expr␣ausdruck␣...

ausdruck

ausdruck besteht entweder nur aus einem Operanden oder aus zwei Operanden, die mit einem Operator verknüpft sind. Als Operanden können Sie beliebige Zeichenketten angeben. Zeichenketten, die nur aus den Ziffern 0 bis 9 bestehen, interpretiert *expr* als ganze Zahlen. Durch ein vorangestelltes Minuszeichen - können Sie ganze Zahlen als negative Zahlen kennzeichnen.

Operanden und Operatoren werden durch ein Trennzeichen voneinander getrennt. Als Trennzeichen wird der Wert der Variablen *IFS* verwendet. Wenn Sie eine Zeichenkette als Operand angeben wollen, die Leer- oder Tabulatorzeichen enthält, müssen Sie deshalb entweder die ganze Zeichenkette oder die Leer- oder Tabulatorzeichen in Hochkommas '...' oder Anführungszeichen "..." einschließen, damit diese nicht als Trennzeichen interpretiert werden. Sonderzeichen der Shell müssen Sie in Hochkommas '...' oder Anführungszeichen "..." einschließen oder mit einem Gegenschrägstrich \ entwerten (siehe *IFS* im Abschnitt „POSIX-Shell-Variablen und Parameter-Ersetzung“ auf Seite 54).

Wenn *ausdruck* nur aus einem Operanden besteht, ist das Ergebnis der Auswertung dieser Operand selber.

Wie zwei Operanden miteinander verknüpft werden können, ist im folgenden beschrieben. Dabei sind die Operatoren nach aufsteigendem Vorrang geordnet, Operatoren mit gleichem Vorrang sind in geschweiften Klammern {...} zusammengefasst.

Das Ergebnis 0 steht für den Wert 0, nicht für eine leere Zeichenkette.

Verknüpfung von Operatoren

op1␣**|**␣op2

Wenn *op1* weder die leere Zeichenkette noch 0 ist, ist das Ergebnis *op1*; sonst *op2*.

op1␣**&**␣op2

Wenn weder *op1* noch *op2* die leere Zeichenkette oder 0 ist, ist das Ergebnis *op1*; sonst 0.

op1␣rel␣op2

rel kann einer der folgenden Vergleichsoperatoren sein:

- < kleiner als
- <= kleiner als oder gleich
- = gleich
- >= größer als oder gleich
- > größer als
- != ungleich

Wenn die Bedingung erfüllt ist, ist das Ergebnis des Vergleichs 1. Wenn die Bedingung nicht erfüllt ist, ist das Ergebnis des Vergleichs 0. Wenn sowohl *op1* als auch *op2* ganze Zahlen sind, werden sie numerisch miteinander verglichen. Ansonsten werden sie als Zeichenketten lexikalisch miteinander verglichen.

op1␣{**+ -**}␣op2

Wenn *op1* und *op2* ganze Zahlen sind, ist das Ergebnis die Summe bzw. Differenz der beiden Zahlen. Wenn eines der Argumente keine ganze Zahl ist, gibt *expr* eine Fehlermeldung aus (siehe *Fehler*).

op1␣{*** / %**}␣op2

Wenn *op1* und *op2* ganze Zahlen sind, ist das Ergebnis das Ergebnis der angegebenen arithmetischen Operation:

- * Multiplikation
- / (ganzzahlige) Division
- % Rest bei (ganzzahliger) Division

Wenn eines der Argumente keine ganze Zahl ist, gibt *expr* eine Fehlermeldung aus (siehe *Fehler*).

op1␣**:**␣op2

Die Zeichenketten *op1* und *op2* werden miteinander verglichen, beginnend mit dem ersten Zeichen in jeder Zeichenkette und endend mit dem letzten Zeichen in *op2*. *op2* können Sie in Form eines einfachen regulären Ausdrucks (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*) angeben. Wenn *op1* und *op2* vom ersten Zeichen in jeder Zeichenkette bis zum letzten Zeichen in *op2* übereinstimmen, wird normalerweise die Anzahl der übereinstimmenden Zeichen ausgegeben. Wenn Sie für *op2* jedoch das Muster \(...\) verwenden, wird der Teil von *op1* ausgegeben, der zu diesem Muster passt (siehe *Beispiele 6* und *7*).

(...**)**

Durch Setzen runder Klammern *(...)* können Sie die Auswertungsreihenfolge verändern. Die in runden Klammern zusammengefassten Ausdrücke werden zunächst ausgewertet, auch wenn der Vorrang der Operatoren etwas anderes aussagt.

Endestatus

0 Ausdruck ist weder fehlerhaft noch leer noch 0

1 Ausdruck ist leer oder 0

2 Ausdruck ist fehlerhaft oder Division durch 0

Fehler

`expr: Non-numeric argument`
Bei den arithmetischen Operatoren +, –, *, /, % dürfen Sie nur ganzzahlige Operanden angeben.

`expr: Division by zero`
Sie haben durch 0 dividiert.

`expr: Syntax error`
Sie haben einen Syntaxfehler beim Aufruf von *expr* gemacht, z.B. Operanden und Operatoren nicht durch Leer- oder Tabulatorzeichen getrennt.

`expr: RE error`
Sie haben beim Mustervergleichsoperator : den zweiten Operanden nicht in korrekter Form als einfachen regulären Ausdruck angegeben.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *expr*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Bestimmt in geklammerten regulären Ausdrücken die Bedeutung von Zeichenbereichen, Äquivalenzklassen und Zeicheneinheiten. *LC_COLLATE* bestimmt zudem das Verhalten von Vergleichsoperatoren beim Vergleich von Zeichenketten. Wenn z.B. die Buchstaben *a* und *ä* in einer Äquivalenzklasse sind, ergibt der Ausdruck `expr $var:'[[=a=]]` den Wert 1, wenn der Wert der Variablen *a* oder *ä* ist.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
: Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH
: Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Einfache Rechnung:

```
$ expr 21 + 9 '*' 2 / 6
24
```

Beispiel 2 Im folgenden Beispiel wird zu der Variablen *a* der Wert 1 addiert.

```
$ echo $a
3
$ a=`expr $a + 1`
$ echo $a
4
```

Beispiel 3 Im folgenden Beispiel werden zwei Umgebungsvariablen verglichen.

```
$ echo $op1 $op2
text1 text2
$ expr $op1 = $op2
0
```

Problematisch wird der Vergleich, wenn der Wert einer Variablen einen Operator für *expr* darstellt. Das Problem kann durch Verbindung der Variablen mit einem unproblematischen Zeichen gelöst werden.

```
$ echo $op1 $op2
= =
$ expr $op1 = $op2
expr: syntax error
$ expr X$op1 = X$op2
1
```

Beispiel 4 Ausgabe der Anzahl der Zeichen in *VAR*:

```
$ echo $VAR
Hallo
$ expr $VAR : '.*'
5
```

Beispiel 5 Vergleich zweier Zeichenketten:

```
$ expr mausoleum : maus
4
```

Beispiel 6 Vergleich zweier Zeichenketten, wobei das Muster als regulärer Ausdruck angegeben wird. Das Zeichen Dach ^ ist ein Sonderzeichen und muss deshalb durch einen Gegenschrägstrich \ entwertet werden.

```
$ expr abc : [\^d-f]
1
```

Beispiel 7 In der Variablen *a* ist der absolute Pfadname einer Datei abgelegt, etwa */usr/anna/parnassum/infinitum*. Um den einfachen Dateinamen, also *infinitum*, auszugeben, geben Sie ein:

```
$ expr $a : '.*/\(.*\)'
infinitum
```

Beispiel 8 Wenn in der Variablen *a* entweder der absolute Pfadname oder der einfache Dateiname einer Datei abgelegt ist, erhalten Sie den einfachen Dateinamen auf folgende Weise:

```
$ expr $a : '.*/\(.*\)' \| $a
```

false Endestatus ungleich 0 zurückgeben (return false value)

Das in die POSIX-Shell *sh* eingebaute Kommando *false* gibt einen Endestatus ungleich 0 zurück. Sie können damit in Shell-Prozeduren die Bedingung *falsch* erzeugen.
Analog erzeugen Sie mit dem Kommando *true* die Bedingung *wahr* (Endestatus gleich 0).

Syntax **false**

Endestatus ungleich 0

Beispiel Der Endestatus von *false* ist:

```
$ false
$ echo $?
1
```

Siehe auch *true*

fc Zugriff auf die History-Datei (process command history list)

Die abgesetzten Kommandos einer interaktiven POSIX-Shell werden in einer *History*-Datei gespeichert. Auf diese kann dann über das eingebaute Kommando *fc* zugegriffen werden. Dabei bietet *fc* folgende Möglichkeiten:

– editieren einzelner Kommandozeilen aus der *History*-Datei vor deren Ausführung (Format 1)
– auflisten der Kommandozeilen auf Standard-Ausgabe (Format 2)
– Änderung des Kommandos vor der erneuten Ausführung ohne Editieren der *History*-Datei (Format 3)

Syntax

Format 1: fc[␣**-r**][␣**-e**␣editor][␣erstes[␣letztes]]

Format 2: fc␣**-l**[␣**-nr**][␣erstes[␣letztes]]

Format 3: fc␣**-s**[␣alt**=**neu][␣erstes]

Format 1 **fc**[␣**-r**][␣**-e**␣editor][␣erstes[␣letztes]]

-r Die Kommandos werden in umgekehrter Reihenfolge zur Verfügung gestellt.

-e␣editor
Durch *editor* wird der zu benutzende Editor bezeichnet. Fehlt diese Angabe, wird der Wert der Variablen *FCEDIT* als Editor verwendet.
Ist *FCEDIT* nicht gesetzt, dann wird standardmäßig */usr/bin/ed* eingesetzt.

erstes[␣letztes]
Aus den letzten *HISTSIZE* Kommandos, die an der Datensichtstation eingegeben wurden, wird der Bereich von *erstes* bis *letztes* Kommando zum Editieren und anschließenden Ausführen ausgewählt. Die Argumente *erstes* und *letztes* können Sie als Zahl oder als Zeichenkette angeben:

[+]zahl	Eine positive Zahl steht für die Nummer des Kommandos. Die Nummer des Kommandos wird mit der Option *-l* ausgegeben.
-zahl	Eine negative Zahl wird als Differenz zur Nummer des aktuellen Kommandos interpretiert. So ist beispielsweise *-1* das unmittelbar vorhergehende Kommando.
string	Eine Zeichenkette wird dazu benutzt, das zuletzt ausgeführte Kommando, das mit dieser Zeichenkette beginnt, auszuwählen.

erstes und *letztes* angegeben
alle Kommandos im Interval von *erstes* bis *letztes* werden zum Editieren ausgewählt. Ist *erstes* ein neueres Kommando als *letztes*, werden die Kommandos in umgekehrter Reihenfolge editiert (analog zur Option *-r*).

letztes nicht angegeben
es wird *erstes* verwendet.

erstes und *letztes* nicht angegeben
es wird das letzte Kommando editiert.

Format 2 **fc␣-l**[␣**-nr**][␣erstes[␣letztes]]

-l Die Kommandos und Kommandonummern werden nur auf die Standard-Ausgabe geschrieben und können nicht editiert und ausgeführt werden.

-n Die Nummern der Kommandos werden beim Schreiben durch die Option *-l* unterdrückt.

-r Die Kommandos werden in umgekehrter Reihenfolge zur Verfügung gestellt.

erstes[␣letztes]
Aus den letzten *HISTSIZE* Kommandos, die an der Datensichtstation eingegeben wurden, wird der Bereich von *erstes* bis *letztes* Kommando zum Editieren und anschließenden Ausführen ausgewählt. Die Argumente *erstes* und *letztes* können Sie als Zahl oder als Zeichenkette angeben:

[+]zahl Eine positive Zahl steht für die Nummer des Kommandos. Die Nummer des Kommandos wird mit der Option *-l* ausgegeben.

-zahl Eine negative Zahl wird als Differenz zur Nummer des aktuellen Kommandos interpretiert. So ist beispielsweise *-1* das unmittelbar vorhergehende Kommando.

string Eine Zeichenkette wird dazu benutzt, das zuletzt ausgeführte Kommando, das mit dieser Zeichenkette beginnt, auszuwählen.

erstes und *letztes* angegeben
Alle Kommandos im Interval von *erstes* bis *letztes* werden aufgelistet. Ist *erstes* ein neueres Kommando als *letztes*, werden die Kommandos in umgekehrter Reihenfolge aufgelistet (analog zur Option *-r*).

letztes nicht angegeben
Es wird das vorhergehende Kommando verwendet.

erstes und *letztes* nicht angegeben
Es werden die letzten 16 Kommandos ausgegeben.

Format 3 **fc**␣**-s**[␣alt**=**neu][␣erstes]

-s Die Kommandos werden wieder ausgeführt. Es wird kein Editor aufgerufen.

alt=neu
Änderung des Kommandos vor der Ausführung. Nach der Ersetzung der Zeichenkette *alt* durch *neu* wird das Kommando erneut ausgeführt.

erstes
Alle Kommandos ab *erstes* werden ausgeführt. Beschreibung siehe Format 2.

erstes nicht angegeben
das vorhergehende Kommando wird benutzt.

Fehler `sh: /bin/ed: not found`
Die Shell-Variable *FCEDIT* ist nicht auf den Wert *ed* gesetzt.

Variable *HISTFILE*
Ist diese Variable bei Aufruf der POSIX-Shell gesetzt, dann wird der Wert als Pfadname für die Datei zur Speicherung der Kommando-history verwendet (siehe Abschnitt „Kommando-Wiederaufruf“ auf Seite 69).

HISTSIZE
Wenn diese Variable bei Aufruf der POSIX-Shell gesetzt ist, dann behält die Shell den Text eingegebener Kommandos in Erinnerung. Sie können mindestens auf die, als Wert angegebene, Anzahl von früher eingegebenen, und der POSIX-Shell zugänglichen, Kommandos zurückgreifen. Der Standard-Wert ist 128.

FCEDIT
Der Name des Standard-Editors für das eingebaute Kommando *fc*.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *fc*:

LANG	Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.
LC_ALL	Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.
LC_CTYPE	Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
: Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH
: Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Auflisten der letzten 16 Kommandos

```
$ ls -l
$ fc -s ls=fc
:
```

ist identisch zu

```
$ fc -l
```

Beispiel 2 Vorhergehendes Kommando nochmals ausführen

```
$ fc -s
```

ist identisch zu

```
$ fc -s -- -1
```

fg Jobs in den Vordergrund bringen (run jobs in the foreground)

Mit dem in die POSIX-Shell *sh* eingebauten Kommando *fg* können Sie Aufträge in den Vordergrund bringen.

Syntax **fg**[␣job-id...]

job-id
Jeder der angegebenen Aufträge wird in den Vordergrund gebracht. Der Abschnitt *Aufträge* im Kapitel „Kommandos von der POSIX-Shell aus eingeben“ enthält eine Beschreibung des Formats von *job-id*.

job-id nicht angegeben: Der aktuelle Auftrag wird in den Vordergrund gebracht.

Endestatus

0 bei erfolgreicher Ausführung

>0 bei Auftreten eines Fehlers

Internationale Umgebung

Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *fg* aus:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Der Job %1 soll in den Vordergrund gebracht werden:

```
$ fg %1
```

Siehe auch *bg, jobs, kill, wait*

fgrep Zeichenketten suchen (search a file for a fixed-string pattern)

fgrep liest Zeilen aus einer oder mehreren Dateien oder von der Standard-Eingabe und durchsucht die Zeilen nach Zeichenketten. Ist mittels Optionen nichts anderes angegeben, so schreibt *fgrep* alle Zeilen, die eine der angegebenen Zeichenketten enthalten, auf die Standard-Ausgabe.
Geben Sie mehrere Eingabedateien an, dann wird jeder Ausgabezeile der Name der betreffenden Datei vorangestellt.

Syntax

Format 1: fgrep[␣**-bchilnrvxy**]␣**-e**␣musterliste[␣datei...]

Format 2: fgrep[␣**-bchilnrvxy**]␣**-f**␣musterdatei[␣datei...]

Format 3: fgrep[␣**-bchilnrvxy**]␣musterliste[␣datei...]

Die Formate werden gemeinsam beschrieben, da die Zeichenketten, nach denen *fgrep* suchen soll, entweder über *musterliste* oder über die Option *-e musterliste* oder *-f musterdatei* angegeben werden muss. Eines dieser Argumente müssen Sie angeben, zwei zusammen oder alle drei sind nicht erlaubt.

Keine Option angegeben
fgrep gibt alle Zeilen aus, die mindestens eine der in *liste* angegebenen Zeichenketten enthalten. Geben Sie mehrere Eingabedateien an, dann wird jeder Ausgabezeile der Name der Datei vorangestellt, aus der die Zeile gelesen wurde.

option

-b (b - block) Jeder Ausgabezeile wird die Nummer des Blocks vorangestellt, in dem sie enthalten ist.
Die Blöcke, aus denen eine Datei besteht, sind je 512 Byte groß und werden, mit 0 beginnend, durchnummeriert.
Die Option *-b* kann hilfreich sein, wenn die Nummern von Blöcken nach dem Kontext ermittelt werden sollen (siehe z.B. das Kommando *od*, Argument *offset*).

-c (c - count) *fgrep* gibt nur die Anzahl der gefundenen Zeilen aus (das sind die Zeilen, die *fgrep* ohne die Option *-c* ausgeben würde, siehe *Beispiel 4*); die Zeilen selbst werden nicht ausgegeben.

-h (h - hidden) Beim Durchsuchen mehrerer Eingabedateien unterlässt *fgrep* die Voranstellung der Dateinamen vor jeder Ausgabezeile.

-i oder **-y**
(i - ignore) *fgrep* unterscheidet beim Vergleich nicht zwischen Groß- und Kleinbuchstaben.

-l (l - list) *fgrep* gibt nur die Namen der Dateien aus, die mindestens eine der gefundenen Zeilen enthalten (das sind die Zeilen, die *fgrep* ohne die Option *-l* ausgeben würde, siehe *Beispiel 5*). Jeder Dateiname wird nur einmal ausgegeben. Die Zeilen selbst gibt *fgrep* nicht aus.

-n (n - number lines) Jeder Ausgabezeile wird die Zeilennummer aus der betreffenden Eingabedatei vorangestellt, wobei von 1 an nummeriert wird. Liest *fgrep* von der Standard-Eingabe, bezieht sich die Zeilennummer auf die Standard-Eingabe.

-r (r - recursive) Jeder Dateiname, der ein Verzeichnisname ist, wird rekursiv durchsucht. Das bedeutet, das alle Dateien und Dateiverzeichnisse, die in diesem Dateiverzeichnis enthalten sind, berücksichtigt werden.

-v (v - vice versa) *fgrep* gibt alle Zeilen aus, die *keine* der angegebenen Zeichenketten enthalten.

Zusammen mit Option *-c*: *fgrep* gibt nur die Anzahl solcher Zeilen aus.
Zusammen mit Option *-l*: *fgrep* gibt nur die Namen der Dateien aus, die solche Zeilen enthalten.

-x (x - exact) *fgrep* gibt nur solche Zeilen aus, die eine der angegebenen Zeichenketten und sonst keine weiteren Zeichen enthalten.

Zusammen mit Option *-c*: *fgrep* gibt nur die Anzahl solcher Zeilen aus.
Zusammen mit Option *-l*: *fgrep* gibt nur die Namen der Dateien aus, die solche Zeilen enthalten.

-e␣musterliste
(e - expression) Diese Option brauchen Sie, wenn der erste Ausdruck in *musterliste* mit einem Bindestrich - beginnt. Zusammen mit *-e* wird eine solche Musterliste nicht als Option interpretiert, sondern als Liste von Mustern, mit denen *fgrep* die Eingabezeilen vergleichen soll.

-f␣musterdatei
(f - file) *fgrep* liest die Suchzeichenketten aus der Datei *musterdatei*. Jede Zeile von *musterdatei* wird als eine Zeichenkette interpretiert.

musterliste
Liste von Zeichenketten, mit denen *fgrep* die Eingabezeilen vergleichen soll. Die Zeichenketten trennen Sie durch Neue-Zeile-Zeichen. Enthält *musterliste* Neue-Zeile-Zeichen oder sonstige Zeichen, die für die Shell eine Sonderbedeutung haben, dann schließen Sie *musterliste* in Hochkommas ein: '*musterliste*'.

datei
Name der Datei, die *fgrep* durchsuchen soll. Pro Aufruf können Sie mehrere Dateinamen angeben.

datei nicht angegeben:
fgrep liest die Eingabezeilen von der Standard-Eingabe.

grep, fgrep und egrep

Die Kommandos *grep*, *fgrep* und *egrep* sind von der Oberfläche her weitgehend identisch. Im Folgenden sind die wichtigsten Unterschiede zwischen diesen Kommandos aufgeführt.

grep verarbeitet einfache reguläre Ausdrücke. Pro Aufruf können Sie nur einen einzigen regulären Ausdruck angeben.

fgrep verarbeitet nur Zeichenketten. Pro Aufruf können Sie jedoch mehrere Zeichenketten angeben: Die Zeichenketten geben Sie entweder direkt in der Aufrufzeile, getrennt durch Neue-Zeile-Zeichen, an oder Sie übergeben sie in einer Datei.
fgrep kann effizient sehr viele Zeichenketten suchen: *fgrep* sucht jede einzelne Zeile nach allen Zeichenketten ab.

egrep verarbeitet erweiterte reguläre Ausdrücke. Diese umfassen u.a. die einfachen regulären Ausdrücke bis auf eine Ausnahme: Der einfache reguläre Ausdruck \(*regausdruck*\) hat bei erweiterten regulären Ausdrücken keine Sonderbedeutung und wird deshalb auch nicht von *egrep* verarbeitet.
Pro Aufruf können Sie mehrere reguläre Ausdrücke, durch Neue-Zeile-Zeichen getrennt, angeben. *egrep* interpretiert diese Neue-Zeile-Zeichen wie einen senkrechten Strich (Zeichen für die Alternative, siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Die regulären Ausdrücke geben Sie entweder direkt in der Aufrufzeile an oder Sie übergeben sie in einer Datei.

Endestatus

0 Zeilen gefunden

1 keine Zeile gefunden

>1 Syntaxfehler oder „Datei kann nicht geöffnet werden“. Dieser Endestatus gilt auch dann, wenn in anderen Eingabedateien Zeilen gefunden wurden

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *fgrep*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE beeinflusst die Sortierreihenfolge.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Grundlage für die Beispiele sind die Dateien *kunden1* und *kunden2*. Sie haben folgenden Inhalt:

kunden1:

```
080685    999.98  20 LE  Art.  038   Fa. Holzinger
120387   1240.25   3 LE  Art.  023   Fa. Wanninger
180588    330.87   1 LE  Art.  332   Fa. Wanninger
```

kunden2:

```
hinterhuber berta, rosenheim, zugspitzstr.1
wanninger herbert, muenchen, kirschstr.3
```

Beispiel 1 Zeilen ausgeben, die eine bestimmte Zeichenkette enthalten (keine Option und Option *-i*):

```
$ fgrep Wanninger kunden1 kunden2
kunden1:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180588    330.87   1 LE  Art.  332   Fa. Wanninger
```

Wenn Sie auch Zeilen mit kleingeschriebenem *wanninger* ausgeben lassen möchten, geben Sie ein:

```
$ fgrep -i wanninger kunden1 kunden2
kunden1:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180588    330.87   1 LE  Art.  332   Fa. Wanninger
kunden2:wanninger herbert, muenchen, kirschstr.3
```

Beispiel 2 Mehrere Zeichenketten suchen (keine Option und Option *-f*):

```
$ fgrep 'Wanninger
> Holzinger' kunden1 kunden2
kunden1:080685    999.98  20 LE  Art.  038   Fa. Holzinger
kunden1:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180588    330.87   1 LE  Art.  332   Fa. Wanninger
```

Sie können die beiden Zeichenketten auch in eine Datei *namen* schreiben (je Zeichenkette eine Zeile) und *fgrep* folgendermaßen aufrufen:

```
$ fgrep -f namen kunden1 kunden2
```

Beispiel 3 Zeilen ausgeben, die bestimmte Zeichenketten nicht enthalten (Option *-v*):

```
$ fgrep -v Wanninger kunden1 kunden2
kunden1:080685    999.98  20 LE  Art.  038   Fa. Holzinger
kunden2:hinterhuber berta, rosenheim, zugspitzstr.1
kunden2:wanninger herbert, muenchen, kirschstr.3
```

Beispiel 4 Anzahl der gefundenen Zeilen ausgeben (Option *-c*):

Zuerst soll für jede Eingabedatei die Anzahl der Zeilen, die die Zeichenkette *Wanninger* enthalten, ausgegeben werden.

```
$ fgrep -c Wanninger kunden1 kunden2
kunden1:2
kunden2:0
```

Nun soll die Anzahl der Zeilen, die die Zeichenkette *Wanninger* nicht enthalten, ausgegeben werden.

```
$ fgrep -c -v Wanninger kunden1 kunden2
kunden1:1
kunden2:2
```

Beispiel 5 Nur Dateinamen ausgeben (Option *-l*):

Zuerst sollen die Namen der Dateien, die die Zeichenkette *Wanninger* enthalten, ausgegeben werden.

```
$ fgrep -l Wanninger kunden1 kunden2
kunden1
```

Nun sollen die Namen der Dateien, die Zeilen ohne die Zeichenkette *Wanninger* enthalten, ausgegeben werden.

```
$ fgrep -l -v Wanninger kunden1 kunden2
kunden1
kunden2
```

Beispiel 6 Gefundene Zeilen mit Zeilennummer ausgeben (Option *-n*):

```
$ fgrep -n -i wanninger kunden1 kunden2
kunden1:2:120387   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:3:180588    330.87   1 LE  Art.  332   Fa. Wanninger
kunden2:2:wanninger herbert, muenchen, kirschstr.3
```

Siehe auch *ed*, *egrep*, *grep*, *sed*

file Art einer Datei bestimmen (determine file type)

file ermittelt in den als Argument angegebenen Dateien, die Art des Inhalts bzw. den Typ des Arguments. *file* unterscheidet z.B. Dateiverzeichnisse, Gerätedateien, FIFO-Dateien, Bibliotheken, C-Quellprogramme, ausführbare Programme, Shell-Prozeduren und normale Textdateien.

Achtung!
file prüft in der *magic*-Datei die *magic*-Nummer, um so die betreffende Datei zu identifizieren. Wird die Datei hierdurch nicht identifiziert, versucht *file* über Plausibilitätstest den Dateityp festzustellen. Dies führt jedoch nicht immer zu korrekten Ergebnissen.

Syntax

Format 1: file[␣**-h**][␣**-m**␣magicfile][␣**-f**␣fdatei]␣datei␣...

Format 2: file[␣**-h**][␣**-m**␣magicfile]␣**-f**␣fdatei[␣datei␣...]

Format 3: file␣**-c**[␣**-m**␣magicfile]

Art einer Datei bestimmen

Format 1 **file**[␣**-h**][␣**-m**␣magicfile][␣**-f**␣fdatei]␣datei␣...

Format 2 **file**[␣**-h**][␣**-m**␣magicfile]␣**-f**␣fdatei[␣datei␣...]

-h (h - hidden) Falls die zu überprüfende Datei ein symbolischer Verweis ist, so wird diesem nicht nachgegangen. *file* gibt in diesem Fall folgende Meldung aus:
datei1: symbolic link to *datei2*

-m␣magicfile
(m - magic) *file* benutzt statt der Systemdatei */etc/magic* die Datei *magicfile*, um die Dateiformatkennungen (magic numbers) der zu überprüfenden Dateien zu bestimmen.

-f␣fdatei
file interpretiert das Argument *fdatei* als Namen einer Datei, die die Namen aller zu überprüfenden Dateien enthält. Wenn Sie diese Option nicht angeben, dann müssen Sie wenigstens eine zu überprüfende Datei *datei* angeben.

datei
Name der Datei, die *file* überprüfen soll. Falls die Datei ein symbolischer Verweis ist, so folgt *file* diesem Verweis und wertet die Ursprungsdatei aus. Pro Aufruf können Sie mehrere Dateinamen angeben. Wenn Sie die Option *-f* nicht angeben, dann müssen Sie wenigstens eine zu überprüfende Datei *datei* angeben.

Format 3 **Überprüfung der magicfile**

file␣**-c**[␣**-m**␣magicfile]

-c (c - check) Standardmäßig überprüft *file* die Systemdatei */etc/magic* auf Formatfehler. Ist die Option *-m* angegeben, dann wird stattdessen die Datei *magicfile* überprüft.

-m␣magicfile
(m - magicfile) *file* überprüft die Datei *magicfile* auf Formatfehler.

Arbeitsweise

file führt für jede Eingabedatei eine Reihe von Tests durch und versucht so, den Dateiinhalt zu klassifizieren. Handelt es sich wahrscheinlich um eine Textdatei, dann prüft *file* deren Anfang (die ersten 512 Byte) und stellt Vermutungen an über die Sprache des Textes. Die Richtigkeit der Ausgabe ist jedoch nicht gewährleistet.

Enthält die Eingabedatei ein ausführbares Programm, dann erkennt *file* dies und gibt weitere Informationen über den Dateiinhalt aus. Dazu durchsucht *file* die Datei nach sogenannten magic numbers, d.h. nach numerischen Konstanten oder Zeichenkettenkonstanten, die Aufschluss über die Art des Dateiinhaltes geben. Eine Erläuterung zu diesen magic numbers steht in der Datei */etc/magic*.

Ausgabe

file schreibt den Typ des Dateiinhalts bzw. der Datei auf die Standard-Ausgabe. Nachfolgend sind Ausgabetext und Bedeutung der wichtigsten Klassifikationen des *file*-Kommandos aufgelistet.

ascii text: ASCII-Textdatei

block special: blockorientierte Gerätedatei

c program text: C-Quellprogramm

character special: zeichenorientierte Gerätedatei

commands text: Shell-Prozedur

cpio archive: Bibliothek, erzeugt mit cpio, pax

current ar archive: Bibliothek (siehe ar)

data: Datei mit Daten

directory: Dateiverzeichnis

executable: Datei ausführbar (z.B. LLMs)

empty file: leere Datei

fifo: FIFO-Datei

fortran program text: FORTRAN-Quellprogramm

tar archive: Bibliothek, erzeugt mit tar, pax

text: EBCDIC-Textdatei

Datei */etc/magic*
enthält Erläuterungen zu den magic numbers.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *file*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Datei *liste* enthält die folgenden Dateinamen:

```
dvz
brief
lib.a
prog.c
```

Mit folgendem Aufruf erhalten Sie Informationen über die Art des Dateiinhalts:

```
$ file -f liste
dvz:            directory
brief:          ascii text
lib.a:          current ar archive
prog.c:         c program text
```

dvz ist also ein Dateiverzeichnis, *brief* enthält normalen ASCII-Text. *lib.a* ist eine Bibliothek und *prog.c* enthält ein C-Quellprogramm.

Die gleiche Ausgabe erhalten Sie mit dem Aufruf

```
$ file dvz brief lib.a prog.c
```

Siehe auch *c89* [5]

find Dateiverzeichnisse durchsuchen (find files)

find durchsucht Dateiverzeichnisse und deren Unterdateiverzeichnisse nach Dateien, die angegebene Bedingungen erfüllen. Die gefundenen Dateien können Sie mit einem Kommando bearbeiten und/oder sich ihre Namen ausgeben lassen.
find sucht nicht nur nach einfachen Dateien, sondern auch nach Dateiverzeichnissen, Gerätedateien oder FIFOs.

Syntax **find**␣dateiverzeichnis␣ausdruck

dateiverzeichnis
: Name des Dateiverzeichnisses, das durchsucht werden soll. *find* durchsucht dieses Dateiverzeichnis sowie rekusiv alle seine Unterdateiverzeichnisse, sofern Sie für die Dateiverzeichnisse Lese- und Ausführrecht besitzen (r- und x-Bit). Sie können mehrere Namen angeben.

ausdruck
: Für *ausdruck* geben Sie entweder einen der unten aufgeführten elementaren Ausdrücke an oder mehrere elementare Ausdrücke, die Sie miteinander verknüpfen.

  find prüft *ausdruck* für jede Datei, die in einem der angegebenen Dateiverzeichnisse bzw. Unterdateiverzeichnisse steht, von links nach rechts ab. Ein elementarer Ausdruck ist für eine Datei entweder wahr oder falsch. Vom Wahrheitswert des Ausdrucks und dem darauffolgenden Verknüpfungsoperator hängt es ab, ob *find* den nächsten elementaren Ausdruck bearbeitet oder nicht.
  Die meisten elementaren Ausdrücke sind Bedingungen, d.h., sie sind wahr, wenn die aktuelle Datei die Bedingung erfüllt. Die Ausdrücke

  -print
  -exec kommando **\;**
  -ok kommando **\;**

  sind keine Bedingungen. Sie bewirken, dass eine Aktion ausgeführt wird, z.B. Ausgeben des Dateinamens oder Löschen der Datei. Wenn *find* einen dieser Ausdrücke bearbeitet, wird die zugehörige Aktion auf jeden Fall ausgeführt, egal, ob der Ausdruck den Wert „wahr“ oder „falsch“ liefert. Wenn Sie keinen der obigen Ausdrücke angeben, bricht *find* mit einer Fehlermeldung ab. Innerhalb von *ausdruck* müssen Sie Sonderzeichen der Shell entwerten, so dass die Shell sie nicht interpretiert, sondern an *find* übergibt.

Elementare Ausdrücke als Bedingungen

Im Folgenden sind die elementaren Ausdrücke beschrieben; weiter unten erfahren Sie, wie Sie elementare Ausdrücke miteinander verknüpfen.

Wenn Sie Optionen angeben, um Dateien mit bestimmten Eigenschaften herauszufinden (z.B. Dateien, die innerhalb einer bestimmten Zeit verändert wurden), so muss eine solche Option vor der Option *-print* (siehe unten) angegeben werden. Ansonsten werden alle Dateien ausgegeben.

Der Parameter *n* steht für eine Dezimalzahl; *+n* bedeutet mehr als *n*, *-n* bedeutet weniger als *n*, und *n* bedeutet genau *n*.

-name␣datei
: Wahr, wenn die aktuelle Datei den Namen *datei* hat.
Für *datei* können Sie die üblichen Sonderzeichen der Shell zur Dateinamen-Generierung verwenden, müssen dann aber *datei* in Hochkommas einschließen, z.B. *find /usr/adam -name 'gruppe.*'* (siehe *Tabellen und Verzeichnisse, Sonderzeichen der POSIX-Shell*). Sie dürfen nur einfache Dateinamen angeben (nicht erlaubt ist z.B. *text/tmp*).

-perm␣modus
: Wahr, wenn die aktuelle Datei die Zugriffsrechte *modus* hat (siehe *chmod*). *modus* kann auch eine Oktalzahl sein. Geben Sie vor *modus* oder der Oktalzahl ein Minuszeichen an, so werden nur die Bits mit den Zugriffsrechten verglichen, die in *modus* gesetzt sind, und der Ausdruck ist wahr, wenn sie übereinstimmen.

-type␣zeichen
: Wahr, wenn die aktuelle Datei vom Typ *zeichen* ist, wobei *zeichen* sein kann:
  - **f** für einfache Datei (ordinary file)
  - **d** für Dateiverzeichnis (directory)
  - **b** für blockorientierte Gerätedatei (block special file)
  - **c** für zeichenorientierte Gerätedatei (character special file)
  - **l** für symbolische Verweise (symbolic links)
  - **p** für FIFO (named pipe)

-fstype␣name
: Wahr, wenn das Dateisystem, zu dem die aktuelle Datei gehört, vom Typ *name* ist, wobei *name* das Dateisystem *ufs* oder die Spezial-Dateisysteme wie *proc* oder *fdfs* sein kann.

-links␣[+-]n
: Wahr, wenn auf die aktuelle Datei (mehr als / weniger als / genau) *n* Verweise bestehen.

-inum␣[+-]n
: Wahr, wenn der Index-Eintrag (inode) der aktuellen Datei größer, kleiner bzw. gleich *n* ist. Mit dem Ausdruck *-inum +n -inum -m* finden Sie alle Index-Einträge zwischen *n* und *m*.

-user benutzerkennung
Wahr, wenn die aktuelle Datei dem Benutzer *benutzerkennung* gehört. Diese Option ist nur für Benutzer mit uid=0 (Kommando *id*) sinnvoll.

-nouser
Wahr, wenn die aktuelle Datei einer nicht vorhandenen Benutzerkennung gehört. Diese Option ist nur für Benutzer mit uid=0 (Kommando *id*) sinnvoll.

-group gruppenname
Wahr, wenn die aktuelle Datei der Gruppe *gruppenname* gehört.

-nogroup
Wahr, wenn die aktuelle Datei einer Gruppe gehört, die nicht in der Datei */etc/group* als Gruppe eingetragen ist.

-size [**+-**]n[**c**]
c nicht angegeben:
Wahr, wenn die aktuelle Datei (mehr / weniger als / genau) *n* Blöcke zu je 512 Byte belegt.

c angegeben:
Wahr, wenn die aktuelle Datei (mehr / weniger als / genau) *n* Byte belegt.

-atime [**+-**]n
(a - access) Wahr, wenn auf die aktuelle Datei zuletzt vor (mehr als / weniger als / genau) *(n-1)* bis *n**24 Stunden zugegriffen wurde. Die Zugriffszeit der Dateiverzeichnisse vom bzw. im durchsuchten Dateiverzeichnis wird von *find* selbst verändert.

-mtime [**+-**]n
(m - modification) Wahr, wenn die aktuelle Datei zuletzt vor (mehr als / weniger als / genau) *(n-1)* bis *n**24 Stunden geändert wurde.

-ctime [**+-**]n
Wahr, wenn der Index-Eintrag (inode) der aktuellen Datei zuletzt vor (mehr als / weniger als / genau) *(n-1)* bis *n**24 Stunden geändert wurde.

-newer datei
Wahr, wenn die aktuelle Datei jünger als die angegebene *datei* ist, d.h., wenn sie zu einem späteren Zeitpunkt erstellt oder geändert wurde.

-local
Wahr, wenn sich die aktuelle Datei physikalisch auf dem lokalen Rechner befindet.

-prune
Immer wahr. Wird dieser Ausdruck bearbeitet, werden keine Dateien oder Dateiverzeichnisse durchsucht, die unterhalb der aktuellen Hierarchiestufe liegen.

-xdev
Immer wahr. Wird dieser Ausdruck bearbeitet, werden keine Dateien oder Dateiverzeichnisse durchsucht, die unterhalb von Verzeichnissen liegen, die eine verschiedene Gerätenummer haben (*st_dev*, siehe Funktion *stat()* [4]). Ist *-xdev* gesetzt, so gilt dies für den gesamten Ausdruck, auch wenn *-xdev* normalerweise nicht bearbeitet würde.

Elementare Ausdrücke, die Aktionen bewirken

-print
Immer wahr. Wird dieser Ausdruck bearbeitet, gibt *find* den Pfadnamen der aktuellen Datei auf die Standard-Ausgabe aus.

-exec kommando **\;**
kommando wird ausgeführt, sobald dieser Ausdruck bearbeitet wird.
kommando müssen Sie mit einem Leerzeichen und einem entwerteten Strichpunkt \; abschließen. Ein Paar geschweifter Klammern {} als Kommandoargument steht für den Namen der aktuellen Datei.

Der Ausdruck *-exec kommando* \; ist wahr, wenn *kommando* den Endestatus 0 liefert. Der Wahrheitswert ist von Bedeutung, wenn dieser Ausdruck mit weiteren Ausdrücken verknüpft ist.

-ok kommando **\;**
Arbeitet wie *-exec*, nur fragt *find* jedesmal, wenn der Ausdruck bearbeitet wird, ob das Kommando ausgeführt werden soll. Das Kommando wird nur dann ausgeführt, wenn Sie mit dem Zeichen bzw. der Zeichenkette antworten, die in der aktuell gültigen Umgebung „y[es]“ bedeutet (siehe Umgebungsvariable *LANG*).

Ausdrücke, die die Arbeitsweise von find beeinflussen

-depth
Immer wahr. Wird dieser Ausdruck bearbeitet, dann bearbeitet *find* die Einträge eines Dateiverzeichnisses eher als das Verzeichnis selbst.

-follow
Immer wahr. Wird dieser Ausdruck bearbeitet, dann wird symbolischen Verweisen nachgegangen. Dieser Ausdruck sollte nicht mit der Option *-type l* kombiniert werden.

-mount
Immer wahr. Wird dieser Ausdruck bearbeitet, dann beschränkt *find* seine Suche auf das Dateisystem, in dem sich das angegebene Dateiverzeichnis befindet.

Ausdrücke verknüpfen

Die oben beschriebenen Ausdrücke können Sie wie folgt miteinander verknüpfen; beachten Sie die Leerzeichen vor und nach den Operatoren!
Die Verknüpfungs-Operatoren sind nach Priorität in absteigender Reihenfolge geordnet.

\(␣ausdruck...␣**\)**
Klammern fassen mehrere Ausdrücke zusammen. Die Klammern müssen mit einem Gegenschrägstrich \ entwertet werden, da sie eine Sonderbedeutung für die Shell haben.

!␣ausdruck
Negation: wahr, wenn der Ausdruck falsch ist.

ausdruck␣ausdruck
logisches UND: wahr, wenn beide Ausdrücke wahr sind. Ist der erste Ausdruck falsch, dann bearbeitet *find* den zweiten Ausdruck nicht.

Beispiel

Ist der zweite Ausdruck
-exec kommando **\;**
dann wird das Kommando nur dann ausgeführt, wenn der erste Ausdruck wahr ist.

ausdruck␣**-o**␣ausdruck
logisches ODER: wahr, wenn einer der Ausdrücke oder beide wahr sind. Ist der erste Ausdruck wahr, dann bearbeitet *find* den zweiten Ausdruck nicht.

Beispiel

Ist der zweite Ausdruck
-exec kommando **\;**
dann wird das Kommando nur dann ausgeführt, wenn der erste Ausdruck falsch ist.

Fehler

Fehlermeldungen, die nicht zum Abbruch führen (Endestatus 0)

`find: cannot read dir` *dvz*`: Permission denied`
Sie haben für das Dateiverzeichnis *dvz* kein Leserecht. Das Dateiverzeichnis kann folglich nicht durchsucht werden. *find* bricht jedoch nach dieser Fehlermeldung nicht ab, sondern durchsucht die Dateiverzeichnisse, für die Sie die erforderlichen Zugriffsrechte (r-Bit) besitzen.

`find: cannot access` *dvz/file*`: permission denied`
Sie haben für das Dateiverzeichnis *dvz* kein Ausführrecht (x-Bit). Das Dateiverzeichnis kann folglich nicht durchsucht werden. *find* bricht jedoch nach dieser Fehlermeldung nicht ab, sondern durchsucht die Dateiverzeichnisse, für die Sie die erforderlichen Zugriffsrechte (r- und x-Bit) besitzen.

`find: cannot open` *dvz*`: No such file or directory`
Das Dateiverzeichnis *dvz* existiert nicht.

Fehlermeldungen, die zum Abbruch führen (Endestatus ungleich 0)

`find: missing conjunction`
Sie haben beim *find*-Aufruf die elementaren Ausdrücke nicht richtig miteinander verknüpft oder bei einem elementaren Ausdruck mehrere Argumente angegeben.

`find: no action specified`
Sie haben keine Aktion angegeben (*print, exec, ok*).

`find: incomplete statement`
Sie haben ein Kommando bei *-exec* oder *-ok* nicht mit entwertetem Strichpunkt \; abgeschlossen.

`find: insufficient number of arguments`
Sie haben zu wenige Argumente angegeben.

`find: path-list predicate-list`
Sie haben vergessen, die Pfadliste anzugeben.

Datei

/etc/group
Datei mit Gruppennamen

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *find*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Wenn Sie Dateinamen in regulären Ausdrücken verwenden (geklammerte Dateinamensmuster wie z.B. *find . -name '[[=a=]]*' -print*), bestimmt *LC_COLLATE* die Bedeutung von Zeichenbereichen, Äquivalenzklassen und Zeicheneinheiten.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Arbeitsweise von *find* - ein einfaches Beispiel:

Die Pfadnamen aller Dateien ausgeben,
– die im aktuellen Dateiverzeichnis oder einem Unterdateiverzeichnis stehen und
– deren Name *tmp* ist.

```
$ find . –name tmp –print
```

Hier besteht *ausdruck* aus zwei elementaren Ausdrücken, die mit einem logischen UND verknüpft sind. Für jede Datei prüft *find* zuerst den elementaren Ausdruck *-name tmp* ab. Ist dieser Ausdruck wahr (d.h. hat die aktuelle Datei den Namen *tmp*), dann bearbeitet *find* auch den Ausdruck *-print*, d.h., *find* gibt den Pfadnamen aus. Andernfalls bearbeitet *find* den Ausdruck *-print* nicht, d.h., *find* gibt nichts aus.

Beachten Sie die Reihenfolge: Wenn Sie stattdessen

```
$ find . –print –name tmp
```

eingeben, dann bearbeitet *find* den Ausdruck *-print* für jede Datei, d.h., *find* gibt die Pfadnamen aller Dateien aus.

Beispiel 2 Arbeitsweise von *find* - ein komplexeres Beispiel:

Die Pfadnamen aller Dateien ausgeben,
– die im aktuellen Dateiverzeichnis oder Unterverzeichnissen stehen und
– deren Name *tmp* ist oder auf *.xx* endet.

```
$ find . \( -name tmp -o -name '*.xx' \) -print
```

Beachten Sie die Leerzeichen zwischen den Argumenten!

Für jede Datei bearbeitet *find* zuerst den Inhalt der Klammern. Dieser besteht aus zwei elementaren Ausdrücken, die mit einem logischen ODER verknüpft sind.
find prüft zuerst den elementaren Ausdruck *-name tmp* ab. Ist dieser Ausdruck wahr (d.h. hat die aktuelle Datei den Namen *tmp*), dann bearbeitet *find* den Ausdruck *-name '*.xx'* gar nicht mehr: das Ergebnis des geklammerten Ausdrucks ist dann auf jeden Fall wahr. Ist *-name tmp* falsch, prüft *find* den zweiten elementaren Ausdruck *-name '*.xx'* ab; ist er wahr (d.h. endet der Dateiname auf *.xx*), dann ist auch der geklammerte Ausdruck wahr, andernfalls ist der geklammerte Ausdruck falsch.
Ist der geklammerte Ausdruck wahr, dann bearbeitet *find* den Ausdruck *-print*, d.h., *find* gibt den Dateinamen aus (UND-Verknüpfung - siehe *Beispiel 1* !). Ist der geklammerte Ausdruck falsch, dann bearbeitet *find* den Ausdruck *-print* nicht, d.h., *find* gibt nichts aus.

Beispiel 3 Alle Einträge im Dateiverzeichnis */usr/nikolaus* und den Unterverzeichnissen ausgeben, deren Eigentümer nicht *nikolaus* ist.

```
$ find /usr/nikolaus ! -user nikolaus -print
```

Beispiel 4 Alle Dateien löschen,
– deren Name *tmp* ist oder auf *.xx* endet und
– auf die mehr als 7 Tage lang nicht zugegriffen wurde.

Vor dem Löschen soll abgefragt werden, ob die Datei tatsächlich gelöscht werden soll.

```
$ find / \( -name tmp -o -name '*.xx' \) -atime +7 -ok rm {} \;
```

Beispiel 5 Rekursives Drucken der Dateiverzeichnis-Struktur mit Auslassen aller SCCS-Dateiverzeichnisse.

```
$ find . -name SCCS -prune -o -print
```

Siehe auch *chmod*, *ln*, *test*
stat() [4]

fold Lange Zeilen zerlegen (filter for folding lines)

fold zerlegt die Zeilen der Textdateien oder der Standard-Eingabe so, dass sie nicht länger sind als die voreingestellte oder angegebene Zeilenlänge. Standardmäßig sind 80 Zeichen pro Zeile vorgegeben.

Die Zeilen werden durch Einfügen eines Neue-Zeile-Zeichens zerlegt, so dass jede Zeile der Ausgabe (weiter unten auch als „Segment“ bezeichnet) nicht länger ist als die voreingestellte oder angegebene Zeilenlänge (bzw. nicht länger als die maximale Anzahl der Bytes). Zeilen werden jedoch nicht mitten in Zeichen zerlegt. Das Verhalten ist nicht vorhersehbar, wenn *anzahl* kleiner ist als die Anzahl der Spalten, die ein einzelnes Zeichen in der Eingabe einnehmen würde.

Wird in der Eingabe das Wagenrücklaufzeichen (Carriage Return), das Rücksetz-Zeichen (Backspace) oder das Tabulatorzeichen (Tab) gelesen, und die Option *-b* ist nicht gesetzt, so gilt für diese Zeichen:

Rücksetz-Zeichen
: Die aktuelle Zeilenlänge wird um Eins verringert, wird aber nie negativ. *fold* fügt kein Neue-Zeile-Zeichen unmittelbar vor oder nach einem Rücksetz-Zeichen ein.

Wagenrücklaufzeichen
: Die aktuelle Zeilenlänge wird auf Null gesetzt. *fold* fügt kein Neue-Zeile-Zeichen unmittelbar vor oder nach einem Wagenrücklaufzeichen ein.

Tabulatorzeichen
: Jedes gelesene Tabulatorzeichen setzt den Spaltenpositionszeiger auf die nächste Tabulatorstop-Position. Tabulatorstop-Positionen befinden sich an jeder Spaltenposition *n*, wobei gilt: *n* modulo 8 gleich 1.

Achtung!
Enthalten die Eingabezeilen Unterstreichungen, so kann dies zu fehlerhaftem Verhalten von *fold* führen.

Syntax

fold[␣**-bs**][␣**-w**␣anzahl][␣datei...]

Keine Option angegeben
: Die eingelesenen Zeilen zerlegt *fold* in Zeilen mit maximal 80 Zeichen (Voreinstellung).

option

-b (b - bytes) Die Zeilenlänge wird in Byte, nicht in Spaltenpositionen, gezählt.

-s (s - segment) Enthält ein Segment einer Zeile ein Leerzeichen innerhalb der ersten *anzahl* Spaltenpositionen (oder Bytes), so wird die Zeile nach dem letzten Leerzeichen, das noch der Bedingung *anzahl* entspricht, zerlegt. Liegt kein Leerzeichen vor, das diese Anforderung erfüllt, so bleibt die Option *-s* für dieses Ausgabesegment der Eingabezeile ohne Wirkung.

-w␣anzahl
(w - width) Die maximale Zeilenlänge soll *anzahl* Zeichen betragen. *anzahl* muss als positive Dezimalzahl angegeben werden.

Falls Tabulatoren in den Eingabezeilen vorkommen, sollte *anzahl* ein Vielfaches von 8 sein.

datei
Name der Textdatei, die von *fold* bearbeitet werden soll. Sie können mehrere Dateien angeben. Die angegebenen Optionen gelten dann für alle Dateien.

datei nicht angegeben:
fold liest von der Standard-Eingabe.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *fold*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien) und bestimmt die Breite von Spaltenpositionen, die jedes Zeichen mit einer konstanten Breite einnimmt.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Sie geben den Inhalt der Datei *notiz* auf die Standard-Ausgabe aus.

Unter Verwendung des Kommandos *cat* erhalten Sie folgendes Ergebnis:

```
$ cat
Notizen:
Zum Projekt Z noch Angebot der Firma Leist & Gutmann einholen. Vergleich mit
den anderen Angeboten muss im Mai abgeschlossen sein.
```

Verwenden Sie jedoch das Kommand *fold*, haben Sie die Möglichkeit, die Länge der ausgegebenen Zeilen neu festzusetzen.

In folgendem Beispiel werden die Zeilen so zerlegt, dass die Ausgabe eine maximale Breite von 40 Zeichen hat:

```
$ fold -w 40 notiz
Notizen:
Zum Projekt Z noch Angebot der Firma Lei
st & Gutmann einholen. Vergleich mit
den anderen Angeboten muss im Mai abgesc
hlossen sein.
```

Siehe auch *pr*

fsck Konsistenzprüfung des Dateisystems und Korrektur im Dialog mit dem Benutzer (filesystem check)

fsck prüft Dateisysteme auf Inkonsistenzen und korrigiert diese im Dialog mit dem Benutzer. In diesem Falle wird die Zustimmung des Benutzers eingeholt, bevor Korrekturmaßnahmen durchgeführt werden. Andere als die oben genannten Inkonsistenzen können zu Datenverlusten führen. Der Umfang und die Schwere der Verluste lässt sich anhand der Diagnosemeldungen bestimmen.

fsck korrigiert weniger schwerwiegende Inkonsistenzen wie I-Nodes ohne Referenz, zu hohe Verweiszähler in I-Nodes, fehlende Blöcke in der Liste der freien Blöcke, Blöcke, die in der Liste der freien Blöcke und gleichzeitig in Dateien erscheinen oder falsche Zähler im Superblock automatisch. Für jede korrigierte Inkonsistenz wird eine Meldung ausgegeben, die die Art der Korrektur und das Dateisystem, auf dem sie ausgeführt wird, anzeigt. Nach der Korrektur eines Dateisystems gibt *fsck* die Anzahl der Dateien in diesem Dateisystem, die Anzahl der belegten und der freien Blöcke und die entsprechenden Angaben in Prozent aus.

Standardmäßig wartet das Kommando vor jeder Korrektur auf die Antwort des Benutzers, die *yes* (Ja) oder *no* (Nein) lauten muss. Hat der Benutzer keine Schreibberechtigung für das Dateisystem, geht *fsck* standardmäßig von *-n* (keine Korrekturen) aus.

Folgende Inkonsistenzen werden überprüft:

1. Blöcke, die von mehreren I-Nodes oder der Liste der freien Blöcke beansprucht werden.
2. Blöcke, die von einem I-Node oder der Liste der freien Blöcke beansprucht werden und außerhalb des Bereichs des Dateisystems liegen.
3. Falsche Verweiszähler.
4. Falsche Verzeichnisgrößen.
5. Fehlerhaftes Format der I-Nodes.
6. Blöcke, die nirgends aufgeführt sind.
7. Verzeichnisprüfungen, Dateien, die auf einen nicht zugewiesenen I-Node zeigen, I-Node-Nummer außerhalb des zugewiesenen Bereichs, Fehlen von `.' und `..' als erste Einträge in den Verzeichnissen.
8. Prüfungen des Superblocks: Mehr Blöcke für I- Nodes, als das Dateisystem enthält.
9. Fehlerhaftes Format der Liste der freien Blöcke.
10. Summe der freien Blöcke und/oder der freien I- Nodes fehlerhaft.

Verwaiste Dateien und Verzeichnisse (d.h. zugewiesen, aber ohne Referenz) werden mit Zustimmung des Benutzers neu verbunden, indem sie in das Verzeichnis *lost+found* geschrieben werden. Der zugewiesene Name entspricht der I-Node-Nummer. Besteht das Verzeichnis *lost+found* noch nicht, wird es nun angelegt. Reicht der Platz nicht aus, wird es vergrößert.

Zur Bezeichnung eines Dateisystems kann der Name des block- oder zeichenorientierten Geräts, auf dem es steht, oder der Name des Einhängepunkts angegeben werden.

Syntax **fsck**[␣**-F**␣**ufs**|**UFS**][␣**-y**|**-n**|**-m**][␣gerätedatei ...]

Folgende Optionen sind möglich:

-F␣ufs|UFS
Gibt *ufs* als Dateisystem-Typ an.

-y Alle Inkonsistenzen werden geprüft. Die von *fsck* gestellten Fragen werden mit "yes" beantwortet.

-n Alle Inkonsistenzen werden geprüft. Die von *fsck* gestellten Fragen werden mit "no" beantwortet.

-m Der Status im Superblock des Dateisystems wird geprüft. Diese Option stellt sicher, dass sich das Dateisystem zum Einhängen eignet.

gerätedatei
gerätedatei steht für eine block- oder zeichenorientierte Gerätedatei (z.B. */dev/dsk/234*). Vorzugsweise ist ein zeichenorientiertes Gerät zu verwenden. *fsck* kann bei eingehängten blockorientierten Gerätedateien nicht ausgeführt werden.

gerätedatei nicht angegeben:
die Datei */etc/vfstab* wird durchsucht. *fsck* wird dann für alle zeichenorientierten Geräte im Feld *fsckdev* der Datei */etc/vfstab* durchgeführt, für die das Feld *fsckpass* einen numerischen Eintrag enthält.

Hinweis In nahezu allen Fällen lässt sich das zeichenorientierte Gerät schneller überprüfen. Für Dateisysteme, die mit Journal eingehängt waren, ist die Option *–m* auch nach einem Systemabsturz ausreichend, um das Dateisystem in kürzester Zeit in einen konsistenten Zustand zu versetzen. Hingegen kann die Prüfung aller Inkonsistenzen abhängig von Größe und Füllgrad des Dateisystems sehr lange dauern.

Das Kommando *fsck* wird für bs2fs-Dateisysteme nicht unterstützt.

Datei *lost+found*

/etc/vfstab
Liste der Standardparameter für jedes Dateisystem

Siehe auch *mount, mountall, umount*

fsexpand Existierende Dateisysteme vergrößern (expand existing file systems)

Mit *fsexpand* können Dateisysteme um Pamseiten oder Zylindergruppen erweitert werden. Die Dateisysteme dürfen nicht gemountet sein.

Syntax **fsexpand**[␣**-i**][␣**-p**␣pamseiten|␣**-c**␣zylindergruppen]␣gerät

Folgende Optionen sind möglich:

-i Ausgabe von Dateisystem-Informationen. Die Ausgabe erfolgt auf stdout. Insbesondere werden Informationen darüber ausgegeben, inwieweit eine Vergrößerung optimal ist, z.B. um ungenutzte PAM-Seiten im erweiterten Dateisystem zu vermeiden. In Verbindung mit der Option *-p* bzw. *-c* werden Informationen vor der Erweiterung (Ausgangsdateisystem) und nach der Erweiterung (Zielsystem) ausgegeben.

-p␣pamseiten
Erweiterung eines Dateisystems um Pamseiten.
Das Dateisystem wird um die angegebene Anzahl Pamseiten erweitert.

-c␣zylindergruppen
Erweiterung eines Dateisystems um Zylindergruppen.
Das Dateisystem wird um die angegebene Anzahl Zylindergruppen erweitert. Die Größe einer Zylindergruppe im Ausgangsdateisystem kann vor der Erweiterung mit *fsexpand -i* ermittelt werden.

Bis zu einer Dateigröße von etwas mehr als 2 GB beträgt die Größe 2048 PAM-Seiten. Ab 2 GB wächst die Größe an. Sie beträgt dann z.B. bei einer Dateigröße von 4 GB 4096 PAM-Seiten.

gerät
Gerätename */dev/rdsk/...* (nur zeichenorierntiertes Gerät, also nicht */dev/dsk*) oder BS2000-Dateiname.

gerät muss beschreibbar sein, wenn eine Erweiterung durchgeführt werden soll. Wenn das Dateisystem auf dem HOME-Pubset liegt, muss der BS2000-Dateiname (mit oder ohne Catid) mit der Notation in der Datei */etc/partitions* übereinstimmen.

Hinweis Das Kommando *fsexpand* wird für bs2fs-Dateisysteme nicht unterstützt.

Das Erweitern eines Dateisystems geschieht in zwei Schritten:

1. die physikalische Erweiterung (Expandieren)
2. Zusammenfassen von (Verwaltungs-)Einheiten des Dateisystems (den sogenannten Zylindergruppen) zu größeren Einheiten (Kompaktifizieren)

   Das Kompaktifizieren wird nur dann ausgeführt, wenn das erweiterte Dateisystem größer als 2 GB und mindestens doppelt so groß wie das ursprüngliche Dateisystem wird.

Die Kompaktifizierung geht erheblich in die Laufzeit von *fsexpand* ein. Wird ein Dateisystem der Größe 1 GB beispielweise um einen Wert erweitert, der geringfügig über 1 GB liegt (also mit Kompaktifizierung), ergibt sich gegenüber einer Erweiterung um einen Wert, der etwas kleiner ist als 1 GB (also ohne Kompaktifizierung), eine Laufzeitverlängerung um den Faktor 3 - 4. Die Kompaktifizierung hat allerdings den Vorteil, dass bei der Nutzung des Dateisystems z.T. erhebliche Performance-Gewinne anfallen, da zur Laufzeit der Bedarf an Speicher und CPU-Zeit wegen der Zusammenfassung von Verwaltungseinheiten kleiner wird.

Beispiel

```
$ fsexpand -i '$BACH.BACH.TEST

device /dev/rdsk/8 in Containerfile: $BACH.BACH.TEST

size of BS2000 Containerfile (PP):     12288
last page used in Containerfile (PP):  12288
size used for POSIX filesystem (PP):   12288
unused in Containerfile (PP):          0
Cylindergroups in filesystem:          6
Cylinders in cylindergroup (PP):       16
size of a cylindergroup (PP):          2048
inodes per cylindergroup:              2048
inodes total:                          12288
datablocks (4K) in filesystem:         5731
        free blocks             directories             free inodes
        1853                    4                       12232
optimal values for expansion of container: 0 PP + N * 2048 PP
```

ftyp Bearbeitungsart für Dateien festlegen (define file processing mode) *(BS2000)*

ftyp legt fest, ob Dateien beim nachfolgenden Kopieren mit *bs2cp* zwischen BS2000 und dem POSIX-Dateisystem als Text- oder Binärdateien interpretiert werden sollen. Das Kommando ist nur für SAM-Dateien und für textartige PLAM-Bibliothekselemente (Elementtypen ungleich LLM) wirksam. PAM-Dateien und werden stets als Binärdateien, ISAM-Dateien stets als Textdateien interpretiert.
Wurde kein Kommando *ftyp* gegeben, werden alle SAM-Dateien als Textdateien interpretiert.

Syntax

ftyp[␣**-h**][␣**text**|**binary**|**textbin**]

Keine Option angegeben
Option *text* wird verwendet.

option

-h Ausgabe der Kommandosyntax mit Erläuterung der Optionen.

text
SAM-Dateien und textartige Bibliothekselemente sollen als Textdateien interpretiert werden.
Beim Schreiben in das BS2000 werden Neue-Zeile Zeichen (x'15') in Zeilenwechsel (Satzwechsel) umgewandelt und Tabulatorzeichen durch die entsprechende Anzahl von Leerzeichen bis zur Tabulatorposition ersetzt.

Beim Lesen einer Datei aus dem BS2000 wird an jeden gelesenen Satz ein Neue-Zeile Zeichen angefügt (x'15').

Diese Option kann für SAM-Dateien mit fester Satzlänge nicht verwendet werden.

binary
Die SAM-Dateien sollen binär interpretiert werden.
Es finden keine Konvertierungen statt.

Diese Option kann auch für SAM-Dateien mit fester Satzlänge verwendet werden. Der letzte Satz wird in diesem Fall mit binär Null aufgefüllt.

textbin

SAM-Dateien und textartige Bibliothekselemente sollen als binäre Textdateien interpretiert werden.
Beim Schreiben in das BS2000 werden Neue-Zeile Zeichen (x'15') in Zeilenwechsel (Satzwechsel) umgewandelt.

Tabulatorzeichen werden nicht konvertiert.

Beim Lesen einer Datei aus dem BS2000 wird an jeden gelesenen Satz ein Neue-Zeile Zeichen angefügt (x'15').

Diese Option kann für SAM-Dateien mit fester Satzlänge nicht verwendet werden.

Beispiel Die Datei *input* des POSIX-Dateisystems soll binär in das BS2000 übertragen werden.

```
$ ftyp binary
$ bs2cp input bs2:output
```

Siehe auch *bs2cp, bs2file*

fuser Dateinutzer anzeigen (display file users)

Mit dem Kommando *fuser* können die Prozesse angezeigt werden, die Dateien oder Dateisysteme benutzen.

Syntax **fuser** [**-cl-f**] [**-u**] [**-k**] path ...

-c Alle Prozesse, die beliebige Dateien des Dateisystems am Einhängepunkt *path* benutzen, werden ausgegeben.

-f Nur die Prozesse werden ausgegeben, die die Datei *path* selbst benutzen (nur für Gerätedateien).

-u Ausgabe der Benutzerkennung der gefundenen Prozesse hinter der Prozess-ID.

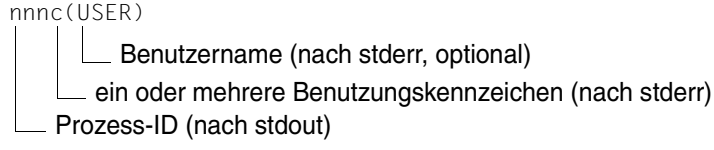
-k Die gefundenen Prozesse werden mit dem Signal SIGKILL beendet.

path
Name der Datei oder Einhängepunkt des Dateisystems.

Optionen *-c* und *-f* nicht angegeben:

Für Gerätedateien, die ein eingehängtes Dateisystem repräsentieren, wird implizit die Option *-c* gesetzt und für alle anderen Dateien die Option *-f*.

Für jeden gefundenen Prozess gibt *fuser* aus:



Benutzungskennzeichen

c aktuelles Verzeichnis

r Root-Verzeichnis

o Datei zum Lesen/Schreiben geöffnet

t Datei zum Ausführen geöffnet

Beispiel 1 Nutzer einer Datei anzeigen:

```
# fuser /var/adm/syslog
/var/adm/syslog:       54o
# ps -fTp 54
     UID   PID   TSN  PPID  C    STIME TTY       TIME CMD
    ROOT    54  0606     1  0 05/06/11 ?         0:02 [syslogd]
#
```

Beispiel 2 Nutzer von zwei Dateisystemen mit Benutzernamen anzeigen:

```
# fuser -cu /opt /home/froede
/opt:      226t(SYSROOT)     240t(SYSROOT)
/home/froede:      628co(FROEDE)
#
```

Beispiel 3 Informationen über alle Prozesse ausgeben, die ein Dateisystem benutzen:

```
# fuser -c /var 2>/dev/null | read PIDLST
# for PID in $PIDLST
> do
> ps -p $PID -o user= -o pid= -o tsn= -o comm=
> done
 SYSROOT   603  07GN in.rlogind
 SYSROOT   226  0653 prngd
 SYSROOT    54  0606 syslogd
  FROEDE   628  07HD sh
 SYSROOT   201  065E cron
 SYSROOT   606  07GR sh
 SYSROOT   625  07HA in.rlogind
#
```

Die betreffenden Prozess-IDs werden in die Variable *PIDLST* eingelesen; dabei wird die Ausgabe der Benutzungskennzeichen (stderr) unterdrückt. Die Prozess-IDs werden dann in der for-Schleife weiter verarbeitet.

gencat Binär codierten Meldungskatalog erzeugen (generate a formatted message catalogue)

Das Kommando *gencat* schreibt den Inhalt einer Meldungstext-Datei in einen binär codierten Meldungskatalog.
Wenn der Meldungskatalog noch nicht existiert, wird er neu erstellt. Wenn der Meldungskatalog bereits existiert, werden die darin enthaltenen Meldungen in den neuen Meldungskatalog mit eingebunden. Wenn die Mengen- und Meldungsnummern von aktuellen und neudefinierten Meldungstexten übereinstimmen, dann ersetzt *gencat* die aktuell im Meldungskatalog stehenden Meldungstexte durch die in der Meldungstext-Datei definierten neuen Meldungstexte.

Mit *gencat* erzeugte Meldungskataloge sind binär codiert, d.h., dass zwischen verschiedenen Maschinentypen keine Kompatibilität garantiert werden kann. Genau wie C-Programme für jeden Maschinentyp neu übersetzt werden müssen, müssen die Meldungskataloge mit *gencat* neu erstellt werden.

Eine Meldungstext-Datei kann einfache Meldungsnummern oder zweistufige Meldungsnummern enthalten. Zweistufige bestehen aus der Meldungsnummer und einer Mengennummer. Bei einfachen Meldungsnummern wird standardmäßig *NL_SETD* für die Mengennummern verwendet.

Syntax **gencat**[␣**-lm**]␣catfile[␣msgfile...]

option

-l Falls *catfile* bereits existiert, werden Informationen über die dort vorhandenen Meldungen auf die Standard-Ausgabe ausgegeben. Die Ausgabe hat folgendes Format:

```
SET mengennr, MESSAGE meldungsnr, OFFSET offset, LENGTH meldungslänge
meldungstext
*
```

offset gibt den Abstand des Meldungstextbeginns vom Anfang an, d.h.
- erste Meldung: offset(1)=0
- i-te Meldung: offset(i)=offset(i-1)+meldungslänge(i)

-m Mit der Option *-m* kann X/Open-konform gearbeitet werden.

Aus Kompatibilitätsgründen zu früheren Versionen von *gencat*, die in einer Reihe von speziellen internationalisierten Produkten veröffentlicht wurden, wird die Option *-m* zur Verfügung gestellt. Sie bewirkt, dass *gencat* eine einzige Datei *catfile* erzeugt, die mit den Formatkatalogen, die von den früheren Versionen erstellt wurden, kompatibel ist. Die Suchroutinen finden heraus, um welchen Katalogtyp es sich handelt, und agieren dementsprechend.

Das Verhalten der Option *-m* ist die Voreinstellung, sie braucht nicht explizit eingeschaltet zu werden.

Keine Option angegeben
: *gencat* verhält sich, wie bei der Option *-m* beschrieben.

catfile
: Name des binär codierten Meldungskatalogs, den *gencat* aus der Meldungstext-Datei *msgfile* erzeugen soll.

  Wenn Sie für *catfile* einen Bindestrich - angeben, schreibt *gencat* auf die Standard-Ausgabe.

msgfile
: Name der Meldungstext-Datei, aus der *gencat* einen binär codierten Meldungskatalog erzeugt.
  Wenn Sie für *msgfile* einen Bindestrich - angeben, liest *gencat* von der Standard-Eingabe.
  Sie können auch mehrere Meldungsquelldateien angeben.

Format eines Meldungskatalogs

Ein von *gencat* mit der Option *-m* erzeugter Meldungskatalog enthält die folgenden Strukturen in der angegebenen Reihenfolge:

- den Katalog-Kopf *CAT_HDR*, bestehend aus:
  - der Dateiformatkennung (magic number)
  - der Anzahl der Mengen (set) in der Meldungsdatei
  - dem Platz (in Byte), den die Datei zum Laden benötigt, die Länge des Dateikopfs nicht mitgerechnet
  - der Position, an der die Meldungsköpfe beginnen, die Länge des Dateikopfs nicht mitgerechnet
  - der Position, an der die Meldungstexte beginnen, die Länge des Dateikopfs nicht mitgerechnet
- einen Mengenkopf *CAT_SET_HDR* für jede vorhandene Menge (set), bestehend aus:
  - der Mengennummer
  - der Anzahl der Meldungen in der Menge
  - dem Start-Offset in der Tabelle
- einen Meldungskopf *CAT_MSG_HDR* für jede vorhandene Meldung, bestehend aus:
  - der Meldungsnummer
  - der Länge der Meldung in Byte
  - dem Meldungs-Offset in der Tabelle
- die einzelnen Meldungstexte, die durch \0 voneinander getrennt sind.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *gencat*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel

Erzeugen des binär codierten Meldungskatalogs *de_wc.cat* aus der Meldungstext-Datei *de_wc.msf* für das Kommando *wc*. Die Meldungstext-Datei hat folgenden Inhalt:

```
1 wc: %s kann nicht geoeffnet werden \n
2 gesamt \n
3 Syntax: wc[␣-c|␣-m][␣-lw][␣datei...] \n
```

Mit folgendem Aufruf wird der Meldungskatalog erzeugt:

```
$ gencat de_wc.cat de_wc.msf
```

Siehe auch

iconv
catopen(), catgets(), catclose(), limit.h, nl_types.h [4]

getconf Konfigurationswerte abrufen (get configuration values)

In der ersten Syntaxform schreibt das Kommando *getconf* den Wert der durch den Operanden *system_var* angegebenen Variablen auf die Standard-Ausgabe.

In der zweiten Syntaxform schreibt das Kommando *getconf* den Wert der durch den Operanden *path_var* angegebenen Variablen auf die Standard-Ausgabe, der innerhalb des durch den Operanden *pathname* angegebenen Pfades gültig ist.

Der Wert jeder Konfigurationsvariablen wird so ausgegeben, wie ihn ein Aufruf der Funktion, von der er definiert wurde, ermittelt. Der Wert gibt die Bedingungen in der aktuellen Ablaufumgebung wieder.

Ist die angegebene Variable gültig, aber auf dem System nicht definiert, schreibt *getconf* `%undefined` auf die Standard-Ausgabe.

Syntax

Format 1: getconf ␣system_var

Format 2: getconf ␣path_var ␣pathname

Format 1 **getconf** ␣system_var

system_var

Name einer Konfigurationsvariablen, deren Wert über die Funktionen *confstr*() oder *sysconf*() [4] abgerufen werden kann. Die folgenden Werte werden unterstützt:

ARG_MAX, BC_BASE_MAX, BC_DIM_MAX, BC_SCALE_MAX, BC_STRING_MAX, CHAR_BIT, CHAR_MAX, CHAR_MIN, CHARCLASS_NAME_MAX, CHILD_MAX, CLK_TCK, COLL_WEIGHTS_MAX, CS_PATH, EXPR_NEST_MAX, INT_MAX, INT_MIN, LINE_MAX, LONG_BIT, LONG_MAX, LONG_MIN, MB_LEN_MAX, NGROUPS_MAX, NL_ARGMAX, NL_ARGMAX, NL_LANGMAX, NL_LANGMAX, NL_MAX, NL_MAX, NL_MSGMAX, NL_MSGMAX, NL_SETMAX, NL_SETMAX, NL_TEXTMAX, NL_TEXTMAX, NZERO, OPEN_MAX, POSIX_ARG_MAX, POSIX_CHILD_MAX, POSIX_JOB_CONTROL, POSIX_LINK_MAX, POSIX_MAX_CANON, POSIX_MAX_INPUT, POSIX_NAME_MAX, POSIX_NGROUPS_MAX, POSIX_OPEN_MAX, POSIX_PATH_MAX, POSIX_PIPE_BUF, POSIX_SAVED_IDS, POSIX_SSIZE_MAX, POSIX_STREAM_MAX, POSIX_TZNAME_MAX, POSIX_VERSION, POSIX2_BC_BASE_MAX, POSIX2_BC_DIM_MAX, POSIX2_BC_SCALE_MAX, POSIX2_BC_STRING_MAX, POSIX2_C_BIND, POSIX2_C_DEV, POSIX2_C_VERSION, POSIX2_CHAR_TERM, POSIX2_COLL_WEIGHTS_MAX, POSIX2_EXPR_NEST_MAX, POSIX2_FORT_DEV, POSIX2_FORT_RUN, POSIX2_LINE_MAX, POSIX2_LOCALEDEF, POSIX2_RE_DUP_MAX, POSIX2_SW_DEV, POSIX2_UPE, POSIX2_VERSION, RE_DUP_MAX, SCHAR_MAX, SCHAR_MIN, SHRT_MAX, SHRT_MIN, SSIZE_MAX, STREAM_MAX,

TMP_MAX, TZNAME_MAX, UCHAR_MAX, UINT_MAX, ULONG_MAX, USHRT_MAX, WORD_BIT, XOPEN_CRYPT, XOPEN_ENH_I18N, XOPEN_SHM, XOPEN_VERSION, XOPEN_XCU_VERSION, XOPEN_XPG2, XOPEN_XPG3, XOPEN_XPG4

Das Symbol PATH wird ebenfalls erkannt. Es liefert denselben Wert wie der Wert CS_PATH von *confstr*() [4]. Außerdem erkennt *getconf* zusätzlich die Variablen

`LOGNAME_MAX, PAGE_SIZE, PAGESIZE` und `PASS_MAX`.

Sie können alle Variablennamen mit oder ohne führenden Unterstrich (_) angeben.

Format 2 **getconf** ␣ path_var ␣ pathname

path_var
Name einer Konfigurationsvariablen, deren Wert über die Funktion *pathconf*() [4] abgerufen werden kann. Die folgenden Werte werden unterstützt:

LINK_MAX, MAX_CANON, MAX_INPUT, NAME_MAX, PATH_MAX, PIPE_BUF, POSIX_CHOWN_RESTRICTED, POSIX_NO_TRUNC, POSIX_VDISABLE

Sie können alle Variablennamen mit oder ohne führenden Unterstrich (_) angeben.

pathname
Pfadname, für den die durch *path_var* angegebene Variable festgestellt werden soll.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *getconf*:

LANG Gibt einen Standardwert für die Internationalisierungsvariablen an, die nicht gesetzt oder null sind. Ist *LANG* nicht gesetzt oder null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als wäre keine der Variablen definiert.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d.h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation von Byte-Folgen als Zeichen fest (z.B. Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt die Position der Meldungskataloge für die Verarbeitung von *LC_MESSAGES* fest.

Endestatus Die folgenden Endewerte werden zurückgegeben:

0 Die angegebene Variable ist gültig, und es wurden Informationen über ihren aktuellen Status auf die Standard-Ausgabe geschrieben.

>0 Ein Fehler ist aufgetreten.

Beispiel 1 Der Wert von {NGROUPS_MAX} wird abgerufen (Format 1):

```
$ getconf NGROUPS_MAX
16
```

Beispiel 2 Der Wert von {NAME_MAX} wird für ein bestimmtes Verzeichnis abgerufen (Format 2):

```
$ getconf NAME_MAX /usr
255
```

Beispiel 3 Dieses Beispiel zeigt den vorsichtigeren Umgang mit möglicherweise unspezifizierten Ergebnissen:

```
if value=$(getconf PATH_MAX /usr); then
        if [ "$value" = "undefined" ]; then
                echo PATH_MAX in /usr is infinite.
        else
                echo PATH_MAX in /usr is $value.
        fi
else
        echo Error in getconf.
fi
```

Beachten Sie, dass folgende Aufrufe in einem C-Programm unterschiedliche Ergebnisse erzielen können:

```
sysconf(_SC_POSIX_C_BIND);
```

und:

```
system("getconf POSIX2_C_BIND");
```

Der Aufruf *sysconf*() [4] ergibt einen Wert, der den Bedingungen bei der Kompilierung oder bei der Ausführung des Programms entspricht. Der Aufruf von *getconf* über *system*() [4] ergibt immer einen Wert, der den Bedingungen bei der Ausführung des Programms entspricht.

Siehe auch *pathconf(), confstr(), sysconf()* [4]

getopts Argumente einer Prozedur nach Optionen durchsuchen (parse utility options)

Das in die POSIX-Shell *sh* eingebaute Kommando *getopts* wird in Shell-Prozeduren verwendet, um die Argumente und Optionen in der Kommandozeile zu analysieren und auf Gültigkeit zu überprüfen. Alle Regeln des Syntax-Standards für Kommandozeilen werden unterstützt.
getopts können Sie in Shell-Prozeduren zur Analyse der beim Prozedur-Aufruf angegebenen Argumente verwenden. Die einzelnen Optionen und Argumente werden der Reihe nach in Shell-Variablen abgelegt und können so einfach abgefragt bzw. überprüft werden. Wenn in der Argumentliste eine Option nicht zulässig ist oder wenn *getopts* für eine Option, die ein Argument verlangt, kein zugehöriges Argument erkennt, dann gibt *getopts* eine entsprechende Fehlermeldung aus.

Damit alle Prozeduren und Kommandos die Argumentliste einheitlich verarbeiten, sollte die Argumentliste immer mit *getopts* analysiert und auf gültige Optionen untersucht werden.

Syntax

getopts optstring name[arg]...

optstring
: Zeichenkette, die aus Buchstaben und Doppelpunkten : bestehen kann. Die in *optstring* angegebenen Buchstaben werden von *getopts* als zulässige Optionen der Shell-Prozedur betrachtet. Wenn hinter einem Buchstaben ein Doppelpunkt steht, so erwartet *getopts* für diese Option ein Argument oder eine Gruppe von Argumenten, die durch Leerzeichen oder Tabulatorzeichen von der Option getrennt sein müssen.

name
: Name der Shell-Variablen, die bei jedem Aufruf von *getopts* mit der jeweils nächsten Option belegt wird.

arg ...
: Argumentliste, die von *getopts* analysiert wird.

  arg nicht angegeben:
  getopts analysiert die Argumentliste der Kommandozeile, mit der die Prozedur aufgerufen wurde.

Arbeitsweise

Bei jedem Aufruf von *getopts* wird die Shell-Variable *name* mit dem Wert der jeweils nächsten Option belegt. Die Shell-Variable *OPTIND* wird mit der Nummer des nächsten Arguments belegt, das noch nicht verarbeitet wurde. *OPTIND* wird immer mit dem Wert 1 initialisiert, wenn der Kommandointerpreter *sh* oder eine Shell-Prozedur aufgerufen wird.

Bei Optionen, die ein Argument verlangen, wird die Shell-Variable *OPTARG* mit diesem Argument belegt. Optionen, die ein Argument verlangen, müssen in *optstring* mit einem Doppelpunkt : gekennzeichnet sein. Liegt keine Option vor, oder hat die Option kein Argument, so wird *OPTARG* zurückgesetzt.

Wird eine ungültige Option erkannt, so wird die Shell-Variable *name* mit einem Fragezeichen ? belegt. Ungültige Optionen sind solche, die nicht in *optstring* vorkommen. In diesem Fall (wenn das erste Zeichen in *optstring* ein Doppelpunkt : ist) wird die Shell-Variable *OPTARG* auf das gefundene Optionszeichen gesetzt, es wird aber keine Ausgabe auf die Standard-Fehlerausgabe geschrieben. Andernfalls wird die Shell-Variable *OPTARG* zurückgesetzt und eine Meldung auf die Standard-Fehlerausgabe geschrieben. Dies wird als erkannter Fehler in der Präsentation der Argumente für die aufrufende Anwendung interpretiert, ist aber kein Fehler von *getopts*.

Fehlt ein Argument einer Option, so gilt:

– Ist das erste Zeichen von *optstring* ein Doppelpunkt :, so wird die durch *name* angegebene Shell-Variable auf den Doppelpunkt und die Shell-Variable *OPTARG* auf das gefundene Optionszeichen gesetzt.

– Andernfalls wird die durch *name* angegebene Shell-Variable auf das Fragezeichen gesetzt, die Shell-Variable *OPTARG* wird zurückgesetzt, und eine Meldung wird auf die Standard-Fehlerausgabe geschrieben. Dies wird als erkannter Fehler in der Präsentation der Argumente für die aufrufende Anwendung interpretiert, ist aber kein Fehler von *getopts*. Es wird wie angedeutet eine Meldung ausgegeben, aber der Endestatus ist Null.

Wenn das Ende der Optionsliste erreicht wurde, beendet sich *getopts* mit einem Endestatus ungleich Null. Das Ende der Optionsliste kann auch mit der speziellen Option -- gekennzeichnet werden.

Wird der Wert der Shell-Variablen *OPTIND* verändert oder *getopts* mit unterschiedlichen Argumentlisten aufgerufen, so kann dies zu unerwarteten Ergebnissen führen.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *getopts*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel

Der folgende Ausschnitt aus einer Shell-Prozedur *proz* zeigt, wie die Argumente einer Prozedur verarbeitet werden können. Für *proz* sind die Optionen *-a*, *-b* und *-o* zulässig, für *-o* ist ein Argument erforderlich. Die Optionen *-a* und *-b* schließen sich gegenseitig aus; werden trotzdem beide angegeben, ist nur die zuletzt angegebene gültig:

```
while getopts abo: c
do
      case $c in
      [ab])           FLAG=$c;;
      o)              OARG=$OPTARG;;
      \?)             echo "usage: $0 [-a| -b] [-o <arg>]"
                      exit 2;;
      esac
done
shift `expr $OPTIND - 1`
```

Durch dieses Programmstück sind z.B. die folgenden Aufrufe von *proz* gleichbedeutend:

```
proz -a -b -o "xxx z yy" datei
proz -a -b -o "xxx z yy" -- datei
proz -ab -o "xxx z yy" datei
```

grep Muster suchen (search a file for a pattern)

grep liest Zeilen aus einer oder mehreren Textdateien oder von der Standard-Eingabe und vergleicht die Zeilen mit einem angegebenen Muster. Ist mittels Optionen nichts anderes angegeben, so schreibt *grep* alle Zeilen, die zu dem Muster passen, auf die Standard-Ausgabe.
Als Muster können Sie einfache reguläre Ausdrücke angeben (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*).
Geben Sie mehrere Eingabedateien an, dann wird jeder Ausgabezeile der Name der betreffenden Datei vorangestellt.

Syntax

Format 1: grep[␣**-E**|␣**-F**][␣**-c**|␣**-l**|␣**-q**][␣**-bihnrsvxy**]␣**-e**␣musterliste
[␣**-f**␣musterdatei][␣datei...]

Format 2: grep[␣**-E**|␣**-F**][␣**-c**|␣**-l**|␣**-q**][␣**-bihnrsvxy**][␣**-e**␣musterliste]
␣**-f**␣musterdatei [␣datei...]

Format 3: grep[␣**-E**|␣**-F**][␣**-c**|␣**-l**|␣**-q**][␣**-bihnrsvxy**]␣musterliste[␣datei...]

Die Formate werden gemeinsam beschrieben, da die Muster, nach denen *grep* die Eingabezeilen vergleichen soll, entweder über *musterliste* oder über die Option *-e musterliste* oder *-f musterdatei* angegeben werden.

Keine Option angegeben
grep gibt alle Zeilen aus, die zu *muster* passen. Geben Sie mehrere Eingabedateien an, dann wird jeder Ausgabezeile der Name der Datei vorangestellt, aus der die Zeile gelesen wurde.

option

-E (E - extended) *grep* behandelt jedes Muster als erweiterten regulären Ausdruck.
Die Option *-E* ist äquivalent zum Kommando *egrep*.

-F (F - fast grep) *grep* sucht nach Zeichenketten.
Die Option *-F* ist äquivalent zum Kommando *fgrep*.

-c (c - count) *grep* gibt nur die Anzahl der gefundenen Zeilen aus (das sind die Zeilen, die *grep* ohne die Option *-c* ausgeben würde, siehe *Beispiel 3*); die Zeilen selbst werden nicht ausgegeben.

-l (l - list) *grep* gibt nur die Namen der Dateien aus, die mindestens eine der gefundenen Zeilen enthalten. (das sind die Zeilen, die *grep* ohne die Option *-l* ausgeben würde, siehe *Beispiel 4*). Jeder Dateiname wird nur einmal ausgegeben. Die Zeilen selbst gibt *grep* nicht aus.

-q (q - quiet) *grep* schreibt nichts auf die Standardausgabe, auch wenn passende Zeilen gefunden werden.

-b (b - block) Jeder Ausgabezeile wird die Nummer des Blockes vorangestellt, in dem sie enthalten ist.
Die Blöcke, aus denen eine Datei besteht, sind je 512 Byte groß und werden, mit 0 beginnend, durchnummeriert.
Option *-b* kann hilfreich sein, wenn die Nummern von Blöcken nach dem Kontext ermittelt werden sollen (siehe z.B. das Kommando *od*, Argument *offset*).

-h (h - hidden) Beim Durchsuchen mehrerer Eingabedateien unterlässt *grep* die Voranstellung der Dateinamen vor jeder Ausgabezeile.

-i oder **-y**
(i - ignore) *grep* unterscheidet beim Vergleich nicht zwischen Groß- und Kleinbuchstaben.

-n (n - number lines) Jeder Ausgabezeile wird die Zeilennummer aus der betreffenden Eingabedatei vorangestellt, wobei von 1 an nummeriert wird. Liest *grep* von der Standard-Eingabe, bezieht sich die Zeilennummer auf die Standard-Eingabe.

-r (r - recursive) Jeder Dateiname, der ein Verzeichnisname ist, wird rekursiv durchsucht. Das bedeutet, das alle Dateien und Dateiverzeichnisse, die in diesem Dateiverzeichnis enthalten sind, berücksichtigt werden.

-s (s - silent) Fehlermeldungen, die sich auf nicht vorhandene Dateien oder auf Dateien, für die Sie kein Leserecht haben, beziehen, werden unterdrückt.

-v (v - vice versa) *grep* gibt alle Zeilen aus, die *nicht* zu dem angegebenen Muster passen.

Zusammen mit Option *-c*:
grep gibt nur die Anzahl solcher Zeilen aus.

Zusammen mit Option *-l*:
grep gibt nur die Namen der Dateien aus, die solche Zeilen enthalten.

-x (x - exact) *fgrep* gibt nur solche Zeilen aus, die eine der angegebenen Zeichenketten und sonst keine weiteren Zeichen enthalten.

-e␣musterliste
(e - expression) Diese Option brauchen Sie, wenn der erste Ausdruck in *musterliste* mit einem Bindestrich - beginnt. Zusammen mit *-e* wird eine solche Musterliste nicht als Option interpretiert, sondern als Liste von Mustern, mit denen *egrep* die Eingabezeilen vergleichen soll.

-f␣musterdatei
(f - file) *egrep* liest die Musterliste aus der Datei *musterdatei*. Jede Zeile von *musterdatei* wird als ein erweiterter regulärer Ausdruck interpretiert.

musterliste

Einfacher regulärer Ausdruck, mit dem *grep* die Eingabezeilen vergleichen soll (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*).
Enthält *musterliste* Zeichen, die für die Shell eine Sonderbedeutung haben, dann schließen Sie *musterliste* in Hochkommas ein: '*musterliste*'.

datei

Name der Datei, die *grep* durchsuchen soll. Pro Aufruf können Sie mehrere Dateinamen angeben.

datei nicht angegeben:
grep liest die Eingabezeilen von der Standard-Eingabe.

grep kann nur auf Textdateien angewendet werden. Wird *grep* auf Binärdateien (z.B. Historydatei) angewendet, ist das Ergebnis undefiniert, da das Auftreten eines Nullbytes eine Eingabezeile logisch beendet.

grep, fgrep und egrep

Die Kommandos *grep*, *fgrep* und *egrep* sind von der Oberfläche her weitgehend identisch. Im Folgenden sind die wichtigsten Unterschiede zwischen diesen Kommandos aufgeführt.

grep verarbeitet einfache reguläre Ausdrücke.

fgrep verarbeitet nur Zeichenketten. Pro Aufruf können Sie jedoch mehrere Zeichenketten angeben: Die Zeichenketten geben Sie entweder direkt in der Aufrufzeile, getrennt durch Neue-Zeile-Zeichen, an oder Sie übergeben sie in einer Datei.
fgrep kann effizient sehr viele Zeichenketten suchen: *fgrep* sucht jede einzelne Zeile nach allen Zeichenketten ab.

egrep verarbeitet erweiterte reguläre Ausdrücke. Diese umfassen u.a. die einfachen regulären Ausdrücke bis auf eine Ausnahme: Der einfache reguläre Ausdruck \(*regausdruck*\) hat bei erweiterten regulären Ausdrücken keine Sonderbedeutung und wird deshalb auch nicht von *egrep* verarbeitet.
Pro Aufruf können Sie mehrere reguläre Ausdrücke, durch Neue-Zeile-Zeichen getrennt, angeben. *egrep* interpretiert diese Neue-Zeile-Zeichen wie einen senkrechten Strich (Zeichen für die Alternative, siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Die regulären Ausdrücke geben Sie entweder direkt in der Aufrufzeile an oder Sie übergeben sie in einer Datei.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *grep*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE beeinflusst die Sortierreihenfolge.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts (Option *-i/-y*) als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Endestatus

0 Zeilen gefunden

1 keine Zeile gefunden

>1 Syntaxfehler oder „Datei kann nicht geöffnet werden“. Dieser Endestatus gilt auch dann, wenn in anderen Eingabedateien Zeilen gefunden wurden

Beispiel Grundlage für die nachfolgenden Beispiele sind die Dateien *kunden1* und *kunden2*. Sie haben folgenden Inhalt:

kunden1:

```
080693    999.98  20 LE  Art.  038   Fa. Holzinger
120394   1240.25   3 LE  Art.  023   Fa. Wanninger
180595    330.87   1 LE  Art.  332   Fa. Wanninger
```

kunden2:

```
hinterhuber berta, rosenheim, zugspitzstr.1
wanninger herbert, muenchen, kirschstr.3
```

Beispiel 1 Zeilen ausgeben, die eine bestimmte Zeichenkette enthalten (keine Option und Option *-i*):

```
$ grep Wanninger kunden1 kunden2
kunden1:120394   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180595    330.87   1 LE  Art.  332   Fa. Wanninger
```

Wenn Sie auch Zeilen mit kleingeschriebenem *wanninger* ausgeben lassen möchten, geben Sie ein:

```
$ grep -i wanninger kunden1 kunden2
kunden1:120394   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:180595    330.87   1 LE  Art.  332   Fa. Wanninger
kunden2:wanninger herbert, muenchen, kirschstr.3
```

Kompliziertere Muster stellen Sie mit Hilfe von regulären Ausdrücken dar, z.B.:

Einträge des Jahres 1994 in der Datei *kunden1* ausgeben - das sind alle Zeilen, die in der 5. und 6. Spalte die Zahl 94 enthalten:

```
$ grep '^....94' kunden1
120394   1240.25   3 LE  Art.  023   Fa. Wanninger
```

Beispiel 2 Zeilen ausgeben, die *nicht* zu dem angegebenen Muster passen (Option *-v*):

```
$ grep -v '^1' kunden1 kunden2
kunden1:080693    999.98  20 LE  Art.  038   Fa. Holzinger
kunden2:hinterhuber berta, rosenheim, zugspitzstr.1
kunden2:wanninger herbert, muenchen, kirschstr.3
```

Beispiel 3 Anzahl der gefundenen Zeilen ausgeben (Option *-c*):

Zuerst soll für jede Eingabedatei die Anzahl der Zeilen, die mit einer 1 beginnen, ausgegeben werden.

```
$ grep -c '^1' kunden1 kunden2
kunden1:2
kunden2:0
```

Nun soll die Anzahl der Zeilen, die *nicht* mit einer 1 beginnen, ausgegeben werden.

```
$ grep -c -v '^1' kunden1 kunden2
kunden1:1
kunden2:2
```

Beispiel 4 Nur Dateinamen ausgeben (Option *-l*):

Zuerst sollen die Namen der Dateien, die Zeilen mit einer 1 am Anfang enthalten, ausgegeben werden.

```
$ grep -l '^1' kunden1 kunden2
kunden1
```

Nun sollen die Namen der Dateien, die Zeilen ohne eine 1 am Anfang enthalten, ausgegeben werden.

```
$ grep -l -v '^1' kunden1 kunden2
kunden1
kunden2
```

Beispiel 5 Gefundene Zeilen mit Zeilennummer ausgeben (Option *-n*):

```
$ grep -n -i wanninger kunden1 kunden2
kunden1:2:120394   1240.25   3 LE  Art.  023   Fa. Wanninger
kunden1:3:180595    330.87   1 LE  Art.  332   Fa. Wanninger
kunden2:2:wanninger herbert, muenchen, kirschstr.3
```

Siehe auch *ed*, *egrep*, *fgrep*, *sed*
stdio [4]

hash Hash-Tabelle der Shell bearbeiten (remember or report utility locations)

Das in die POSIX-Shell *sh* eingebaute Kommando *hash* hat zwei Funktionen:
- Es schreibt den Inhalt der Hash-Tabelle auf die Standard-Ausgabe oder trägt das angegebene Kommando in die Hash-Tabelle ein (Format 1).
- Es löscht den Inhalt der Hash-Tabelle (Format 2).

Jede Shell hat ihre eigene Hash-Tabelle, in die sie alle Kommandos einträgt, die mit ihrem einfachen Dateinamen aufgerufen wurden. Wenn Sie ein Kommando mit dem einfachen Dateinamen aufrufen, sucht die Shell dieses Kommando zuerst in der Hash-Tabelle. Dadurch beschleunigt sich der Suchvorgang. Steht dieses Kommando noch nicht in der Hash-Tabelle der aktuellen Shell, wird es neu eingetragen.

Wenn Sie eine Subshell starten, ist die Hash-Tabelle der neuen Shell noch leer. Wenn Sie die Subshell beenden, meldet sich die übergeordnete Shell mit ihrer Hash-Tabelle zurück. Die Hash-Tabelle der Subshell ist gelöscht.

Syntax

Format 1: hash[␣name]...

Format 2: hash␣-r

Format 1 **Hash-Tabelle ausgeben oder erweitern**

hash[␣name]...

name

einfacher Dateiname eines Kommandos, einer ausführbaren Shell-Prozedur oder eines ausführbaren Programms. Die Shell sucht diese Datei entsprechend dem Inhalt der Variablen *PATH*. Wenn die Shell die Datei gefunden hat, übergibt sie an *hash*:
- den entsprechenden relativen Pfadnamen der Form *./name*, wenn das aktuelle Dateiverzeichnis der Variablen *PATH* zugewiesen ist und *name* enthält, oder
- den absoluten Pfadnamen.

Das Kommando *hash* trägt diesen Pfadnamen in die Hash-Tabelle ein. Ist *name* in keinem Dateiverzeichnis enthalten, das der Variablen *PATH* zugewiesen ist, so erhalten Sie eine Fehlermeldung.

Für *name* können Sie nicht angeben:
- Shell-Prozeduren, für die Sie kein Ausführrecht haben
- Kommandos, deren *name* einen Schrägstrich / enthält. Sie können also keine absoluten oder relativen Pfadnamen angeben.
- eingebaute *sh*-Kommandos, da sie Unterprogramme von *sh* sind. Ihr Name ist also nicht der Name einer ausführbaren Datei.

Die Shell trägt ebenfalls nur ausführbare Shell-Prozeduren und Kommandos ein, wenn Sie sie mit ihrem einfachen Dateinamen aufrufen.

name nicht angegeben:
Das Kommando *hash* schreibt den Inhalt der Hash-Tabelle auf die Standard-Ausgabe. Die Ausgabe sieht beispielsweise wie folgt aus:

```
ls=/usr/bin/ls
cat=/usr/bin/cat
chmod=/usr/bin/chmod
```

Format 2 **Hash-Tabelle löschen**

hash␣-r

-r Den Inhalt der Hash-Tabelle löschen.

Wenn Sie den Wert der Variablen *PATH* ändern, wird der Inhalt der Hash-Tabelle automatisch gelöscht. Wenn Sie die aktuelle Shell beenden, ist auch die Hash-Tabelle als Bestandteil dieser Shell gelöscht.

Sie sollten den Inhalt der Hash-Tabelle in der aktuellen Shell löschen, wenn folgender Fall eingetreten ist:

> Sie haben eine ausführbare Datei in einem Dateiverzeichnis erstellt, dessen Pfadname der Variablen *PATH* zugewiesen ist. Ein anderes Dateiverzeichnis, dessen Pfadname ebenfalls der Variablen *PATH* zugewiesen ist, enthält bereits eine Kommando-Datei gleichen Namens, die in der aktuellen Hash-Tabelle eingetragen ist. Dann führt die Shell immer das ältere Kommando aus, auch wenn in der Variablen *PATH* das zugehörige Dateiverzeichnis hinter dem zuerst genannten eingetragen ist.

Wenn Sie die Hash-Tabelle mit *hash -r* löschen, wird beim nächsten Aufruf das Kommando gestartet, das die Shell zuerst findet. In diesem Fall entscheidet also die Reihenfolge der Einträge in der Variablen *PATH*. Der Pfadname des zuerst gefundenen Kommandos wird in die Hash-Tabelle eingetragen; d.h., dieses Kommando wird ab jetzt immer ausgeführt, wenn Sie es mit seinem einfachen Dateinamen aufrufen.

Fehler *name*`: not found`
Diese Fehlermeldung kann eine der folgenden Ursachen haben:
– *name* ist in keinem der Dateiverzeichnisse enthalten, deren Pfad der Variablen *PATH* zugewiesen ist.
– *name* ist zwar in einem dieser Dateiverzeichnisse enthalten, aber nicht ausführbar.
– *name* ist ein eingebautes *sh*-Kommando oder eine Shell-Funktion.

Variable *PATH*
Suchpfad der Shell

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *hash*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Den Inhalt der Hash-Tabelle ausgeben:

```
$ echo $PATH
/bin:/usr/bin:.
$ hash
cat=/usr/bin/cat
ls=/usr/bin/ls
```

Beispiel 2 Ein Kommando in die Hash-Tabelle eintragen:

```
$ hash cat
$ hash
cat=/usr/bin/cat
```

Das Kommando */usr/bin/cat* wurde neu in die Hash-Tabelle eingetragen.

Beispiel 3 Der folgende Bildschirm-Dialog soll zeigen, wann die Hash-Tabelle gelöscht wird:

```
$ hash
cat=/usr/bin/cat
ls=/usr/bin/ls
$ sh
$ hash
$ cp <datei1> <datei2>
...
$ ls
...
$ hash
cp=/usr/bin/cp
ls=/usr/bin/ls
$ exit
$ hash
cat=/usr/bin/cat
ls=/usr/bin/ls
```

In der neu gestarteten Subshell ist die Hash-Tabelle leer. Wenn die Subshell beendet wird, gilt wieder die Hash-Tabelle der übergeordneten Shell. Die Hash-Tabelle der Subshell ist gelöscht.

hd Dateiinhalt hexadezimal ausgeben (hex dump)

hd gibt den Inhalt von Dateien hexadezimal, oktal, dezimal oder als Zeichenfolge aus. Die Lage von Zeichen innerhalb eines Zeichensatzes ist ebenfalls darstellbar.

Syntax **hd**[␣**-**format]...[␣**-A**][␣**-t**][␣**-s**␣offset[*****][**wlbk**]][␣**-n**␣zähler[*****][**wlbk**]][␣datei]...

Weder Format noch Offset noch Zähler angegeben
hd ist identisch mit *hd -abx -A*. D.h., Adressen und Bytes werden hexadezimal ausgegeben. Zusätzlich gibt *hd* alle Bytes, die druckbare Zeichen darstellen, als solche aus und schreibt für die nichtdruckbaren einen Punkt. Dabei stehen die Adressen am Anfang jeder Zeile, die Hexadezimaldarstellung der Bytes folgt in den nächsten Spalten und die Darstellung als Buchstaben bzw. Punkt steht in jeder Zeile ganz rechts.

Die Adressen werden relativ zum Dateianfang gezählt. Fehlt die Angabe einer Datei, dann wird von der Standard-Eingabe gelesen, sonst wird der Inhalt der angegebenen Dateien aufgelistet.

-format
Format, das festlegt, wie einzelne Byteblöcke interpretiert und ausgegeben werden sollen (siehe Abschnitt „Formatbeschreibung“ auf Seite 447)

-A (A - ASCII) *hd* gibt alle druckbaren Zeichen aus, für nichtdruckbare Zeichen steht ein Punkt. Die Zeichen werden in der Spalte ausgegeben, die rechts auf das erste Ausgabeformat folgt.

-t (t - text) Ist diese Option gesetzt, dann ignoriert *hd* alle Formatangaben, die keine Adressen betreffen. *hd* druckt jede Textzeile mit der Adressangabe am Zeilenanfang aus. Zu lange Zeilen werden unterteilt. Kontrollzeichen (Wert 0x00 bis 0x1f) werden entsprechend als Zeichen ^@ bis ^_ ausgegeben. Bytes, bei denen das höchste Bit gesetzt ist, werden, mit einer Tilde ~ davor, ohne das höchste Bit ausgegeben. Den Zeichen Dach ^, Tilde ~ und Gegenschrägstrich \ wird bei der Ausgabe ein Gegenschrägstrich vorangestellt.

In Spezialfällen werden Werte numerisch repräsentiert, z.B. DELETE (127) 7-Bit als \177 und DELETE (255) 8-Bit als \377.

-s␣offset[*****][**wlbk**]
Relative Adresse, ab der die Ausgabe des Dateiinhalts beginnen soll. Falls Sie keine Datei angeben oder über eine Pipe eingeben, werden entsprechend viele Bytes überlesen. *hd* bricht die Behandlung der aktuellen Datei ab, falls die Adressangabe fehlerhaft ist.

Die relative Adresse besteht aus einer Zahl, die Sie dezimal, hexadezimal (mit *0x* davor) oder oktal (mit *0* davor) angeben und optional einer Maßeinheit, die Sie direkt an die Zahl anhängen.

Mögliche Einheiten sind:

w 2 Byte (d.h. ein Wort)
l 4 Byte (d.h. ein langes Wort)
b 512 Byte (d.h. ein halbes KByte)
k 1024 Byte (d.h. ein KByte)

Um eine Hexadezimalzahl, die ja die Ziffer *b* enthalten darf, von der Einheit *b* zu unterscheiden, schreiben Sie in diesem Fall zwischen die Zahl und die Einheit *b* einen Stern *.

Mögliche Offsetangaben sind z.B.:
-s 111 (111 Byte), *-s 124l* (496 Byte), *-s 0xa*b* (5120 Byte), *-s 011k* (9216 Byte).

-s␣offset[*][wlbk] nicht angegeben:
hd gibt ab Dateianfang aus.

-n␣zähler[*****][**wlbk**]
Anzahl der Bytes, die *hd* ausgeben soll. Die Byte-Anzahl geben Sie ebenso wie die Relativadresse an: dezimal, hexadezimal oder oktal mit eventuell folgendem *w*, *l*, *b* oder *k* (siehe *-s*).

datei
Name der Datei, die *hd* auflisten soll. Pro Aufruf können Sie mehrere Dateinamen angeben.

datei nicht angegeben:
hd liest die Eingabezeilen von der Standard-Eingabe.

Formatbeschreibung

Ein Format setzt sich zusammen aus
– der Byteblockangabe (*a*, *b*, *c*, *l* oder *w*) und
– der Interpretationsart, gemäß der ein Byteblock auszugeben ist: hexadezimal (*x*), dezimal (*d*) oder oktal (*o*).

Innerhalb eines Formats werden sämtliche angegebenen Interpretationsarten auf alle angegebenen Byteblöcke angewendet. Formatangaben können zusammengesetzt und wiederholt werden, um Adressen, Zeichen, Wörter etc. verschieden auszugeben. Sie können z.B. *-ax -bx* zusammenfassen zu *-abx*. Oder Sie können mit *-cxdo* alle Zeichen hexadezimal, dezimal und oktal ausgeben lassen.

Byteblockangabe

a (a - address) Formatangabe für Adressen. Adressen werden nur auf eine Art interpretiert, hexadezimal, oktal oder dezimal. Die Adresse steht immer am Anfang jeder auszugebenden Zeile bzw. in der ersten Zeile eines Ausgabeblocks, falls die Formate mehrere Zeilen beanspruchten.

b (b - byte) Formatangabe für Bytes.

c (c - character) Formatangabe für Zeichen. Alle druckbaren Zeichen werden ausgegeben. C-Escape-Zeichen gibt *hd* so aus, wie sie in der Sprache definiert sind und die restlichen Zeichen oktal, hexadezimal oder dezimal, je nach Interpretationsart.

l (l - long word) Formatangabe für 4 Byte.

w (w - word) Formatangabe für 2 Byte.

Interpretationsart

x (x - hexadecimal) *hd* interpretiert Adressen oder Byteblöcke als Hexadezimalzahlen.

d (d - decimal) *hd* interpretiert Adressen oder Byteblöcke als Dezimalzahlen.

o (o - octal) *hd* interpretiert Adressen oder Byteblöcke als Oktalzahlen.

Keine Interpretationsart, aber Byteblockangabe angegeben:
hd interpretiert gemäß *-xdo*.

Keine Byteblockangabe, außer Adressen, angegeben:
hd verwendet zusätzlich zum angegebenen Adressformat *-bx*.

Keine Byteblockangabe, aber Interpretationsart angegeben:
hd interpretiert *-acbwl*.

-format nicht angegeben:
hd verhält sich wie *hd -abx -A*.

Siehe auch *od*

head Anfang einer Datei ausgeben (copy the first part of files)

head schreibt die ersten Zeilen aus dem Inhalt einer Datei auf die Standard-Ausgabe. Wenn Sie keine Datei angegeben haben, liest *head* von der Standard-Eingabe.

Syntax

Format 1: head[␣**-n**␣nummer][␣datei]

Format 2: head[␣**-**nummer][␣datei]

Format 1 **head**[␣**-n**␣nummer][␣datei]

-n␣nummer
Anzahl der auszugebenden Zeilen. *nummer* muss eine positive Dezimalzahl sein. Das Leerzeichen ␣ zwischen *-n* und *nummer* ist optional.

-n␣*nummer* nicht angegeben:
Die ersten 10 Zeilen werden ausgegeben.

datei
Name der Eingabedatei. Sie können auch mehrere Dateien angeben, die nacheinander bearbeitet werden. Wenn mehrere Dateien angegeben werden, dann beginnt die Ausgabe mit:
==>*datei*<==

Format 2 **head**[␣**-**nummer][␣datei]

-nummer
Anzahl der auszugebenden Zeilen. *nummer* muss eine positive Dezimalzahl sein.

-nummer nicht angegeben:
Die ersten 10 Zeilen werden ausgegeben.

datei
Name der Eingabedatei. Sie können auch mehrere Dateien angeben, die nacheinander bearbeitet werden. Wenn mehrere Dateien angegeben werden, dann beginnt die Ausgabe mit:
==>*datei*<==

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *head*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Sie wollen sich die ersten 5 Zeilen von drei Dateien ansehen. Dazu geben Sie das Kommando in folgender Form an:

```
$ head -n5 datei1 datei2 datei3
```

Die ersten fünf Zeilen der drei Dateien werden jeweils folgendermaßen auf die Standard-Ausgabe geschrieben:

```
==>datei1<==
Zeile 1-5 von Datei 1

==>datei2<==
Zeile 1-5 von Datei 2

==>datei3<==
Zeile 1-5 von Datei 3
```

Siehe auch *cat*, *more*, *tail*

iconv Code konvertieren (codeset conversion)

iconv liest Eingabezeichen aus einer Datei oder von der Standard-Eingabe, codiert die Eingabezeichen um und schreibt das Ergebnis auf die Standard-Ausgabe.

Die mit *iconv* möglichen Konvertierungen legen Sie durch Konvertierungstabellen fest, die sich unter */usr/lib/iconv* befinden.

So wandelt *iconv* z.B. die Zeichen aus dem Zeichensatz ISO 8859-1 in landesspezifische Zeichen aus den nationalen Varianten des Zeichensatzes ISO 646 (ASCII-Derivate) um, oder unterstützt die Umwandlung in umgekehrter Richtung (siehe *Beispiele*).

BS2000 Mit *iconv* können Sie auch Codekonvertierungen zwischen ISO646 und EDF03, also zwischen dem ASCII-7-bit-Code und EBCDIC vornehmen.

Syntax **iconv** ␣ **-f** ␣ ausgangscode ␣ **-t** ␣ zielcode [␣ datei]

-f ␣ ausgangscode
-t ␣ zielcode
(f - from, t - to) *iconv* erwartet die Konvertierungstabelle in der Datei */usr/lib/iconv/ausgangscode.zielcode.t*. Zeichen, die im Ziel-Zeichensatz nicht existieren, werden in Unterstrich _ verwandelt.

datei
Name der Datei, deren Inhalt umcodiert werden soll.

datei nicht angegeben:
iconv liest von der Standard-Eingabe.

Fehler `Not supported` *xx* `to` *yy*
Die gewünschte Umwandlung des Zeichensatzes *xx* in den Ziel-Zeichensatz *yy* wird von *iconv* nicht unterstützt.

Datei */usr/lib/iconv*
In diesem Dateiverzeichnis befinden sich die Standard-Konvertierungstabellen für die Umcodierung

/usr/lib/iconv/iconv_data
Hilfsdatei für *iconv*

/usr/lib/iconv/.t*
Konvertierungstabellen

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *iconv*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten). Während der Konvertierung der Datei wird diese Variable durch die *ausgangscode* Option aufgehoben.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Sie wollen alle Konvertierungstabellen auflisten:

```
$ ls /usr/lib/iconv
646.edf03.t   646es.8859.t  8859.646.t    8859.646es.t  8859.edf04.t
646da.8859.t  646fr.8859.t  8859.646da.t  8859.646fr.t  edf03.646.t
646de.8859.t  646it.8859.t  8859.646de.t  8859.646it.t  edf04.8859.t
646en.8859.t  646sv.8859.t  8859.646en.t  8859.646sv.t  iconv_data
```

Beispiel 2 Sie wollen den Inhalt der Datei *brief* konvertieren und das Ergebnis in die Datei *brief.conv* schreiben. Der Ausgangs-Zeichensatz ist die deutsche Variante des Zeichensatzes ISO 646, der Ziel-Zeichensatz soll der Zeichensatz ISO 8859-1 sein:

```
$ iconv -f 646de -t 8859 brief > brief.conv
```

Beispiel 3 Sie wollen den Inhalt der Datei *bs2000* von ASCII nach EBCDIC konvertieren und das Ergebnis in die Datei *bs2000.conv* schreiben:

```
$ iconv -f 646 -t edf03 bs2000 > bs2000.conv
```

id Benutzer-Identifikation ausgeben (return user identity)

Für den Prozess, der *id* aufgerufen hat, schreibt *id* folgendes auf die Standard-Ausgabe:
– die Benutzernummer (UID)
– die Benutzerkennung
– die Gruppennummer (GID)
– den Gruppennamen.

Stimmen die effektiven und realen Nummern bzw. Kennungen nicht überein, so gibt *id* beide aus.

Syntax

Format 1: id[␣**-a**][␣user]

Format 2: id␣**-G**[␣**-n**][␣user]

Format 3: id␣**-g**[␣**-nr**][␣user]

Format 4: id␣**-u**[␣**-nr**][␣user]

Format 1 **id**[␣**-a**][␣user]

-a (a - all) *id* gibt neben der Kennung und dem Namen des Benutzers alle Gruppen an, zu denen der aufrufende Prozess gehört. Auch alle Gruppen, zu denen der aufrufende Benutzer gehört, werden angegeben.

user
Loginname, für den die Informationen ausgegeben werden. Wird *user* angegeben, und der Prozess verfügt über die entsprechenden Zugriffsrechte, so werden die Benutzernummer und die Gruppennummer des ausgewählten Benutzers ausgegeben. In diesem Fall wird angenommen, dass die effektiven und die realen Nummern identisch sind. Wenn für den ausgewählten Benutzer mehr als eine zulässige Gruppenzugehörigkeit in der Datenbank aufgeführt ist, so werden diese ebenfalls ausgegeben.

user nicht angegeben: Wird der Operand *user* nicht angegeben, so gibt *id* die Benutzer- und Gruppennummer sowie die entsprechende Benutzerkennung und den Gruppennamen des aufrufenden Prozesses auf Standard-Ausgabe aus.

Format 2 **id**␣**-G**[␣**-n**][␣user]

-G Nur die verschiedenen Gruppennummern (effektive, reale und supplementary) werden im Format „%u\n“ ausgegeben. Liegen mehrere Gruppenzugehörigkeiten vor, so werden alle Gruppenzugehörigkeiten im Format „%u“ vor dem Neue-Zeile-Zeichen ausgegeben.

-n Gibt den Namen im Format „%s“ statt der Nummer im Format „%u“ aus.

user
siehe Format 1

Format 3 **id**␣**-g**[␣**-nr**][␣user]

-g Nur die effektive Gruppennummer wird ausgegeben.

-n Gibt den Namen als Zeichenkette aus.

-r Nur die reale Nummer wird ausgegeben.

user
siehe Format 1

Format 4 **id**␣**-u**[␣**-nr**][␣user]

-u Nur die effektive Benutzernummer wird ausgegeben.

-n Gibt den Namen als Zeichenkette aus.

-r Nur die reale Nummer wird ausgegeben.

user
siehe Format 1

Datei */etc/group*
Gruppendatei. Sie enthält die Gruppennamen sowie die zugehörigen Gruppennummern und Benutzerkennungen.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *id*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Wenn Sie Ihre aktuelle Benutzernummer, die zugehörige Benutzerkennung sowie Ihre aktuelle Gruppennummer und den Gruppennamen wissen wollen, dann geben Sie ein:

```
$ id
```

id gibt dann z.B. aus:

```
uid=227(USER1) gid=100(USROTHER) groups=100(USROTHER)
```

Siehe auch *logname, newgrp, who*
getuid() [4]

info Online-Diagnosetool

info gibt charakteristische Daten des POSIX-Subsystems während einer POSIX-Session aus. Neben allgemeinen Informationen über den Zustand des Subsystems werden schwerpunktmäßig Prozessinformationen zu den POSIX-Anwendern in verschiedenen Ausführlichkeitsstufen zur Verfügung gestellt.

Der POSIX-Anwender erhält Informationen über die allgemeinen Zustand des POSIX-Subsystems (Option *-s*), über die POSIX-Steuerparameter (Option *-p*) sowie Informationen über seine eigenen Prozesse.
Der POSIX-Verwalter erhält zusätzlich Informationen über alle im System laufenden Anwender-Prozesse.

Syntax

info␣**-d**|␣**-h**|␣**-s**|␣**-t**|␣**-p**[␣**-f**][␣**-u**␣user][␣**-g**␣gid][␣**-p**␣pid]

-d Erzeugen eine Dumps für das POSIX-Subsystem

-h Anzeigen der Hilfe

-p Kurzinformation über alle POSIX-Anwender-Prozesse

-s Allgemeine Informationen zum Zustand des POSIX-Subsystems

-t Ausgabe der POSIX-Steuerparameter (siehe Handbuch „Grundlagen für Anwender und Systemverwalter“ [1])

-pf Langinformation über alle POSIX-Anwender-Prozesse

-p␣**-u**␣user
Kurzinformation über alle Prozesse von *user*. *user* kann ein Benutzername oder eine Userid (uid) sein.

-p␣**-g**␣gid
Kurzinformation über alle Prozesse mit der Gruppenkennung *gid*.

-pf␣**-u**␣user
Langinformation über alle Prozesse von *user*. *user* kann ein Benutzername oder eine Userid (uid) sein.

-pf␣**-g**␣gid
Langinformation über alle Prozesse mit der Gruppenkennung *gid*.

-p␣**-p**␣pid
Ausführliche Information über den Prozess mit der Prozessnummer *pid*.

Beispiel 1 Der POSIX-Anwender lässt sich Informationen über das POSIX-Subsystem ausgeben:

```
$ info -s

GLOBALE SYSTEMINFORMATIONEN:
init-TSN: 38YK
System-Status: System ist aktiv !
Rootfsname: :POSX:$SYSROOT.FS.ROOT
Anzahl der POSIX-Anwender (ohne 'init' und 'info'):
  TU-Anwendungen       : 6
  TPR-Anwendungen      : 0
  Konnektierte Tasks   : 6

$
```

Beispiel 2 Der POSIX-Verwalter lässt sich Informationen über die POSIX-Anwender-Prozesse ausgeben:

```
# info -p

PROZESSINFORMATION ALLGEMEIN:

USERNAME    PID    PPID   TU/TPR       TSN       SYSCALL
ROOT      18989       1   TU-Task      8RJZ            3
ROOT          1       0   TU-Task      38YK            0
ROOT         16       1    ------      8RJ5           87
ROOT         23       1    ------      8RKC            3
ROOT       9059    9043    ------      5YQU           54
ROOT      18001      16    ------      9VKQ           87
VSX0       7050    7039    ------      5W3F            3
:
#
```

ipcrm Einrichtungen zur Interprozess-Kommunikation löschen (remove inter-process communication facilities)

ipcrm entfernt ein oder mehrere Semaphore, Nachrichten-Warteschlangen, gemeinsam benutzten Speicher (shared memory) oder Einrichtungen zur Interprozess-Kommunikation (IPC-Einrichtungen). Diese können Sie entweder durch deren Bezeichner oder durch den Schlüssel angeben, der bei der Erzeugung der jeweiligen IPC-Einrichtung verwandt wurde.

Bezeichner und Schlüssel von IPC-Einrichtungen erfahren Sie mit Hilfe des Kommandos *ipcs* (Ausgabe-Spalten *ID* bzw. *KEY*). Informationen über die Vorgänge bei der Entfernung einer Nachrichten-Warteschlange, eines gemeinsam benutzten Speichersegments oder eines Semaphors erhalten Sie in der Beschreibung von *msgctl()*, *shmctl()* bzw. *semctl()* im Referenzhandbuch „C-Bibliotheksfunktionen (BS2000/OSD)“ [4].

Syntax

ipcrm[␣option]...

option

-q␣msgqid
(q - queue) Die Nachrichten-Warteschlange mit der Kennzahl *msgqid* wird aus dem System entfernt und die damit verbundenen Datenstrukturen werden zerstört.

msgqid
Kennzahl der Nachrichten-Warteschlange, die entfernt werden soll. Diese Kennzahl gibt das Kommando *ipcs* in der Spalte *ID* aus.

-Q␣msgkey
(Q - queue) Die Nachrichten-Warteschlange mit dem Schlüssel *msgkey* wird aus dem System entfernt und die damit verbundenen Datenstrukturen werden zerstört.

msgkey
Schlüssel der Nachrichten-Warteschlange, die entfernt werden soll. Diesen Schlüssel gibt das Kommando *ipcs* in der Spalte *KEY* aus.

-m␣shmid
(m - memory) Der gemeinsam benutzte Speicher (shared memory) mit der Kennzahl *shmid* wird aus dem System entfernt, und die damit verbundenen Datenstrukturen werden zerstört.

shmid
Kennzahl des gemeinsam benutzten Speichers, der entfernt werden soll. Diese Kennzahl gibt das Kommando *ipcs* in der Spalte *ID* aus.

-M␣shmkey
(M - memory) Der gemeinsam benutzte Speicher (shared memory) mit dem Schlüssel *shmkey* wird aus dem System entfernt, und die damit verbundenen Datenstrukturen werden zerstört.

shmkey
Schlüssel des gemeinsam benutzten Speichers, der entfernt werden soll. Diesen Schlüssel gibt das Kommando *ipcs* in der Spalte *KEY* aus.

-s␣semid
(s - semaphor) Das Semaphor mit der Kennzahl *semid* wird aus dem System entfernt, und die damit verbundenen Datenstrukturen werden zerstört.

semid
Kennzahl des Semaphors, das entfernt werden soll. Diese Kennzahl gibt das Kommando *ipcs* in der Spalte *ID* aus.

-S␣semkey
(S - semaphor) Das Semaphor mit dem Schlüssel *semkey* wird aus dem System entfernt, und die damit verbundenen Datenstrukturen werden zerstört.

semkey
Schlüssel, mit dem das Semaphor erzeugt wurde, das Sie entfernen möchten. Diesen Schlüssel gibt das Kommando *ipcs* in der Spalte *KEY* aus.

Beispiel Zuerst lassen Sie mit *ipcs* einen Bericht über den Zustand von Einrichtungen zur Interprozess-Kommunikation ausgeben. Danach entfernen Sie die Nachrichten-Warteschlange mit der Kennzahl 40 aus dem System:

```
$ ipcs
IPC status from /dev/kmem as of Mon Mar 9 08:40:41 2009
T     ID     KEY        MODE       OWNER    GROUP
Message Queues:
q     40  0x0000004b -Rrw-rw-rw-  user1   usrother
Shared Memory:
Semaphores:
$ ipcrm -q 40
```

Siehe auch *ipcs*
msgctl(), *msgget()*, *semctl()*, *semget()*, *semop()*, *shmctl()*, *shmget()* [4]

ipcs Zustand von Interprozess-Kommunikationseinrichtungen anzeigen (inter-process communication status)

ipcs gibt Informationen über aktive Einrichtungen zur Interprozess-Kommunikation (IPC-Einrichtungen) aus.

Durch die Angabe von Optionen können Sie steuern,
- über welche Art von IPC-Einrichtungen Informationen ausgegeben werden
- welche Informationen ausgegeben werden.

Da sich die Zustände der IPC-Einrichtungen ändern können, während *ipcs* läuft, ist die ausgegebene Information nur zum Zeitpunkt der Abfrage aktuell.

Syntax

ipcs[␣option]...

Keine Option angegeben

ipcs gibt in Kurzform Informationen aus über
- Nachrichten-Warteschlangen (message queues)
- gemeinsam benutzten Speicher (shared memory)
- Semaphore

die derzeit im System aktiv sind. Die Bedeutung der Ausgabe-Spalten ist im Abschnitt „Ausgabe“ auf Seite 462 beschrieben.

option

Es gibt Optionen für die Art der IPC-Einrichtung und Optionen für die Art der Informationen.

Art der IPC-Einrichtung festlegen

-q (q - message queues) Informationen über aktive Nachrichten-Warteschlangen ausgeben.

-m (m - memory) Informationen über aktiven gemeinsam benutzten Speicher (shared memory) ausgeben.

-s (s - semaphores) Informationen über aktive Semaphore ausgeben.

Ist eine der Optionen *-q*, *-m* oder *-s* gesetzt, so werden nur Informationen über die entsprechenden IPC-Einrichtung ausgegeben. Die Kombination der Optionen ist möglich. Ist keine der Optionen gesetzt, so werden Informationen über alle Arten von IPC-Einrichtungen ausgegeben.

Art der Informationen festlegen

Im Folgenden sind die Optionen beschrieben, die festlegen, welche Informationen für die durch die Optionen *-q*, *-m* und *-s* festgelegten IPC-Einrichtungen ausgegeben werden. Die Bedeutung der Ausgabe-Spalten ist im Abschnitt „Ausgabe“ auf Seite 462 beschrieben.

-a (a - all) Alle Optionen, die die Art der ausgegebenen Informationen bestimmen, sind gesetzt. Dies ist die Kurzform für *-b*, *-c*, *-o*, *-p* und *-t*.

-b (b - biggest) Es wird die maximal zulässige Größe der jeweiligen IPC-Einrichtung ausgegeben:
- für Nachrichten-Warteschlangen die maximale Anzahl von Bytes in einer Nachricht, die in diese Nachrichten-Warteschlange eingereiht werden soll
- für gemeinsam benutzten Speicher (shared memory) die Größe der Speichersegmente
- für Semaphore die Zahl der Semaphore in jeder Semaphor-Gruppe

-c (c - creator) Die Benutzerkennung und der Gruppen-Name des Erzeugers der IPC-Einrichtung werden ausgegeben.

-o (o - outstanding) Informationen über noch zu bearbeitende IPC-Einrichtungen werden ausgegeben:
- Zahl der Nachrichten in Nachrichten-Warteschlangen
- Gesamtzahl der Bytes von Nachrichten in Nachrichten-Warteschlangen
- Anzahl von Prozessen, die einem gemeinsam benutzten Speichersegment zugeordnet sind.

-p (p - process) Es werden Informationen über Prozessnummern ausgegeben:
- Prozessnummer des letzten Prozesses, der eine Nachricht gesandt hat
- Prozessnummer des letzten Prozesses, der eine Nachricht empfangen hat
- Prozessnummer des Prozesses, der ein gemeinsam benutztes Speichersegment erzeugt hat
- Prozessnummer des letzten Prozesses, der ein gemeinsam benutztes Speichersegment benutzt hat (attach, detach).

-t (t - time) Zeit-Informationen ausgeben. Es wird der Zeitpunkt der letzten Änderung der Zugriffsrechte für alle IPC-Einrichtungen ausgegeben:
- Für Nachrichten-Warteschlangen wird der Zeitpunkt ausgegeben, zu dem zuletzt die Systemaufrufe *msgrcv()* [4] (Nachricht aus einer Nachrichten-Warteschlange lesen) oder *msgsnd()* [4] (Nachricht in eine Nachrichten-Warteschlange eintragen) angewandt wurden.
- Für gemeinsam benutzten Speicher wird der Zeitpunkt ausgegeben, zu dem zuletzt die Systemaufrufe *shmat()* [4] oder *shmdt()* [4] angewandt wurden.
- Für Semaphore wird der Zeitpunkt ausgegeben, zu dem zuletzt der Systemaufruf *semop()* [4] auf ein Semaphor angewandt wurde.

Ausgabe

Im Folgenden sind die Spalten-Überschriften und die Bedeutung der Spalten in der Ausgabe eines *ipcs*-Aufrufs aufgeführt. Die in runden Klammern (...) eingeschlossenen Buchstaben stehen dabei für die Optionen, die die Ausgabe der entsprechenden Spalten-Überschrift zur Folge haben. *alle* bedeutet, dass die betreffende Spalten-Überschrift in jedem Fall ausgegeben wird.

CBYTES (a,o)
: Die Anzahl der Bytes in den Nachrichten, die in der zugehörigen Nachrichten-Warteschlange auf ihre Bearbeitung warten.

CGROUP (a,c)
: Der Name der Gruppe zu der der Erzeuger der IPC-Einrichtung gehört.

CREATOR (a,c)
: Die Benutzerkennung des Erzeugers der IPC-Einrichtung.

CTIME (a,t)
: Der Zeitpunkt, zu dem der Eintrag für die zugehörige IPC-Einrichtung erstellt oder geändert wurde.

GROUP (alle)
: Der Name der Gruppe zu der der Eigentümer der IPC-Einrichtung gehört.

ID (alle)
: Die Kennzahl der IPC-Einrichtung.

KEY (alle)
: Der Schlüssel, der beim Erzeugen der IPC-Einrichtung mit *msgget()* (Nachrichten-Warteschlange einrichten) oder *semget()* (Semaphor-Menge anlegen) als Argument benutzt wurde.

LRPID (a,p)
: Die Prozessnummer des Prozesses, der zuletzt eine Nachricht aus der zugehörigen Nachrichten-Warteschlange empfangen hat.

LSPID (a,p)
: Die Prozessnummer des Prozesses, der zuletzt eine Nachricht an die zugehörige Nachrichten-Warteschlange geschickt hat.

MODE (alle)
Die Zugriffsmodi und Status-Anzeigen für die IPC-Einrichtung. Der Modus besteht aus 11 Zeichen, die folgende Bedeutung haben:

Das erste Zeichen kann sein:

S wenn ein Prozess auf ein *msgsnd()* wartet
D wenn das betreffende gemeinsam benutzte Speichersegment freigegeben wurde. Es verschwindet, wenn es vom letzten Prozess, der dem Segment zugeordnet ist, freigegeben wird.
– wenn kein Prozess auf ein *msgsnd()* wartet.

Das zweite Zeichen kann sein:

R wenn ein Prozess auf ein *msgrcv()* wartet
C Das betreffende gemeinsam benutzte Speichersegment wird bereinigt, wenn der erste Zugriff (attach) ausgeführt wird.
– wenn kein Prozess auf ein *msgrcv()* wartet.

Die folgenden 9 Zeichen werden in drei Gruppen zu je drei Bits unterteilt:
– Die erste Gruppe steht für die Zugriffsrechte des Eigentümers der IPC-Einrichtung.
– Die zweite steht für die Zugriffsrechte der Benutzer, die zur selben Benutzer-Gruppe wie der Eigentümer der IPC-Einrichtung gehören.
– Die dritte Gruppe steht für die Zugriffsrechte aller anderen Benutzer.

In jeder Gruppe steht das erste Zeichen für das Leserecht, das zweite Zeichen für das Schreibrecht bzw. für das Recht, den Eintrag für die IPC-Einrichtung zu ändern. Das dritte Zeichen in jeder Gruppe wird zur Zeit noch nicht benutzt.

Die Zugriffsrechte werden folgendermaßen angegeben:

r Leserecht ist vorhanden
w Schreibrecht ist vorhanden
a Recht, den Eintrag für die IPC-Einrichtung zu ändern, ist vorhanden.
– Das betreffende Recht ist nicht vorhanden.

NSEMS (a,b)
Die Zahl der Semaphore in der zugehörigen Semaphor-Menge.

OTIME (a,t)
Der Zeitpunkt, zu dem die letzte Semaphor-Operation auf die zugehörige Semaphor-Menge angewandt wurde.

OWNER (alle)
Die Benutzerkennung des Eigentümers der IPC-Einrichtung.

QBYTES (a,b)
Die Anzahl der Bytes, die diejenigen Nachrichten maximal enthalten dürfen, die in der Nachrichten-Warteschlange auf ihre Bearbeitung warten.

QNUM (a,o)
Die Anzahl der Nachrichten, die momentan in der zugehörigen Nachrichten-Warteschlange enthalten sind.

RTIME (a,t)
Der Zeitpunkt, zu dem die letzte Nachricht aus der zugehörigen Nachrichten-Warteschlange empfangen wurde.

STIME (a,t)
Der Zeitpunkt, zu dem die letzte Nachricht an die zugehörige Nachrichten-Warteschlange geschickt wurde.

NATTCH (a,o)
Die Anzahl der Prozesse, die dem betreffenden gemeinsam benutzten Speichersegment zugeordnet sind.

SEGSZ (a,b)
Die Größe des betreffenden gemeinsam benutzten Speichersegments.

CPID (a,p)
Die Nummer des Prozesses, der das gemeinsam benutzte Speichersegment angelegt hat.

LPID (a,p)
Die Nummer des letzten Prozesses, der ein gemeinsam benutztes Speichersegment benutzt hat (attach, detach).

ATIME (a,t)
Der Zeitpunkt, zu dem der letzte Zugriff (attach) auf das betreffende gemeinsam benutzte Speichersegment beendet war.

DTIME (a,t)
Der Zeitpunkt, zu dem die letzte Freigabe (detach) eines gemeinsam benutzten Speichersegment beendet war.

T (alle)
Typ der IPC-Einrichtung:

- *q* steht für Nachrichten-Warteschlange
- *m* steht für gemeinsam benutzten Speicher (shared memory)
- *s* steht für Semaphor.

Datei *etc/group*

Die Datei */etc/group* enthält alle eingerichteten Benutzergruppen.

Beispiel Das Programm *server* richtet eine Nachrichten-Warteschlange ein. Sie starten dieses Programm im Hintergrund und fragen dann mit *ipcs* den Zustand von Einrichtungen zur Interprozess-Kommunikation ab:

```
$ server &
$ ipcs
IPC status from /dev/kmem as of Mon Mar 9 08:40:41 2009
T     ID     KEY        MODE       OWNER    GROUP
Message Queues:
q     40  0x0000004b -Rrw-rw-rw-  user1   usrother
Shared Memory:
Semaphores:
```

Siehe auch *ipcrm*

msgctl(), *msgget()*, *semctl()*, *semget()*, *semop()*, *shmctl()* [4]

jobs Auftragsinformationen ausgeben (display status of jobs in the current session)

jobs schreibt auf die Standard-Ausgabe. Es gibt Informationen zu den angegebenen Aufträgen aus oder, falls *job-id* fehlt, zu allen aktiven Aufträgen.

Syntax **jobs**[␣**-l**|**p**][␣**-n**][␣job-id]...

-l Diese Option gibt zusätzlich noch die Prozessnummern aus.

-p Nur die Prozessgruppe wird ausgegeben.

-n Nur die bereits beendeten Aufträge werden ausgegeben.

job-id
Informationen über die angegebenen Aufträge werden ausgegeben. Der Abschnitt *Aufträge* im Kapitel „Kommandos von der POSIX-Shell aus eingeben“ enthält eine Beschreibung des Formats von *job-id*.

job-id nicht angegeben: Informationen über alle aktiven Aufträge werden ausgegeben.

Internationale Umgebung
Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *jobs* aus:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *bg, fg, kill, wait*

join Zwei Dateien nach Vergleichsfeldern verbinden (relational database operator)

join vergleicht zwei Dateien nach Vergleichsfeldern und verbindet alle Zeilenpaare, deren Vergleichsfeld identisch ist. Das Ergebnis gibt *join* auf die Standard-Ausgabe aus.

Beim *join*-Aufruf legen Sie für jede der beiden Dateien fest, welches Feld das Vergleichsfeld sein soll. Ein Feld wird von zwei Feldtrennzeichen begrenzt. *join* vergleicht jede Zeile der ersten Datei mit den Zeilen der zweiten Datei. Für jedes Zeilenpaar mit identischem Vergleichsfeld gibt *join* auf die Standard-Ausgabe eine Ausgabezeile aus, die sich aus bestimmten Feldern beider Zeilen zusammensetzt.

Vor dem Aufruf beachten:
Jede Eingabedatei muss in ihrem Vergleichsfeld gemäß der gültigen Sortier-Reihenfolge sortiert sein (siehe *sort*). Bei Standard-Feldtrennung (*join* ohne Option *-t*) dürfen beim Sortieren führende Feldtrennzeichen nicht berücksichtigt werden (siehe *sort*, Option *-b*). Geben Sie dagegen *join* mit Option *-t* an, müssen Sie führende Trennzeichen beim Sortieren berücksichtigen (siehe *sort* ohne Option *-b*).

Syntax

Format 1: join[␣**-a**␣n|␣**-v**␣n][␣**-e**␣zeichenkette][␣**-o**␣liste][␣**-t**␣c][␣**-1**␣feld]
[␣**-2**␣feld]␣datei1␣datei2

Format 2: join[␣**-a**␣n][**-e**␣zeichenkette][**-j**␣feld][**-j1**␣feld][**-j2**␣feld]
[␣**-o**␣liste...][␣**-t**␣c]␣datei1␣datei2

Die Formate werden gemeinsam beschrieben, da die Option *-j␣field* in Format 2 den Optionen *-1␣feld -2␣ feld* in Format 1 entspricht. *-j1␣feld* ist äquivalent zu *-1␣feld* und *-j2␣ feld* ist äquivalent zu *-2␣ feld*.

Keine Option angegeben

Vergleichsfeld für beide Dateien ist das erste Feld. Feldtrennzeichen für die Eingabezeilen sind Leerzeichen, Tabulatorzeichen und Neue-Zeile-Zeichen. Aufeinanderfolgende Feldtrennzeichen werden als ein einziges Feldtrennzeichen interpretiert; führende Feldtrennzeichen werden ignoriert.
Für jedes Zeilenpaar mit identischem Vergleichsfeld gibt *join* auf die Standard-Ausgabe eine Ausgabezeile aus. Diese Ausgabezeile enthält in dieser Reihenfolge:

– das gemeinsame Vergleichsfeld
– den Rest der Eingabezeile aus der ersten Datei
– den Rest der Eingabezeile aus der zweiten Datei.

Die Felder der Ausgabezeilen sind durch ein Leerzeichen voneinander getrennt.

option

-a␣n
(a - additional output) *join* gibt zusätzlich zur normalen Ausgabe alle Zeilen der *n*-ten Eingabedatei aus, deren Vergleichsfeld mit keinem Vergleichsfeld der anderen Datei übereinstimmt.
Für *n* können Sie 1 oder 2 angeben. Soll die Ausgabe für beide Vergleichsdateien erfolgen, dann geben Sie *-a␣1 -a␣2* an.

Die Option *-a* darf nicht zusammen mit der Option *-v* angegeben werden.

-v␣n
Statt der Standardausgabe wird für jede Zeile in *n* ohne Entsprechung eine Zeile erzeugt. *n* kann 1 oder 2 sein. Wenn sowohl *-v␣1* als auch *-v␣2* angegeben wird, werden alle Zeilen ohne Entsprechung ausgegeben.

-e␣zeichenkette
(e - empty output fields) *join* ersetzt leere Ausgabefelder durch die angegebene Zeichenkette.

-j[n]␣m
Als Vergleichsfeld für die *n*-te Datei wird das *m*-te Feld festgelegt. Für *n* können Sie 1 oder 2 angeben, für *m* eine ganze Zahl größer gleich 1.
Wenn Sie für die andere Datei keine Option *-j* angeben, dann ist das Vergleichsfeld für diese andere Datei das 1. Feld.

n nicht angegeben:
Vergleichsfeld für beide Dateien ist das *m*-te Feld.

-j nicht angegeben:
Vergleichsfeld für beide Dateien ist das 1. Feld.

-o␣liste
(o - output format) *join* ändert das Format der Ausgabezeilen: Die Ausgabezeilen enthalten dann der Reihe nach die in *liste* angegebenen Felder. Das gemeinsame Vergleichsfeld wird nur dann ausgegeben, wenn Sie es ausdrücklich in *liste* angegeben haben.

Für *liste* geben Sie eine Liste an, die aus Elementen der Form *n.m* besteht, wobei *n* gleich 1 oder 2 und *m* größer gleich 1 ist. Ein Element *n.m* steht für das *m*-te Feld der *n*-ten Datei. Die Elemente trennen Sie durch Leer- oder Tabulatorzeichen.

-t␣c
Das Zeichen *c* wird als Feldtrennzeichen sowohl für die Eingabe- als auch für die Ausgabezeilen definiert. Jedes Vorkommen von *c* wird als Feldtrennzeichen interpretiert, d.h.
– zwei aufeinanderfolgende *c* kennzeichnen ein leeres Feld und
– ein führendes *c* kennzeichnet ein leeres erstes Feld.

Zusätzlich ist das Neue-Zeile-Zeichen Feldtrennzeichen für die Eingabezeilen.
Die Standard-Feldtrennzeichen Leer- und Tabulatorzeichen werden nur dann als Feldtrennzeichen interpretiert, wenn Sie diese Zeichen für *c* angeben.

-1␣feld
Verbindet das Feld *feld* aus Datei 1. Felder sind dezimale Ganzzahlen ab 1.

-2␣feld
Verbindet das Feld *feld* aus Datei 2. Felder sind dezimale Ganzzahlen ab 1.

datei1 datei2
Namen der beiden Dateien, die *join* nach Vergleichsfeldern verbinden soll.
Wenn Sie für *datei1* einen Bindestrich - angeben, liest *join* von der Standard-Eingabe.

Wenn die Dateien nicht nach ihrem Vergleichsfeld sortiert sind, dann bearbeitet *join* nicht alle Zeilen!

Wenn Sie für *datei1* einen numerischen Dateinamen (z.B. 1.2) angeben und diesem Dateinamen die Option *-o* unmittelbar voranstellen, dann kann das zu Problemen führen. Wenn Sie also einen numerischen Dateinamen angeben wollen, dann geben Sie ihn mit Schrägstrich (z.B. ./1.2) an.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *join*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE beeinflusst die Sortierreihenfolge.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 In der Datei *ort* ist einem Namen ein Ort zugeordnet, in der Datei *betrag* sind denselben Namen je ein Betrag und ein Datum zugeordnet. Beide Dateien sind nach den Namen sortiert. *join* soll beide Dateien nach den Namen verbinden.

Inhalt der Datei *ort*:

```
Albert München
Hugo Stuttgart
Ilse Hamburg
```

Inhalt der Datei *betrag*:

```
Albert   287.56  20.03.94
Hugo      23.15  25.06.93
Hugo     167.87  16.12.93
Ilse    1212.12  12.12.94
Ilse       1.98  01.01.94
```

Verbinden der beiden Dateien nach dem ersten Vergleichsfeld:

```
$ join ort betrag
Albert Muenchen 287.56 20.03.94
Hugo Stuttgart 23.15 25.06.93
Hugo Stuttgart 167.87 16.12.93
Ilse Hamburg 1212.12 12.12.94
Ilse Hamburg 1.98 01.01.94
```

Verbinden der beiden Dateien und spaltenweise formatieren mit *awk*:

```
$ join ort betrag | awk '{printf("%-10s %-15s %-10s %-10s\n",$1,$2,$3,$4)}'
Albert    Muenchen       287.56    20.03.94
Hugo      Stuttgart       23.15    25.06.93
Hugo      Stuttgart      167.87    16.12.93
Ilse      Hamburg       1212.12    12.12.94
Ilse      Hamburg          1.98    01.01.94
```

Beispiel 2 In der Datei *stadt* ist einer Stadt ein Name zugeordnet, in der Datei *betrag* (siehe *Beispiel 1*) ist einem Namen ein Betrag und ein Datum zugeordnet. Die Datei *stadt* ist nach den Städten, die Datei *betrag* nach den Namen sortiert. *join* soll beide Dateien nach den Namen verbinden.

Inhalt der Datei *stadt*:

```
Augsburg     Egon
Hamburg      Ilse
Muenchen     Albert
Muenchen     Franz
Stuttgart    Hugo
```

Vergleichsfeld für die Datei *stadt* ist das 2. Feld, für die Datei *betrag* das 1. Feld.
Bevor Sie die Dateien verbinden, müssen Sie die Datei *stadt* nach dem 2. Feld sortieren. Formatieren Sie anschließend wieder spaltenweise mit *awk*:

```
$ sort -b +1 stadt | join -j1 2 - betrag | \
> awk '{printf("%-10s %-15s %-10s %-10s\n",$1,$2,$3,$4)}'
Albert     Muenchen        287.56     20.03.94
Hugo       Stuttgart       23.15      25.06.93
Hugo       Stuttgart       167.87     16.12.93
Ilse       Hamburg         1212.12    12.12.94
Ilse       Hamburg         1.98       01.01.94
```

Siehe auch *awk*, *comm*, *sort*, *uniq*

kill Signale an Prozesse senden (terminate or signal processes)

Das Kommando *kill* sendet ein Signal an eine durch die Prozessnummer (PID) bestimmte Menge von Prozessen. Um die Prozessnummer des Prozesses zu erfahren, an den Sie ein Signal senden wollen, verwenden Sie das Kommando *ps*.

Syntax

Format 1: kill[␣**-**signal]␣prozessnummer␣...

Format 2: kill␣**-s**␣signal␣prozessnummer␣...

Format 3: kill␣**-**signal␣**-**gruppennummer␣...

Format 4: kill␣**-l**␣[exit-status]

Format 1 **Signale an Prozesse senden**

kill[␣**-**signal]␣prozessnummer␣...

-signal

Signal, das an die Prozesse gesendet werden soll. Sie können dieses Signal in Form einer Zahl oder als symbolischen Namen angeben. Ein symbolischer Name ist die Bezeichnung eines Signals entsprechend der Definition in der Include-Datei <*sys/signal.h*>; das Präfix SIG wird allerdings nicht angegeben. Die Liste dieser Namen können Sie sich mit der Option *-l* ausgeben lassen.

Alle in der Include-Datei <*sys/signal.h*> definierten Signale können angegeben werden. Die Signale mit folgenden Nummern und symbolischen Namen sind auf Kommando-Ebene von Bedeutung:

1 - SIGHUP:
Verbindung zu Datensichtstation unterbrechen (hangup)

2 - SIGINT:
unterbrechen durch DEL (interrupt)

3 - SIGQUIT:
abbrechen (quit)

9 - SIGKILL:
unbedingter Prozessabbruch (kill)

15 - SIGTERM:
Programmbeendigung (software termination)

Zusätzlich wird der symbolische Name 0 erkannt, der den Signalwert Null darstellt.

signal nicht angegeben:
kill sendet das Signal SIGTERM (15) an die angegebenen Prozesse. Dadurch werden Prozesse, die dieses Signal nicht abfangen oder ignorieren, in der Regel abgebrochen.

prozessnummer
Nummer des Prozesses, an den Sie ein Signal senden wollen. Benutzer können nur an eigene Prozesse Signale senden.
Der POSIX-Verwalter kann Signale an alle Prozesse senden.

Die aktuellen Prozessnummern gibt das Kommando *ps* aus.

0 als Prozessnummer bedeutet: Das angegebene Signal wird an alle Prozesse aus Ihrer Prozessgruppe gesendet.

Format 2 **Signale an Prozesse senden**

kill ␣ **-s** ␣ signal ␣ prozessnummer ␣ ...

-s ␣ signal
Signal, das an die Prozesse gesendet werden soll. Sie können dieses Signal in Form einer Zahl oder als symbolischen Namen angeben. Ein symbolischer Name ist die Bezeichnung eines Signals entsprechend der Definition in der Include-Datei <*sys/signal.h*>; das Präfix SIG wird allerdings nicht angegeben. Die Liste dieser Namen können Sie sich mit der Option *-l* ausgeben lassen.

Format 3 **Signale an Prozessgruppen senden**

kill ␣ **-**signal ␣ **-**gruppennummer ␣ ...

-signal
Signal, das an die Prozesse einer Prozessgruppe gesendet werden soll. Sie können dieses Signal in Form einer Zahl oder als symbolischen Namen angeben. Ein symbolischer Name ist die Bezeichnung eines Signals entsprechend der Definition in der Include-Datei <*sys/signal.h*>; das Präfix SIG wird allerdings nicht angegeben. Die Liste dieser Namen können Sie sich mit der Option *-l* ausgeben lassen.

-gruppennummer
kill sendet das Signal an alle Prozesse der Prozessgruppe *gruppennummer*.

Benutzer können nur an eigene Prozessgruppen Signale senden.
Der POSIX-Verwalter kann Signale an alle Prozessgruppen senden.

Wenn Sie für *gruppennummer* die Zahl *1* angeben, so wird *signal* an alle Prozesse gesendet, deren realer User gleich dem effektiven User des kill-Kommandos ist. Als privilegierter Benutzer senden Sie das Signal an alle Prozesse, mit Ausnahme bestimmter Systemprozesse.

Format 4 **Symbolische Signalnamen auflisten**

kill␣**-l**␣[exit-status]

-l *kill* listet die symbolischen Signalnamen auf.

exit-status
Signalnummer oder Endestatus des Prozesses, der durch ein Signal beendet wird.

Fehler `no such process`
Sie haben für *prozessnummer* einen ungültigen Wert angegeben.

`no such process group`
Sie haben für *gruppennummer* einen ungültigen Wert angegeben.

Datei *<sys/signal.h>*
Include-Datei, in der die symbolischen Namen der Signale definiert sind.

Internationale Umgebung
Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *kill* aus:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Der Prozess mit der Nummer 312 wird durch das Signal SIGKILL (9) beendet:

```
$ kill -9  312
```

Beispiel 2 Das folgende Beispiel gibt den Status eines beendeten Auftrags aus:

```
job
stat=$?
if [ $stat -eq 0 ]
then
      echo job completed successfully.
elif [ $stat -gt 128 ]
then
      echo job terminated by signal SIG$(kill -l $stat).
else
      echo job terminated with error code $stat.
fi
```

Um zu vermeiden, dass bei einem Anfangsargument mit einer negativen Zahl entweder eine Signalnummer oder eine Prozessgruppe angegeben werden kann und dadurch Zweideutigkeiten entstehen, schreibt ISO POSIX-2 DIS vor, dass immer ersteres angenommen wird. Wenn also das Standardsignal an eine Prozessgruppe gesendet werden soll (z.B. 123), muss eine Anwendung ein Kommando ähnlich dem folgenden verwenden:

```
kill -TERM -123
kill -- -123
```

Siehe auch *ps, trap*
kill(), signal() [4]

last Zuletzt angemeldete Benutzer anzeigen (display last logged in users)

Mit dem Kommando *last* können die zuletzt (und gegenwärtig) angemeldeten Benutzer angezeigt werden. Das Kommando gibt den Benutzernamen, den Terminal-Namen, den fernen Rechnernamen (falls vorhanden) und die An- und Abmeldezeiten aus.

Syntax

last [**-n** number] [**-f** filename] [**-a**] [name | tty]

-n number
Maximale Anzahl anzuzeigender Datensätze. Standardmäßig werden alle Datensätze angezeigt.

-f filename
Name der zu verwendenden wtmpx/utmpx-Datei; Standardwert: */var/adm/wtmpx*.

-a Alternatives Anzeige-Layout; der Rechnername wird in der rechten Spalte ausgegeben. Auf diese Weise wird der Rechnername nicht abgeschnitten, falls es sich um einen vollqualifizierten DNS-Namen handelt.

name | tty
Nur Datensätze, die den angegebenen Benutzernamen oder Terminal-Namen betreffen, werden ausgegeben.

Beim POSIX-Start wird ein spezieller Datensatz mit dem Benutzernamen `reboot` geschrieben.

Datei

/var/adm/wtmpx
Datei mit allen gegenwärtig und zuvor angemeldeten Benutzern.

/var/adm/utmpx
Datei mit allen gegenwärtig angemeldeten Benutzern.

Beispiel 1 Alle Informationen aus */var/adm/wtmpx* anzeigen:

```
$ last
froede    pts/1        mch7509d.mch.fsc Tue May 10 13:00   still logged in
FROEDE    pts/0        MCH4987D         Tue May 10 13:00   still logged in
TSOS      term/001     ~                Mon May  9 12:46 - 12:58 (1+00:11)
TSOS      term/002     ~                Fri May  6 10:47   still logged in
TSOS      term/002     ~                Wed May  4 21:53 - 21:55  (00:02)
SYSROOT   sf/003       ~                Wed May  4 21:39 - 21:39  (00:00)

...
```

```
SYSROOT   sf/002       ~               Thu Apr 21 20:41 - 20:41  (00:00)
reboot    ~            ~               Thu Apr 21 20:40

/var/adm/wtmpx begins Thu Apr 21 20:40
$
```

Beispiel 2 Informationen über alle gegenwärtig angemeldeten Benutzer anzeigen:

```
$ last -f /var/adm/utmpx
froede    pts/1        mch7509d.mch.fsc Tue May 10 13:00   still logged in
FROEDE    pts/0        MCH4987D         Tue May 10 13:00   still logged in
TSOS      term/002     ~                Fri May  6 10:47   still logged in
reboot    ~            ~                Thu Apr 21 20:40

/var/adm/utmpx begins Thu Apr 21 20:40
$
```

Beispiel 3 Anmeldeinformationen über den Benutzer FROEDE im alternativen Ausgabeformat anzeigen:

```
$ last -a froede
froede    pts/1        Tue May 10 13:00   still logged in
mch7509d.mch.fsc.net
FROEDE    pts/0        Tue May 10 13:00   still logged in  MCH4987D
FROEDE    term/001     Wed May  4 10:28 - 10:34  (00:06)   ~
FROEDE    pts/0        Mon May  2 17:10 - 15:27 (1+22:17)  MCH4987D
FROEDE    term/002     Fri Apr 29 13:06 - 13:08  (00:02)   ~

/var/adm/wtmpx begins Thu Apr 21 20:40
$
```

Beispiel 4 Information über den letzten Neustart anzeigen:

```
$ last reboot
reboot    ~            ~               Thu Apr 21 20:40

/var/adm/wtmpx begins Thu Apr 21 20:40
$
```

Siehe auch *who*

let Arithmetische Berechnungen (integer arithmetic)

Durch das eingebaute Shell-Kommando *let* steht Ganzzahl-Arithmetik zur Verfügung. Die Berechnungen werden auf der Basis von *Long*-Arithmetik durchgeführt. Konstanten werden in der Form [*basis*#]*n* dargestellt. Dabei ist *basis* eine ganze Zahl zwischen 2 und 36 und gibt die Basis an, und *n* ist eine Zahl zu dieser Basis. Fehlt *basis*, dann wird im Zehnersystem gerechnet.

Syntax

Format 1: **let**␣ausdruck␣...

Format 2: **((**ausdruck**))**␣...

Format 1 **let**␣ausdruck

ausdruck
: Jedes Argument ist ein arithmetischer Ausdruck. Die berechneten Ergebnisse werden ausgegeben.

Format 2 **((**ausdruck**))**

Da einige der arithmetischen Operatoren für die POSIX-Shell entwertet werden müssen, wurde eine alternative Form zum eingebauten Kommando *let* eingeführt. Bei jedem Kommando, das mit doppelter runder Klammer auf *((* beginnt, werden alle Zeichen bis zum schließenden runden Klammerpaar *))* als entwertet genommen.
((a=a+b)) entspricht *let "a=a+b"*.

Arithmetische Berechnungen

Der arithmetischer Ausdruck ist stark an die Programmiersprache C angelehnt. Er benutzt dieselbe Syntax, gleiche Vorrangregeln und Assoziativität. Alle unerlässlichen Operatoren außer ++, --, *?:* und Komma , sind vorhanden. Auf den Wert von Variablen kann über deren Namen zugegriffen werden, Sie müssen kein Dollarzeichen verwenden. Wenn der Wert einer Variablen eingesetzt wird, dann wird ihr Wert als arithmetischer Ausdruck berechnet.

Durch die Option *-i* des eingebauten Kommandos *typeset* kann als Attribut für die interne Darstellung des Wertes einer Variablen die Ganzzahldarstellung gewählt werden. Bei jeder Wertzuweisung auf eine Variable mit dem *-i*-Attribut wird eine arithmetische Berechnung durchgeführt. Wird keine Basis für die Berechnungen angegeben, dann wird die Basis der ersten Wertzuweisung an die Variable verwendet. Diese Basis wird auch bei der Durchführung von Parameterersetzung verwendet.

Endestatus

0 wenn der Wert des letzten Ausdrucks ungleich 0 war

1 sonst.

Fehler `sh: *ausdruck*: bad number`
Fehlerhafter Ausdruck

Beispiel Das nachfolgende Beispiel zeigt eine einfache Rechenoperation. Es werden beide Schreibweisen von *let* verwendet.

```
$ let var=10
$ echo $var
10
$ ((var=var-3))
$ echo $var
7
```

lex Scanner erstellen (generate programs for lexical tasks)

lex erzeugt ein C-Programm aus einer *Datei*, die den „*lex*-Quelltext“ enthält, den Sie für das vorliegende Problem entwickelt haben. Ein *lex*-Quelltext besteht aus höchstens drei Abschnitten: Definitionen, Regeln und Benutzerfunktionen. Die Regeln geben an, welche Muster in einem Eingabetext gesucht und welche Aktionen ausgeführt werden sollen, wenn ein Muster gefunden wurde. Sie müssen angegeben werden. Die Definitionen und Benutzerfunktionen sind optional.

lex erzeugt eine Datei mit dem Namen *lex.yy.c*. Wenn *lex.yy.c* mit der Lex-Bibliothek übersetzt und gebunden wird, kopiert es die Eingabe auf die Ausgabe, es sei denn, ein in der Datei angegebenes Muster wird gefunden. In diesem Fall wird der entsprechende Programmtext ausgeführt. Das Muster, für das eine Übereinstimmung gefunden wurde, befindet sich in *yytext[]*, einem externen Zeichenfeld. Die Prüfung auf Übereinstimmung wird in der Reihenfolge der Suchmuster in der Eingabedatei durchgeführt.

Syntax

lex[␣**-ctvnV**][␣**-Q**[**y**|**n**]][␣datei ...]

Die *lex*-Optionen haben folgende Bedeutung:

-c steht für die Verwendung von C-Aktionen und ist der Standard

-t das Programm wird in die Datei *lex.yy.c*, nicht auf Standardausgabe geschrieben

-v liefert eine zweizeilige Statistik-Zusammenfassung

-n verhindert Ausdrucken der Zusammenfassung von *-v*

-V gibt Versionsinformationen auf die Standard-Fehlerausgabe aus

-Q[**y**|**n**]
legt fest, ob Versionsinformationen an die Ausgabedatei *lex.yy.c* ausgegeben werden sollen. *y*|*n* steht für eine ja/nein-Angabe in der jeweils eingestellten Sprachumgebung. In einer englischsprachigen Umgebung geben Sie *-Qy* an, um Versionsinformationen in die Datei *lex.yy.c* zu schreiben und *-Qn* um keine Versionsinformationen auszugeben. In einer deutschsprachigen Umgebung müssen Sie beispielsweise *-Qj* oder *-Qn* angeben. Standardmäßig werden keine Versionsinformationen ausgegeben.

datei
Eingabedatei. Mehrere Dateien werden wie eine Einzeldatei behandelt.

datei nicht angegeben
Wenn keine Datei angegeben wird, wird die Standardeingabe verwendet.

Bestimmte Standard-Tabellengrößen sind für einige Benutzer zu klein. Die Tabellengrößen für den erzeugten endlichen Automaten können im Definitionsabschnitt gesetzt werden:

%p *n*	Anzahl der Positionen ist *n* (Standard 2500)
%n *n*	Anzahl der Zustände (Standard 500)
%e *n*	Anzahl der Knoten des Syntaxbaums ist *n* (1000)
%a *n*	Anzahl der Übergänge ist *n* (2000)
%k *n*	Anzahl der gepackten Zeichenklassen ist *n* (2500)
%o *n*	Größe des Ausgabefelds ist *n* (3000)

Die Verwendung einer oder mehrerer Größen zieht automatisch die Option *-v* nach sich, wenn die Option *-n* nicht verwendet wird.

Der Regelteil der *Datei* beginnt mit dem Begrenzungssymbol %%. Sie können im Regelteil lokale Variablen für *yylex()* vereinbaren. Alle Zeilen im Regelteil, die mit einem Leerzeichen oder Tabulator beginnen und vor der ersten Regel stehen, werden an den Anfang der Funktion *yylex()* kopiert, direkt hinter die erste geöffnete Klammer.

Jede Regel besteht aus einem regulären Ausdruck, der ein aufzufindendes Muster beschreibt, und Aktionen, die ausgeführt werden sollen, wenn das Muster gefunden wird. Eingabetext, der keinem aufzufindenden Muster entspricht, wird von *lex* unverändert an die Ausgabedatei weitergegeben.

Ein regulärer Ausdruck besteht aus Textzeichen mit oder ohne zusätzliche Operatoren.

Folgende Operatoren können bei *lex* verwendet werden:

`\x`	x
`"xy"`	xy, auch wenn x und/oder y lex-Operatoren sind (außer \)
`[xy]`	x oder y
`[x-z]`	x, y oder z
`[^x]`	jedes Zeichen außer x
`.`	jedes Zeichen außer Neue-Zeile-Zeichen
`^x`	x am Zeilenanfang
`<y>x`	x wenn lex sich im Startzustand y befindet
`x$`	x am Ende einer Zeile
`x?`	x einmal oder keinmal
`x*`	leere Zeichenkette oder mehrfaches Vorkommen von x
`x+`	ein- oder mehrfaches Vorkommen von x
`x{m,n}`	m bis n Vorkommen von x
`xx|yy`	xx oder yy
`x |`	die Aktion von x ist auch die Aktion für die nächste Regel
`(x)`	x
`x/y`	x wenn y folgt
`{xx}`	Ersetzung für xx aus dem Definitionsteil

Im Aktionsteil einer Regel können spezielle Aufgaben durchgeführt werden. Folgende Makros werden dafür von *lex* zur Verfügung gestellt:

input()	ein weiteres Zeichen wird vom Eingabestrom gelesen
unput()	ein Zeichen wird für einen späteren Lesevorgang zurückgestellt
output()	ein Zeichen wird in den Ausgabestrom geschrieben

Sie können diese Makros umdefinieren, wenn Sie die Ein-/Ausgabe selbst steuern möchten. Achten Sie dabei aber auf Konsistenz.

Abgesehen vom Abspeichern gefundener Muster in *yytext[]* gibt es weitere Möglichkeiten, mit *lex*-Funktionen die gefundenen Textmuster zu bearbeiten:

yymore()	Neu erkannte Zeichen werden an die bereits in *yytext[]* befindlichen angehängt (normalerweise wird *yytext[]* mit den nächsten gefundenen Zeichen überschrieben).
yyless(*n*)	Nur die ersten *n* Zeichen in *yytext[]* werden berücksichtigt.

REJECT Zeichenketten, die sich überlappen oder die zum Teil in einer anderen Zeichenkette enthalten sind, werden verarbeitet. *REJECT* springt direkt zur nächsten Regel, ohne den Inhalt von *yytext[]* zu ändern.

Hinweis Wird mit *c89* [5] ein *lex*-Programm gebunden, muss als Bibliotheksparameter *-ll* angegeben werden.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *lex*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Bestimmt in regulären Ausdrücken die Bedeutung von Zeichenbereichen, Äquivalenzklassen und Zeicheneinheiten.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *yacc*

ln Verweis auf eine Datei eintragen (link files)

ln erzeugt Verweise auf vorhandene Dateien oder Dateiverzeichnisse. Sie können dann auf diese Dateien bzw. Dateiverzeichnisse mit verschiedenen Namen bzw. Pfadnamen zugreifen (siehe *Arbeitsweise*).

Um mit *ln* einen Verweis zu erzeugen, benötigen Sie Schreibrecht für das Dateiverzeichnis, in das der Verweis eingetragen werden soll.

Es gibt zwei verschiedene Verweisarten:

– Der einfache Verweis (hard link):
Wenn Sie einen einfachen Verweis auf eine Datei erzeugen, dann ist diese Datei als Eintrag mehrmals in einem Dateiverzeichnis bzw. einmal oder mehrmals in verschiedenen Dateiverzeichnissen vorhanden; physikalisch existiert diese Datei aber nur einmal. Der Indexeintrag jeder Datei enthält einen Verweiszähler; erst wenn alle Verweise auf eine Datei gelöscht sind, wird die Datei selbst gelöscht.
Mit einem einfachen Verweis können Sie nicht auf Dateiverzeichnisse oder Dateien in anderen Dateisystemen verweisen.

– Der symbolische Verweis (symbolic link):
Ein symbolischer Verweis ist eine Datei, die einen Pfadnamen enthält. Wenn die Shell auf einen Dateinamen stößt, der zu einem symbolischen Verweis gehört, ersetzt sie diesen Namen durch den angegebenen Pfadnamen. Sie greifen also nicht auf den symbolischen Verweis zu, sondern auf die Datei, zu der der Pfadname führt.
Symbolische Verweise können Sie auf beliebige Dateien oder Dateiverzeichnisse über Dateisystemgrenzen hinaus einrichten.

Syntax

Format 1: ln[␣option]␣datei␣verweis

Format 2: ln[␣option]␣datei␣...␣dateiverzeichnis

Format 3: ln␣-s␣name␣verweis

Format 4: ln␣-s␣name␣...␣dateiverzeichnis

Format 1 **Einfachen Verweis eintragen**

ln[␣option]␣datei␣verweis

ln erzeugt den Verweis *verweis* auf die Datei *datei*. Die Datei ist dann sowohl unter dem Namen *datei* als auch unter dem Namen *verweis* ansprechbar.

Keine Option angegeben
Existiert bereits eine Datei mit dem Namen, den der zu erzeugende Verweis erhalten soll, und haben Sie für diese Datei kein Schreibrecht, dann gibt *ln* die Zugriffsrechte aus und fragt, ob es den Verweis erzeugen soll. *ln* erzeugt den Verweis nur dann, wenn Sie die Frage bejahen (siehe *verweis* und *Beispiel 2*).

Achtung!
Ist die Standard-Eingabe keine Datensichtstation, dann unterbleibt die Frage und der Verweis wird nicht erzeugt.

option

-f (f - force) Existiert bereits eine Datei mit dem Namen *verweis*, dann erzeugt *ln* den Verweis ohne Rückfrage, egal, ob Sie für die Datei Schreibrecht besitzen oder nicht.

-n Existiert bereits eine Datei mit dem Namen *verweis*, dann wird der Inhalt der Datei nicht überschrieben.
Die Option *-f* überlagert die Option *-n*.

datei
Name der Datei, für die Sie einen Verweis erzeugen möchten. Die Datei muss vorhanden sein. Für *datei* dürfen Sie kein Dateiverzeichnis angeben.

verweis
Name des Verweises, den Sie für *datei* eintragen möchten. Sie können für *verweis* einen einfachen Dateinamen oder einen absoluten oder relativen Pfadnamen angeben.

einfacher Dateiname:
ln trägt den einfachen Dateinamen *verweis* in das aktuelle Dateiverzeichnis ein.

absoluter oder relativer Pfadname *präfix/verweis*:
ln trägt den einfachen Dateinamen *verweis* in das Dateiverzeichnis *präfix* ein.

Existiert bereits eine Datei mit dem Namen *verweis* und haben Sie für diese Datei Schreibrecht, dann erzeugt *ln* den Verweis ohne Rückfrage. Das heißt, mit dem Namen *verweis* wird dann nicht mehr die ursprüngliche Datei angesprochen, sondern die Datei *datei*. War *verweis* der einzige Verweis auf die ursprüngliche Datei, ist nach der Ausführung von *ln* der zugehörige Dateiinhalt gelöscht.
Haben Sie für die Datei *verweis* kein Schreibrecht, dann fragt *ln*, ob es den Verweis erzeugen soll (siehe Option *-f* und *Beispiel 2*).

Wenn das übergeordnete Dateiverzeichnis von *verweis* schreibbar ist, aber das t-Bit (sticky-Bit) gesetzt hat, muss eine der folgenden Bedingungen erfüllt sein, um *verweis* anzulegen:
- die Datei muss dem Benutzer gehören
- das Dateiverzeichnis muss dem Benutzer gehören
- der Benutzer muss Schreibberechtigung für die Datei haben
- der Benutzer muss ein privilegierter Benutzer sein

ln in dieser Form legt keine Verweise über Dateisystemgrenzen hinweg an.
Verweise über Dateisystemgrenzen hinweg können Sie mit der Option *-s* einrichten (siehe *Format 3* und *Format 4*).

Format 2 **Einfache Verweise unter gleichem Namen in ein anderes Dateiverzeichnis eintragen**

ln[␣option]␣datei␣...␣dateiverzeichnis

ln trägt für jede angegebene Datei *datei* einen Verweis in das angegebene Dateiverzeichnis ein. Die Datei ist dann in zwei verschiedenen Dateiverzeichnissen unter dem gleichen einfachen Dateinamen ansprechbar.

Keine Option angegeben
Existiert bereits eine Datei mit dem Namen, den der zu erzeugende Verweis erhalten soll, und haben Sie für diese Datei kein Schreibrecht, dann gibt *ln* die Zugriffsrechte aus und fragt, ob es den Verweis erzeugen soll. *ln* erzeugt den Verweis nur dann, wenn Sie die Frage bejahen (siehe *verweis* und *Beispiel 2*).

Achtung!
Ist die Standard-Eingabe keine Datensichtstation, dann unterbleibt die Frage und der Verweis wird nicht erzeugt.

option

-f (f - force) Existiert in *dateiverzeichnis* bereits eine Datei mit dem Namen *datei*, dann erzeugt *ln* den Verweis ohne Rückfrage, egal, ob Sie für die Datei Schreibrecht besitzen oder nicht.

-n Existiert in *dateiverzeichnis* bereits eine Datei mit dem Namen *datei*, dann wird der Inhalt der Datei nicht überschrieben.
Die Option *-f* überlagert die Option *-n*.

datei
Name der Datei, für die Sie einen Verweis erzeugen möchten. Die Datei muss vorhanden sein. Ein Dateiverzeichnis dürfen Sie nicht angeben. Pro Aufruf können Sie mehrere Dateinamen angeben. Sie können für *datei* einen einfachen Dateinamen oder einen absoluten oder relativen Pfadnamen angeben.

einfacher Dateiname:
ln trägt in *dateiverzeichnis* den Verweis *datei* ein.

absoluter oder relativer Pfadname *präfix/name*:
ln trägt in *dateiverzeichnis* den einfachen Dateinamen *name* ein.

Existiert in *dateiverzeichnis* bereits eine Datei, die den gleichen einfachen Dateinamen wie *datei* hat, und haben Sie für diese Datei Schreibrecht, dann erzeugt *ln* den Verweis ohne Rückfrage. Das heißt, mit dem Verweis wird dann nicht mehr die ursprüngliche Datei angesprochen, sondern die Datei *datei*. War der Verweis der einzige Verweis auf die ursprüngliche Datei, ist nach der Ausführung von *ln* der zugehörige Dateiinhalt gelöscht.

Haben Sie für die Datei, die bereits in *dateiverzeichnis* existiert, kein Schreibrecht, dann fragt *ln*, ob es den Verweis erzeugen soll (siehe Option *-f* und *Beispiel 2*).

dateiverzeichnis
: Name des Dateiverzeichnisses, in das Sie einen Verweis eintragen möchten. Das Dateiverzeichnis muss existieren.

ln in dieser Form legt keine Verweise über Dateisystemgrenzen hinweg an.

Format 3 **Symbolischen Verweis eintragen**

ln␣-s␣name␣verweis

ln -s erzeugt einen symbolischen Verweis *verweis* auf *name*, wobei *name* eine Datei oder ein Dateiverzeichnis sein kann. Die Besonderheit von *ln -s* liegt darin, dass über verschiedene Dateisysteme hinweg verwiesen werden kann, was bei einfachen Verweisen nicht möglich ist.

name
: Name der Datei oder des Dateiverzeichnisses, für das Sie einen symbolischen Verweis erzeugen möchten. Für *name* kann ein beliebiger Pfadname angegeben werden, der nicht existieren muss. *name* kann in einem anderen Dateisystem als *verweis* liegen.

verweis
: Name des symbolischen Verweises, den Sie für *name* einrichten möchten. Sie können für *verweis* einen einfachen Dateinamen oder einen absoluten oder relativen Pfadnamen angeben.

  einfacher Dateiname:
  ln trägt den einfachen Dateinamen *verweis* als symbolischen Verweis in das aktuelle Dateiverzeichnis ein.

  absoluter oder relativer Pfadname *präfix/verweis*:
  ln trägt den einfachen Dateinamen *verweis* in das Dateiverzeichnis *präfix* ein.

  Existiert bereits eine Datei mit dem Namen *verweis*, erfolgt eine Fehlermeldung (siehe *Fehler*); die bereits existierende Datei wird nicht überschrieben.

Format 4 **Symbolische Verweise unter gleichem Namen in anderes Dateiverzeichnis eintragen**

ln␣-s␣name␣...␣dateiverzeichnis

ln trägt für jede angegebene Datei oder jedes angegebene Dateiverzeichnis *name* einen symbolischen Verweis in das angegebene Dateiverzeichnis *dateiverzeichnis* ein.
ln legt Verweise über Dateisystemgrenzen hinweg an.

name
: Name der Datei oder des Dateiverzeichnisses, für das Sie einen symbolischen Verweis erzeugen möchten. Pro Aufruf können Sie mehrere Namen angeben.
  Sie können für *name* einen absoluten oder relativen Pfadnamen angeben.

absoluter oder relativer Pfadname *präfix/name*:
ln trägt in *dateiverzeichnis* den einfachen Dateinamen *name* als Verweis auf *präfix/name* ein.

Existiert in *dateiverzeichnis* bereits eine Datei mit dem gleichen einfachen Dateinamen wie *name*, erfolgt eine Fehlermeldung (siehe *Fehler*); die bereits existierende Datei wird nicht überschrieben.

dateiverzeichnis
Name des Dateiverzeichnisses, in das die symbolischen Verweise eingetragen werden sollen. Das Dateiverzeichnis muss existieren.

Arbeitsweise

- Einfache Verweise

  Wenn *ln* einen Verweis auf eine Datei erzeugt, dann wird der zum Verweis gehörige einfache Dateiname in das entsprechende Dateiverzeichnis eingetragen. Der Eintrag erhält die gleiche Index-Nummer wie der alte Dateiname. Für beide Dateinamen ist also der gleiche Index-Eintrag gültig (und damit sind auch Zugriffsrechte, Eigentümer, Daten usw. identisch). Die Datei, auf die sich die Dateinamen beziehen, ist physikalisch nur einmal vorhanden. Sie können nun ein und dieselbe Datei unter verschiedenen Namen bzw. Pfadnamen ansprechen (siehe *Beispiel 1*).

  An der Index-Nummer können Sie erkennen, ob zwei Dateinamen auf dieselbe Datei verweisen (siehe *ls -i*). Der Verweiszähler gibt an, wie viele Verweise auf die Datei bestehen, d.h. wie viele Dateiverzeichnis-Einträge für die Datei existieren (siehe *ls -l*).

  Mit *rm* löschen Sie den Eintrag im Dateiverzeichnis. Bestanden auf die Datei mehrere Verweise, so ist sie unter den nicht gelöschten Namen weiterhin ansprechbar. Erst mit dem letzten Verweis löscht *rm* die Datei.

- Symbolische Verweise

  Ein symbolischer Verweis ist eine Datei, die einen Pfadnamen enthält. Diesen Pfadnamen können Sie mit dem Kommando *ls -l* lesen. Wenn die Shell einen Dateinamen findet, der zu einem symbolischen Verweis gehört, ersetzt sie diesen Namen durch den angegebenen Pfadnamen. Ein Pfadname wird also auf einen anderen abgebildet. Hier gibt es keinen Verweiszählmechanismus; mit dem symbolischen Verweis wird die Datei gelöscht, die den Pfadnamen enthält.
  Bei einem symbolischen Verweis enthält der Indexeintrag der Datei, auf die verwiesen wird, keine Information über diese Tatsache. Der Verweiszähler zählt nur die einfachen Verweise. Wenn Sie also das Ziel eines symbolischen Verweises löschen, existiert der symbolische Verweis zwar immer noch, aber er verweist auf eine Datei ohne Inhalt und Indexeintrag.
  Symbolische Verweise sind nicht an Dateisystemgrenzen gebunden. Der Pfadname kann der Name einer Datei oder eines Dateiverzeichnisses sein.

Symbolische Verweise können mit dem *ls*-Kommando sichtbar gemacht werden:

- Mit dem Kommando *ls -l* stellen Sie fest, welche Dateien im angegebenen Dateiverzeichnis symbolische Verweise sind. Außerdem können Sie mit diesem Kommando den Inhalt dieser Dateien lesen. Das ist der Pfadname, der auf das Zeichen '->' folgt.
- Mit dem Kommando *ls -L* erhalten Sie Informationen über die Datei, auf die der symbolische Verweis zeigt.

Operationen in symbolisch verwiesenen Dateiverzeichnissen, die '..' beinhalten, wie z.B. *'cd ..'*, referenzieren das Original-Dateiverzeichnis!

Fehler

`ln: Cannot link` *datei* `to` *verweis*`: Permission denied`
ln kann den Verweis *verweis* nicht anlegen, da Sie für das Dateiverzeichnis, in das der Verweis eingetragen werden soll, kein Schreibrecht besitzen.

`ln: cannot access` *datei*`: no such file or directory`
ln kann auf *datei* nicht zugreifen, da sie nicht existiert oder da Sie für ein Dateiverzeichnis, das im Pfadnamen *datei* vorkommt, kein Ausführrecht (x-Bit) besitzen.

`ln:` *dvz* `is a directory`
Anstelle einer Datei haben Sie das Dateiverzeichnis *dvz* angegeben. *ln* erzeugt jedoch nur Verweise auf einfache Dateien, Gerätedateien und FIFO-Dateien, nicht aber auf Dateiverzeichnisse.

`ln: are on different file systems`
Sie wollten einen Verweis erzeugen und in einem anderen Dateisystem eintragen, als sich die betreffende Datei befindet. *ln* erzeugt jedoch keine Verweise über Dateisystemgrenzen hinweg. Versuchen Sie es mit der *-s* Option.

`ln: cannot create`
`ln: File exists`
Sie haben versucht, über die Option -s in einem Dateiverzeichnis einen Verweis *datei* anzulegen, wobei *datei* in dem betreffenden Dateiverzeichnis bereits existiert.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *ln*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Der Benutzer Max besitzt die Dateien *bolte* und *laempel*; sie stehen im Dateiverzeichnis */HOME/MAX*. Nun soll auch Moritz mit diesen Dateien arbeiten. Damit Moritz weiterhin in seinem eigenen Dateiverzeichnis */HOME/MORITZ* arbeiten kann und dort nicht immer die langen Pfadnamen eingeben muss, legt Moritz im Dateiverzeichnis */HOME/MORITZ* Verweise auf diese Dateien an. Voraussetzung dafür ist, dass Moritz
– für sein Dateiverzeichnis */HOME/MORITZ* Schreibrecht hat
– für das Dateiverzeichnis */HOME/MAX* Ausführrecht (x-Bit) hat.

Um die Dateien lesen und verändern zu können, benötigt Moritz außerdem Lese- und Schreibrecht für die Dateien. Im Beispiel ist das erfüllt, wenn Moritz der Gruppe *proj* angehört (siehe unten).
Am schnellsten legt Moritz die Verweise mit einem *ln*-Aufruf gemäß Format 2 an:

```
$ ln /HOME/MAX/bolte /HOME/MAX/laempel /HOME/MORITZ
```

Dasselbe leisten die folgenden zwei *ln*-Aufrufe in Format 1:

```
$ ln /HOME/MAX/bolte /HOME/MORITZ/bolte
$ ln /HOME/MAX/laempel /HOME/MORITZ/laempel
```

Wie die folgenden *ls*-Aufrufe zeigen, hat *ln* die Dateinamen *bolte* und *laempel* in das Dateiverzeichnis */HOME/MORITZ* eingetragen. Der Verweiszähler für die Dateien wurde auf 2 erhöht (*ls -l*). Den *bolte*- bzw. *laempel*-Dateinamen wurde jeweils die gleiche Index-Nummer zugeordnet (*ls -i*).

```
$ ls -l /HOME/MORITZ
total 6
-rw-rw----   2 MAX      proj          34   Mar 09 15:08 bolte
-rw-rw----   2 MAX      proj        1217   Mar 09 18:59 laempel

$ ls -i /HOME/MAX/bolte /HOME/MORITZ/bolte
16435 /HOME/MAX/bolte    16435 /HOME/MORITZ/bolte

$ ls -i /HOME/MAX/laempel /HOME/MORITZ/laempel
24766 /HOME/MAX/laempel  24766 /HOME/MORITZ/laempel
```

Der obige Aufruf *ls -l* zeigt auch, dass Max Eigentümer der Dateien bleibt. Max kann Moritz jederzeit die Erlaubnis, mit den Dateien zu arbeiten, entziehen, z.B. mit

```
$ chmod 600 bolte laempel
```

Will Moritz für die beiden Dateien nicht gleichnamige, sondern anderslautende Verweise anlegen, muss er *ln* wie folgt im Format 1 aufrufen:

```
$ ln /HOME/MAX/bolte /HOME/MORITZ/spitz
$ ln /HOME/MAX/laempel /HOME/MORITZ/pfeife
```

Beispiel 2 Das folgende Beispiel zeigt, was bei einem *ln*-Aufruf *ln datei verweis* geschieht, wenn es bereits eine Datei namens *verweis* gibt.

Im aktuellen Dateiverzeichnis sind drei Dateinamen eingetragen: *brief*, *liste* und *text*. *brief* verweist auf die Datei1, *liste* und *text* verweisen beide auf die Datei2. Für die Datei2 haben Sie kein Schreibrecht:

```
$ ls -l
total 3
-rw-r--r--   1  BERTA    other     57  Jul 17 14:29 brief
-r--r--r--   2  EMIL     other    103  Jul 16 15:30 liste
-r--r--r--   2  EMIL     other    103  Jul 16 15:30 text
```

Geben Sie nun ein:

```
$ ln brief liste
ln: liste: 444 mode (y/n) ?n
```

Da Sie mit *n* geantwortet haben, erzeugt *ln* den Verweis nicht.

Wenn Sie für die Datei2 Schreibrecht haben, erzeugt *ln* den Verweis ohne Rückfrage:

```
$ ls -l
total 3
-rw-r--r--   1  BERTA    other     57  Jul 17 14:29 brief
-rw-rw-rw-   2  EMIL     other    103  Jul 16 15:30 liste
-rw-rw-rw-   2  EMIL     other    103  Jul 16 15:30 text
$ ln brief liste
$ ls -l
total 3
-rw-r--r--   2  BERTA    other     57  Jul 17 14:29 brief
-rw-r--r--   2  BERTA    other     57  Jul 17 14:29 liste
-rw-rw-rw-   1  EMIL     other    103  Jul 16 15:30 text
```

Da *ln* den Verweis *liste* für die Datei *brief* (Datei1) erzeugt hat, verweist *liste* nun nicht mehr auf die Datei2, sondern auf die Datei1. *text* ist nun der einzige Verweis auf die Datei2.

```
$ ln brief text
$ ls -l
total 3
-rw-r--r--   3  BERTA    other     57  Jul 17 14:29 brief
-rw-r--r--   3  BERTA    other     57  Jul 17 14:29 liste
-rw-r--r--   3  BERTA    other     57  Jul 17 14:29 text
```

Nun verweist auch *text* auf die Datei1. Damit ist die Datei2 gelöscht.

Beispiel 3 Der Benutzer Norbert will von seinem Dateiverzeichnis */HOME/NORBERT* möglichst einfach auf das Dateiverzeichnis */HOME2/ANDREA* seiner Kollegin Andrea zugreifen. Er erstellt deshalb einen symbolischen Verweis durch einen Aufruf von *ln* im Format 3.

```
$ ln -s /HOME2/ANDREA /HOME/NORBERT/and
```

Der Benutzer Norbert kann nun von seinem Dateiverzeichnis */HOME/NORBERT* aus über den symbolischen Verweis *and* direkt auf Andreas Dateiverzeichnis zugreifen, obwohl sich dieses in einem anderen Dateisystem befindet:

```
$ ls -lL and
total 16
drwxr-xr-x   2 ANDREA   usrother    2560 Feb 27 13:20 PASCAL
drwxr-xr-x   2 ANDREA   usrother    2048 Mar  5 17:32 KURSE
drwxr-xr-x   2 ANDREA   usrother     512 Mar  5 12:07 BRIEFE
drwxr-xr-x   2 ANDREA   usrother     512 Feb 26 10:05 bin
-rw-r-xr-x   1 ANDREA   usrother     148 Feb 20 09:28 test
```

Beispiel 4 Der Benutzer Norbert hat in seinem Dateiverzeichnis */HOME/NORBERT* die Dateiverzeichnisse *briefe89*, *briefe90* und *briefe91* angelegt:

```
$ ls /HOME/NORBERT
briefe89  briefe90  briefe91
```

Er will nun von seinem Dateiverzeichnis */HOME/NORBERT/briefe91* ohne Umwege auf die beiden anderen Dateiverzeichnisse zugreifen. Hierzu legt er zwei symbolische Verweise durch einen Aufruf von *ln* im Format 4 an:

```
$ cd /HOME/NORBERT
$ ln -s /HOME/NORBERT/briefe89 /HOME/NORBERT/briefe90 briefe91
$ cd briefe91
$ ls -og briefe??
lrwxrwxrwx   1     25 Mar  6 12:42 briefe89 -> /HOME/NORBERT/briefe89
lrwxrwxrwx   1     25 Mar  6 12:42 briefe90 -> /HOME/NORBERT/briefe90
$ ls briefe90
angebot     brief.huber  brief.maier  text
```

Siehe auch *chmod, cp, ls, mv, rm*
link(), readlink(), stat(), symlink() [4]

locale Informationen über die internationale Umgebung abrufen (get locale-specific information)

Das Kommando *locale* schreibt Informationen über die aktuelle internationale Umgebung oder über sonstige öffentliche internationale Umgebungen auf die Standard-Ausgabe. Im Folgenden wird eine öffentliche Umgebung als eine von der Implementierung zur Verfügung gestellte Umgebung angesehen, auf die die Anwendung zugreifen kann.

Wird *locale* ohne Argumente aufgerufen, fasst es die aktuelle internationale Umgebung für jede Umgebungskategorie so zusammen, wie durch die Einstellungen der Umgebungsvariablen festgelegt ist.

Wird das Kommando mit Operanden aufgerufen, gibt es die Werte, die den Schlüsselwörten in den Umgebungskategorien zugeordnet wurden, folgendermaßen aus:

– Ist ein Schlüsselwortname angegeben, wird dieses Schlüsselwort sowie die Kategorie, die das jeweilige Schlüsselwort enthält, ausgegeben.

– Ist ein Kategoriename angegeben, werden diese Kategorie sowie alle in ihr enthaltenen Schlüsselwörter ausgegeben.

Syntax

Format 1: locale[␣**-a**|␣**-m**]

Format 2: locale[␣**-ck**]␣name...

optionen

-a Gibt Informationen über alle verfügbaren öffentlichen internationalen Umgebungen aus. Die verfügbaren internationalen Umgebungen schließen POSIX ein.

-c Gibt die Namen der ausgewählten Umgebungskategorien aus (siehe Abschnitt „Standard-Ausgabe (stdout)“ auf Seite 494).
Die Option *-c* verbessert die Lesbarkeit, wenn mehrere Kategorien ausgewählt sind (beispielsweise über mehrere Schlüsselwortnamen oder über einen Kategorienamen). Diese Option ist sowohl mit als auch ohne Angabe der Option *-k* gültig.

-k Gibt die Namen und Werte der ausgewählten Schlüsselwörter aus.

-m Gibt die Namen der Zeichentabellen aus.

name
Der Name einer Umgebungskategorie (siehe *localedef)*, der Name eines Schlüsselworts in einer Umgebungskategorie oder der reservierte Name *charmap*. Die angegebene Kategorie oder das angegebene Schlüsselwort wird für die Ausgabe ausgewählt. Kategorie- und Schlüsselwortnamen können als Operanden *name* in beliebiger Reihenfolge angegeben werden.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *locale*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Datei **Standard-Ausgabe (stdout)**

Wenn *locale* ohne Optionen oder Operanden aufgerufen wird, werden die Namen und Werte der Umgebungsvariablen *LANG* und *LC_** auf die Standard-Ausgabe geschrieben. Hierbei wird für jede Variable eine Zeile verwendet, *LANG* wird zuerst ausgegeben. Nur Variablen, die in der Umgebung gesetzt sind und von *LC_ALL* nicht überschrieben werden, werden im folgenden Format ausgegeben:

```
"%s=%s\n", variable_name, value
```

Die Namen der Variablen *LC_**, die nicht in der Umgebung gesetzt sind oder von der Variablen *LC_ALL* überschrieben werden, werden im folgenden Format ausgegeben:

```
"%s=\"%s\"\n", variable_name, implied_value
```

implied_value ist der Name der internationalen Umgebung, die von der Implementierung auf der Grundlage der Werte in *LANG* und *LC_ALL* für diese Kategorie ausgewählt wurde.

Die Werte *value* und *implied_value* werden in Anführungszeichen gesetzt, damit sie zu einem späteren Zeitpunkt wieder als Eingabe verwendet werden können. *value* wird dabei nicht in doppelte Anführungszeichen gesetzt (zur Unterscheidung von *implied_value*, der immer in doppelte Anführungszeichen gesetzt wird).

Die Variable *LC_ALL* wird als letzte ausgegeben. Hierbei wird das erste oben aufgeführte Format verwendet. Ist diese Variable nicht gesetzt, wird sie folgendermaßen ausgegeben:

```
"LC_ALL=\n"
```

Bei Angabe von Argumenten gilt Folgendes:

1. Wenn die Option *-a* angegeben ist, werden die Namen aller öffentlichen internationalen Umgebungen im folgenden Format ausgegeben:
   ```
   "%s\n", locale_name
   ```
2. Wenn die Option *-c* angegeben ist, werden die Namen aller ausgewählten Kategorien im folgenden Format ausgegeben:
   ```
   "%s\n", category_name
   ```

   Wenn auch Schlüsselwörter ausgewählt wurden, werden diese unmittelbar nach der zugehörigen Kategorie ausgegeben.

   Ist die Option *-c* nicht angegeben, werden die Kategorienamen nicht ausgegeben, sondern nur die Schlüsselwörter.
3. Wenn die Option *-k* angegeben ist, werden die Namen und Werte der ausgewählten Schlüsselwörter ausgegeben. Ist der Wert nicht numerisch, wird er im folgenden Format ausgegeben:
   ```
   "%s=\"%s=\"\n", keyword_name, keyword_value
   ```

   Bei dem Schlüsselwort *charmap* wird der Name der Zeichentabelle ausgegeben, falls über die Option *localedef -f* bei der Erstellung der internationalen Umgebung eine Zeichentabelle angegeben wurde. Hierbei wird das Wort *charmap* als *keyword_name* verwendet.

   Numerische Werte werden in einem der folgenden Formate ausgegeben:
   ```
   "%s=%d\n", keyword_name, keyword_value
   "%s=%c%o\n", keyword_name, escape_character, keyword_value
   "%s=%cx%x\n", keyword_name, escape_character, keyword_value
   ```

   Hierbei ist *escape_character* das vom Schlüsselwort *escape_char* in der aktuellen internationalen Umgebung definierte Zeichen (siehe XBD-Spezifikation, Abschnitt 5.3, Locale Definition [15]). Standardmäßig wird der Backslash \ als *escape_character* verwendet.

   Zusammengesetzte Schlüsselwortwerte (Listeneinträge) werden in der Ausgabe durch Semikolons getrennt. Enthalten Schlüsselwortwerte Semikolons, doppelte Anführungszeichen, Gegenschrägstriche und/oder Steuerzeichen, werden diese Zeichen durch Voranstellen des Escape-Zeichens entwertet.
4. Ist die Option *-k* nicht angegeben, werden die Werte aller ausgewählten Schlüsselwörter im folgenden Format ausgegeben:
   ```
   "%s\n", keyword_value
   ```

   Bei dem Schlüsselwort *charmap* wird der Name der Zeichentabelle ausgegeben (falls vorhanden).

5. Wenn die Option *-m* angegeben ist, wird eine Liste aller verfügbaren Zeichentabellen im folgenden Format ausgegeben:
   `"%s\n", charmap`

   Die Ausgabe kann als Argument für die Option *-f* von *localedef* verwendet werden.

Endestatus

0 Alle angeforderten Informationen wurden gefunden und ausgegeben.

>0 Ein Fehler ist aufgetreten.

Beispiel In den Beispielen wird von der folgenden internationalen Umgebung ausgegangen:

```
LANG=locale_x
LC_COLLATE=locale_Y
```

Das Kommando *locale* hätte folgende Ausgabe:

```
LANG=locale_x
LC_CTYPE="locale_x"
LC_COLLATE=locale_Y
LC_TIME="locale_x"
LC_NUMERIC="locale_x"
LC_MONETARY="locale_x"
LC_MESSAGES="locale_x"
LC_ALL=
```

Das Kommando
`LC_ALL=POSIX locale -ck decimal_point`
würde folgende Ausgabe erzeugen:

```
LC_NUMERIC
decimal_point="."
```

Das folgende Kommando zeigt eine Anwendung von *locale*, mit der festgestellt werden kann, ob eine vom Benutzer eingegebene Antwort eine Ja-Antwort (bestätigend) ist:

```
if printf "%s\n" "$response" | grep -Eq "$(locale yesexpr)"
then
      affirmative processing goes here
else
      non-affirmative processing goes here
fi
```

Siehe auch *localedef*

localedef Internationale Umgebung definieren (define local environment)

Das Kommando *localedef* konvertiert Quelldefinitionen für Kategorien der internationalen Umgebungen in ein Format, das von den Funktionen und Kommandos verwendet werden kann, deren Ausführungsverhalten durch die Einstellungen der Variablen für die internationale Umgebung bestimmt wird. Dieses Format ist in der XBD-Spezifikation, Kapitel 5, Locale [15] definiert.

Jeder Benutzer eines POSIX-Systems hat die Möglichkeit, lokal neue internationale Umgebungen zu erstellen. Jedoch können nur Benutzer mit entsprechender Berechtigung (normalerweise Systemadministratoren) diese internationalen Umgebungen auf dem System installieren, so dass sie von Funktionen, Anwendungen und Kommandos verwendet werden können. Dies geschieht durch Kopieren der lokal erstellten Dateien nach */usr/lib/locale/name* und */usr/lib/charmap/charmap*.

localedef liest Quelldefinitionen für eine oder mehrere Kategorie(n) der internationalen Umgebungen, die zu derselben internationalen Umgebung gehören, aus der mit der Option *-i* angegebenen Datei (falls angegeben) oder von der Standard-Eingabe.

Der Operand *name* kennzeichnet die Zielumgebung. Das Kommando unterstützt die Erstellung internationaler Umgebungen, die „öffentlich“ sind (d.h. allgemein zugänglich), sowie internationaler Umgebungen, die privat sind (d.h. mit eingeschränkten Zugriffsrechten). Auf POSIX-Systemen dürfen allgemein zugängliche, internationale Umgebungen nur von Benutzern mit den entsprechenden Berechtigungen erstellt bzw. geändert werden.

Jede Quelldefinition einer Kategorie wird durch den Namen der zugehörigen Umgebungsvariablen gekennzeichnet und mit einer Anweisung `END` *category-name* abgeschlossen. Die folgenden Kategorien werden unterstützt:

LC_CTYPE Definiert die Klassifizierung von Zeichen und die Umsetzung von Klein-/Großschreibung.

LC_COLLATE Definiert die Sortierregeln.

LC_MONETARY
Definiert das Format und die Symbole, die für die Formatierung von Währungsangaben verwendet werden.

LC_NUMERIC Definiert das Dezimalzeichen, das Tausender-Trennzeichen und die Zifferngruppierung für das Editieren von numerischen Angaben (keine Währungsangaben).

LC_TIME Definiert Format und Inhalt von Datums- und Uhrzeitangaben.

LC_MESSAGES
Definiert Format und Werte für Ja-/Nein-Antworten (Bestätigungen und Ablehnungen).

Syntax **localedef**[␣**-c**][␣**-f**␣charmap][␣**-i**␣sourcefile]␣name

optionen

-c Erstellt eine Ausgabedatei, auch wenn Warnmeldungen ausgegeben wurden.

-f␣charmap
Gibt den Pfadnamen einer Datei an, die eine Zuordnung von den symbolischen "character symbols" und "collating element symbols" zu tatsächlichen Zeichencodierungen enthält (Zeichentabelle). Das Format von *charmap* ist in X/Open CAE Specification, System Interface Definitions [15] beschrieben. Diese Option muss angegeben werden, wenn symbolische Namen verwendet werden, die nicht mittels Schlüsselwort collating-symbol definiert wurden.

-f nicht angegeben:
Die in der Datei */usr/lib/charmap/posix* definierte Zeichenzuordnung (ISO 8859-1) wird verwendet.

-i␣sourcefile
Der Pfadname einer Datei, die die Quelldefinitionen enthält.

-i nicht angegeben:
Die Quelldefinitionen werden von der Standard-Eingabe gelesen. Das Format der Eingabedatei ist in X/Open CAE Specification, System Interface Definitions [15] beschrieben.

name
Gibt die internationale Umgebung an. Die Verwendung dieses Namens kann in X/Open CAE Specification, System Interface Definitions [15] nachgelesen werden.
- Wenn *name* einen oder mehrere Schrägstriche enthält, wird *name* als Pfadname interpretiert, in dem die erstellten Definitionen für die internationalen Umgebungen gespeichert werden.
- Wenn *name* keine Schrägstriche enthält, ist die internationale Umgebung privat und wird im aktuellen Verzeichnis generiert.

Da nur ein Name angegeben werden kann, können bei einem Aufruf nur Kategorien verarbeitet werden, die derselben internationalen Umgebung angehören.

Internationale Umgebung

Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *localedef* aus:

LANG Gibt einen Standardwert für Internationalisierungsvariablen an, die nicht gesetzt oder null sind. Ist *LANG* nicht gesetzt oder null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE (Diese Variable hat keine Auswirkungen auf *localedef*; für diese Kategorie wird die internationale Umgebung von POSIX verwendet.)

LC_CTYPE Legt die internationale Umgebung für die Interpretation von Byte-Folgen als Zeichen fest (zum Beispiel Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien). Diese Variable wirkt sich nicht auf die Verarbeitung von Eingabedaten des Kommandos aus. Hierfür wird die internationale Umgebung von POSIX verwendet, und zwar unabhängig vom Wert dieser Variablen.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt die Position der Meldungskataloge für die Verarbeitung von *LC_MESSAGES* fest.

Datei

Standardeingabe

Wenn die Option *-i* nicht angegeben ist, muss die Standardeingabe eine Textdatei sein, die eine oder mehrere Quelldefinitionen für Kategorien für internationale Umgebungen enthält, wie in der XBD-Spezifikation, Abschnitt 5.3, Locale Definition [15], beschrieben. Wenn Zeilen mit Hilfe des Escape-Mechanismus fortgesetzt werden, gibt es keine Längenbeschränkungen für die gesamte Eingabezeile.

Eingabedateien

Die Datei, die die Beschreibung des Zeichensatzes enthält, (Argument charmap bei *-f*) ist in X/Open CAE Specification, System Interface Definitions [15] beschrieben. Wenn eine Quelldefinition einer Kategorie für eine internationale Umgebung die Anweisung *copy* enthält (wie in X/Open CAE Specification, System Interface Definitions [15] beschrieben) und diese Anweisung eine gültige internationale Umgebung angibt, verhält sich *localedef* so, als enthielte die Quelldefinition eine gültige Quelldefinition der Kategorie für die angegebene internationale Umgebung.

Ausgabedateien

Das Format der erstellten Ausgabe entspricht dem Format der Internationalisierungsdatenbank auf POSIX-Systemen:

name (falls Schrägstriche enthalten sind) oder *./name* (ohne Schrägstriche) ist der Pfadname eines Verzeichnisses, das die folgenden neu generierten Dateien enthält:

LC_COLLATE	Datei mit Sortierangaben
LC_CTYPE	Datei mit Angaben zum Zeichentyp (ctype)
LC_MONETARY	Datei mit Währungsangaben
LC_NUMERIC	Datei mit numerischen Angaben
LC_TIME	Datei mit Datums- und Uhrzeitangaben
LC_MESSAGES	Verzeichnis mit Meldungsangaben
LC_MESSAGES/Xopen_info	Datei mit Ja-/Nein-Angaben
loc_info	Datei mit Angaben zu Name und Pfad der Zeichentabelle

Ferner wird eine Kopie der Zeichentabelle in */usr/lib/charmap* gestellt, wenn die Option *-f* verwendet wird.

Endestatus Die folgenden Endewerte werden übergeben:

0 Es sind keine Fehler aufgetreten und die internationalen Umgebungen wurden erfolgreich erstellt.

1 Es wurden zwar Warnungen ausgegeben, die internationalen Umgebungen wurden jedoch erfolgreich erstellt.

2 Die Angabe der internationalen Umgebung hat Grenzen der Implementierung überschritten oder der codierte Zeichensatz bzw. die verwendeten Zeichensätze wurden von der Implementierung nicht unterstützt. Es wurde keine internationale Umgebung erstellt.

4 Es sind Warnungen oder Fehler aufgetreten, und es wurde keine Ausgabe erzeugt.

Fehler

Wird ein Fehler in den Eingabedateien gefunden, wird keine Ausgabedatei erstellt.

Wenn Warnungen auftreten, wird eine Ausgabedatei erstellt, sofern die Option *-c* angegeben wurde. Bei folgenden Bedingungen werden Warnmeldungen ausgegeben:

- Ein in der Datei *charmap* nicht enthaltener symbolischer Name wurde für die Beschreibungen der Kategorien *LC_CTYPE* oder *LC_COLLATE* verwendet (bei anderen Kategorien kommt es hierdurch zu einer Fehlerbedingung).
- Die Anzahl der Operanden für das Schlüsselwort order überschreitet die {COLL_WEIGHTS_MAX}-Grenze.
- Die Quelle enthält optionale, von der Implementierung nicht unterstützte Schlüsselwörter.

Hinweis

Die Definition der Zeichentabelle ist optional und ist nicht in der Definition der internationalen Umgebung enthalten. Hierdurch sind sowohl vollständig selbstdefinierte Quelldateien als auch allgemeine Quellen (die auf mehrere codierte Zeichensätze anwendbar sind) möglich. Zur besseren Portabilität müssen alle Zeichentabellen-Definitionen dieselben symbolischen Namen für portierbare Zeichensätze haben. Wie in X/Open CAE Specification, System Interface Definitions [15] beschrieben, ist es von der jeweiligen Implementierung abhängig, ob Benutzer oder Anwendungen zusätzliche Beschreibungsdateien für Zeichensätze einsetzen können. Daher arbeitet die Option *-f* möglicherweise nur dann korrekt, wenn eine von der Implementierung unterstützte *charmap* angegeben wurde.

Siehe auch

locale
X/Open CAE Specification, System Interface Definitions [15]

logger Meldungen protokollieren (log messages)

logger speichert Meldungen in die Datei */var/adm/logger*, die später vom POSIX-Verwalter ausgewertet werden können. Die Meldungen werden ohne Formatdefinition gespeichert.

Dies ist vor allem für nicht-interaktive Utilities geeignet, die Informationen über einen Fehler ausgeben.

Syntax **logger**␣string

string
ein oder mehrere Stringargumente, dessen Inhalt aneinander gefügt wird.

string erscheint in der Ausgabe (siehe *Beispiel*)

Internationale Umgebung
Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *logger* aus:

LANG Gibt einen Standardwert für Internationalisierungsvariablen an, die nicht gesetzt oder null sind. Ist *LANG* nicht gesetzt oder null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation von Byte-Folgen als Zeichen fest (zum Beispiel Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Diagnose-Meldungen fest, die auf die Standard-Fehlerausgabe geschrieben werden. Dies betrifft Diagnose-Meldungen, die *logger* an den Benutzer oder die Anwendung sendet und nicht Diagnose-Meldungen, die der Benutzer an den POSIX-Verwalter sendet.

NLSPATH Legt die Position der Meldungskataloge für die Verarbeitung von *LC_MESSAGES* fest.

Beispiel Ein Batch-Job, der nicht interaktiv abläuft, versucht vergeblich, eine Konfigurationsdatei zu lesen. Dann würde der POSIX-Verwalter folgende Meldung erhalten:

```
logger myname: unable to read file foo. [timestamp]
```

Siehe auch *mailx, write*

logname Login-Kennung abfragen (return user's login name)

logname schreibt Ihre Login-Benutzerkennung auf die Standard-Ausgabe.

Syntax **logname**

Datei */etc/profile*
Datei, die von jeder Login-Shell ausgewertet wird. Sie dient u.a. zur Einstellung einer Shell-Umgebung, abhängig vom Inhalt der von login übergebenen Umgebungsvariablen LOGNAME (Login-Kennung des Benutzers). Diese Datei erstellt der POSIX-Verwalter.

Internationale Umgebung
Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *logname* aus:

LANG Gibt einen Standardwert für Internationalisierungsvariablen an, die nicht gesetzt oder null sind. Ist *LANG* nicht gesetzt oder null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation von Byte-Folgen als Zeichen fest (zum Beispiel Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Diagnose-Meldungen fest, die auf die Standard-Fehlerausgabe geschrieben werden.

NLSPATH Legt die Position der Meldungskataloge für die Verarbeitung von *LC_MESSAGES* fest.

Beispiel Sie haben sich unter der Benutzerkennung POSUSER1 angemeldet und wollen sich diese nun anzeigen lassen.

```
$ logname
POSUSER1
```

Siehe auch *env, id, who*

logrotate Wechsel der Protokolldateien des syslog-Dämonen (change logging files of the syslog daemon)

Mit *logrotate* können Protokolldateien gewechselt werden. *logrotate* analysiert die Konfigurationsdatei des syslog-Dämonen (standardmäßig */etc/syslog.conf*) und sucht darin alle Protokolldateien, die vom syslog-Dämon verwendet werden. *logrotate* legt Kopien dieser Protokolldateien an, deren Dateiname aus dem Originaldateinamen mit dem Suffix „.0“ gebildet wird. Falls von vorangegangenen Wechseln der Protokolldateien bereits Kopien existieren, wird das Suffix in deren Namen beginnend mit dem numerisch größten jeweils um 1 erhöht. Also wird <logfile>.<n> zu <logfile>.<n+1>, ..., <logfile>.0 zu <logfile>.1 und <logfile> zu <logfile>.0. Maximal vier vorhergehende Prokolldateien (Suffix ".0" bis ".3") werden aufbewahrt. Ältere Potokolldateien werden gelöscht.

Die aktuelle Datei <logfile> wird nicht gelöscht, aber nach dem Kopieren geleert.

Anschließend wird der syslogd-Dämon veranlasst, die offenen Protokolldateien zu schließen und neu zu öffnen.

Beim Beenden und Starten des POSIX-Subsystems wird automatisch das Kommando `logrotate -n 1` aufgerufen.

Weitere Informationen finden Sie im Abschnitt „POSIX-Protokolldateien“ im POSIX-Handbuch „Grundlagen für Anwender und Systemverwalter“ [1].

Syntax

logrotate [**-f** config_file] [**-n** line_num]

-f Mit dieser Option muss die Konfigurationsdatei angegeben werden, wenn für den syslog-Dämonen eine andere Konfigurationsdatei als */etc/syslog.conf* festgelegt wurde, z.B. durch Abändern seiner *rc*-Skripts.

-n Die Ausführung von *logrotate* wird auf Dateien beschränkt, die mindestens die angegebene Anzahl von Zeilen enthält.
(Standardwert: 4 Zeilen).

Datei

/usr/sbin/logrotate

/etc/syslog.conf

lp Dateien ausdrucken (send files to a printer)

Mit *lp* können Sie:

- Dateien am Drucker ausdrucken lassen (Druckauftrag)
- den RSO-Drucker bzw. die Druckergruppe auswählen, auf der ein Druckauftrag ausgeführt wird
- eine Kopfseite ausgeben lassen
- Druckeroptionen für Druckaufträge ändern

lp richtet für jede mit Namen angegebene Datei einen Druckauftrag ein. Geben Sie keine Datei an, erwartet *lp* die Druckdaten von der Standard-Eingabe. Die Dateien werden in der Reihenfolge gedruckt, in der Sie sie aufgelistet haben. *lp* ordnet jedem Druckauftrag eine eindeutige Auftragsnummer zu und gibt diese auf die Standard-Ausgabe aus. Diese Nummer können Sie verwenden, um einen Druckauftrag abzubrechen oder zu löschen oder Informationen über den Druckauftrag ausgeben zu lassen.

Mit *cancel* können Sie laufende Druckaufträge abbrechen oder gestellte, aber noch nicht gestartete Druckaufträge löschen.

Mit *lpstat* können Sie sich Informationen über Druckaufträge ausgeben lassen.

Syntax

lp[␣**-c**][␣**-d**␣dest][␣**-n**␣copies][␣**-s**][␣**-o**␣par] ...[␣**-t**␣title][␣dateil**-**...]

Keine Option angegeben
: Die angegebenen Dateien werden auf einem freien Drucker der Standard-Druckergruppe (standardmäßig ALLE) ausgedruckt.

option

-c (c - copy) Die angegebenen Dateien werden beim Aufruf von *lp* kopiert, und diese Kopien werden ausgedruckt.
Die Option *-c* ist implizit immer eingeschaltet.

-d␣dest
: Der Druckauftrag wird auf den RSO-Drucker oder auf der Druckergruppe *dest* ausgeführt. Wenn zu einer Druckergruppe mehrere Drucker gehören, wird der Auftrag auf dem ersten freien Drucker dieser Druckergruppe ausgeführt.
Für *dest* geben Sie den Namen eines RSO-Druckers oder einer Druckergruppe an.

  Mit *lpstat -a* erhalten Sie die Namen und den Status der zulässigen RSO-Drucker bzw. Druckergruppen.

  -d␣dest nicht angegeben:
  Wenn die Umgebungsvariable *LPDEST* gesetzt ist, wird deren Wert eingesetzt.
  Wenn die Umgebungsvariable *LPDEST* nicht gesetzt ist, wird der Druckauftrag auf der Standard-Druckergruppe ausgeführt. Damit ist entweder die Druckergruppe ALLE oder die dem Benutzer/Terminal zugeordnete Druckergruppe gemeint.

-n␣copies
Mit dieser Option legen Sie fest, wie oft die Dateien ausgedruckt werden. Die größte wirksame Angabe für *copies* ist 99, die kleinste ist 1.

-n␣*copies* nicht angegeben:
Für *copies* wird 1 angenommen.

-o␣par
par bestimmt die Druckparameter *line-spacing, control-char-pos* und *control-mode* oder wählt die Druckparameter für die Ausgabe von PostScript-Dateien aus.
Sollen mehrere Parameter miteinander kombiniert werden, muss für jeden Parameter die Option *-o*␣*parameter* angegeben werden.

Die Parameter können folgende Werte annehmen:

```
line-spacing = 1 | 2 | 3 | *s[td] | *e[bcdic] | *a[sa] | *i[bm]

control-char-pos = *std | <int 1..2040>

control-mode = *pa[ge-mode] | *li[ne-mode] | *lo[gical-mode] | *ap[a] |
               *ph[ysical]
postscript | ps
```

Bei Angabe der Druckparameter *line-spacing, control-char-pos* und *control-mode* werden die angegebenen Werte an die entsprechenden Operanden des BS2000-Kommandos PRINT-DOCUMENT durchgereicht (DOCUMENT-FORMAT), siehe Handbuch „Kommandos“ [11].

Sind in der auszudruckenden Datei Steuerzeichen vorhanden (z.B. \f für den Seitenvorschub), dann werden diese Steuerzeichen nur ausgewertet, wenn die beiden folgenden Bedingungen erfüllt sind:

– die Datei wird auf einem RSO-Drucker ausgegeben
– der Parameter *control-mode* ist auf den Wert **line-mode* oder **physical* gesetzt

Ohne Angabe von *control-mode* werden die in der Datei vorhandenen Steuerzeichen als nicht druckbare Zeichen („Schmierzeichen“) dargestellt.

Bei Angabe des Parameters *postscript* (bzw. *ps*) werden die BS2000-Druckparameter für die Ausgabe von PostScript-Dateien eingestellt. Die Angabe der anderen Druckparameter ist dann nicht sinnvoll. Der ausgewählte Drucker muss in der Lage sein, PostScript-Dateien zu drucken.

-s (s - silent) Die Meldung des Kommandos *lp* „`request id is <request-id>`“ wird unterdrückt.

-t␣title
(t - title) Der Ausdruck beginnt mit einer zusätzlichen Kopfseite, auf der *title* ausgedruckt wird.

datei | **-**
Name der auszudruckenden Datei. Sie können mehrere Dateien angeben. Wenn Sie mehrere Dateien angeben, werden diese in der beim Aufruf angegebenen Reihenfolge ausgedruckt.
Wenn Sie für *datei* einen Bindestrich angeben, liest *lp* von der Standard-Eingabe und legt eine temporäre Datei im Verzeichnis */var/spool/lp/temp* an.

Für jede auszudruckende Datei gibt *lp* eine Meldung der folgenden Form aus:
`request id is <request-id> (<basename>)`

<request-id> wird im Format „TSN-nnnn“ ausgegeben, wobei *nnnn* der BS2000-TSN entspricht.
<basename> ist der Basisname von *datei*, d.h. der Teil des Pfadnamens nach dem letzten Schrägstrich („/“).

Wenn *lp* von der Standard-Eingabe liest, heißt die Meldung:
`request id is druckergruppe-id (standard input)`

datei nicht angegeben:
lp liest von der Standard-Eingabe.

Fehler

`lp: Cannot access the file: datei.`
`lp: Make sure file names are valid.`
`lp: No (or empty) input files.`
datei existiert nicht oder Sie haben kein Leserecht für *datei*. *lp* hat den Druckauftrag für *datei* nicht angenommen.

`lp: standard input is empty`
`lp: request not accepted`
Sie haben beim *lp*-Aufruf keine Datei oder einen Bindestrich - für *datei* angegeben, aber nichts über die Standard-Eingabe eingegeben. *lp* hat den Druckauftrag nicht angenommen.

`lp: Requests for destination "XYZ" aren't being accepted.`
`lp: Use the "lpstat -a" command to see why this destination is not accepting requests.`
Sie haben beim *lp*-Aufruf bei der Option *-d* für *dest* XYZ angegeben, XYZ ist aber kein zulässiger RSO-Drucker oder Druckergruppe.

Es kann auch sein, dass die Umgebungsvariable *LPDEST* den Wert XYZ hat. Dieser Wert wird dann bei jedem *lp*-Aufruf für *dest* eingesetzt, wenn Sie nicht explizit etwas anderes angeben.

`lp: unrecognized option "-x"`
Sie haben beim *lp*-Aufruf eine ungültige Option, in diesem Fall *-x*, angegeben.

Datei *var/spool/lp/temp/**
Temporäre Dateien, die beim Lesen von der Standard-Eingabe angelegt werden.

Variable *LPDEST*
Name einer Druckergruppe.
Wenn Sie *-d␣dest* nicht angeben, wird *-d␣$LPDEST* angenommen.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *lp*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

LC_TIME Legt das Format der Datums- und Zeitangaben für eine evtl. vorhandene Kopfseite fest.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Es sollen drei Dateien, *t1*, *t2* und *t3* ausgedruckt werden. Von den Dateien *t2* und *t3* sollen jeweils drei Kopien ausgedruckt werden:

```
$ lp t1 -n3 t2 t3
request id is TSN-153D (t1)
request id is TSN-153E (t2)
request id is TSN-1540 (t3)
```

Für jede auszudruckende Datei gibt *lp* eine Meldung aus.

Beispiel 2 Es sollen zwei Dateien, *t1* und *t2*, ausgedruckt werden. Es sind zwei Druckergruppen definiert, G001 und G002. Der Wert der Umgebungsvariablen *LPDEST* ist G001. *t1* soll auf der Druckergruppe G001, *t2* auf der Druckergruppe G002 ausgedruckt werden:

```
$ lp t1
request id is TSN-234A (t1)
$ lp -dG002 t2
request id is TSN-234C (t2)
```

Da die Umgebungsvariable *LPDEST* auf G001 gesetzt ist, brauchen Sie für *t1* nicht anzugeben, auf welcher Druckergruppe die Datei ausgedruckt werden soll. Wäre *LPDEST* nicht definiert oder hätte sie nicht den Wert G001, müsste der Aufruf folgendermaßen heißen:

```
$ lp -dG001 t1
$ lp -dG002 t2
```

Siehe auch *cancel*, *lpstat*

lpstat Informationen über Druckaufträge ausgeben (report line printer status information)

lpstat gibt Informationen über alle mit *lp* gestellten Druckaufträge aus.

Nichtprivilegierte Benutzer erhalten nur Informationen über die von der eigenen Benutzerkennung gestarteten Druckaufträge.
Privilegierte Benutzer ($TSOS) erhalten Informationen über alle Druckaufträge.

Mit *cancel* können Sie laufende Druckaufträge abbrechen oder gestartete Druckaufträge löschen.

Syntax

lpstat[␣**-drst**][␣**-a**[list]][␣**-c**[list]][␣**-o**[list]][␣**-p**[list]][␣**-u**[list]][␣ID....]

Keine Option angegeben

Für jeden mit *lp* gestellten und noch nicht vollständig ausgeführten Druckauftrag gibt *lpstat* eine Zeile mit folgenden Informationen aus:

- die Auftragsnummer, die *lp* nach seinem Aufruf ausgibt
- die Benutzerkennung des Auftraggebers
- die Größe der auszudruckenden Datei in Byte
- den Namen des ausführenden RSO-Druckers, insofern der Druckauftrag gerade ausgeführt wird

option

list kann entweder als Aufzählung, durch Komma getrennt, angegeben werden oder als Aufzählung in Anführungszeichen, wobei die Listenelemente durch Komma oder Leerzeichen voneinander getrennt sein können.

-a[list]
Druckerstatus.
list bezeichnet RSO-Druckernamen und Druckergruppennamen.

-c[list]
Druckergruppenname.
list bezeichnet Druckergruppernamen.

-d Ausgabe der Standard-Druckergruppe in folgender Form:

```
system default destination: *CENTRAL
```

-o[list]
Ausgabe des Status der Druckaufträge.
list bezeichnet RSO-Druckernamen, Druckergruppen und Auftragsnummern.

-p[list]
Ausgabe des Druckerstatus.
list bezeichnet RSO-Druckernamen.

-r Ausgabe des Spooler-Status:

```
scheduler is running
```

-s Ausgabe des Spooler- und Druckerstatus:

```
scheduler is running
system default destination: *CENTRAL
```

-t Ausgabe der gesamten Status-Information (siehe *Beispiel 2*).

-u[list]
Benutzerkennung. Nichtprivilegierte Benutzer dürfen nur die eigene Benutzerkennung angeben.
list bezeichnet login-Namen.

ID Auftragsnummer, die von *lp* vergeben wird.

Variable *TZ*
bestimmt die Zeitzone, die bei Datums- und Zeitangaben benötigt wird.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *lpstat*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

LC_TIME Legt das Format der Datums- und Zeitangaben fest, wenn die Informationen über die Druckaufträge mit den Optionen *-a, -o, -p, -t* oder *-u* ausgegeben wird.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Informationen über anstehende und zur Zeit laufende Druckaufträge ausgeben:

```
$ lpstat
TSN-6XDZ QM212JNA 12837 *ACT LE       *CENTRAL HP
```

Beispiel 2 Ausgabe aller Informationen

```
$ lpstat -t
scheduler is running
system default destination: *CENTRAL
DROS336  accepting requests
DRS03    accepting requests
DR9001   accepting requests
DR9022   accepting requests
EMDRS001 accepting requests
:
:
TC12G002 accepting requests
DROS336   9001RP     idle
DRS03     9022RP     idle
EMDRS003  9022RP     stopped
EMDRS016  9022RP     idle
LB        HP-PRINT   printing
LC        HP-PRINT   printing
LD        HP-PRINT   printing
LE        HP-PRINT   printing
MDRS05    9001RP     idle
:
TSN-6XK4 QM212JNA 329853 *ACT LE       *CENTRAL HP
```

Siehe auch *lp*, *cancel*
„Referenzhandbuch für Benutzer“ [12]

ls Informationen über Dateiverzeichnisse und Dateien ausgeben (list directory contents)

Mit *ls* können Sie sich Informationen über Dateien und Dateiverzeichnisse ausgeben lassen. Art und Umfang der Information sowie das Ausgabeformat können Sie festlegen, indem Sie Optionen angeben.

Syntax **ls**[␣option]...[␣datei]...

Kein Argument angegeben
ls gibt aus: Namen der Dateien und Dateiverzeichnisse des aktuellen Dateiverzeichnisses, alphabetisch sortiert.

Keine Option angegeben
datei ist ein Dateiverzeichnis:
ls gibt einspaltig aus: Namen der Dateien und Dateiverzeichnisse, die darin eingetragen sind, alphabetisch sortiert. Dateien, deren Name mit einem Punkt beginnt, z.B. *.profile*, werden nicht ausgegeben.

datei ist eine Datei:
ls gibt aus: Name der Datei. Die Datei kann mit ihrem teilqualifizierten Dateinamen angegeben werden. Wenn für *datei* ein Stern * angegeben wird, gibt *ls* alle Dateien und Dateiverzeichnisse (mit Inhalt) aus.

option

-a Alle Dateinamen, auch die, die mit einem Punkt . beginnen, werden aufgelistet.

-b Nicht druckbare Zeichen in Dateinamen werden oktal im Format *\ddd* ausgegeben. Anstelle von *ddd* steht der oktale Zahlenwert des Zeichens (siehe *Tabellen und Verzeichnisse, Zeichensatz EBCDIC*).

-c Zusammen mit Option *-l*:
Statt des Zeitpunkts der letzten Änderung der Datei wird der Zeitpunkt der letzten Indexänderung ausgegeben (Dateierstellung, Änderung der Zugriffsberechtigung etc).

Zusammen mit Option *-t*:
Statt des Zeitpunkts der letzten Änderung der Datei wird der Zeitpunkt der letzten Indexänderung als Sortierschlüssel verwendet.

-C Die Einträge werden mehrspaltig ausgegeben (Standardwert bei Ausgabe auf Terminal). Die Einträge innerhalb der Spalten sind alphabetisch sortiert.

ls verwendet die Umgebungsvariable *COLUMNS*, um die Anzahl Zeichen pro Zeile zu ermitteln. Ist diese nicht gesetzt, versucht *ls*, die Information in der Geräte-Datenbank zu finden, wobei der Wert der Umgebungsvariablen *TERM* als Suchschlüssel verwendet wird. Wenn das auch nicht erfolgreich ist, werden 80 Zeichen pro Zeile ausgegeben.

-d Ist *datei* ein Dateiverzeichnis, gibt *ls* dessen Namen (nicht seinen Inhalt) aus; diese Option wird oft zusammen mit *-l* gesetzt, um Informationen über den Status eines Dateiverzeichnisses zu erhalten. Wird *datei* nicht angegeben, gibt *ls* einen Punkt für das aktuelle Dateiverzeichnis aus.

-f *ls* behandelt jedes Argument als Dateiverzeichnis und gibt die Einträge jedes Dateiverzeichnisses in der Reihenfolge aus, in der sie erfasst wurden. Wird kein Argument angegeben, gibt *ls* die Einträge des aktuellen Dateiverzeichnisses aus.
Diese Option deaktiviert die Optionen *-l*, *-t*, *-F*, *-s* und *-r* und aktiviert die Option *-a*.

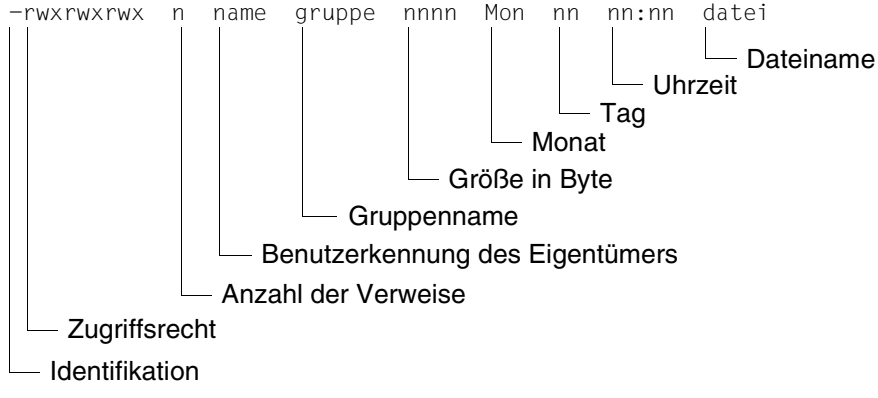
-F Dateiverzeichnisse werden mit einem Schrägstrich / hinter ihren Namen gekennzeichnet. Dateien, die als ausführbar deklariert sind, kennzeichnet *ls* mit dem Suffix *. Ein Klammeraffe @ bezeichnet einen symbolischen Verweis, ein senkrechter Strich | eine FIFO-Datei.

-g Die Benutzerkennung des Eigentümers wird nicht ausgegeben; wirkt im übrigen wie *-l*.

-i Die Indexnummer (inode) jeder Datei bzw. jedes Dateiverzeichnisses wird in der ersten Spalte der Ausgabe angezeigt.

-l Sie erhalten in einer langen Liste ausführliche Informationen.

Die Informationen werden in folgender Form ausgegeben:



Identifkation

- *d* (directory) Dateiverzeichnis
- *l* (symbolic link) Symbolischer Verweis
- *b* (block device special file) Gerätedatei, auf die blockweise zugegriffen werden kann
- *c* (character device special file) Gerätedatei, auf die zeichenweise zugegriffen werden kann

m (shared memory) Datei, auf die verteilter Zugriff möglich ist

p (named pipe) FIFO

s (semaphore) Semaphore-Datei

\- einfache Datei

Zugriffsrecht

3 Gruppen mit je 3 Zeichen informieren über die Zugriffsrechte
– des Eigentümers der Datei (Zeichen 1 bis 3),
– der Gruppenmitglieder (Zeichen 4 bis 6),
– der anderen Benutzer (Zeichen 7 bis 9).

In jeder Gruppe steht an

1. Stelle:

r für Leserecht oder - für kein Leserecht

2. Stelle:

w für Schreibrecht oder - für kein Schreibrecht

3. Stelle:

x für Ausführungsrecht

s für Ausführungsrecht plus gesetztes s-Bit

t für Ausführungsrecht plus gesetztes t-Bit (Sticky-Bit)

T für t-Bit (Sticky-Bit) ohne Ausführungsrecht

S für l-Bit (gesetztes s-Bit und nicht gesetztes Ausführungsrecht)

\- für weder Ausführungsrecht noch ein gesetztes spezielles Bit

Wenn für den Eigentümer oder die Gruppe das s-Bit und gleichzeitig das entsprechende x-Bit (für Ausführungsrecht) gesetzt ist, steht anstelle von *x* ein *s*. Das s-Bit kann ohne das entsprechende x-Bit nicht gesetzt sein.

Anstelle des x-Bits kann bei der Gruppe auch ein *S* stehen, falls das l-Bit für die Datei gesetzt ist, d.h., wenn die obligatorische Sperre (mandatory locking) für diese Datei eingeschaltet ist. In diesem Fall kann weder das x-Bit noch das s-Bit für die Gruppe gesetzt sein.
Die Lese- und Schreibrechte einer Datei mit gesetztem l-Bit können von einem Programm, das die Funktion *lockf()* verwendet, solange gesperrt werden, wie dieses Programm auf die Datei zugreift (siehe *chmod*).

Anstelle des x-Bits bei den anderen Benutzern kann auch ein t oder T stehen. T bzw. t bezeichnen den Status des Sticky-Bits (t-bits). T steht für das gesetzte Sticky-Bit ohne x-Bit, t steht für gesetztes Sticky-Bit mit gesetztem x-Bit (siehe *chmod*).

Anzahl der Verweise
Dezimalzahl, die die Anzahl der Verweise auf die Datei angibt, mindestens 1.

Benutzerkennung des Eigentümers
Login-Name des Eigentümers der Datei.

Gruppenname
Gruppenname des Eigentümers der Datei.

Größe in Byte
Dezimalzahl, die die Größe der Datei in Byte angibt.
Wenn eine Gerätedatei angegeben ist, so wird nicht die Größe der Datei ausgegeben, sondern Gerätetyp (major device number) und Gerätenummer (minor device number).

Monat, Tag, Uhrzeit
gibt das Datum und die Uhrzeit der letzten Änderung der Datei an. Wenn das Datum der letzten Änderung länger als ein halbes Jahr zurückliegt, wird anstelle der Uhrzeit die Jahreszahl ausgegeben.

Dateiname
Name der Datei.

Bei Dateiverzeichnissen wird zusätzlich zur Größe der einzelnen Dateien die Größe des ganzen Dateiverzeichnisses in Blöcken zu 512 Byte angegeben (siehe *Beispiel 4*).

Falls die Datei oder das Dateiverzeichnis ein symbolischer Verweis ist, dann wird hinter dem Dateinamen ein Pfeil -> angegeben, gefolgt von dem Pfadnamen der Datei, auf die verwiesen wird.

Beachten Sie auch, dass Sie in einer Umgebung mit Remote File Sharing nicht immer die Rechte haben, die Ihnen die Ausgabe des Kommandos *ls -l* vermittelt.

-L Bei symbolischen Verweisen wird der Verweisname für die Ursprungsdatei oder das Ursprungsverzeichnis ausgegeben.

Beispiel:

```
$ ls -l
-rw------- 1   HUGO     other       7593   Nov 30 10:48 datei
-rw------- 1   HUGO     other          3   Dec 17 10:53 linker -> datei

$ ls -lL
-rw------- 1   HUGO     other       7593   Nov 30 10:48 datei
-rw------- 1   HUGO     other       7593   Nov 30 10:48 linker
```

-m Ausgabe in Listenform. Die Einträge werden hintereinander, durch Kommas voneinander getrennt, aufgelistet. *ls* verwendet die Umgebungsvariable *COLUMNS*, um die Anzahl Zeichen pro Zeile zu ermitteln. Ist diese nicht gesetzt, versucht *ls*, die Information in der Geräte-Datenbank zu finden, wobei der Wert der Umgebungsvariablen *TERM* als Suchschlüssel verwendet wird. Wenn das auch nicht erfolgreich ist, werden 80 Zeichen pro Zeile ausgegeben.

-n Anstelle des Namens des Eigentümers bzw. des Gruppennamens wird die Benutzer- bzw. die Gruppennummer ausgegeben; wirkt im übrigen wie *-l*.

-o Die Gruppe wird nicht ausgegeben; wirkt im übrigen wie *-l*.

-p Ein Schrägstrich / hinter Dateinamen kennzeichnet Dateiverzeichnisse.

-q Nicht druckbare Zeichen in Dateinamen werden in der Ausgabe als Fragezeichen ? ausgegeben.

-r Die aktuell gültige Sortierreihenfolge wird umgekehrt.

-R ohne Angabe von *datei*
Die Namen aller Dateien und Dateiverzeichnisse des aktuellen Dateiverzeichnisses werden ausgegeben, danach werden alle Unterdateiverzeichnisse und die darin enthaltenen Dateien und Dateiverzeichnisse rekursiv aufgelistet.

mit Angabe von *datei*
datei ist der Name eines Dateiverzeichnisses.

Die Namen aller Dateien und Unterdateiverzeichnisse des angegebenen Dateiverzeichnisses werden rekursiv ausgegeben.

-s Zusätzlich zu den Dateinamen wird in der ersten Spalte die Größe jeder Datei in Einheiten zu je 512 Byte ausgegeben.

-t Als Sortierschlüssel wird anstelle des Dateinamens das Datum der letzten Änderung verwendet: die Datei mit dem jüngsten Datum steht an erster Stelle.

-u Zusammen mit Option *-t*:
Als Sortierschlüssel wird der Zeitpunkt des letzten Zugriffs verwendet.

Zusammen mit Option *-l*:
Statt des Zeitpunkts der letzten Änderung wird der Zeitpunkt des letzten Zugriffs auf die Datei ausgegeben.

-x Mehrspaltige Ausgabe; die Einträge sind innerhalb der einzelnen Zeilen sortiert.
ls verwendet die Umgebungsvariable *COLUMNS*, um die Anzahl Zeichen pro Zeile zu ermitteln. Ist diese nicht gesetzt, versucht *ls*, die Information in der Geräte-Datenbank zu finden, wobei der Wert der Umgebungsvariablen *TERM* als Suchschlüssel verwendet wird. Wenn das auch nicht erfolgreich ist, werden 80 Zeichen pro Zeile ausgegeben.

-1 Es wird nur ein Eintrag pro Ausgabezeile ausgegeben.

datei
Name der Datei oder des Dateiverzeichnisses, über die/das Sie Informationen ausgeben möchten. Sie können auch mehrere Dateien oder Dateiverzeichnisse angeben.

datei nicht angegeben:
Der Inhalt des aktuellen Dateiverzeichnisses wird aufgelistet.

Datei */etc/group*
enthält alle eingerichteten Benutzergruppen.

Variable *COLUMNS*
Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Definition der Spaltenbreite bei mehrspaltiger Ausgabe verwendet. Enthält diese Variable eine Dezimalzahl, dann berechnet *ls*, wieviele Textspalten möglich sind (siehe auch Option *-C*). Dateinamen werden nicht abgeschnitten, um bei mehrspaltiger Ausgabe in eine Spalte zu passen.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *ls*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die Sortierreihenfolge der Ausgabe von *ls* fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten). *LC_CTYPE* bestimmt, welche Zeichen im Zusammenhang mit der Option *-q* als nichtdruckbare Zeichen definiert werden.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_TIME Legt das Format und den Inhalt der Datums- und Zeitausgabe bei den Optionen *-g*, *-l*, *-n* und *-o* fest.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Ausgeben des Inhaltsverzeichnisses des aktuellen Dateiverzeichnisses in ausführlicher Form. Benutzerkennung: *hugo*.

```
$ ls -l
total 142
drwx--x--x 2   HUGO     other          48   Dec  1 16:13 ADRESSEN
-rw------- 1   HUGO     other         593   Nov 30 10:48 brief1
-rw------- 1   HUGO     other         837   Dec 17 10:53 brief2
-rw------- 1   HUGO     other        3247   Dec 17 13:46 brief3
-rw------- 1   HUGO     other        5222   Nov 30 10:48 brief4
-rw-rw-rw- 1   HUGO     other        4687   Dec 21 11:15 brief5
-rw------- 1   HUGO     other         228   Nov 30 10:48 brief6
-rw------- 1   HUGO     other         356   Dec 17 13:58 kart_a
-rw------- 1   HUGO     other       24802   Dec  1 12:13 kart_b
-rw------- 1   HUGO     other        7890   Nov 30 10:48 kart_c
-rwx--xr-x 1   HUGO     other        7253   Dec 21 13:37 kart_d
-rw------- 1   HUGO     other        9476   Dec 21 13:37 kart_e
-rwx--x--x 1   HUGO     other           0   Dec 18 13:16 kart_f
-r-------- 1   HUGO     other         105   Dec 21 13:39 typescript
```

Beispiel 2 Mehrspaltige Ausgabe des aktuellen Dateiverzeichnisses auf einer Breite von 80 Zeichen pro Ausgabezeile. Diese Form der Ausgabe ist zugleich Standard bei Ausgabe auf Terminal).

```
$ ls -C
ADRESSEN    brief3      brief6      kart_c     kart f
brief1      brief4      kart_ a     kart_d     typescript
brief2      brief5      kart_ b     kart_e
```

Beispiel 3 Mehrspaltige Ausgabe, nachdem die Zahl der Zeichen pro Ausgabezeile auf 40 festgelegt wurde.

```
$ COLUMNS=40
$ export COLUMNS
$ ls -C
ADRESSEN    brief5      kart d
brief1      brief6      kart e
brief2      kart a      kart f
brief3      kart b      typescript
brief4      kart c
```

Beispiel 4 Ausgeben des aktuellen Dateiverzeichnisses in ausführlicher Form (Option *-l*), mit allen Punktdateien (Option *-a*) und mit Kennzeichnung der Dateiverzeichnisse (Option *-p*). Benutzerkennung: *sysiphus*.

```
$ ls -pla
total 120
drwxr-xr-x  10 SYSIPHUS   other       5720   Nov 17 08:00 ./
drwxr-xr-x  13 SYSROOT    SYSROOT     3380   Nov 04 11:48 ../
-rw-------   1 SYSIPHUS   other         79   Jul 19 14:21 .profile
-rw-------   1 SYSIPHUS   other         14   Oct 21 08:56 .rhosts
-rwx------   1 SYSIPHUS   other        125   May 25 10:29 anfang
drwx------   2 SYSIPHUS   other       1560   Oct 11 15:36 bildschirme/
drwx------   2 SYSIPHUS   other       1820   Oct 11 15:35 briefe/
drwx--x--x   2 SYSIPHUS   other       3380   Nov 09 10:30 kommandos/
drwxr-----   3 SYSIPHUS   other       2340   Oct 11 15:35 lingua/
-rw-------   1 SYSIPHUS   other       2082   Nov 08 12:29 ls.ex
-rw-------   1 SYSIPHUS   other      11597   Nov 17 07:59 ls.rc.1
-rw-------   1 SYSIPHUS   other       1351   Jul 19 15:14 plural
drwx------   2 SYSIPHUS   other       3380   Oct 11 15:36 post/
drwx------   2 SYSIPHUS   other       1560   Oct 11 15:36 pro/
drwx--x--x   2 SYSIPHUS   other       2080   Nov 07 10:43 proz/
drwx--x--x   2 SYSIPHUS   other       1040   Nov 15 08:23 sdg/
```

Beispiel 5 Ausgeben aller Dateien und Dateiverzeichnisse (mit Inhalt). Die Namen der Dateiverzeichnisse werden mit einem Doppelpunkt beendet.

```
$ ls -C *
Postkasten autoren   bvp.col

bildschirme:
aus        pause     sinix

lingua:
beispiele  car.cdr   engl.dt    fachwort
```

Siehe auch *chmod, find, ln*

mailx Nachrichten interaktiv bearbeiten (process messages)

Die Beschreibung ist in folgende Abschnitte unterteilt:

- Einführung
- Überblick über die *mailx*-Formate
- Beschreibung der Formate:
  - Format 1: Lesemodus (ab Seite 523)

    mailx-Kommandos im Lesemodus
    Eingabeformat
    Funktionale Übersicht
    Alphabetische Beschreibung
    Arbeitsweise im Lesemodus
  - Format 2: Sendemodus (ab Seite 543)

    mailx-Kommandos im Sendemodus (Tilde-Kommandos)
    Eingabeformat
    Funktionale Übersicht
    Alphabetische Beschreibung
    Arbeitsweise im Sendemodus
- *mailx*-Kommando- und Startdateien (ab Seite 550)
- Variablen:
  - *mailx*-Variablen (ab Seite 552)
  - Umgebungsvariablen (ab Seite 557)
- Hinweis zu den Fehlermeldungen
- Dateien
- Beispiele
- Siehe auch

Einführung

Mit *mailx* können Sie elektronische Post senden und empfangen.
Wenn MAIL der interNet Services (ehemals interNet Value Edition) installiert ist, können Sie mit anderen Benutzern desselben Rechners Nachrichten austauschen.

Bereits bei der Anmeldung erfahren Sie, ob Sie Post bekommen haben. Auch während der Anwendung von *mailx* wird gemeldet, wenn neue Nachrichten eintreffen.

Nachrichten enthalten einen Nachrichtenkopf mit Informationen, die zum Weiterleiten der Nachricht dienen.

Der Nachrichtenkopf enthält folgende Felder:

– Message Nummer der Nachricht

– From Sender der Nachricht, Datum und Uhrzeit

– To Empfänger der Nachricht

– Subject Titel der Nachricht (siehe Option *-s*)

– Status Status der Nachricht (siehe Beschreibung auf Seite 542)

Auf den Nachrichtenkopf folgt eine Leerzeile und anschließend der Text der Nachricht.

Mit *mailx* können Sie

– prüfen, ob Nachrichten vorliegen (Lesemodus, Format 1, Option *-e*)

– Nachrichten lesen (Lesemodus, Format 1)

– Nachrichten mit Shell-Kommandos weiterverarbeiten (Lesemodus, Format 1, *mailx*-Kommandos *!*, | und *pipe*)

– Nachrichten senden (Sendemodus, Format 2)

– Nachrichten während des *mailx*-Dialogs mit einem Editor bearbeiten (Lese- und Sendemodus, *mailx*-Kommandos *edit, visual,* ~*e* und ~*v*)

mailx sammelt gelesene Nachrichten automatisch in einem benutzereigenen Briefkasten (standardmäßig *$HOME/mbox*).

Überblick über die mailx-Formate

Syntax

Format 1: mailx[␣option]

Format 2: mailx[␣option]...␣empfänger...

Format 1 **Lesemodus**

mailx[␣option]

Keine Option angegeben
mailx verhält sich wie im Abschnitt *Arbeitsweise im Lesemodus* beschrieben.

option

-e *mailx* prüft, ob Nachrichten vorliegen. Wenn dies der Fall ist, gibt *mailx* den Endestatus 0 zurück, sonst 1. Dann beendet sich *mailx*.
mailx durchläuft keine Startdatei (siehe Abschnitt *mailx-Kommando- und Startdateien*).

-f[␣datei]

datei nicht angegeben:
mailx liest die Nachrichten aus dem benutzereigenen Briefkasten *$HOME/mbox*.

-f datei nicht angegeben:
mailx liest die Nachrichten aus dem Standard-Briefkasten */var/mail/$USER*.

-H (H - header) *mailx* gibt nur Übersichtszeilen aus und beendet sich. Der Endestatus ist 0, wenn Nachrichten vorliegen, sonst 1.

Der Aufbau der Übersichtszeilen ist im Abschnitt *Arbeitsweise im Lesemodus* beschrieben.

-i (ignore) *mailx* ignoriert das Signal SIGINT (siehe *mailx*-Variable *ignore*).

-n *mailx* durchläuft nicht die systemweite Startdatei */etc/mail/mailx.rc* (siehe Abschnitt *mailx-Kommando- und Startdateien*).

-N *mailx* unterdrückt die Ausgabe der Meldungs- und Übersichtszeilen nach dem *mailx*-Aufruf.

-u␣benutzerkennung
mailx liest die Nachrichten aus dem Standard-Briefkasten des angegebenen Benutzers, vorausgesetzt, Sie haben Leserecht.

-V *mailx* gibt seine Versionsnummer aus und beendet sich.

mailx-Kommandos im Lesemodus

Eingabeformat

mailx-Kommandos im Lesemodus haben das folgende Format:

```
[kommando][␣nachrichtenliste][␣argument]...
```

kommando
: Name eines *mailx*-Kommandos. Die meisten Kommandonamen können Sie in abgekürzter Form angeben. Die Kurzform ist im Abschnitt *Alphabetische Beschreibung* durch Fettdruck hervorgehoben.

  kommando nicht angegeben:
  mailx führt das Kommando *print* aus, wenn Sie nach dem *mailx*-Prompt nur ↵ drücken.

nachrichtenliste
: Nachricht, die bearbeitet werden soll. Sie können mehrere Nachrichten angeben, jeweils durch Leerzeichen voneinander getrennt.

  nachrichtenliste kann sein:

  n — die Nachricht mit der Nummer *n*

  . — die aktuelle Nachricht (in der Übersichtszeile gekennzeichnet durch >)

  ^ — die erste nicht gelöschte Nachricht

  $ — die letzte Nachricht

  ***** — alle Nachrichten

  + — die nächste Nachricht

  – — die vorhergehende Nachricht

  n-m — alle Nachrichten der laufenden Nummern *n* bis *m* (einschließlich)

  benutzerkennung
  : alle Nachrichten des angegebenen Benutzers

  /zeichenkette
  : alle Nachrichten, die *zeichenkette* im Subject-Feld enthalten. Groß- und Kleinschreibung werden ignoriert.

:nachrichtentyp
die Nachricht vom Typ *nachrichtentyp*. Folgende Angaben sind möglich:

d (deleted)
gelöschte Nachrichten (nur bei Kommando undelete sinnvoll)

n (new)
neue Nachrichten

o (old)
alte Nachrichten (alle Nachrichten, die nicht vom Typ r oder n sind)

r (read)
gelesene Nachrichten

u (unread)
noch nicht gelesene Nachrichten

nachrichtenliste nicht angegeben:
mailx nimmt die aktuelle Nachricht an.

argument
Beliebige Zeichenkette, die jeweils bei den betreffenden Kommandos beschrieben ist. Wenn *argument* ein Dateiname ist, können Sie die üblichen Sonderzeichen der Shell verwenden. Zeichenketten, die Leerzeichen enthalten, müssen Sie in Anführungszeichen „...“ einschließen, wenn sie als ein einziges Argument interpretiert werden sollen.

Funktionale Übersicht

In diesem Abschnitt erhalten Sie eine Übersicht über alle *mailx*-Kommandos im Lesemodus, sortiert nach ihren wichtigsten Funktionen. Dabei kann es vorkommen, dass einige Kommandos mehrmals aufgeführt werden.
Im Anschluss an diese Übersicht werden diese Kommandos in alphabetischer Reihenfolge beschrieben.

Die meisten Kommandos sind abkürzbar. Die Kurzformen sind im Abschnitt „Alphabetische Beschreibung“ auf Seite 529 durch Fettdruck hervorgehoben.

Hilfsinformationen ausgeben

?	Übersicht über die *mailx*-Kommandos ausgeben
help	Übersicht über die *mailx*-Kommandos ausgeben
list	Namen aller *mailx*-Kommandos ausgeben
=	Nummer der aktuellen Nachricht ausgeben
size	Größe einer Nachricht ausgeben
from	Übersichtszeilen ausgeben
z+	Nächste Seite mit Übersichtszeilen ausgeben
z-	Vorhergehende Seite mit Übersichtszeilen ausgeben

headers	Bildschirmseite mit Übersichtszeilen ausgeben
top	Die ersten 5 Zeilen des Nachrichtenkopfes ausgeben
folders	Inhalt des Dateiverzeichnisses ausgeben, das mit der *mailx*-Variablen *folder* festgelegt wurde
version	Versionsnummer von *mailx* ausgeben

mailx beenden

exit	*mailx* beenden, ohne den Briefkasten zu verändern
xit	*mailx* beenden, ohne den Briefkasten zu verändern
quit	*mailx* beenden

Übersichtszeilen ausgeben

from	Übersichtszeilen ausgeben
headers	Bildschirmseite mit Übersichtszeilen ausgeben
z+	Nächste Seite mit Übersichtszeilen ausgeben
z-	Vorhergehende Seite mit Übersichtszeilen ausgeben

Nachrichtenkopf bearbeiten, ausgeben

discard	Felder des Nachrichtenkopfes unterdrücken
undiscard	Wirkung von *discard* aufheben
ignore	Felder des Nachrichtenkopfes unterdrücken
unignore	Wirkung von *ignore* aufheben
retain	Nur die angegebenen Felder des Nachrichtenkopfes ausgeben
top	Die ersten 5 Zeilen des Nachrichtenkopfes ausgeben

Nachricht ausgeben

print	Nachricht ausgeben
type	Nachricht ausgeben
next	Die nächste, passende der spezifizierten Nachrichten ausgeben
Print	Nachricht mit ganzem Nachrichtenkopf ausgeben, trotz *discard*
Type	Nachricht mit ganzem Nachrichtenkopf ausgeben, trotz *discard*

Nachricht editieren

edit	Nachricht mit einem Editor bearbeiten (Wert der *mailx*-Variablen *EDITOR*, Standard: *ed*)
visual	Nachricht mit einem Editor bearbeiten (Wert der *mailx*-Variablen *VISUAL*, Standard: *vi*)

Briefkasten wechseln

file	Aktuellen Briefkasten schließen und angegebenen öffnen
folder	Aktuellen Briefkasten schließen und angegebenen öffnen

Nachricht sichern

hold	Nachrichten im Briefkasten halten
preserve	Nachrichten im Briefkasten halten
save	Nachricht in eine Datei schreiben
copy	Nachricht in eine Datei schreiben
write	Nachricht ohne Nachrichtenkopf in eine Datei schreiben
mbox	Nachricht in den benutzereigenen Briefkasten schreiben
touch	Nachricht in den benutzereigenen Briefkasten schreiben
Save	Nachrichten in eine Datei schreiben, deren Name gleich dem Namen des Absenders der ersten angegebenen Nachricht ist
Copy	Nachrichten in eine Datei schreiben, deren Name gleich dem Namen des Absenders der ersten angegebenen Nachricht ist

Nachricht löschen

delete	Nachricht löschen
dp	Nachricht löschen, nächste Nachricht ausgeben
dt	Nachricht löschen, nächste Nachricht ausgeben

In den Sendemodus wechseln und Nachricht senden oder beantworten

mail	Nachricht senden
Mail	Nachricht senden und in einer Datei protokollieren
reply	Eine Nachricht beantworten
respond	Eine Nachricht beantworten
followup	Eine Nachricht beantworten und Antwort protokollieren
Reply	Mehrere Nachrichten beantworten
Respond	Mehrere Nachrichten beantworten
Followup	Mehrere Nachrichten beantworten und Antwort protokollieren

mailx-Kommandos während der mailx-Sitzung rückgängig machen

undelete	Gelöschte Nachrichten zurückholen
touch	Wirkung von *hold* aufheben
hold	Wirkung von *touch* aufheben
unalias	Alias-Namen löschen
undiscard	Wirkung von *discard* aufheben
unignore	Wirkung von *ignore* aufheben
unset	Variable zurücksetzen

Kommandointerpreter aufrufen, Shell-Kommando ausführen

!	Shell-Escape
!!	Das zuletzt ausgeführte Shell-Kommando wiederholen
shell	Kommandointerpreter aufrufen

pipe	Nachrichten als Standard-Eingabe an Shell-Kommando übergeben
|	Nachrichten als Standard-Eingabe an Shell-Kommando übergeben

Verschiedenes

#	Leeres Kommando (Kommentare in Kommandodateien einfügen)
=	Nummer der aktuellen Nachricht ausgeben
alias	Alias-Namen für Empfänger definieren (wie *group*)
alternates	Alternative Namen für die eigene Benutzerkennung definieren
cd	Dateiverzeichnis wechseln
chdir	Dateiverzeichnis wechseln
echo	Zeichenkette ausgeben (wie SINIX-Kommando *echo*)
folders	Inhalt des Dateiverzeichnisses ausgeben, das mit der *mailx*-Variablen *folder* festgelegt wurde
group	Alias-Namen für Empfänger definieren (wie *alias*)

if modus cmdlist1 else cmdlist2 endif

if-Anweisung, die je nach Modus (senden, lesen) eine der Kommandolisten ausführt

set	Variable setzen
size	Größe einer Nachricht ausgeben
source	Kommandodatei lesen und ausführen
unset	Variablen zurücksetzen

mailx-Kommandos, die nicht in Kommandodateien stehen dürfen

!	Shell-Escape
Copy	Nachrichten in eine Datei schreiben, deren Name gleich dem Namen des Absenders der ersten angegebenen Nachricht ist
edit	Nachricht mit einem Editor bearbeiten
followup	Eine Nachricht beantworten und Antwort protokollieren
Followup	Mehrere Nachrichten beantworten und Antwort protokollieren
hold	Nachrichten im Briefkasten halten
mail	Nachricht senden
Mail	Nachricht senden und protokollieren
reply	Eine Nachricht beantworten
preserve	Nachrichten im Briefkasten halten
Reply	Mehrere Nachrichten beantworten
respond	Eine Nachricht beantworten
Respond	Mehrere Nachrichten beantworten
shell	Kommandointerpreter aufrufen
visual	Nachricht mit einem Editor bearbeiten

Alphabetische Beschreibung

Für einige Kommandonamen gibt es Synonyme. Die ausführliche Beschreibung finden Sie immer beim alphabetisch ersten Kommando.

Die meisten dieser *mailx*-Kommandos können Sie sowohl im Dialog als auch in Kommandodateien benutzen. Auf Ausnahmen davon ist beim jeweiligen Kommando verwiesen (siehe auch Abschnitt *Funktionale Übersicht*).

Der Fettdruck bei den Kommandonamen kennzeichnet ihre Kurzform.

!shell-kommando
führt *shell-kommando* aus. Standardmäßig wird der durch die Umgebungsvariable *SHELL* definierte Kommandointerpreter aufgerufen und der angegebene Kommandoaufruf übergeben. Falls *SHELL* nicht gesetzt ist, wird */bin/sh* aufgerufen.

Wenn die *mailx*-Variable *bang* gesetzt ist, speichert *mailx* das zuletzt ausgeführte Shell-Kommando. Mit *!!* können Sie es wiederholen.

Das Kommando *!* darf nicht in einer Kommandodatei stehen.

#kommentar
ist ein leeres Kommando. Sie können damit Kommentare in Kommandodateien (z.B. *.mailrc*) einfügen.

= gibt die Nummer der aktuellen Nachricht aus.

? gibt eine Übersicht über *mailx*-Kommandos aus.

alias[␣alias-name[␣empfänger]]...
group[␣alias-name[␣empfänger]]...
definiert einen Alias-Namen für die angegebenen Empfänger. *mailx* setzt die definierten Empfänger ein, wenn Sie die Alias-Namen als Empfänger angeben.

alias-name
Beliebige Zeichenkette.

alias-name nicht angegeben:
mailx gibt eine Liste der definierten Alias-Namen aus.

empfänger nicht angegeben:
mailx gibt die Definitionen zu *alias-name* aus.

alternates[␣name]...
definiert alternative Namen für die eigene Benutzerkennung. Wenn Sie eine Nachricht beantworten, streicht *mailx* diese alternativen Namen aus der Liste der Empfänger.

name: Zeichenkette für den alternativen Namen.

name nicht angegeben:
mailx gibt die aktuelle Liste der alternativen Namen aus.

cd␣[dateiverzeichnis]
chdir␣[dateiverzeichnis]
wechselt ins angegebene Dateiverzeichnis.

dateiverzeichnis nicht angegeben:
mailx wechselt in *$HOME*.

copy[[␣nachrichtenliste]␣datei]
schreibt die angegebenen Nachrichten in die Datei *datei*. Die Datei wird erweitert, falls sie existiert.
Die Nachrichten werden als gelesen gekennzeichnet (*R*) und beim Schließen des Standard-Briefkastens in den benutzereigenen Briefkasten geschrieben.

Kein Argument angegeben:
mailx schreibt die aktuelle Nachricht ans Ende der Datei *$HOME/mbox*. Wenn Sie die Nachricht anschließend nicht löschen, wird sie beim Schließen des Standard-Briefkastens nochmals gesichert!

Copy[␣nachrichtenliste]
schreibt die angegebenen Nachrichten in eine Datei im aktuellen Dateiverzeichnis, deren Name gleich dem Namen des Absenders der ersten Nachricht in der Nachrichtenliste gesetzt wird (From-Eintrag).
Die Datei wird erweitert, falls sie existiert.

Die Nachrichten werden als gelesen gekennzeichnet (*R*) beim Schließen des Standard-Briefkastens in den benutzereigenen Briefkasten geschrieben.

Das Kommando *Copy* darf nicht in einer Kommandodatei stehen.

delete[␣nachrichtenliste]
löscht die angegebenen Nachrichten aus dem aktuellen Briefkasten. Wenn die *mailx*-Variable *autoprint* gesetzt ist, wird die nächste Nachricht nach der gelöschten ausgegeben.

Eine gelöschte Nachricht können Sie innerhalb einer *mailx*-Sitzung mit *undelete* zurückholen.

discard[␣feld]...
ignore[␣feld]...
unterdrückt die angegebenen Felder des Nachrichtenkopfes bei der Ausgabe, wenn die Felder am Beginn einer Zeile stehen und mit Doppelpunkt enden, z.B.: *Cc:, Date:, Status:, Subject:, To:*. Den Doppelpunkt müssen Sie nicht angeben. *mailx* unterscheidet nicht zwischen Groß- und Kleinschreibung von *feld*.

discard wirkt auf die *mailx*-Kommandos *next*, *pipe* (bzw. |), *print*, *type*, *~f* und *~m*, nicht jedoch auf *Print*, *Type, ~F* und *~M*.

Beim Sichern einer Nachricht werden die unterdrückten Felder mitgesichert.

Die Wirkung von *discard* lässt sich mit *undiscard* bzw. *unignore* rückgängig machen. *retain* hebt die Wirkung von *discard* auf; es unterdrückt alle Felder außer denen, die explizit angegeben werden.

feld nicht angegeben:
discard gibt die aktuelle Liste der zu unterdrückenden Felder aus, falls vorhanden.

dp[␣nachrichtenliste]
dt[␣nachrichtenliste]
(dp - delete and print) löscht Nachrichten aus dem Briefkasten und gibt die nächste Nachricht nach der zuletzt gelöschten aus.

Eine gelöschte Nachricht können Sie innerhalb einer *mailx*-Sitzung mit *undelete* zurückholen.

echo␣zeichenkette␣...
gibt *zeichenkette* auf die Standard-Ausgabe aus (wie das SINIX-Kommando *echo*).
Mit *$name* können Sie auf den Wert der Umgebungsvariablen *name* zugreifen. *echo* gibt immer deren Wert aus, auch wenn eine *mailx*-Variable gleichen Namens definiert ist.

edit[␣nachrichtenliste]
ruft den mit der *mailx*-Variablen *EDITOR* eingestellten Editor auf (standardmäßig *ed*) und lädt die angegebenen Nachrichten.

Nach Beenden der Editorsitzung liegt die bearbeitete Nachricht wieder im Briefkasten vor.

Der Text wird in einer temporären Datei bearbeitet. Der Dateiname ist */tmp/Rz$$* (*$$* ist die Prozessnummer des *mailx*-Prozesses).

Das Kommando *edit* darf nicht in einer Kommandodatei stehen.

exit
xit
beendet *mailx*. Der aktuell bearbeitete Briefkasten bleibt beim Schließen unverändert, d.h.
- gelöschte Nachrichten bleiben erhalten
- gelesene Nachrichten werden nicht nach *$HOME/mbox* gesichert
- bearbeitete Nachrichten behalten den alten Stand

Siehe auch *mailx*-Kommando *quit*.

file[␣datei]
folder[␣datei]
schließt den aktuellen Briefkasten (wie bei *quit*) und öffnet die angegebene Datei als Briefkasten. Dabei zeigt *mailx* die entsprechenden Übersichtszeilen an.

datei
Name des zu bearbeitenden Briefkastens oder eines der folgenden Sonderzeichen:
% der aktuelle Briefkasten
%benutzerkennung
der Standard-Briefkasten des angegebenen Benutzers (*/var/mail/$USER*)
+datei die Datei *datei* im *folder* Dateiverzeichnis (siehe mailx-Variable *folder*)
der zuvor bearbeitete Briefkasten
& der benutzereigene Briefkasten (*$HOME/mbox* bzw. der durch die *mailx*-Variable *MBOX* festgelegte Briefkasten)

datei nicht angegeben:
mailx bleibt im aktuellen Briefkasten und meldet nur die Anzahl der darin enthaltenen Nachrichten.

Durch die Angabe *file %* wird der aktuelle Briefkasten geschlossen und wieder geöffnet. So können Sie anschließend die Nachrichten lesen, die während Ihrer *mailx*-Sitzung neu eingetroffen sind.

folders
gibt die Dateinamen des Dateiverzeichnisses aus, das durch die *mailx*-Variable *folder* festgelegt ist (in dieses Dateiverzeichnis sichert bzw. protokolliert *mailx* Nachrichten).

followup[␣nachricht]
beantwortet die angegebene Nachricht wie das *mailx*-Kommando *reply*.

mailx geht in den Sendemodus und nimmt als Empfänger

– den Absender der angegebenen Nachricht, d.h. der Eintrag im From-Feld wird zum Eintrag in der To-Liste

– die weiteren Empfänger der Nachricht, d.h. die Einträge im To-Feld werden in die To-Liste übernommen, die Einträge im Cc-Feld in die Cc-Liste

Wenn Sie die Texteingabe beendet haben, sendet *mailx* die Nachricht ab.

Im Unterschied zu *reply* protokolliert *followup* die Antwort in eine Datei, deren Name gleich dem Namen des Empfängers gesetzt wird (Netz-Pfade werden abgetrennt). Wo diese Protokolldatei abgelegt wird, hängt davon ab, ob die *mailx*-Variablen *folder* und *outfolder* gesetzt sind. Wenn beide gesetzt sind, wird die Protokolldatei in das Dateiverzeichnis geschrieben, das durch *folder* festgelegt ist. Ansonsten wird die Datei im aktuellen Dateiverzeichnis abgelegt. Die Datei wird erweitert, falls sie existiert.

Das Kommando *followup* darf nicht in einer Kommandodatei stehen.

Followup[␣nachrichtenliste]
beantwortet die erste in der Nachrichtenliste angegebene Nachricht wie das *mailx*-Kommando *Reply*.

mailx geht in den Sendemodus und nimmt als Empfänger alle Absender aus *nachrichtenliste*.

Wenn Sie die Texteingabe beenden, sendet *mailx* die Nachricht ab.

Im Unterschied zu *Reply* protokolliert *Followup* die Antwort in eine Datei, deren Name gleich dem Namen des Absenders der ersten Nachricht gesetzt wird (Netz-Pfade werden abgetrennt). Wo diese Protokolldatei abgelegt wird, hängt davon ab, ob die *mailx*-Variablen *folder* und *outfolder* gesetzt sind. Wenn beide gesetzt sind, wird die Protokolldatei in das Dateiverzeichnis geschrieben, das durch *folder* festgelegt ist. Ansonsten wird die Datei im aktuellen Dateiverzeichnis abgelegt. Die Datei wird erweitert, falls sie existiert.

Das Kommando *Followup* darf nicht in einer Kommandodatei stehen.

from [␣nachrichtenliste]
gibt die Übersichtszeile jeder angegebenen Nachricht auf die Standard-Ausgabe aus.

group␣alias-name␣empfänger...
definiert Alias-Namen für die angegebenen Empfänger (siehe *alias*).

headers[␣nachricht]
gibt eine Bildschirmseite mit den Übersichtszeilen aus, die *nachricht* enthalten.
Eine Bildschirmseite umfasst 20 Zeilen oder die mit der Variablen *screen* festgelegte Anzahl.

nachricht nicht angegeben:
mailx nimmt die aktuelle Nachricht an.

help
gibt eine Übersicht über die *mailx*-Kommandos aus (siehe auch *?*).

hold[␣nachrichtenliste]
preserve[␣nachrichtenliste]
hält die angegebenen Nachrichten im Briefkasten.

Die Nachrichten sind in der Übersichtszeile mit *H* gekennzeichnet und bleiben im Briefkasten, auch wenn sie gelesen oder gesichert wurden.

Mit *touch* lässt sich die Wirkung von *hold* aufheben und umgekehrt.

Die Kommandos *hold* und *preserve* dürfen nicht in einer Kommandodatei stehen.

if modus
kommando-liste1
else
kommando-liste2
endif

if-Anweisung, die je nach angegebenem Modus eine der Kommandolisten ausführt.

modus

modus bezeichnet den Modus (lesen oder senden), in dem Sie *mailx* aufgerufen haben. *modus* kann sein:

s (send) *kommando-liste1* wird ausgeführt, wenn Sie *mailx* im Sendemodus aufgerufen haben, sonst wird *kommando-liste2* ausgeführt.

r (read) *kommando-liste1* wird ausgeführt, wenn Sie *mailx* im Lesemodus aufgerufen haben, sonst wird *kommando-liste2* ausgeführt.

kommando-liste1
kommando-liste2

Listen von *mailx*-Kommandos. Nicht zulässig sind hier Kommandos, die auch nicht in Kommandodateien stehen dürfen: *!, edit, followup, Followup, mail, Mail, reply, Reply, respond, Respond, shell* und *visual*.

if, *else*, *endif* und jedes Kommando müssen jeweils in einer Zeile stehen.

ignore[␣feld]...

unterdrückt die angegebenen Felder des Nachrichtenkopfes bei der Ausgabe (siehe *discard*).

list

gibt die Namen aller verfügbaren *mailx*-Kommandos auf die Standard-Ausgabe aus (siehe auch *help* und *?*).

mail␣empfänger...

sendet eine Nachricht an *empfänger*.

mailx geht in den Sendemodus. Wenn Sie die Texteingabe beendet haben, sendet *mailx* die Nachricht ab.
Wenn die *mailx*-Variable *record* gesetzt ist, schreibt *mailx* die Nachricht in die dort angegebene Datei. Die Datei wird erweitert, falls sie existiert.

Das Kommando *mail* darf nicht in einer Kommandodatei stehen.

Mail␣empfänger
sendet eine Nachricht an *empfänger*.

mailx geht in den Sendemodus. Wenn Sie die Texteingabe beendet haben, sendet *mailx* die Nachricht ab.
mailx protokolliert Ihre Nachricht in einer Datei im aktuellen Dateiverzeichnis, deren Name gleich dem des Empfängers ist. Die Datei wird erweitert, falls sie existiert.

Das Kommando *Mail* darf nicht in einer Kommandodatei stehen.

mbox[␣nachrichtenliste]
schreibt die angegebenen Nachrichten beim Schließen des aktuell bearbeiteten Briefkastens in den benutzereigenen Briefkasten und löscht sie anschließend aus dem aktuellen, auch wenn sie nicht gelesen wurden. Sie sind in der Übersichtszeile mit *M* gekennzeichnet.

Der benutzereigene Briefkasten ist *$HOME/mbox* bzw. die in der *mailx*-Variablen *MBOX* festgelegte Datei.

next[␣nachricht]
gibt die nächste Nachricht aus, die durch *nachricht* charakterisiert ist. Für *nachricht* können Sie Merkmale wie bei einer *nachrichtenliste* angeben.

Wenn Sie z.B. als nächste Nachricht die eines bestimmten Absenders lesen wollen, geben Sie *next Absender* an.
Absender kann eine Email-Adresse oder eine lokale Benutzerkennung sein.

nachricht nicht angegeben:
mailx gibt die auf die aktuelle Nachricht folgende aus.

Ansonsten wirkt *next* wie *print*.

pipe[[␣nachrichtenliste]␣shell-kommando]
| [[␣nachrichtenliste]␣shell-kommando]
übergibt die angegebenen Nachrichten an die Standard-Eingabe von *shell-kommando*. Die Nachrichten werden in den Übersichtszeilen als gelesen (*R*) gekennzeichnet. Wenn die *mailx*-Variable *page* gesetzt ist, fügt *mailx* nach jeder Nachricht einen Formularvorschub ein (FF = CTRL L = X'0C').

Kein Argument angegeben:
mailx entnimmt den Kommandonamen der *mailx*-Variablen *cmd* und übergibt die aktuelle Nachricht. Wenn *cmd* nicht gesetzt ist, wird das Kommando *pipe* ignoriert.

preserve[␣nachrichtenliste]
hält die angegebenen Nachrichten im Briefkasten (siehe *hold*).

print[␣nachrichtenliste]
type[␣nachrichtenliste]
gibt die angegebenen Nachrichten auf die Standard-Ausgabe aus.

Die Nachrichten werden in den Übersichtszeilen als gelesen gekennzeichnet (*R*). Sie werden beim Schließen des Briefkastens in den benutzereigenen Briefkasten geschrieben und anschließend aus dem aktuellen gelöscht. Der benutzereigene Briefkasten ist *$HOME/mbox* bzw. die in der Variablen *MBOX* festgelegte Datei.
Wenn die *mailx*-Variable *crt* gesetzt ist, übergibt *mailx* Ausgaben, die mehr Zeilen haben als dort festgelegt, an das POSIX-Kommando *more*. Mit der Variablen *PAGER* können Sie ein anderes POSIX-Kommando als *more* festlegen.

Print[␣nachrichtenliste]
Type[␣nachrichtenliste]
wirkt wie *print*, gibt jedoch immer den ganzen Nachrichtenkopf aus. Das heißt, *Print* unterdrückt die Wirkung von *discard* bzw. *ignore*.

quit
beendet *mailx*. Der aktuell bearbeitete Briefkasten wird geschlossen.

Wurde der Standard-Briefkasten bearbeitet, gilt folgendes:
- Gelesene Nachrichten (*O*) und mit *mbox* bearbeitete Nachrichten (*M*) werden in den benutzereigenen Briefkasten geschrieben und anschließend gelöscht. Der benutzereigene Briefkasten ist *$HOME/mbox* bzw. die in der *mailx*-Variablen *MBOX* festgelegte Datei.
- Ungelesene Nachrichten (*U*) und mit *hold* oder *preserve* bearbeitete Nachrichten (*H*) bleiben im Standard-Briefkasten.
- Gesicherte Nachrichten (*S*) werden aus dem Briefkasten gelöscht, falls die Variable *keepsave* nicht gesetzt ist.

Siehe auch *mailx*-Kommando *exit* und *xit*.

reply[␣nachricht]
respond[␣nachricht]
beantwortet die angegebene Nachricht.

mailx geht in den Sendemodus und nimmt als Empfänger
- den Absender der angegebenen Nachricht, d.h. der Eintrag im From-Feld wird zum Eintrag in der To-Liste
- die weiteren Empfänger der Nachricht, d.h. die Einträge im To-Feld werden in die To-Liste übernommen, die Einträge im Cc-Feld in die Cc-Liste

Wenn Sie die Texteingabe beendet haben, sendet *mailx* die Nachricht ab.

Im Unterschied zu *followup* legt *reply* nicht automatisch eine Protokolldatei Ihrer Antwort an. Nur wenn die *mailx*-Variable *record* gesetzt ist, schreibt *mailx* die Antwort in die dort angegebene Datei. Die Datei wird erweitert, falls sie existiert.

Die Kommandos *reply* und *respond* dürfen nicht in einer Kommandodatei stehen.

Reply[␣nachrichtenliste]
Respond[␣nachrichtenliste]
beantwortet die erste in der Nachrichtenliste angegebene Nachricht.

mailx geht in den Sendemodus und sendet die Antwort an alle Absender aus *nachrichtenliste*.

Wenn Sie die Texteingabe beenden, sendet *mailx* die Nachricht ab.

Im Unterschied zu *Followup* legt *Reply* nicht automatisch eine Protokolldatei Ihrer Antwort an. Nur wenn die *mailx*-Variable *record* gesetzt ist, schreibt *mailx* die Antwort in die dort angegebene Datei. Die Datei wird erweitert, falls sie existiert.

Die Kommandos *Reply* und *Respond* dürfen nicht in einer Kommandodatei stehen.

retain[␣feld]
gibt nur die angegebenen Felder des Nachrichtenkopfes aus. Alle übrigen Felder werden unterdrückt. *retain* gibt die Felder auch dann aus, wenn sie in der Liste der zu unterdrückenden Felder stehen, d h. *retain* hebt die Wirkung von *discard* bzw. *ignore* auf.

feld nicht angegeben:
retain gibt die aktuelle Liste der beizubehaltenden Felder aus, falls vorhanden.

save[[␣nachrichtenliste]␣datei]
schreibt die angegebenen Nachrichten in die Datei *datei*. Die Datei wird erweitert, falls sie existiert.

Die Nachrichten werden als gesichert gekennzeichnet (*S*). Das heißt, sie werden aus dem Standard-Briefkasten gelöscht, sobald Sie *mailx* beenden, es sei denn, Sie haben die *mailx*-Variable *keepsave* gesetzt.

Kein Argument angegeben:
mailx schreibt die aktuelle Nachricht ans Ende der Datei *$HOME/mbox*.

Save[␣nachrichtenliste]
schreibt die angegebenen Nachrichten in eine Datei im aktuellen Dateiverzeichnis, deren Name gleich dem Namen des Absenders der ersten Nachricht aus *nachrichtenliste* gesetzt wird (From-Eintrag, Netz-Pfade werden abgetrennt). Die Datei wird erweitert, falls sie existiert.

Die Nachrichten werden als gesichert gekennzeichnet (*S*). Das heißt, sie werden aus dem Standard-Briefkasten gelöscht, sobald Sie *mailx* beenden, es sei denn, Sie haben die *mailx*-Variable *keepsave* gesetzt.

set[␣name[=wert]]
setzt die Variable *name*.

name — Name einer *mailx*-Variablen oder einer frei definierten Variable.

wert — Beliebige Zeichenkette oder ein numerischer Wert. Die Angabe \n innerhalb von *wert* wird als Neue-Zeile-Zeichen interpretiert, \t als Tabulatorzeichen.

wert nicht angegeben:
name wird mit der leeren Zeichenkette belegt.

Kein Argument angegeben:
mailx gibt alle gesetzten Variablen mit ihren Werten aus. Der Wert ist dabei in Anführungszeichen eingeschlossen.

Den Wert einer Umgebungsvariablen können Sie nicht verändern. Wenn Sie jedoch eine gleichlautende Variable definieren, gilt für *mailx* (außer für das *mailx*-Kommando *echo*) solange deren Wert, bis Sie ihn wieder zurücksetzen.

Mit dem Tilde-Kommando *~i variable* können Sie den Wert von *variable* in den Nachrichtentext einfügen.

Mit *unset* können Sie Variablen zurücksetzen.

shell
ruft standardmäßig den durch die Umgebungsvariable *SHELL* definierten Kommandointerpreter auf. Falls *SHELL* nicht gesetzt ist, wird */bin/sh* aufgerufen.

Das Kommando *shell* darf nicht in einer Kommandodatei stehen.

size[␣nachrichtenliste]
gibt die Größe der angegebenen Nachrichten auf die Standard-Ausgabe aus in der Form *Nachrichtennummer: Zeichenanzahl*.

source␣datei
liest die angegebene Datei als Kommandodatei und führt die darin enthaltenen *mailx*-Kommandos aus. Anschließend kehrt *mailx* in den Dialog zurück (siehe Abschnitt *mailx-Kommando- und Startdateien*).

top[␣nachrichtenliste]
gibt die ersten 5 Zeilen des Nachrichtenkopfes für jede angegebene Nachricht auf die Standard-Ausgabe aus. Die Anzahl der ausgegebenen Zeilen können Sie mit der *mailx*-Variablen *toplines* verändern.

touch[␣nachrichtenliste]
bewirkt, dass die angegebenen Nachrichten als gelesen behandelt werden, d.h. sie werden beim Schließen des Briefkastens in den benutzereigenen Briefkasten geschrieben und anschließend aus dem aktuellen gelöscht. Der benutzereigene Briefkasten ist *$HOME/mbox* bzw. die in der *mailx*-Variablen *MBOX* festgelegte Datei.

Dies gilt nicht für Nachrichten, die mit *save* oder *Save* gesichert wurden.

Mit *touch* lässt sich die Wirkung von *hold* aufheben und umgekehrt.

type[␣nachrichtenliste]
gibt die angegebenen Nachrichten auf die Standard-Ausgabe aus (siehe *print*).

Type[␣nachrichtenliste]
wirkt wie *print*, gibt jedoch immer den ganzen Nachrichtenkopf aus (siehe *Print*).

unalias[␣alias-name]...
löscht den angegebenen Alias-Namen.

undelete[␣nachrichtenliste]
holt die angegebenen Nachrichten zurück, wenn sie in der aktuellen Sitzung gelöscht wurden. Die Nachrichten werden als gelesen gekennzeichnet (*R*).

Wenn die *mailx*-Variable *autoprint* gesetzt ist, wird die letzte der zurückgeholten Nachrichten ausgegeben.

nachrichtenliste nicht angegeben:
Es wird die erste gelöschte Nachricht nach der aktuellen Nachricht zurückgeholt, falls vorhanden. Andernfalls wird die letzte gelöschte Nachricht vor der aktuellen Nachricht zurückgeholt.

undiscard[␣feld]...
unignore[␣feld]...
löscht die angegebenen Felder des Nachrichtenkopfes aus der Liste der zu ignorierenden Felder.

feld nicht angegeben:
Die gesamte Liste der zu ignorierenden Felder wird gelöscht.

unset[␣name...]
set[␣noname...]
setzt die angegebenen Variablen zurück.

Wenn Sie eine Variable löschen, deren Name gleichlautend mit einer Umgebungsvariablen ist, können Sie anschließend wieder auf den Wert dieser Umgebungsvariablen zugreifen.

version
gibt die aktuelle Versionsnummer von *mailx* aus.

visual[␣nachrichtenliste]
ruft den mit der *mailx*-Variablen *VISUAL* eingestellten Editor auf (standardmäßig *vi*) und lädt die angegebenen Nachrichten.

Nach Beenden der Editorsitzung liegt die bearbeitete Nachricht im Briefkasten vor.

Der Text wird in einer temporären Datei bearbeitet. Der Dateiname ist */tmp/Rz$$* (*$$* ist die Prozessnummer des *mailx*-Prozesses).

Das Kommando *visual* darf nicht in einer Kommandodatei stehen.

write[␣nachrichtenliste]␣datei
schreibt die angegebenen Nachrichten in die Datei *datei*. Die Datei wird erweitert, falls sie existiert.

write lässt den Nachrichtenkopf und die letzte Leerzeile weg.

Die Nachrichten werden in den Übersichtszeilen als gesichert (*S*) gekennzeichnet. Sie werden aus dem Briefkasten gelöscht, sobald Sie *mailx* beenden, es sei denn, Sie haben die *mailx*-Variable *keepsave* gesetzt.

xit beendet *mailx*. Der aktuell bearbeitete Briefkasten bleibt beim Schließen unverändert (siehe *exit*).

z[+|–] zeigt die nächste (*z*+) bzw. vorhergehende Seite (*z*-) der Übersichtszeilen an. Die Anzahl von Zeilen einer Seite entnimmt *mailx* der *mailx*-Variablen *screen*. Ist *screen* nicht gesetzt, so gibt *mailx* 20 Zeilen aus.

+|– nicht angegeben:
wie *z*+.

Arbeitsweise im Lesemodus

mailx führt während der Systemeinleitung folgende Schritte in der angegebenen Reihenfolge durch:

1. Alle Variablen auf ihren Standardwert setzen
2. Kommandozeilen-Optionen ausführen. Dabei können auch Standardwerte überschrieben werden.
3. Die Variablen DEAD, EDITOR, MBOX, LISTER, PAGER, SHELL und VISUAL importieren, falls sie in der aktuellen Umgebung definiert wurden. Dabei werden deren Standardwerte überschrieben.
4. Dann arbeitet *mailx* die systemweite Startdatei ab, sofern nicht die Option *-n* gesetzt ist. Dadurch werden die internen *mailx*-Variablen und Aliase versorgt.
5. *mailx* arbeitet dann die benutzereigene Startdatei *$HOME/.mailrc* ab, die in der Umgebungsvariablen *MAILRC* festgelegt ist.

Wenn keine Nachrichten vorliegen, meldet *mailx*:

```
No mail for benutzerkennung
```

Wenn Nachrichten vorliegen, meldet sich *mailx* mit einer Meldungszeile, einer Übersicht über die im Briefkasten vorhandenen Nachrichten und dem *mailx*-Prompt *?*. Sie können nun *mailx*-Kommandos eingeben. Mit einem der Kommandos *?, help* oder *list* können Sie sich alle verfügbaren Kommandos listen lassen.

Übersichtszeilen

Pro vorhandener Nachricht gibt *mailx* nach dem Aufruf oder einem der *mailx*-Kommandos *from, headers* oder *z* eine Übersichtszeile aus. Eine Übersichtszeile besteht aus maximal 9 durch Leerzeichen getrennten Feldern, z.B.:

```
N  1    hadea         Mon Sep 21 13:05   10/164      Nicowerfel
```

Die Einträge bedeuten folgendes:

`N`	`B`earbeitungsstatus (siehe nächster Abschnitt)
`1`	`N`achrichtennummer. Nachrichten werden bei jedem *mailx*-Aufruf neu durchnummeriert. Die älteste Nachricht erhält die Nummer 1.
`hadea`	Absender
`Mon Sep 21`	Eingangsdatum
`13:05`	Eingangszeit
`10/164`	Nachrichtengröße in Zeilen/Zeichen
`Nicowerfel`	Titel (die ersten 25 Zeichen des Subject-Eintrages)

Bearbeitungsstatus

Der Bearbeitungsstatus ist der Eintrag im ersten Feld einer Übersichtszeile. Für diesen Status kann angegeben sein:

O (old) Die Nachricht wurde bei einem vorangegangenen *mailx*-Aufruf gelesen. Sie wird in *$HOME/mbox* gesichert, wenn Sie *mailx* oder den Standard-Briefkasten verlassen.

U (unread) Die Nachricht steht schon seit dem letzten *mailx*-Aufruf im Briefkasten, d.h. sie ist nicht neu dazugekommen, wurde aber noch nicht gelesen. Sie wird im aktuellen Briefkasten gehalten, wenn Sie *mailx* mit *quit* verlassen.

R (read) Die Nachricht wurde gelesen. Sie wird in *$HOME/mbox* gesichert, wenn Sie *mailx* oder den Standard-Briefkasten verlassen.

N (new) Die Nachricht ist seit dem letzten *mailx*-Aufruf oder seit dem letzten Briefkastenwechsel neu eingetroffen. Sie wird im aktuellen Briefkasten gehalten, wenn Sie *mailx* mit *quit* verlassen.

M (mbox) Die Nachricht wurde mit *mbox* gesichert.

H (hold) Die Nachricht wurde durch eines der Kommandos *hold* oder *preserve* gekennzeichnet. Sie bleibt im Standard-Briefkasten, wenn Sie ihn schließen.

S (save) Die Nachricht wurde mit einem der Kommandos *save*, *Save* oder *write* gesichert. Sie wird aus dem Standard-Briefkasten gelöscht, wenn Sie ihn schließen.

>c Diese Nachricht ist die aktuelle. Auf diese Nachricht beziehen sich *mailx*-Kommandos, wenn Sie für das Argument *nachrichtenliste* nichts angeben.
Anstelle des Zeichens *c* steht eines der genannten Statuszeichen.

Benutzereigener Briefkasten

Nachrichten werden in den benutzereigenen Briefkasten geschrieben, wenn Sie
– sie gelesen, aber nicht gelöscht oder explizit gesichert haben
– sie mit den Kommandos *mbox* oder *touch* bearbeitet haben
– mit einem der Kommandos *file* oder *folder* vom Standard-Briefkasten in einen anderen gewechselt haben
– *mailx* mit *quit* beendet haben (es sei denn, die Variable *hold* ist gesetzt (dann verbleiben die Nachrichten im Standard-Briefkasten).

Der benutzereigene Briefkasten ist *$HOME/mbox* oder die mit der *mailx*-Variablen *MBOX* festgelegte Datei. Die Datei wird erweitert, falls sie bereits existiert. Wenn Sie *mailx* mit der Option *-f* aufrufen, können Sie diese Datei ebenso mit *mailx*-Kommandos bearbeiten wie den Standard-Briefkasten.

Format 2 **Sendemodus**

mailx[␣option]...␣empfänger␣...

Keine Option angegeben
mailx verhält sich wie im Abschnitt *Arbeitsweise im Sendemodus* beschrieben.

option

-F (F - file) *mailx* protokolliert alle gesendeten Nachrichten in einer Datei. Die Datei trägt den Namen des ersten angegebenen Empfängers. Sie wird in ihrem HOME-Dateiverzeichnis angelegt und kann mit *mailx* wie ein Briefkasten bearbeitet werden.

-F nicht angegeben:
mailx sucht den Namen der Protokolldatei in der *mailx*-Variablen *record*. Wenn diese nicht gesetzt ist, wird nichts protokolliert.

-i (i - ignore) *mailx* ignoriert das Signal SIGINT (siehe *mailx-Variable ignore*).

-n *mailx* durchläuft nicht die systemweite Startdatei */etc/mail/mailx.rc* (siehe Abschnitt *mailx-Kommando- und Startdateien*).

-s␣subject
(s - subject) *mailx* trägt *subject* in das Feld *Subject:* im Nachrichtenkopf ein. Damit können Sie der Nachricht einen Titel geben.

subject
Eine beliebige Zeichenkette. Wenn sie Leer- oder Sonderzeichen enthält, müssen Sie sie in Anführungszeichen setzen.

empfänger
Ein oder mehrere Empfänger. *empfänger* kann sein:
- Email-Adresse (wenn MAIL der interNet Services im Einsatz ist)
- Benutzerkennung im lokalen System
- Alias-Namensgruppe (siehe *mailx*-Lesekommando *alias*)
- Pipe-Symbol mit anschließendem Shell-Kommando

Beginnt *empfänger* mit einem Pipe-Symbol (|), dann wird der Rest der Zeile als Shell-Kommando interpretiert, zu dem die Nachricht durch eine Pipe gesendet wird.

mailx-Kommandos im Sendemodus (Tilde-Kommandos)

Eingabeformat

mailx-Kommandos im Sendemodus haben bis auf das vorangestellte Escape-Zeichen (Fluchtsymbol) das gleiche Format wie die Kommandos im Lesemodus:

```
[kommando][␣nachrichtenliste][␣argument]...
```

~ Escape-Zeichen Tilde. Mit der *mailx*-Variablen *escape* können Sie das Zeichen umdefinieren. Es darf jedoch nicht in ein Zeichen umdefiniert werden, welches ein Kommando beschreibt (z.B. ? oder !)

kommando nachrichtenliste argument
Beschreibung siehe oben, Abschnitt *mailx-Kommandos im Lesemodus, Eingabeformat*.

Funktionale Übersicht

In diesem Abschnitt erhalten Sie eine Übersicht über alle *mailx*-Kommandos im Sendemodus, sortiert nach ihren wichtigsten Funktionen. Dabei kann es vorkommen, dass einige Kommandos mehrmals aufgeführt werden. Im Anschluss an diese Übersicht werden diese Kommandos in alphabetischer Reihenfolge beschrieben.

Hilfsinformationen ausgeben

~?	Übersicht über die Tilde-Kommandos ausgeben
~p	Bisher eingegebenen Text anzeigen

Texteingabe beenden, abbrechen

~.	Texteingabe beenden und absenden
~x	Texteingabe abbrechen, nicht absenden
~q	Texteingabe abbrechen, nicht absenden und Text sichern

Variablenwerte, alte Nachrichten, Inhalt von Dateien einfügen

~a	Wert der Variablen *sign* einfügen
~A	Wert der Variablen *Sign* einfügen
~i	Wert einer *mailx*- oder Umgebungsvariablen einfügen
~d	Den Inhalt von *$HOME/dead.letter* einfügen
~f	Empfangene Nachrichten einfügen
~F	Empfangene Nachrichten einfügen
~m	Empfangene Nachrichten einfügen
~M	Empfangene Nachrichten einfügen
~r	Inhalt der angegebenen Datei einfügen
~<	Inhalt der angegebenen Datei einfügen

Kommandointerpreter aufrufen, Shell-Kommando ausführen

~!	Shell-Escape
~<!	Shell-Kommando ausführen und die Ausgabe in den Text einfügen
~|	Den Text an ein Shell-Kommando übergeben und durch dessen Ausgabe ersetzen

Text anzeigen

~p	Den bisher eingegebenen Text anzeigen

Text editieren

~e	Text mit einem Editor bearbeiten (Standard: *ed*)
~v	Text mit einem Editor bearbeiten (Standard: *vi*)

Empfängerkreis ändern

~b	Namen zur Bcc-Liste hinzufügen
~c	Namen zur Cc-Liste hinzufügen
~t	Namen zur To-Liste hinzufügen
~h	To-, Subject-, Cc- und Bcc-Angaben ändern

Nachrichtenkopf bearbeiten

~c	Namen zur Cc-Liste hinzufügen
~t	Namen zur To-Liste hinzufügen
~h	To-, Subject-, Cc- und Bcc-Angaben ändern
~s	Inhalt des Subject-Feldes ersetzen

Text protokollieren

~w	Bisher eingegebenen Text ohne Nachrichtenkopf in eine Datei schreiben
~q	Texteingabe abbrechen, nicht absenden und Text sichern

Sendekommandos, die den mailx-Aufruf im Lesemodus voraussetzen

~_	mailx-Kommando ausführen
~:	mailx-Kommando ausführen
~f	Empfangene Nachrichten einfügen
~F	Empfangene Nachrichten einfügen
~m	Empfangene Nachrichten einfügen
~M	Empfangene Nachrichten einfügen

mailx-Lesekommandos ausführen

~_	mailx-Lesekommando ausführen
~:	mailx-Lesekommando ausführen

Alphabetische Beschreibung

mailx-Kommandos im Sendemodus (Tilde-Kommandos) müssen ab der ersten Spalte mit dem Escape-Zeichen Tilde ~ als erstes Zeichen eingegeben werden. Dieses Zeichen, auch Fluchtsymbol genannt, können Sie mit der *mailx*-Variable *escape* umdefinieren.

Tilde-Kommandos dürfen nicht in einer Kommandodatei stehen.

~!shell-kommando
führt *shell-kommando* aus. Standardmäßig wird der durch die Umgebungsvariable *SHELL* definierte Kommandointerpreter aufgerufen und der angegebene Kommandoaufruf übergeben. Falls *SHELL* nicht gesetzt ist, wird */bin/sh* aufgerufen.

~. beendet die Texteingabe und sendet die Nachricht ab.

Wenn Sie mit *rlogin* an einem fernen Rechner arbeiten, wird dieses Tilde-Kommando als Kommando zum Verbindungsabbau interpretiert, d.h. die Sitzung am fernen Rechner wird sofort abgebrochen. Dies können Sie mit einer der folgenden Maßnahmen verhindern:
- die Tilde mit der *mailx*-Variablen *escape* umdefinieren
- die *mailx*-Variable *dot* setzen und dann die Texteingabe nur mit einem Punkt beenden
- die Texteingabe mit der Taste END beenden

~:mailx-lesekommando
~_mailx-lesekommando
führt das angegebene *mailx*-Lesekommando aus.

Sie müssen *mailx* im Lesemodus aufgerufen haben (und sind z.B. durch das *mailx*-Kommando *mail* in den Sendemodus gelangt). Sonst führt *mailx* nur solche Kommandos aus, die nichts mit dem Bearbeiten eines Briefkastens zu tun haben, wie z.B. *set* oder *exit*.

Der Unterstrich im zweiten Format muss mit angegeben werden.

~? gibt eine Übersicht über die Tilde-Kommandos aus.

~a (a - autograph) fügt den Wert der *mailx*-Variablen *sign* in den Text ein.

~A (A - autograph) fügt den Wert der *mailx*-Variablen *Sign* in den Text ein.
Damit können Sie z.B. eine weitere Briefsignatur definieren.

~b␣name...
(b - blind carbon copy) fügt einen oder mehrere Namen zur Bcc-Liste hinzu. Die Bcc-Liste (verdeckter Verteiler) enthält die Namen weiterer Empfänger der Nachricht. Diese Namen werden nicht in den Nachrichtenkopf eingefügt.

~c name...
(c - carbon copy) fügt einen oder mehrere Namen zur Cc-Liste hinzu. Die Cc-Liste (offener Verteiler) enthält die Namen weiterer Empfänger der Nachricht. Diese Namen werden in den Nachrichtenkopf zur Information eingefügt (Cc-Eintrag).

~d (d - dead.letter) fügt den Inhalt der Datei *$HOME/dead.letter* in den Text ein. Diese Datei enthält Nachrichten, die *mailx* nicht absenden konnte oder deren Eingabe Sie mit *~q* abgebrochen haben.

~e (e - editor ed) ruft den mit der *mailx*-Variablen *EDITOR* eingestellten Editor auf (standardmäßig *ed*) und lädt den bisher eingegebenen Text. Nach Beenden der Editorsitzung kann der bearbeitete Text wie vorher weitergeschrieben werden.

~f[nachrichtenliste]
(f - file) fügt die angegebenen Nachrichten unverändert in den Text ein.

Das Kommando wird nur ausgeführt, wenn *mailx* im Lesemodus (Format 1) aufgerufen wurde.

~F[nachrichtenliste]
wirkt wie ~f, fügt jedoch immer den ganzen Nachrichtenkopf ein. *discard*, *ignore* und *retain* werden nicht berücksichtigt.

~h (h - header) fordert nacheinander folgende Angaben an:

To:	Empfänger
Subject:	Titel
Cc:	Cc-Liste (Carbon Copy). Das sind weitere Empfänger der Nachricht; die Namen der Cc-Liste erscheinen im Cc-Feld des Nachrichtenkopfes (offener Verteiler).
Bcc:	Bcc-Liste (Blind carbon copy). Wie Cc; die Namen erscheinen jedoch nicht im Nachrichtenkopf (verdeckter Verteiler).

Bereits vorhandene Angaben werden angezeigt. Sie können Sie verändern, als hätten Sie sie eben eingegeben.

~i variable
(i - insert) fügt den Wert von *variable* in den Text ein. *variable* kann eine *mailx*-Variable oder eine Umgebungsvariable sein.

~m[nachrichtenliste]
(m - move) fügt die angegebenen Nachrichten in den Text ein. Vor jeder Zeile wird der Wert der Variablen *indentprefix* eingefügt.

Das Kommando wird nur ausgeführt, wenn *mailx* im Lesemodus (Format 1) aufgerufen wurde.

~M[␣nachrichtenliste]
wirkt wie ~m, fügt jedoch immer den ganzen Nachrichtenkopf ein. *discard*, *ignore* und *retain* werden nicht berücksichtigt.

~p (p - print) zeigt den bisher eingegebenen Text am Bildschirm an.

~q (q - quit) bricht die Texteingabe ab. Der bisher eingegebene Text wird nicht abgeschickt, sondern in die Datei *$HOME/dead.letter* geschrieben. *~q* wirkt wie die Taste DEL, lässt sich aber nicht mit der Variablen *ignore* unterdrücken.

~r␣datei
~r␣**!**shell-kommando
~<␣datei
~<␣**!**shell-kommando
(r - read) fügt den Inhalt von *datei* oder die Ausgabe von *shell-kommando* in den Text ein.

~s␣zeichenkette...
(s - subject) ersetzt den Inhalt des Subject-Feldes (Titel) im Nachrichtenkopf durch den Inhalt der Zeichenkette. Mehrere, durch Leerzeichen voneinander getrennte Zeichenketten müssen *nicht* in Anführungszeichen stehen.

~t␣empfänger...
(t - to) fügt die Namen eines oder mehrerer Empfänger hinzu (To-Liste). Mehrere Namen sind durch Leerzeichen zu trennen.

~v (v - editor vi) ruft den mit der *mailx*-Variablen *VISUAL* eingestellten Editor auf (standardmäßig *vi*) und lädt den bisher eingegebenen Text. Nach Beenden der Editorsitzung kann der bearbeitete Text wie vorher weitergeschrieben werden.

~w␣datei
(w - write) schreibt den bisher eingegebenen Text in die angegebene Datei. Der Kopf wird nicht mitgeschrieben. Falls die angegebene Datei noch nicht existiert, wird sie angelegt, andernfalls wird sie fortgeschrieben.

~x (x - xit) bricht die Texteingabe ab. Der bisher eingegebene Text wird nicht abgeschickt und nicht gesichert.

~!␣shell-kommando
übergibt den bisher eingegebenen Text an die Standard-Eingabe von *shell-Kommandos*. Ist der Endestatus dieses Kommandos 0, dann wird der bisher eingegebene Text durch die Standard-Ausgabe des Kommandos ersetzt.

Standardmäßig wird der durch die Umgebungsvariable *SHELL* definierte Kommandointerpreter aufgerufen und der angegebene Kommandoaufruf übergeben. Falls *SHELL* nicht gesetzt ist, wird */bin/sh* aufgerufen.

Arbeitsweise im Sendemodus

mailx arbeitet nach dem Aufruf zunächst die Startdateien ab, in denen Sie z.B. *mailx*-Variablen initialisieren können (siehe *mailx-Kommando- und Startdateien*). Wenn Sie beim *mailx*-Aufruf keinen Nachrichtentitel angeben (Option *-s*), gibt *mailx* anschließend aus:

```
Subject:
```

und erwartet die Eingabe des Titels der Nachricht. Dies ist eine Zeile mit maximal 1024 Zeichen Text, die *mailx* im Subject-Feld des Nachrichtenkopfes einträgt. Ist der eingegebene Text zu lang, so wird die Fehlermeldung `mail: ERROR signal 10` ausgegeben und die Nachricht nicht abgeschickt.

Anschließend befindet sich *mailx* im Sendemodus und Sie können den Nachrichtentext eingeben. Während der Texteingabe sind die alle *mailx*-Tilde-Kommandos erlaubt. Sie müssen ab Spalte 1 eingegeben werden. Sobald Sie ein Tilde-Kommando mit ↵ abgeschickt haben, gibt *mailx* das gesamte eingegebene Kommando nochmals in der gleichen Zeile aus und nach Kommando-Beendigung die Zeichenkette `(continue)`. Wenn Sie eines der Kommandos eingeben, mit denen Sie anderen Text in Ihren Nachrichtentext einfügen können, z.B. ~a, gibt *mailx* diesen Text nicht am Bildschirm aus. Den bisher eingegebenen sowie eingefügten Text können Sie sich mit *~p* ausgeben lassen.

Den Eingabetext speichert *mailx* in einer temporären Datei im Dateiverzeichnis */tmp*.

Das Kommando ~. oder die Taste END beendet die Texteingabe.

Tilde-Kommandos im Lesemodus

Einige der Tilde-Kommandos setzen für ihre volle Funktionalität voraus, dass Sie *mailx* im Lesemodus (Format 1) aufgerufen und von dort vorübergehend in den Sendemodus gewechselt haben. Es sind dies: ~_ und *~:* (mailx-Kommando ausführen) und *~f* und *~m* (empfangene Nachrichten einfügen).

Die Kommandos im Lesemodus, mit denen Sie vorübergehend in den Sendemodus wechseln können, um eine Nachricht zu senden oder zu beantworten, sind *followup, Followup, mail, Mail, reply, Reply, respond* und *Respond*.

mailx-Kommando- und Startdateien

Kommandodateien

Kommandodateien sind Dateien mit *mailx*-Kommandos. Jedes *mailx*-Kommando muss in einer eigenen Zeile stehen. Kommandodateien können Sie während der *mailx*-Sitzung mit dem Kommando *source* ausführen oder Sie verwenden sie als Startdateien (siehe unten).

Unzulässige Kommandos sind alle Tilde-Kommandos und *!, edit, followup, Followup, mail, Mail, reply, Reply, respond, Respond, shell, visual.*

Die Kommandos *copy, Copy, hold* und *preserve* sind zwar zulässig, jedoch werden darauffolgende Kommandos, die mit einer Nachrichtenliste arbeiten, nicht mehr ausgeführt.

Wenn in einer Kommandodatei ein Fehler auftritt, ignoriert *mailx* alle folgenden Kommandos in dieser Datei.
Zu einem Fehler führt z.B. auch, wenn in einer Nachrichtenliste auf eine nicht vorhandene Nachricht Bezug genommen wird (siehe *Beispiel 2*).

Startdateien

Startdateien sind Kommandodateien, die *mailx* nach dem Aufruf abarbeitet, sofern Sie nicht *mailx* im Lesemodus mit den Optionen *-e* oder *-n* aufgerufen haben.

mailx arbeitet erst die systemweite Startdatei */etc/mail/mailx.rc* ab, dann die benutzereigene Startdatei *$HOME/.mailrc*, soweit diese vorhanden sind.
Den Pfadnamen der benutzereigenen Startdatei können Sie mit der Umgebungsvariablen *MAILRC* neu festlegen.

Die mailx-Kommandos in den Startdateien dürfen *nachrichtenliste* nicht verwenden, da diese Information zum Zeitpunkt der Abarbeitung der Startdateien noch nicht zur Verfügung steht.

Variablen

mailx verwendet Umgebungsvariablen, *mailx*-Variablen und frei definierte Variablen.

Variablen können Sie importieren, während der *mailx*-Sitzung mit dem Kommando *set* setzen bzw. verändern und mit *unset* zurücksetzen.

mailx-Variablen, d.h. alle nur aus Kleinbuchstaben bestehenden Variablen, können Sie nur innerhalb von *mailx* setzen (z.B. in Startdateien); Werte von gleichnamigen Shell-Variablen werden für diese Variablen nicht übernommen.

Wenn Sie eine importierte Variable mit *set* verändern, so gilt der geänderte Wert so lange, bis Sie ihn erneut ändern, zurücksetzen oder die *mailx*-Sitzung beenden.

Das *mailx*-Kommando *echo* greift immer auf den originalen Wert einer importierten Variable zu.

Standardmäßig gesetzt sind *asksub*, *header* und *save*.

Von den Variablen, denen ein Wert zugewiesen wird, haben folgende Variablen folgende Standardwerte:

- *DEAD=$HOME/dead.letter*
- *EDITOR=ed*
- *escape=~*
- *MBOX=$HOME/mbox*
- *LISTER=ls*
- *PAGER=more*
- *prompt=?*
- *screen=20*
- *SHELL=/bin/sh*
- *toplines=5*
- *VISUAL=vi*

Die Angaben für *Relevante Kommandos* in der folgenden Beschreibung beziehen sich auf diejenigen *mailx*-Kommandos, die durch das Setzen oder Zurücksetzen der Variablen insbesondere betroffen sind.

mailx-Variablen

allnet *mailx* behandelt alle Netz-Pfadnamen als gleich, die auf denselben Benutzernamen enden. Als Netz-Pfadnamen gelten Adressen der Form *...rechner!rechner!benutzerkennung* (siehe auch Variable *metoo*).

Standard-Wert: Die Variable ist nicht gesetzt.

append Nachrichten, die in den benutzereigenen Briefkasten (standardmäßig *$HOME/mbox*) geschrieben werden, werden an das Dateiende angehängt.

Relevante Kommandos: *copy, Copy, file, folder, mbox, next, print, Print, type, Type, quit, touch*

Standard-Wert: Die Variable ist nicht gesetzt.

ask
asksub *mailx* fordert beim Aufruf die Eingabe für das Subject-Feld an (siehe auch Option *-s*).

Relevante Kommandos: *~h, ~s*

Standard-Wert: Die Variable ist gesetzt.

askbcc Nach Eingabe des Subject-Feldes fordert *mailx* die Eingabe der Bcc-Liste.
Relevante Kommandos: *~c, ~h*

Standard-Wert: Die Variable ist nicht gesetzt.

askcc Nach Eingabe des Subject-Feldes fordert *mailx* die Eingabe der Cc-Liste.

Relevante Kommandos: *~c, ~h*

Standard-Wert: Die Variable ist nicht gesetzt.

autoprint Nach dem Kommando *delete* wird die nächste Nachricht und nach dem Kommando *undelete* die zurückgeholte Nachricht ausgegeben.

Standard-Wert: Die Variable ist nicht gesetzt.

bang Ein ! innerhalb von *shell-kommando* in *!shell-kommando* oder *~!shell-kommando* wird durch das zuletzt auf diese Weise aufgerufene *shell-kommando* ersetzt.

Standard-Wert: Die Variable ist nicht gesetzt.

cmd=shell-kommando
mailx führt *shell-kommando* beim Kommando *pipe* bzw. | aus, falls dort kein Kommando angegeben ist.

Standard-Wert: keiner.

crt=anzahl
: Wenn eine Nachrichtenausgabe mehr als *anzahl* Zeilen enthält, übergibt *mailx* sie an das Kommando, das in der Variablen *PAGER* festgelegt ist (Standard-Wert *PAGER=more*).

  Relevante Kommandos: *dp, dt, next, print, Print, type, Type, ~p*

  Standard-Wert: Die Variable ist nicht gesetzt.

debug
: *mailx* gibt Meldungen zur Fehlerdiagnose aus. Wenn Sie diese Variable setzen, werden Nachrichten nicht abgeschickt.

  Standard-Wert: Die Variable ist nicht gesetzt.

dot
: Ein Punkt in der ersten Spalte einer eigenen Zeile beendet die Texteingabe anstelle des Kommandos ~. .

  Standard-Wert: Die Variable ist nicht gesetzt.

escape=c
: Das Escape-Zeichen Tilde ~ bei den Tilde-Kommandos wird durch das Zeichen *c* ersetzt. *c* darf kein Zeichen sein, das ein Kommando beschreibt (z.B. ? oder !).

  Standard-Wert: ~

flipr
: Die Wirkung der Kommandos *reply*, *respond* und *Reply*, *Respond* wird vertauscht.

  Relevante Kommandos: *reply, respond, Reply, Respond*

  Standard-Wert: Die Variable ist nicht gesetzt.

folder=dateiverzeichnis
: Wenn *folder* zusammen mit der Variablen *outfolder* gesetzt ist, werden die Antworttexte der Kommandos *followup* und *Followup* in *dateiverzeichnis* und nicht im aktuellen Dateiverzeichnis protokolliert. Beginnt *dateiverzeichnis* nicht mit Schrägstrich, so setzt *mailx* den Namen des Verzeichnisses auf *$HOME/dateiverzeichnis*.

  Sie können den Namen einer solchen Protokolldatei bei allen *mailx*-Kommandos, die Dateinamen erwarten, auch in der Form *+dateiname* angeben. *mailx* expandiert dann den Namen mit *dateiverzeichnis*.

  Standard-Wert: Die Variable ist nicht gesetzt.

header
: *mailx* gibt nach dem Aufruf die erste Seite mit den Übersichtszeilen aus sowie die aktuelle *mailx*-Version und die Anzahl der Nachrichten.

  Standard-Wert: Die Variable ist gesetzt.

hold Gelesene Nachrichten verbleiben im Standard-Briefkasten und werden nicht in den benutzereigenen Briefkasten gesichert (siehe auch Variable *MBOX*).

Relevante Kommandos: *copy, hold, mbox, next, preserve, print, Print, quit, touch, type, Type*

Standard-Wert: Die Variable ist nicht gesetzt.

ignore Das Signal SIGINT soll bei der Texteingabe ignoriert werden.

Relevante Kommandos: *followup, Followup, mail, Mail, reply, Reply, respond, Respond, ~q* (DEL)

Standard-Wert: Die Variable ist nicht gesetzt.

ignoreeof Dateiende (EOF, Taste END) soll bei der Texteingabe ignoriert werden (siehe auch Variable *dot*).

Relevante Kommandos: *followup, Followup, mail, Mail, reply, Reply, respond, Respond, ~.*

Standard-Wert: Die Variable ist nicht gesetzt.

indentprefix=zeichenkette
Beim Einfügen einer Nachricht in einen Text beginnt jede Zeile der Nachricht mit *zeichenkette.*

Relevante Kommandos: *~m, ~M*

Standard-Wert: Tabulatorzeichen.

keep Wenn der Briefkasten leer ist, soll er nicht gelöscht werden.

Standard-Wert: Die Variable ist nicht gesetzt.

keepsave Dateien, die in den Übersichtszeilen als gesichert (*S*) gekennzeichnet sind, sollen nicht aus dem Standard-Briefkasten gelöscht werden.

Relevante Kommandos: *save, Save, write*

Standard-Wert: Die Variable ist nicht gesetzt.

metoo Wenn die eigene Benutzerkennung in der Empfängerliste (To-Liste) erscheint, soll sie nicht daraus gestrichen werden.

Relevante Kommandos: *alias, alternates, followup, group, reply, respond, ~h*

Standard-Wert: Die Variable ist nicht gesetzt.

outfolder Wenn *outfolder* zusammen mit der Variablen *folder* gesetzt ist, werden Antwortdateien der Kommandos *followup* und *Followup* in dem Dateiverzeichnis abgelegt, das durch *folder* definiert wurde (siehe auch Variable *record*).

Wenn nur *outfolder* (oder nur *folder*) gesetzt ist, werden diese Dateien im aktuellen Dateiverzeichnis abgelegt.

Standard-Wert: Die Variable ist nicht gesetzt.

page *mailx* fügt beim Kommando *pipe* bzw. | nach jeder Nachricht ein Zeichen Formularvorschub (FF = CTRL L = X'0C') ein.

Standard-Wert: Die Variable ist nicht gesetzt.

prompt=zeichenkette
Setzt das Bereitzeichen für *mailx*-Kommandos im Lesemodus auf *zeichenkette*.

Standard-Wert: *?*

quiet Unterdrückt die Ausgabe der Versions-Zeile beim *mailx*-Aufruf.

Standard-Wert: Die Variable ist nicht gesetzt.

record=datei In der Datei *datei* protokolliert *mailx* alle abgehenden Nachrichten. Die Datei wird erweitert, falls sie existiert.

Relevante Kommandos: *mail, Mail, reply, Reply, respond, Respond, ~.*

Standard-Wert: Die Variable ist nicht gesetzt.

save *mailx* sichert Nachrichten, die z.B. wegen eines Fehlers nicht abgeschickt werden können oder für die die Texteingabe abgebrochen wurde. Gesichert wird in die Datei, die in der Variablen *DEAD* festgelegt ist.

Relevante Kommandos: *followup, Followup, mail, Mail, reply, Reply, respond, Respond, ~d, ~q* (DEL)

Standard-Wert: Die Variable ist gesetzt.

screen=anzahl
mailx gibt *anzahl* Übersichtszeilen aus.

Relevante Kommandos: *header*

Standard-Wert: abhängig von dem durch *TERM* definierten Terminaltyp, meistens 20

sendwait *mailx* kehrt nach einem Sende-Kommando erst in den Lesemodus zurück, wenn die Nachricht abgeschickt wurde.

Relevante Kommandos: *followup, Followup, mail, Mail, reply, Reply, respond, Respond*

Standard-Wert: Die Variable ist nicht gesetzt.

showto Bei Nachrichten, deren Absender Sie selbst sind, wird bei der Ausgabe der Übersichtszeilen der erste Empfänger genannt, anstelle Ihrer Benutzerkennung. Wenn es mehrere Empfänger gibt, so nimmt *mailx* den ersten Eintrag aus der Empfängerliste (To-Liste).

Relevante Kommandos: *from, headers, z+, z-*

Standard-Wert: Die Variable ist nicht gesetzt.

sign=zeichenfolge

sign ist vorgesehen für eine Briefsignatur, die in eine Nachricht eingefügt wird.
Relevante Kommandos: *~a, ~i*

Standard-Wert: Die Variable ist nicht gesetzt.

Sign=zeichenfolge

Sign ist vorgesehen für eine (zusätzliche) Briefsignatur, die in eine Nachricht eingefügt wird.

Relevante Kommandos: *~A, ~i*

Standard-Wert: Die Variable ist nicht gesetzt.

toplines=anzahl

mailx gibt *anzahl* Zeilen aus dem Nachrichtenkopf aus.

Relevante Kommandos: *top*

Standard-Wert: 5

Fehler *mailx* gibt Fehlermeldungen aus, die selbsterklärend sind.

Datei *\/etc/mail/mailx.rc*
Systemweite Startdatei

$HOME/.mailrc
Benutzereigene Startdatei

/var/mail/$USER
Standard-Briefkasten. Darin sucht *mailx* ankommende Nachrichten.

$HOME/mbox
Benutzereigener Briefkasten. Dorthin sichert *mailx* Nachrichten, die gelesen wurden.

$HOME/dead.letter
In dieser Datei sichert *mailx* Nachrichten, die z.B. wegen eines Fehlers nicht abgeschickt werden können oder für die die Texteingabe mit DEL abgebrochen wurde. Die Datei wird dabei jedesmal überschrieben, falls sie existiert.

./<username>
Dateien im aktuellen Dateiverzeichnis, deren Name gleich einer Benutzerkennung lautet, legt *mailx* bei folgenden Kommandos an:
Copy, followup, Followup, Save.

Anstelle des aktuellen Dateiverzeichnisses können Sie auch ein anderes Dateiverzeichnis wählen (siehe *mailx*-Variablen *folder* und *outfolder*).

*/tmp/R[emsxz]**
Temporäre Dateien

/tmp/Rz$$
Temporäre Datei, die von den Kommandos *edit*, *visual*, *~e* und *~v* benutzt wird. *$$* ist die Prozessnummer des *mailx*-Prozesses.

*/usr/share/lib/mailx/mailx.help**
Hilfedateien

Variable **Umgebungsvariablen**

DEAD=datei In die Datei *datei* sichert *mailx* Nachrichten, die z.B. wegen eines Fehlers nicht abgeschickt werden können oder für die die Texteingabe abgebrochen wurde. Die Datei wird erweitert, falls sie bereits existiert.

Relevante Kommandos: *followup, Followup, mail, Mail, reply, Reply, respond, Respond, ~d, ~q* (DEL)

Standard-Wert: *$HOME/dead.letter*

EDITOR=shell-kommando
Nachrichtentexte können während der *mailx*-Sitzung mit dem Editor *shell-kommando* bearbeitet werden (siehe auch Variable *VISUAL*).

Relevante Kommandos: *edit, ~e*

Standard-Wert: *ed*

HOME=dateiverzeichnis
HOME-Dateiverzeichnis.

Darin sucht *mailx* die Dateien *dead.letter, mbox, .mailrc* und die Protokolldateien (siehe Kommandos *followup*, *Followup*, und *~f*), bzw. legt sie dort an.

LISTER=shell-kommando
mailx benutzt *shell-kommando* zum Auflisten der Dateien des in *folder* genannten Dateiverzeichnisses.

Relevante Kommandos: *folders*

Standard-Wert: *ls*

MAILRC=datei
datei definiert den Namen der benutzereigenen Startdatei (Standard: *$HOME/.mailrc*, siehe *mailx-Kommando- und Startdateien*).

MBOX=datei *datei* bezeichnet den benutzereigenen Briefkasten. Dorthin werden gelesene Nachrichten geschrieben, bevor sie *mailx* aus dem Standard-Briefkasten entfernt. Die Datei wird erweitert (siehe auch Variable *hold*).

Relevante Kommandos: *copy, hold, mbox, next, preserve, print, Print, quit, save, touch, type, Type*

Standard-Wert: *$HOME/mbox*

PAGER=shell-kommando
mailx übergibt an *shell-kommando* Ausgaben, die länger sind, als die in der Variablen *crt* festgelegte Anzahl von Zeilen.

Relevante Kommandos: *dp, dt, next, print, Print, type, Type, ~p*

Standard-Wert: *more*

SHELL=shell-kommando
shell-kommando legt den Kommando-Interpreter fest, den *mailx* benutzt, um POSIX-Kommandos auszuführen.

Relevante Kommandos: *!, shell*

Standard-Wert: */bin/sh*

TERM Enthält Angaben über den Terminaltyp. Die Variable *TERM* wird ausgewertet, wenn die *mailx*-Variable *screen* nicht gesetzt ist.

USER Der Variablen *USER* entnimmt *mailx* den Benutzernamen, um z.B. den Standard-Briefkasten zu finden.

VISUAL=shell-kommando
Nachrichtentexte können während der *mailx*-Sitzung mit dem Editor *shell-kommando* bearbeitet werden.

Relevante Kommandos: *visual, ~v*

Standard-Wert: *vi*

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *mailx*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE beeinflusst die Sortierreihenfolge.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_TIME Legt das Format der Datums- und Zeitangaben fest, die von *mailx* geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Beispielsitzung mit Beantwortung einer Nachricht

Benutzer FELIX hat die Meldung *you have mail* erhalten und ruft nun *mailx* ohne Option auf. Er erhält einige Meldungs- und Übersichtszeilen, lässt sich alle Nachrichten hintereinander ausgeben (z.B. mit ↵) und beantwortet schließlich die vierte Nachricht. Sein Anworttext soll im aktuellen Dateiverzeichnis protokolliert werden (*followup* bzw. *fo*). Während des Sendemodus verändert er den von *mailx* automatisch erstellten Subject-Eintrag (~*s*) und fügt am Ende des Textes seine zuvor mit der Variablen *sign* vereinbarte Briefsignatur ein (~*a*). Bevor er den Brief absendet (~.), lässt er sich seine Antwort nochmals anzeigen (~*p*) und verlässt anschließend *mailx* mit *xit* bzw. *x*, um alle Nachrichten im Briefkasten zu erhalten.

```
mailx
mailx version 4.0 Type ? for help
"var/mail/FELIX": 4 messages 4 new
>N 1   HELMUT     Fri Sep  6   9:21    13/373      oldenburg
 N 2   BERTL      Fri Sep  6  12:00    13/365      sysadm
 N 3   PETER      Mon Sep 16  10:01     9/232      qed
 N 4   BIENE      Tue Sep 17  16:43    21/593      Projekt S
 ↵
 :
 fo BIENE
 To: <BIENE>
 Subject: Re: Projekt S

 ~s Abschlussbericht zu P S~s Abschlussbericht zu P S
 .
 :
 ~a~a

 ~p~p
 --------
 Message contains:
 To: <BIENE>
 Subject: Abschlussbericht zu P S

 Gruess Dich, Biene!
 Danke fuer Dein Protokoll. Aber ich brauche auch den Abschlussbericht,
 am besten vorgestern und am liebsten auf Papier. Wir beide, mein Tee
 und ich, warten auf Dich.
 Ciao,
 (-: felix :-)
 (continue)
 ~.

 ? x
 Held 4 messages in /var/mail/FELIX
 $
```

Beispiel 2 Beispiel für eine Startdatei

Es sollen Variablen gesetzt und, falls *mailx* im Lesemodus aufgerufen wird, alle Nachrichten von *winni* ausgedruckt werden (mit *lp*).

```
# Variablen fuer Bearbeitung
set page crt=24 cmd=lp VISUAL
set sign="\n\tL. van Pelt\n\tManualredaktion SINIX\n\tMuenchen-Perlach"

# Verteiler: Systemverwalter im Netz
alias sys root\muenchen root\nuernberg root\frankfurt

# Bestimmte Post ausdrucken
if r
pipe winni lp
from winni
endif
```

Beachten Sie, dass *mailx* die Prozedur abbricht, wenn ein Kommando nicht ausgeführt werden kann. Dies könnte hier z.B. beim Kommando *pipe* der Fall, wenn keine Nachrichten von *winni* vorliegen. *from* (und eventuelle weitere Kommandos) führt *mailx* dann nicht aus.

Siehe auch *ed, ls, more, sh, vi*

make Gruppen von Dateien verwalten (maintain, update and regenerate groups of programs)

Bei der modularen Programmierung bestehen Programme meist aus mehreren Dateien. *make* bietet eine Methode zum Aktualisieren solcher Programme.

make verwendet ein *makefile*, in dem Sie Ziele festlegen und angeben können, wie ein Ziel von anderen Zielen abhängt. Wurde eine Quelldatei geändert, erzeugt *make* das Programm neu, indem es genau die Teile neu übersetzt, die direkt oder indirekt von der geänderten Datei abhängen.

make erstellt das Ziel neu, wenn es älter ist als mindestens eine der Dateien, von denen es abhängt. *make* berücksichtigt die Abhängigkeitsbeziehungen der Dateien untereinander und ermittelt Datum und Uhrzeit der letzten Änderung einer Datei.

Das *makefile* heißt normalerweise `makefile`, `Makefile`, `s.makefile` oder `s.Makefile`. Folgt man dieser Benennungskonvention, können Sie *make* ohne Angabe von Argumenten aufrufen. *make* sucht nach dem *makefile* im aktuellen Dateiverzeichnis bzw. im Verzeichnis `SCCS` und generiert das Ziel neu, wenn mindestens eine Änderung stattgefunden hat.

Syntax **make**[␣**-einpqrst**][␣**-f**␣makefile]... [␣**-k**|**-S**] [makro=name]... [ziel]...

Die *make*-Optionen haben folgende Bedeutung:

-e überschreibt die Definitionen in *makefiles* durch Umgebungsvariablen.

-f␣makefile
makefile wird als Name einer Beschreibungsdatei angesehen.

-i ignoriert Fehlermeldungen von Kommandos, die in einer Kommandofolge stehen. *make* setzt die Bearbeitung auch nach Fehlern fort. *.IGNORE* im *makefile* bewirkt dasselbe.

-k bricht die Kommandofolge ab, wenn in einem Eintrag des *makefiles* ein Fehler auftritt. Die Bearbeitung wird bei allen anderen Zielen fortgesetzt, die vom fehlerhaften Ziel unabhängig sind. *-k* überschreibt vorangegangene *-S*-Optionen.

-n dient als Testhilfe. Alle Kommandos werden ausgegeben, auch die mit vorangestelltem @. Ausgeführt werden nur Kommandos mit vorangestelltem Pluszeichen +.

-p gibt eine vollständige Liste der Makrodefinitionen und der Beschreibung der Zieldateien aus.

-q gibt den Status Null zurück, wenn die Zieldatei aktuell ist. Ein Status ungleich Null gibt an, dass die Zieldatei neu erstellt werden muss. Die Zieldatei wird nicht verändert, jedoch werden Kommandos mit vorangestelltem Pluszeichen + ausgeführt.

-r ignoriert die vordefinierten Regeln.

-s unterdrückt das Setzen eines neuen Zeitstempels und die Ausgabe der ausgeführten Kommandos. `.SILENT` als Ziel in dem *makefile* bewirkt dasselbe.

-S beendet sich, wenn in einem Eintrag des *makefiles* ein Fehler auftritt. Dies ist die Standardeinstellung. *-S* überschreibt vorangegangene *-k*-Optionen.

-t gibt den Zielen des *makefiles* einen neuen Zeitstempel und diese werden als aktuell angenommen. Ausgeführt werden nur Kommandos mit vorangestelltem Pluszeichen +.

Erzeugen eines makefiles

Das mit der Option *-f* angegebene *makefile* ist eine sorgfältig strukturierte Datei mit expliziten Anweisungen zum Aktualisieren von Programmen. Die Datei enthält eine Reihe von Einträgen, die Abhängigkeiten angeben. Die erste Zeile eines Eintrags ist eine durch Leerzeichen getrennte, nicht leere Liste von Zielen, dann folgt ein :, danach eine (möglicherweise leere) Liste der erforderlichen Dateien oder Abhängigkeiten. Auf ein ; folgender Text und alle nachfolgenden Zeilen, die mit einem Tabulator beginnen, sind Shell-Kommandos, die zur Aktualisierung des Ziels ausgeführt werden sollen. Die erste nicht leere Zeile, die nicht mit einem Tabulator oder # anfängt, beginnt eine neue Abhängigkeit oder Makrodefinition. Shell-Kommandos können durch die Angabe von <Backslash><Neue-Zeile-Zeichen> über mehrere Zeilen gehen. Alles, was von *make* ausgegeben wird (mit Ausnahme des ersten Tabulators), geht direkt und unverändert an die Shell. Daher erzeugt

```
echo a\
b
```

die Ausgabe von *a b* genau wie dies auch mit der Shell erfolgen würde.

Kommandos, die über mehrere Zeilen gehen, dürfen insgesamt maximal `LINE_MAX` lang sein, wenn die Anweisung *.POSIX* verwendet wird.

Kommentare werden von # und dem Neue-Zeile-Zeichen umschlossen.

Das folgende *makefile* besagt, dass *pgm* von den beiden Dateien *a.o* und *b.o* abhängig ist, die ihrerseits von ihren jeweiligen Quelldateien (*a.c* und *b.c*) sowie von der gemeinsamen Datei `incl.h` abhängen:

```
pgm: a.o b.o
      c89 a.o b.o -o pgm
a.o: incl.h a.c
       c89 -c a.c
b.o: incl.h b.c
       c89  -c b.c
```

Ausführung eines makefiles

Kommandozeilen werden nacheinander jeweils von einer eigenen Shell ausgeführt. Mit der SHELL-Umgebungsvariablen bzw. dem SHELL-Makro kann die von *make* zur Ausführung von Kommandos verwendete Shell angegeben werden. Der Standard ist `/usr/bin/sh`.

Die folgenden Anweisungen können in einem *makefile* angegeben werden, um die Ausführung zu steuern:

.POSIX *make* unterstützt in vollem Umfang die XPG4 Beschreibung. Andernfalls weicht *make* in einigen Punkten von der Beschreibung ab, um die Ablauffähigkeit bestehender *makefiles* zu gewährleisten.

.DEFAULT Wenn eine Datei angelegt werden muss, jedoch keine ausdrücklichen Kommandos oder relevanten Standardregeln vorliegen, werden die bei *.DEFAULT* angegebenen Kommandos verwendet.

.IGNORE Für alle von diesem Ziel abhängigen Dateien werden Fehlermeldungen von Kommandos ignoriert. Ist keine Datei angegeben, entspricht die Wirkung der Option *-i*.

.PRECIOUS Von diesem Ziel abhängige Dateien werden nicht gelöscht, falls das Quit- oder Interrupt-Signal empfangen wird. Ist keine Datei angegeben, werden alle Dateien beibehalten.

.SILENT Für alle von diesem Ziel abhängigen Dateien wird die Ausgabe der ausgeführten Kommandos unterdrückt. Ist keine Datei angegeben, entspricht die Wirkung der Option *-s*.

Die Ausführung einzelner Kommandos kann durch Voranstellen der Zeichen @, - oder + gesteuert werden. Wird ein Kommando auf Standardausgabe geschrieben, werden diese Zeichen nicht mit ausgegeben.

@ die Ausgabe des Kommandos wird unterdrückt

- Fehler werden ignoriert

+ das Kommando wird immer ausgeführt, auch wenn eine der Optionen *-n*, *-q* oder *-t* gesetzt ist

Eine Zeile wird ausgegeben, wenn sie ausgeführt wird, es sei denn, die Option *-s* ist vorhanden, oder der Eintrag *.SILENT* ist für die Datei gültig oder die Anfangszeichenkette enthält ein @. Option *-n* gibt das Kommando aus, ohne es auszuführen, außer wenn der Kommandozeile ein + vorangestellt ist (dann wird die Zeile immer ausgeführt). Die Option *-t* aktualisiert das geänderte Datum einer Datei ohne Ausführen von Kommandos (außer wenn einem Kommando ein + vorangestellt ist).

make wird normalerweise (bzw. wenn die Option *-S* gesetzt ist) durch Kommandos beendet, die einen Status ungleich Null zurückgeben. Wenn die Option *-i* vorhanden ist, oder der Eintrag *.IGNORE* für die Datei gilt, oder die erste Zeichenkette des Kommandos - enthält, wird

der Endestatus ignoriert. Wenn die Option *-k* vorhanden ist, wird die Arbeit am aktuellen Eintrag verlassen, jedoch in anderen Verzweigungen, die nicht von diesem Eintrag abhängen, fortgeführt.

Die Zieldatei wird durch Unterbrechung und Verlassen gelöscht, außer der Eintrag *.PRECIOUS* ist für sie gültig.

Umgebung

Die Umgebung wird von *make* gelesen. Alle Variablen werden als Makrodefinitionen verstanden und als solche verarbeitet. Die Umgebungsvariablen werden vor jedem *makefile* und unmittelbar nach den vordefinierten Regeln verarbeitet. Daher werden Umgebungsvariablen von Makrozuweisungen in einem *makefile* überschrieben. Die Option *-e* bewirkt, dass die Umgebung die Makrozuweisungen in einem *makefile* überschreibt. Dateinamen-Zusätze und ihre entsprechenden Regeln in *makefile* überschreiben vordefinierte Regeln für diese Erweiterungen.

Die Umgebungsvariable `MAKEFLAGS` kann Makros und alle Eingabeoptionen außer *-f* und *-p* enthalten, die für die Kommandozeile gültig sind. *make* wertet die Variable vor der Kommandozeile aus. Das gleichnamige Makro `MAKEFLAGS` (und die Variable, falls sie nicht definiert ist) wird automatisch bei Aufruf von *make* mit den aktuellen Optionen und Makros belegt bzw. ergänzt und bei Aufruf der Kommandos weitergereicht. Auf diese Weise enthält das Makro `MAKEFLAGS` stets die aktuellen Definitionen. Dies erweist sich als sehr nützlich für „super-makes“. Tatsächlich wird `$(MAKE)` bei Verwendung der Option *-n* auf jeden Fall ausgeführt. Daher kann man ein *make -n* rekursiv bei einem ganzen Softwaresystem ausführen, um zu sehen, was ausgeführt worden wäre. Dies ist deshalb möglich, weil das *-n* in `MAKEFLAGS` eingetragen wird und an weitere Aufrufe von $(MAKE) weitergeleitet wird. Dies ist eine Möglichkeit der Fehlersuche in allen *makefiles* für ein Softwareprojekt, ohne tatsächlich etwas auszuführen.

Die Umgebungsvariable PROJECTDIR gibt an, in welchem Verzeichnis SCCS-Dateien gesucht werden sollen, falls sie nicht im aktuellen Verzeichnis liegen. SCCS-Dateien werden jeweils in einem Unterverzeichnis SCCS gesucht. Bei Angabe von relativen Pfadnamen wird zunächst das HOME-Verzeichnis nach Unterverzeichnissen *src* bzw. *source* durchsucht. Existieren diese Verzeichnisse nicht, wird der Pfad relativ zum aktuellen Verzeichnis gesetzt.

Include-Dateien

Wenn *include* am Anfang einer Zeile in *makefile* erscheint und danach ein Leerzeichen oder ein Tabulator folgt, wird der Rest der Zeile als Dateiname interpretiert, dessen zugehörige Datei vom aktuellen Aufruf nach Ersetzen von Makros gelesen wird.

Makro-Definitionen

Einträge der Form *Zeichenkette1 = Zeichenkette2* sind Makrodefinitionen. *Zeichenkette2* wird als Folge aller Zeichen bis zu einem Kommentarzeichen oder einem nicht entwerteten Neue-Zeile-Zeichen definiert. Bei jedem anschließenden Auftreten wird $(*Zeichenkette1*[:*Ersatz1*=[*Ersatz2*]]) durch Zeichenkette2 ersetzt. Die runden Klammern sind optional, wenn ein Makroname mit einem Zeichen verwendet wird und keine Ersetzungs-Regel vorhanden ist. Das optionale :*Ersatz1=Ersatz2* ist eine Ersetzungs-Regel. Bei Angabe dieser Regel wird jedes sich nicht überschneidende Auftreten von *Ersatz1* im angegebenen Makro durch *Ersatz2* ersetzt. Zeichenketten für diese Art von Ersetzung werden durch Leerzeichen, Tabulatorzeichen, Neue-Zeile-Zeichen und Zeilenanfänge begrenzt. Ein Beispiel für die Verwendung der Ersetzungs-Regeln wird im Abschnitt *Bibliotheken* gezeigt.

Interne Makros

Es gibt insgesamt fünf intern verwaltete Makros, die zum Schreiben von Regeln für den Aufbau von Zielen nützlich sind.

$* stellt den Stamm des Dateinamens der aktuellen abhängigen Datei dar, wobei der Zusatz gelöscht ist. Die Auswertung erfolgt nur bei Ausführung der Abhängigkeitsregeln.

$@ stellt den vollen Zielnamen des aktuellen Ziels dar. Die Auswertung erfolgt nur für ausdrücklich angegebene Abhängigkeiten.

$< wird nur bei Anwendung der Abhängigkeitsregeln oder der *.DEFAULT*-Regel ausgewertet. Es ist das Modul, das in Bezug auf das Ziel veraltet ist, d.h. der „angefertigte“ abhängige Dateiname. Auf diese Weise würde Makro *$<* in der Regel `.c.o` als .c-Datei ausgewertet werden. Ein Beispiel für die Erstellung von optimierten .o-Dateien aus .c-Dateien ist:

```
.c.o:
  c89 -c -O $*.c
```

oder:

```
.c.o:
  c89 -c -O $<
```

$? wird ausgewertet, wenn die expliziten Regeln des *makefiles* ausgewertet werden. Es ist die Liste der Vorbedingungen, die in Hinblick auf das Ziel veraltet sind.

$% wird nur ausgewertet, wenn das Ziel ein Teil einer Archivbibliothek in der Form `lib(datei.o)` ist. In diesem Fall wird *$@* als *lib* ausgewertet, und *$%* als die Bibliotheksdatei `datei.o`.

Die Makros können alternative Formen aufweisen. Wenn ein Großbuchstabe D oder F angehängt wird, ändert sich die Bedeutung in „Dateiverzeichnis-Teil“ für D und „Dateiteil“ für F. Daher bezieht sich `$(@D)` auf den Dateiverzeichnis-Teil der Zeichenkette `$@`. Ist kein Dateiverzeichnis-Teil vorhanden, wird ./ erstellt. Wenn das Makro $? mehr als einen Dateinamen enthält, expandieren die Makros `$(?D)` und `$(?F)` (bzw. `${?D}` und `${?F}`) zu einer Liste Dateiverzeichnis-Teile und Dateinamen-Teile.

Standardregeln

Bestimmte Namen, wie z. B. solche, die auf .o enden, haben ableitbare Abhängigkeitsbeziehungen wie z.B. .c, .s usw. Wenn für eine solche Datei keine Aktualisierungskommandos in *makefile* definiert sind, werden Dateien, die den Standard-Abhängigkeitsbeziehungen entsprechen gesucht und übersetzt, um das Ziel zu erzeugen. Für diesen Fall hat *make* Abhängigkeitsregeln, die durch Prüfung der Suffixe und Bestimmung einer zur Anwendung geeigneten Abhängigkeitsregel den Aufbau von Dateien aus anderen Dateien ermöglichen. Folgende Standard-Abhängigkeitsregeln existieren:

```
.c      .c~      .f     .f~      .s       .s~      .sh     .sh~    .C      .C~
.c.a    .c.o     .c~.a  .c~.c   .c~.o    .f.a     .f.o    .f~.a   .f~.f   .f~.o
.h~.h   .l.c     .l.o   .l~.c   .l~.l    .l~.o    .s.a    .s.o    .s~.a   .s~.o
.s~.s   .sh~.sh  .y.c   .y.o    .y~.c    .y~.o    .y~.y   .C.a    .C.o    .C~.a
.C~.C   .C~.o    .L.C   .L.o    .L~.C    .L~.L    .L~.o   .Y.C    .Y.o    .Y~.C
.Y~.o   .Y~.Y
```

Der Benutzer kann zu dieser Liste Regeln hinzufügen, indem diese in die Datei *makefile* geschrieben werden.

Die Ableitung von Abhängigkeitsbeziehungen kann gesteuert werden. Die Regel zur Erstellung einer Datei mit dem Suffix .o aus einer Datei mit dem Suffix .c ist als Eintrag mit .c.o: als Ziel und ohne Abhängigkeiten spezifiziert. Mit dem Ziel in Zusammenhang stehende Shell-Kommandos definieren die Regel zur Erstellung einer .o-Datei aus einer .c-Datei. Ein Ziel, das keine Schrägstriche aufweist und mit einem Punkt beginnt, wird als Regel angesehen und nicht als echtes Ziel.

Die Standardregeln für *make* sind in der Quelldatei rules.c für das Programm *make* enthalten. Sie können lokal verändert werden. Um die in *make* eingebauten Regeln auf einem beliebigen Gerät in einer für eine Neuübersetzung geeigneten Form auszugeben, wird nachstehendes Kommando verwendet:

```
make -pf - 2>/dev/null </dev/null
```

Eine Tilde in den obigen Regeln bezieht sich auf eine SCCS-Datei. Daher würde die Regel .c~.o eine SCCS-C-Quelldatei in eine Objektdatei (.o) umwandeln. Da das s. der SCCS-Dateien ein Dateiname-Präfix ist, ist es nicht mit dem Begriff von Suffix im Sinn von *make* vereinbar. Daher stellt die Tilde eine Möglichkeit zur Änderung eines Dateiverweises in einen SCCS-Dateiverweis dar.

Eine Regel mit nur einem Zusatz (z.B. .c:) ist die Definition dafür, wie man x aus x.c aufbaut. Tatsächlich ist das andere Suffix leer. Dies ist nützlich für den Aufbau von Zielen aus nur einer Quelldatei (z.B. Shell-Prozeduren, einfache C-Programme).

Zusätzliche Suffixe können als Liste bei *.SUFFIXES* definiert werden. Die Reihenfolge ist von Bedeutung; der erste mögliche Name, für den eine Datei und eine Regel vorhanden sind, wird für die Erzeugung ausgewählt. Die Standardliste lautet:

```
.SUFFIXES .o .c .c~ .y .y~ .l .l~ .s .s~ .sh .sh~ .h .h~ .f .f~
.C .C~ .Y .Y~ .L .L~
```

Das obige Kommando zur Ausgabe der internen Regeln zeigt auch diese Liste der auf der aktuellen Maschine implementierten Zusätze an. Mehrfach-Suffix-Listen sind akkumulierend; *.SUFFIXES:* ohne Abhängigkeiten löscht die Suffix-Liste.

Das bei „Erzeugen eines makefiles“ angegebene Beispiel lässt sich damit kürzer formulieren:

```
pgm: a.o b.o
      c89 a.o b.o -o pgm
a.o b.o: incl.h
```

Die Standard-Abhängigkeitsregeln verwenden gewisse Makros, um die Aufnahme von optionalen Teilen in die erzeugte Kommandofolge zu ermöglichen. Zum Beispiel werden `CFLAGS`, `LFLAGS` und `YFLAGS` für Übersetzeroptionen von *c89* [5], *lex* bzw. *yacc* verwendet.

Mit der Anweisung `.SCCS_GET` können die Standardkommandos zum Zugriff auf `SCCS`-Dateien, die nicht im aktuellen Verzeichnis liegen, geändert werden. Standardregel ist:

```
.SCCS_GET: sccs $(SCCSFLAGS) get $(SCCSGETFLAGS) $@
```

Bibliotheken

Wenn eine Ziel- oder eine Abhängigkeitsbezeichnung Klammern enthält, wird sie als Archivbibliothek angesehen, wobei die in Klammern stehende Zeichenkette auf einen Eintrag in der Bibliothek verweist.

So verweisen `lib(datei.o)` und `$(LIB)(datei.o)` auf eine Archivbibliothek, die `datei.o` enthält.
Hierbei wird davon ausgegangen, dass das Makro `LIB` vorher definiert wurde. Der Ausdruck `$(LIB)(datei1.o datei2.o)` ist nicht zulässig. Zu Archivbibliotheken gehörende Regeln weisen die Form *.XX.a* auf, wobei *XX* das Suffix der Datei ist, aus der der Archiveintrag erstellt werden soll. Leider ist es bei der gegenwärtigen Implementierung erforderlich, dass *XX* sich vom Suffix des Archiveintrags unterscheiden muss. Daher kann man nicht angeben, dass `lib(datei.o)` von `datei.o` abhängig ist.

Es folgt die Beschreibung der häufigsten Anwendung der Archivschnittstelle. Hier soll davon ausgegangen werden, dass alle Quelldateien C-Quellen sind:

```
lib:  lib(file1.o) lib(file2.o) lib(file3.o)
       @echo lib is now up-to-date
.c.a:
      $(CC) -c $(CFLAGS) $<
      $(AR) $(ARFLAGS) $@ $*.o
      rm -f $*.o
```

Tatsächlich ist diese Regel `.c.a` in *make* vordefiniert und hier überflüssig. Ein interessanteres, allerdings stärker begrenztes Beispiel für eine Archivdatei-Wartung:

```
lib:  lib(file1.o) lib(file2.o) lib(file3.o)
      $(CC) -c $(CFLAGS) $(?:.o=.c)
      $(AR) $(ARFLAGS) lib $?
      rm $?
      @echo lib is now up-to-date .c.a:;
```

In diesem Fall wird der Makro-Ersetzungsmechanismus verwendet. Die Liste *$?* ist als der Satz von Objektdateinamen definiert (innerhalb von `lib`), deren entsprechende C-Quelldateien veraltet sind. Der Makro-Ersetzungsmechanismus ersetzt `.o` durch `.c`. Leider kann eine Umwandlung in `.c` noch nicht vorgenommen werden, könnte aber in Zukunft implementiert werden. Auch ist die Abschaltung der Regel `.c.a:` zu beachten, die jede Objektdatei der Reihe nach erstellt hätte. Diese besondere Konstruktion beschleunigt die Wartung der Archivbibliothek beträchtlich. Sie wird recht umständlich, wenn die Archivbibliothek eine Mischung von Assembler- und C-Programmen enthält.

Datei `[Mm]akefile` und `s.[Mm]akefile`
/usr/bin/sh

Hinweis Einige Kommandos geben ungerechtfertigt einen Status ungleich Null zurück; zur Lösung dieses Problems verwendet man *-i* oder die Option - in der Kommandozeile.

Dateinamen mit den Zeichen = : @ können nicht verarbeitet werden.
Kommandos, die direkt von der Shell ausgeführt werden, insbesondere *cd*, sind in *make* über Neue-Zeile-Zeichen hinweg wirkungslos.
Die Syntax `lib(datei1.o datei2.o datei3.o)` ist unzulässig. Es ist nicht möglich, `lib(datei.o)` aus `datei.o` aufzubauen.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *make*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *ar, lex, yacc*
c89 [5]

man Online-Dokumentation nutzen (display system documentation)

Mit dem Kommando *man* können Sie die POSIX-Online-Dokumentation nutzen, d.h. die Syntax und Beschreibung eines POSIX-Kommandos auf die Standard-Ausgabe ausgeben lassen.

Syntax

man[␣**-x**]␣kommando ...

man -k␣ausdruck

-x Nur die Kommandosyntax wird auf die Standardausgabe geschrieben. Ist diese Option nicht angegeben, wird die gesamte Beschreibung des Kommandos ausgegeben.

-k Die POSIX-eigene Datenbasis wird durchsucht. Die Datenbasis enthält für jedes Kommando eine Kommandokurzbeschreibung. Jede Zeile, die den angegebenen Ausdruck enthält, wird ausgegeben.

kommando
: Name eines oder mehrerer POSIX-Kommandos. Ist für eines der Kommandos keine Beschreibung verfügbar, wird folgende Fehlermeldung ausgegeben:
`Manual-Eintrag nicht gefunden: /usr/share/man/De/`*`text`*`.txt.Z`
Wenn mehrere Kommandos angegeben sind und die Option *-x* nicht gesetzt ist, wird das durch die Variable *PAGER* festgelegte Ausgabekommando für jedes Kommando einzeln aufgerufen.

ausdruck
: Ausdruck gemäß der Syntax des Kommandos *grep*. Es wird beim Durchsuchen der Datenbasis nicht zwischen Groß- und Kleinschreibung unterschieden (`grep -i`).

Datei

/usr/share/man/En/.txt.Z*
/usr/share/man/De/.txt.Z*
Diese Dateien enthalten die Manual-Einträge; eine Datei pro Kommando und Sprache.

/usr/share/man/En/man.index
/usr/share/man/De/man.index
Kurzbeschreibungen der Kommandos.

Variable

PAGER
Legt das Kommando fest, das die Informationen anzeigt oder verarbeitet. Die Informationen werden an die Standardeingabe des Kommandos gesendet. Der Standardwert ist `more -d`.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *man*:

LANG Gibt die Sprache für die Ausgabe der Dokumentation und von Meldungen an. Beginnt der Inhalt der Variablen mit "De" oder "de", erfolgen die Ausgaben in Deutsch, ansonsten in Englisch.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt ihr Wert den Wert der Variablen *LANG*.

Beispiel 1 Informationen zum Kommando *mkdir* ausgeben:

```
$ man mkdir
mkdir - Dateiverzeichnis erzeugen
(make directories)
================================

Mit mkdir können Sie ein neues Dateiverzeichnis einrichten.
...
(ENDE) [RETURN]
$
```

Beispiel 2 Alle Kommandos ausgeben, die die Zeichenfolge "attrib" in ihrer Kurzbeschreibung enthalten:

```
$ man attrib
Manual-Eintrag nicht gefunden: /usr/share/man/De/attrib.txt.Z
$ man -k attrib
bs2file - Dateiattribute für BS2000-Dateien festlegen
typeset - Attribute für Shell-Variable setzen
(ENDE) [RETURN]
$
```

Beispiel 3 Kurzbeschreibung aller Kommandos auf den Standarddrucker ausgeben:

```
$ PAGER=lp man -k .
lp: request id is TSN-OV02 (TSNOV01.2011-04-27.144908-1.standard_input)
$
```

Beispiel 4 Syntax des Kommandos head ausgeben:

```
$ man -x head
+-- Syntax ------------------------------------------------------------------+
|                                                                             |
| Format 1: head[ -n nummer][ datei]                                          |
|                                                                             |
| Format 2: head[ -nummer][ datei]                                            |
|                                                                             |
+-----------------------------------------------------------------------------+
$
```

Siehe auch *cat*, *more*, *lp*

mesg Nachrichtenempfang verbieten oder erlauben (permit or deny message)

mesg regelt den Empfang von Nachrichten auf Terminals, die sich mit *rlogin* oder *telnet* an POSIX angemeldet haben . Mit *mesg* können Sie entweder abfragen, ob Ihre Datensichtstation Nachrichten empfangen kann, oder festlegen, dass andere Benutzer mit *write* Nachrichten auf Ihren Bildschirm schreiben dürfen bzw. nicht schreiben dürfen.

Syntax

mesg[␣**y**|␣**n**]

Keine Option angegeben

mesg gibt die aktuelle Einstellung für Ihren Bildschirm aus und liefert den Endestatus 0, falls der Empfang von Nachrichten erlaubt ist, sonst 1.

option

y Andere Benutzer dürfen Nachrichten an die Datensichtstation des aufrufenden Benutzers senden.

Dies entspricht der alten Option *-y*, die weiterhin unterstützt wird.

n Andere Benutzer dürfen keine Nachrichten an die Datensichtstation des aufrufenden Benutzers senden. Dieses Verbot gilt nicht für Nachrichten, die der POSIX-Verwalter mit *wall* [13] oder *write* verschickt. Diese Nachrichten werden trotzdem empfangen.

Dies entspricht der alten Option *-n*, die weiterhin unterstützt wird.

Endestatus

0 Senden von Nachrichten erlaubt

1 Senden von Nachrichten nicht erlaubt

>1 Fehler

Datei

*/dev/term/**
Bildschirmdateien

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *mesg*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung fest, die für das Format und den Inhalt der Diagnosemeldungen verwendet werden soll, die von *mesg* in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die Benutzerin *herta* ist an der Datensichtstation *tty001* angemeldet und fragt die Einstellung der Datensichtstation ab:

```
$ mesg
is n
```

Die Datensichtstation *tty001* ist für den Empfang von Nachrichten gesperrt. Sie können daher an *herta* an der Datensichtstation *tty001* keine Nachricht schicken.

```
$ write herta
Permission denied.
```

Beispiel 2 Einstellung an der Datensichtstation *tty001* ändern:

```
$ mesg y
```

Siehe auch *tty*, *write*
wall [13]

mkdir Dateiverzeichnis erzeugen (make directories)

Mit *mkdir* können Sie ein neues Dateiverzeichnis einrichten.

mkdir trägt in das neue Dateiverzeichnis folgende Verweise ein:

. (Punkt) für das Dateiverzeichnis selbst

.. (Punkt Punkt) für das übergeordnete Dateiverzeichnis

mkdir kann nur ausgeführt werden, wenn Sie in dem Dateiverzeichnis, das dem einzurichtenden Dateiverzeichnis übergeordnet ist, das Schreibrecht haben.

Syntax

mkdir[␣**-m**␣modus][␣**-p**]␣dateiverzeichnis␣...

Keine Option angegeben
: *mkdir* legt die in *dateiverzeichnis* angegebenen Dateiverzeichnisse mit den Zugriffsrechten *777* (siehe *chmod*) an. Mit dem Befehl *umask* kann die Voreinstellung der Zugriffsrechte jedoch verändert werden (siehe *umask*).

option

-m␣modus
: (m - mode) *mkdir* legt das neu anzulegende *dateiverzeichnis* mit den in *modus* angegebenen Zugriffsrechten an (siehe *chmod*).

-p (p - parent) *mkdir* legt zuerst alle nicht existierenden übergeordneten Dateiverzeichnisse an, die im Pfadnamen von *dateiverzeichnis* enthalten sind, bevor *dateiverzeichnis* selbst angelegt wird.

dateiverzeichnis
: Name des Dateiverzeichnisses, das Sie einrichten möchten. Sie können mehrere Dateiverzeichnisse angeben.

  Sie können für *dateiverzeichnis* sowohl den relativen als auch den absoluten Pfadnamen angeben.

  Das neue Dateiverzeichnis erhält die realen Benutzer- und Gruppennummern des aufrufenden Prozesses.

Endestatus

0 *mkdir* konnte alle angegebenen Dateiverzeichnisse einrichten.

≠ 0 Ein Fehler ist aufgetreten. *mkdir* gibt eine Fehlermeldung aus.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *mkdir*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Anlegen des Dateiverzeichnisses *briefe* im Dateiverzeichnis */home/sysiphus/sonstiges*:
Stellen Sie zunächst fest, in welchem Dateiverzeichnis Sie sich befinden. Die Voreinstellung der Zugriffsrechte für Dateiverzeichnisse ist auf den Wert *777* gesetzt.

```
$ pwd
/home/sysiphus
```

Sie lassen sich den Inhalt des Dateiverzeichnisses ausgeben.

```
$ ls -l
total 145
drwx------   2 SYSIPHUS   gruppe1     1560   Oct 11 15:36 bildschirme
-rw-r--r--   1 SYSIPHUS   gruppe1     5329   Nov 03 09:54 diff.rc.1
                  .
                  .
                  .
drwxr-----   3 SYSIPHUS   gruppe1     2340   Jun 11 15:35 lingua
drwx------   2 SYSIPHUS   gruppe1     3380   Oct 11 15:36 post
drwx--x--x   2 SYSIPHUS   gruppe1     2080   Nov 04 16:08 proz
drwx--x--x   2 SYSIPHUS   gruppe1     2589   Aug 03 15:08 sonstiges
```

Sie legen das neue Dateiverzeichnis an.

```
$ mkdir sonstiges/briefe
```

Prüfen Sie, ob das Dateiverzeichnis *briefe* angelegt wurde.

```
$ cd sonstiges
$ ls -l
total 5
drwxrwxrwx   2 SYSIPHUS   gruppe1      520   Jan 22 16:21  briefe
```

Siehe auch *rm, rmdir, umask*

mkfifo FIFO erstellen (make FIFO special files)

mkfifo erstellt die FIFOs, die in seiner Argumentenliste angegeben sind.

Für jedes Argument *datei* verhält sich das Kommando *mkfifo* so, als ob die Funktion *mkfifo* (siehe *mkfifo*[4]) aufgerufen wurde, und zwar mit folgenden Argumenten:

– das Argument *path* von *mkfifo*[4] wird auf *datei* gesetzt.

– das Argument *mode* von *mkfifo*[4] erhält den Wert *0666* oder den Wert von *modus*, falls die Option *-m* angegeben wurde.

Treten bei der Erzeugung einer FIFO Fehler auf, schreibt *mkfifo* eine Diagnosemeldung auf die Standard-Fehlerausgabe und bearbeitet dann ggf. weitere Argumente.

Syntax **mkfifo**[␣**-m**␣modus]␣datei...

Keine Option angegeben
mkfifo legt die bei datei angegebenen FIFOs mit den Zugriffsrechten 666 an, modifiziert durch die aktuelle Schutzbit-Maske (siehe *umask*).

-m␣modus
mkfifo legt die neue FIFO mit den in *modus* angegebenen Zugriffsrechten an (siehe *chmod*). Bei symbolischer Angabe werden dabei die Operanden + und - relativ zu einem Ausgangswert von a=rw interpretiert.

datei
Name der FIFO, die Sie einrichten möchten. Sie können mehrere FIFOs angeben.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *mkfifo*:

LANG	Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.
LC_ALL	Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.
LC_CTYPE	Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
: Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH
: Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *chmod*, *umask*
mkfifo [4]

mkfs Dateisystem erstellen (make file system)

Mit *mkfs* wird für die Gerätedatei *datei* ein ufs-Dateisystem angelegt. *mkfs* wartet 10 Sekunden, bevor das Dateisystem erstellt wird. Während dieses Intervalls kann das Kommando durch Drücken der DEL-Taste abgebrochen werden.

Syntax

mkfs[␣**-F**␣ufs][␣**-V**][␣**-m**|**-s**][␣**-q**][␣**-y**][␣–**o**␣spezifische optionen]␣datei[groesse]

Achtung!
Die Funktionalität des Kommandos *mkfs* wird bereits durch das POSIX-Installationsprogramm abgedeckt (siehe POSIX- Handbuch „Grundlagen für Anwender und Systemverwalter“ [1]). Deswegen wird es hier nicht mehr ausführlich beschrieben. Sollten Sie dennoch explizit mit dem Shell-Kommando *mkfs* arbeiten wollen, finden Sie eine Beschreibung des Kommandos und dessen Optionen im Reliant UNIX Referenzhandbuch für Systemverwalter [13].

mknod Gerätedatei anlegen (make an inode)

Das Kommando *mknod* erstellt einen Verzeichniseintrag für eine Gerätedatei.

Syntax

Format 1: mknod ␣ name[␣**b**|␣**c**]␣geräteklasse␣gerätenummer

Format 2: mknod ␣ name ␣ **p**

Format 1 **mknod** ␣ name[␣**b**|␣**c**]␣geräteklasse␣gerätenummer

Dieses Format darf nur der POSIX-Verwalter eingeben.

name
Name der zu erstellenden Gerätedatei

option

b für eine blockorientierte Gerätedatei

c für eine zeichenorientierte Gerätedatei

geräteklasse␣gerätenummer
Zahl für die Geräteklasse. Sie kann in Dezimal- oder Oktalschreibweise eingegeben werden. Bei Oktalschreibweise muss eine führende Null angegeben werden. Die Zuweisung der Geräteklassennummer ist systemspezifisch.

Beispiel für mögliche Geräteklassen (major device number):

58 Terminal-Datei (*/dev/term/*...)

59 SF-Terminal-Datei (*/dev/sf/*...)
(SYSFILE - wird bei Zugriff auf Terminals aus BS2000-Batchprozeduren verwendet)

Es ist nicht ausreichend, die TERM- und SF-Gerätedateien nur einzurichten. Um sie verwenden zu können, müssen die entsprechenden POSIX-Parameter gesetzt sein:

Datei: $TSOS.SYSSSI.POSIX-BC.*nnn* (*nnn* = Version, siehe POSIX-Handbuch „Grundlagen für Anwender und Systemverwalter“ [1])

Parameter:

NOTTY Anzahl der maximal verwendbaren Terminal-Verbindungen

NOSSTY Anzahl der maximal verwendbaren SF-Terminal-Verbindungen

Die Parameter werden nur durch erneutes Hochfahren des POSIX-Subsystems wirksam.

Format 2 **mknod**␣name␣**p**

name
Name der Gerätedatei

p Erstellen einer FIFO-Datei (benannte Pipe)

Beispiel Einrichten einer zusätzlichen Terminaldatei (*/dev/term*). Das Beispiel wird unter der POSIX-Verwalterkennung durchgeführt.

```
# cd /dev/term
/dev/term # ls -l /dev/term/511
crw-rw-rw- 1 SYSROOT  TTY   58,511 Jan 14  2008 /dev/term/511
/dev/term # mknod /dev/term/512 c 58 512
/dev/term # chmod a+w /dev/term/512
/dev/term # ls -l /dev/term/512
crw-rw-rw- 1 SYSROOT SYSROOT 58,512 Jan 27 14:14 /dev/term/512
```

Siehe auch *mknod* [4]

more Bildschirmausgabe steuern (display files on a page-by-page basis)

Das Kommando *more* erlaubt das Durchblättern von Dateien an der Datensichtstation. *more* realisiert die Ausgabe durch Hochschieben der Bildschirmzeilen.

Syntax

Format 1: more[␣**-cdefisu**][␣**-n**␣zeile][␣**-p**␣kommando][␣datei]...

Format 2: more[␣**-cdefisu**][␣**-n**␣zeile][␣**+**kommando][␣-zeilen][␣+zeilennummer] [␣+/muster][␣datei]...

Keine Option angegeben

Standardmäßig wird die Bildschirmausgabe nach jeder Seite unterbrochen. Mit den unten beschriebenen Kommandos kann die Ausgabe weiter gesteuert werden. Wenn Sie keine Optionen oder eingebaute Kommandos angeben, unterbricht *more* die Ausgabe nach jeder Bildschirmseite.
Am unteren Bildschirmrand erscheint nach der ersten Bildschirmseite das Bereitzeichen `(<dateiname>..%)`.
Drücken Sie die Leertaste ␣ (bei Blockterminals ␣↵), dann blättert *more* eine ganze Bildschirmseite weiter. Durch Drücken von nur ↵ wird der Bildschirm um eine Zeile nach oben geschoben.
Um beim Durchblättern nicht den Überblick zu verlieren, überschneiden sich die Bildschirmseiten jeweils um zwei Zeilen. Erhält *more* den Input aus einer Datei, wird der Anteil der bereits gezeigten Zeilen am Bildschirm dargestellt (. .%). Bei *more* in Verbindung mit einer Pipe | ist dies nicht der Fall.

option

-c Löscht vor Ausgabe einer neuen Seite den Bildschirm und ermöglicht damit ein schnelleres Durchblättern. *-c* wird nur ausgeführt, wenn für die Datensichtstation eine Funktion zum Löschen des Bildschirms vorhanden ist.

-d Bei Blockterminals werden die Kommentare und Warnungen unterdrückt, die auf die nicht vorhandenen Steuermöglichkeiten der Bildschirmsteuerung hinweisen.

-e Veranlasst *more*, sich beim ersten Erreichen des Dateiendes automatisch zu beenden. Standardmäßig beendet sich *more* beim zweiten Erreichen des Dateiendes.

-f Macht keinen Zeilenumbruch, wenn eine Zeile über den rechten Bildschirmrand hinausgeht. Den vom Bildschirm hardwaremäßig automatisch ausgeführten Zeilenumbruch kann *-f* nicht verhindern. Dennoch zählt *more* die Zeilen, als ob kein Umbruch stattgefunden hätte.

-i Bei Suchvorgängen wird die Groß-/Kleinschreibung ignoriert, d. h. Groß- und Kleinbuchstaben werden als identisch betrachtet. Außerdem kann nach über- oder unterstrichenem Text gesucht werden. Diese Option wird ignoriert, wenn das Suchmuster selbst Großbuchstaben enthält.

-s Die Ausgabedatei wird komprimiert, indem mehrere Leerzeichen zu einem zusammengefasst werden.

-u Die Hervorhebung von Escape-Sequenzen wird unterdrückt.

-u nicht angegeben:
Hervorhebungen, wie sie von einigen Textformatierern (wie z.B. *nroff*) produziert werden, gibt *more* je nach Datensichtstation unterschiedlich aus. Nur wenn an der Datensichtstation Hervorhebungen oder auch ein Stand-Out-Modus vorgesehen sind, werden Escape-Sequenzen, wie sie im Dateitext stehen, am Bildschirm abgebildet.

-n␣zeile
Ändert den Standardwert für das Blättern auf *zeile*. Standardwert ist jeweils ein Bildschirm.

-p␣kommando
Die Option *-p* in der Kommandozeile entspricht der Angabe von *+/kommando*, d. h. *more* führt jedesmal, wenn eine neue Datei angezeigt wird, *kommando* aus.

Dies entspricht der alten Option *+kommando*, die weiterhin unterstützt wird.

-zeilen
Mit dem Wert für *zeilen* geben Sie die Anzahl der Zeilen an, die bei der Ausgabe am Bildschirm weitergeblättert wird.

+zeilennummer
Die Ausgabe der Datei beginnt bei der mit *zeilennummer* angegebenen Zeile.

+/muster
muster ist ein regulärer Ausdruck, nach dem Sie eine Textdatei durchsuchen können. Die Bildschirmausgabe beginnt zwei Zeilen über der zu *muster* passenden Zeichenfolge.
Anders als bei Editoren darf die Zeichenfolge nicht mit einem Schrägstrich / enden, da sonst der Schrägstrich als Zeichen in der gesuchten Zeichenfolge interpretiert wird.

datei
Name der Datei, die ausgegeben werden soll. Sie können mehrere Namen angeben: vor der Ausgabe jeder Datei erscheint dann in einer Kopfzeile der Name der aktuellen Datei.

Kommandos

Wenn *more* die Ausgabe am Ende einer Bildschirmseite unterbrochen hat, können Sie die weitere Ausgabe von *more* durch folgende Kommandos steuern:

n Leertaste
more gibt *n* weitere Zeilen aus. Wenn Sie nur die Leertaste (bei Blockterminals ␣ ↵) drücken, wird eine neue Bildschirmseite ausgegeben.

n ↵
more gibt *n* weitere Zeilen aus. Wenn Sie nur ↵ drücken, wird die nächste Zeile ausgegeben.

n CTRL **d**
n**d**
more gibt *n* weitere Zeilen aus. Gleichzeitig wird die Zeilenanzahl, um die weitergeblättert werden soll, auf *n* gesetzt.

n**z** *more* setzt die Bildschirmgröße auf *n* und gibt den nächsten Bildschirm aus.

n**s** *more* überspringt bei der Ausgabe *n* Zeilen und gibt den darauf folgenden Text aus. Die Auslassung wird an der übersprungenen Stelle kommentiert.

n**f** Mit *n* geben Sie die Anzahl der Bildschirmseiten an, die ausgelassen werden sollen. An der übersprungenen Stelle wird die Auslassung der Bildschirmseiten in Zeilen angegeben.

n CTRL **b**
n**b**
n legt fest, um wieviel Bildschirmseiten zurückgeblättert werden soll.

q
Q
:q
:Q
Q in den vier aufgeführten Schreibformen beendet *more*.

= *more* gibt die aktuelle Zeilennummer aus.

v *more* ruft den Editor *vi* auf und springt in die zuletzt bearbeitete Datei. Die aktuelle Zeile der Datei ist die letzte mit *more* auf dem Bildschirm ausgegebene.

h Auf dem Bildschirm erscheint eine Übersichtstabelle über alle *more*-Kommandos mit einer kurzen Funktionsbeschreibung.

n/muster
more sucht nach dem *n*-ten Auftreten des regulären Ausdrucks *muster*. Tritt dieses Muster weniger als *n*-mal auf und liest *more* von einer Datei und nicht von einer Pipeline, bleibt die Position in der Datei unverändert. Liest *more* dagegen von einer Pipeline, so wird *more* in diesem Fall beendet.
Ist die Suche erfolgreich, wird ein neuer Bildschirm ausgegeben, der zwei Zeilen vor der Zeile beginnt, in der der reguläre Ausdruck das *n*-te Mal auftritt.

n**n** *more* sucht nach dem *n*-ten Auftreten des zuletzt eingegebenen regulären Ausdrucks.

' (Hochkomma)
more springt zu dem Punkt, an dem die letzte Suche nach einem regulären Ausdruck begann. Alle folgenden Hochkommata werden ignoriert. Falls noch keine Suchfunktion aufgerufen wurde, geht *more* an den Anfang der Datei.

!kommando

more startet eine neue Shell, die das Shell-Kommando *kommando* ausführt. Zur Formulierung dieses Kommandos können zusätzlich zwei Variablen verwendet werden: % wird zum Namen der aktuellen Datei expandiert; ! wird zum zuletzt eingegebenen Shell-Kommando expandiert; Wollen Sie ein *%* oder ein *!* in das Kommando einfügen, müssen Sie das Zeichen durch einen vorangestellten Gegenschrägstrich \ entwerten.

n**:n**

more springt zur *n*-ten Datei aller in der Kommandozeile angegebenen Dateien. Sind weniger als *n* Dateien aufgeführt, springt *more* zur letzten Datei. Wenn Sie dagegen kein *n* angeben, springt *more* zu der in der Reihenfolge nächsten Datei. Befinden Sie sich bereits in der letzten Datei, beendet sich *more*.

n**:p**

more springt zur *n*-ten vorhergehenden Datei aller in der Kommandozeile angegebenen Dateien. Sind in der Kommandozeile weniger als *n* Dateien aufgeführt, springt *more* zur ersten Datei.
Das Kommando funktioniert nur, wenn nicht von einer Pipeline gelesen wird.

:f *more* zeigt den Namen der aktuellen Datei und die aktuelle Zeilennummer an.

. (Punkt) *more* führt das letzte eingegebene Kommando nochmals aus.

Arbeitsweise

Das Kommando *more* setzt die Datensichtstation auf einen Nicht-Echo-Modus, so dass die Datei ungehindert ausgegeben wird. Deshalb werden *more*-Kommandos nicht am Bildschirm abgebildet, mit Ausnahme von regulären Ausdrücken, dem Ausrufezeichen ! und dem Schrägstrich /.
more arbeitet im cbreak-Modus, d.h., sobald eines der oben genannten Kommandos vollständig angegeben ist, wird es verarbeitet, ohne mit ↵ bestätigt worden zu sein. So beenden sich beide Kommandos automatisch am Ende der Textdatei.
Rückwärtsblättern mit *more* ist langsamer als Vorwärtsspringen; deshalb ist der Rückwärtsgang zum Durchsuchen großer Dateien weniger geeignet.
Das Kommando *more* holt sich die Daten zur Bildschirmsteuerung aus den *terminfo*-Dateien.

Datei */usr/share/lib/terminfo/**
Diese Datei enthält Daten zur Bildschirmsteuerung.

/usr/lib/more.hlp
HELP-Datei

Variable *LINES*
Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Berechnung der Spaltenzahl für die Ausgabe von *select*-Listen verwendet. *select*-Listen werden vertikal ausgegeben, bis ungefähr zwei drittel der Zeilenzahl *LINES* gefüllt sind. Wird die Option *-n* angegeben, hat sie Vorrang vor dieser Variablen.

COLUMNS
Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Definition der Breite des Editierfensters beim Editiermodus der POSIX-Shell und für die Ausgabe der *select*-Liste verwendet. Diese Variable ist sinnvoll, wenn der Zugang zur POSIX-Shell über *rlogin* erfolgt.

EDITOR
Diese Variable wird vom *more*-Kommando *v* verwendet. Diese Variable ist sinnvoll, wenn der Zugang zur POSIX-Shell über *rlogin* erfolgt.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *more*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Bestimmt in regulären Ausdrücken die Bedeutung von Zeichenbereichen, Äquivalenzklassen und Zeicheneinheiten.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Datei *test* soll ab der Stelle, an der das Wort *Beispiel* steht, ausgegeben werden. Zu lange Zeile sollen dabei nicht umgebrochen werden:

```
$ more -f +/Beispiel test ↵
```

Die Ausgabe beginnt zwei Zeilen über der Zeile mit dem Wort *Beispiel*.

Siehe auch *cat*, *ed*, *man*, *sh*

mount Dateisysteme und ferne Ressourcen einhängen (mount a filesystem)

mount (Format 2 und Format 3, siehe Seite 590) hängt ein ufs-Dateisystem in die Dateisystemhierarchie an der Pfadnamenposition *einhängepunkt* ein, die bereits vorhanden sein muss. Hat *einhängepunkt* vor der mount-Operation bereits einen Inhalt, bleibt dieser verdeckt, bis das Dateisystem wieder ausgehängt wird.

mount (Format 4 und Format 5, siehe Seite 592) hängt ein bs2fs-Dateisystem an einer bestimmten Stelle im POSIX-Dateisystem ein. Unter einem bs2fs-Dateisystem versteht man eine auswählbare Menge von Dateien im BS2000, die transparent in POSIX zur Verfügung gestellt werden, so dass auf sie mit POSIX-Mitteln (Kommandos, Programmschnittstellen) zugegriffen werden kann. Die Auswahl der Dateien erfolgt über Benutzer- und Katalogkennung sowie Wildcard-Symbole.

Außerdem kann mit *mount* (Format 1) eine Liste aller eingehängten Dateisysteme ausgegeben werden.

Zum Mounten von nfs-Dateisystemen siehe Handbuch *NFS* [9].

Syntax

Format 1: mount[␣**-v**␣|␣**-p**]

Format 2: mount[␣**-F**␣**ufs**][␣**-V**][␣**-r**][␣**-o**␣spez_optionen]{␣ressource␣|␣einhängepunkt}

Format 3: mount[␣**-F**␣**ufs**][␣**-V**][␣**-r**][␣**-o**␣spez_optionen]␣ressource␣einhängepunkt

Format 4: mount[␣**-F**␣**bs2fs**][␣**-V**][␣**-r**][␣**-o**␣spez_optionen]
{␣ressource␣|␣einhängepunkt␣}

Format 5: mount[␣**-F**␣**bs2fs**][␣**-V**][␣**-r**][␣**-o**␣spez_optionen]␣ressource␣einhängepunkt

Liste der eingehängten Dateisysteme ausgeben

Format 1

mount[␣**-v**␣|␣**-p**]

Keine Option angegeben
mount gibt eine Liste aller eingehängten Dateisysteme aus (siehe Beispiel).

option

-v gibt die Ausgaben in einer neuen Darstellung aus. Bei der neuen Ausgabe werden dateisystemtyp und Optionen zusätzlich zu den Angaben der alten Ausgabe angezeigt. Die Felder *einhängepunkt* und *ressource* sind vertauscht (siehe Beispiel).

-p Gibt die Liste der eingehängten Dateisysteme im Format von */etc/vfstab* aus (siehe Beispiel).

ufs-Dateisysteme einhängen

Format 2 **mount**[␣**-F**␣**ufs**][␣**-V**][␣**-r**][␣**-o**␣spez_optionen]{␣ressource␣|␣einhängepunkt}

Format 3 **mount**[␣**-F**␣**ufs**][␣**-V**][␣**-r**][␣**-o**␣spez_optionen]␣ressource␣einhängepunkt

Format 2 und Format 3 werden gemeinsam beschrieben, da sie sich nur durch die (optionale) Angabe von *ressource* und *einhängepunkt* unterscheiden.

Format 2 kann nur verwendet werden, wenn in der Datei */etc/vfstab* bereits ein Eintrag für das entsprechende Dateisystem existiert. Aus diesem wird dann die fehlende Angabe für *ressource* oder *einhängepunkt* ergänzt.

Formate 2 und 3 dürfen nur vom POSIX-Verwalter eingegeben werden.

Keine Option angegeben
: *mount* gibt eine Liste aller eingehängten Dateisysteme aus.

-F␣ufs
: Gibt *ufs* als Dateisystem-Typ an.

-V Gibt die gesamte Kommandozeile auf dem Bildschirm aus, führt das Kommando jedoch nicht aus. Die Kommandozeile wird mit den vom Benutzer eingegebenen Optionen und Argumenten sowie aus */etc/vfstab* abgeleiteten Werte erstellt. Diese Option sollte verwendet werden, um die Kommandozeile einer allgemeinen Prüfung und einer Gültigkeitsprüfung zu unterziehen.

-**r** Einhängen des Dateisystems mit Lesezugriff.

-**o** Angeben *ufs*-dateisystemspezifischer Optionen. Werden mehrere Optionen angegeben, werden sie durch Komma getrennt. Werden ungültige Optionen angegeben, wird eine Warnung ausgegeben, und die ungültigen Optionen werden ignoriert. Folgende Optionen sind verfügbar:

f	Täuscht einen */etc/mntab*-Eintrag vor, hängt aber keine Dateisysteme ein. Parameter werden nicht überprüft.
n	Einhängen des Dateisystems, ohne einen Eintrag in */etc/mnttab* zu stellen.
journal	Falls es nicht nicht existiert, wird beim Einhängen des Dateisystems ein Journal angelegt. Dieses Journal dient dem schnellen Wiederanlauf nach einem Systemabsturz. Bei Angabe der Option *ro* ist die Angabe *journal* wirkungslos, d. h. bei Read-Only-Zugriffen gibt es kein Journaling.
rw | **ro**	Lese-/Schreib- oder Lesezugriff. Standardwert ist *rw*.

nosuid Standardmäßig wird das Dateisystem so eingehängt, dass das s-Bit für Benutzer gesetzt wird. Durch Angeben von *nosuid* wird der Standardwert außer Kraft gesetzt, und das Dateisystem wird ohne Setzen des s-Bits für Benutzer eingehängt.

remount Wird zusammen mit *rw* verwendet. Ein mit Lesezugriff eingehängtes Dateisystem kann mit Lese-/Schreibzugriff neu eingehängt werden. Diese Option schlägt fehl, wenn das Dateisystem aktuell nicht oder mit *rw* eingehängt ist.

bs2fscontainer

Vereinbart das einzuhängende Dateisystem als bs2fs-Container, d.h. als Dateisystem, das temporär Dateien von bs2fs-Dateisystemen aufnimmt.

Diese Option darf nur für ein einziges *ufs*-Dateisystem angegeben werden. Jedes weitere *mount*-Kommando mit dieser Option wird abgewiesen.

Die Optionen *-r*, *-o ro*, *-o journal* und *-o* remount dürfen nicht gemeinsam mit der Option *bs2fscontainer* angegeben werden.

Beim Einsatz des POSIX-Installationsprogramm kann diese Option über die Optionszeile eingegeben werden.

Diese Option kann nur für ein Dateisystem angegeben werden, das beim Einrichten mit dem POSIX-Installationsprogramm als bs2fs-Container gekennzeichnet wurde. Bei der *append*-Funktion eines neu zu erstellenden oder zu überschreibenden ufs-Dateisystems muss dazu in der Optionszeile die Option *bs2fscontainer* angegeben werden.
Ist dies nicht der Fall, so wird das Einhängen mit Fehler abgebrochen und das Dateisystem bleibt samt Inhalt erhalten.

ressource

Gibt das Dateisystem an, das eingehängt werden soll.

einhängepunkt

Gibt an, wo die Ressource lokal eingehängt werden soll. Es muss ein absoluter Pfadname angegeben werden. Handelt es sich bei *einhängepunkt* um einen symbolischen Verweis, wird das Dateisystem in das Verzeichnis, auf das sich der symbolische Verweis bezieht, eingehängt und nicht zusätzlich zu dem symbolischen Verweis.

bs2fs-Dateisysteme einhängen

Format 4 **mount**[␣**-F**␣**bs2fs**][␣**-V**][␣**-r**][␣**-o**␣spez_optionen] {␣ressource␣|␣einhängepunkt␣}

Format 5 **mount**[␣**-F**␣**bs2fs**][␣**-V**][␣**-r**][␣**-o**␣spez_optionen]␣ressource␣einhängepunkt

Format 4 und Format 5 werden gemeinsam beschrieben, da sie sich nur durch die (optionale) Angabe von *ressource* und *einhängepunkt* unterscheiden.

Format 4 kann nur verwendet werden, wenn in der Datei */etc/vfstab* bereits ein Eintrag für das entsprechende Dateisystem existiert. Aus diesem wird dann die fehlende Angabe für *ressource* oder *einhängepunkt* ergänzt (siehe auch Hinweis auf Seite 595).

Formate 4 und 5 dürfen nur vom POSIX-Verwalter eingegeben werden.

Voraussetzung für die Eingabe eines *mount*-Kommandos des Formats 4 oder 5 ist, dass bereits ein bs2fs-Container eingehängt ist.

Keine Option angegeben
: *mount* gibt eine Liste aller eingehängten Dateisysteme aus.

-F␣bs2fs
: Gibt *bs2fs* als Dateisystem-Typ an.

-V Gibt die gesamte Kommandozeile auf dem Bildschirm aus, führt das Kommando jedoch nicht aus. Die Kommandozeile wird mit den vom Benutzer eingegebenen Optionen und Argumenten sowie aus */etc/vfstab* abgeleiteten Werte erstellt. Diese Option sollte verwendet werden, um die Kommandozeile einer allgemeinen Prüfung und einer Gültigkeitsprüfung zu unterziehen.

-r Einhängen des Dateisystems mit Lesezugriff.

-o Angeben *bs2fs*-dateisystemspezifischer Optionen. Werden mehrere Optionen angegeben, werden sie durch Komma getrennt. Werden ungültige Optionen angegeben, wird eine Warnung ausgegeben, und die ungültigen Optionen werden ignoriert. Folgende Optionen sind verfügbar:

rw | **ro** Lese-/Schreib- oder Lesezugriff. Standardwert ist *rw*.

nosuid Das Dateisystem wird ohne Setzen des s-Bits für Benutzer eingehängt. Für bs2fs-Dateisysteme ist diese Option standardmäßig aktiviert und kann nicht deaktiviert werden.

remount Wird zusammen mit *rw* verwendet. Ein mit Lesezugriff eingehängtes Dateisystem kann mit Lese-/Schreibzugriff neu eingehängt werden. Diese Option schlägt fehl, wenn das Dateisystem aktuell nicht oder mit *rw* eingehängt ist.

ftyp={**text**|**binary**|**textbin**}

Die Wirkung dieser Option entspricht der des Kommandos *ftyp* beim Kopieren von Dateien mit dem *bs2cp*-Kommando. Die Option legt fest, ob BS2000-SAM-Dateien und textartige PLAM-Bibliothekselemente (Elementtyp ungleich L) in POSIX als Text- oder Binärdateien interpretiert werden. PAM-Dateien werden stets als Binärdateien, ISAM-Dateien werden stets als Textdateien interpretiert.

Diese Option sollte nur einmal angegeben werden. Bei mehrfacher Angabe gilt die Angabe mit der höchsten Priorität, wobei *ftyp=textbin* die höchste, *ftyp=text* die nächsthöchste und *ftyp=binary* die niedrigste Priorität besitzt.

Default ist *ftyp=text*.

ftyp=text

SAM-Dateien und textartige Bibliothekselemente werden als Textdateien interpretiert. Beim Schreiben in eine bs2fs-Datei werden Zeilenende-Zeichen (X'15') in einen Satzwechsel umgewandelt und Tabulatorzeichen (X'05') in die entsprechende Anzahl Leerzeichen.

ftyp=binary

SAM-Dateien und textartige Bibliothekselemente werden als Binärdateien interpretiert. Es erfolgt eine 1:1-Übertragung ohne Interpretation und Umsetzung von Daten (Satzwechsel/Zeilenende-Zeichen, Tabulator/Leerzeichen, usw.).

ftyp=textbin

SAM-Dateien und textartige Bibliothekselemente werden als binäre Textdateien interpretiert. Beim Schreiben in eine bs2fs-Datei werden nur Zeilenende-Zeichen (X'15') in einen Satzwechsel umgewandelt. Tabulatoren (X'05') werden nicht in Leerzeichen umgesetzt.

ressource

Legt fest, welche BS2000-Dateien gemounted werden sollen. Die Option ist in folgender Syntax anzugeben:

:cat**:$**user**.**filename-with-wild

Sie kann in Groß- oder Kleinschreibung oder auch in gemischter Form angegeben werden. Sonderzeichen der POSIX-Shell wie '$' oder '*' müssen explizit entwertet werden.

cat Katalog-Kennung

user BS2000-Benutzerkennung

filename-with-wild

BS2000-Dateiname mit Wildcard-Symbolen

*

Ersetzt eine beliebige, auch leere Zeichenfolge.

/
: Ersetzt genau ein beliebiges Zeichen.

Punkt am Ende
: Teilqualifizierte Angabe eines Namens.
Entspricht implizit der Zeichenfolge „./*“, d.h. nach dem Punkt folgt mindestens ein beliebiges Zeichen.

<s_x:s_y>
: Ersetzt eine Zeichenfolge, für die gilt:
  – sie ist mindestens so lang wie die kürzeste Zeichenfolge (s_x oder s_y)
  – sie ist höchstens so lang wie die längste Zeichenfolge (s_x oder s_y)
  – sie liegt in der alphabetischen Sortierung zwischen s_x und s_y; Zahlen werden hinter Buchstaben sortiert (A…Z, 0…9)
  – s_x darf auch die leere Zeichenfolge sein, die in der alphabetischen Sortierung an erster Stelle steht
  – s_y darf auch die leere Zeichenfolge sein, die an dieser Stelle für die Zeichenfolge mit der höchst möglichen Codierung steht (enthält nur die Zeichen X’FF’)

<s_1,…>
: Ersetzt alle Zeichenfolgen, auf die eine der mit s angegebenen Zeichenkombinationen zutrifft. s kann auch die leere Zeichenfolge sein. Jede Zeichenfolge s kann auch eine Bereichsangabe „s_x:s_y“ sein.

-s
: Ersetzt alle Zeichenfolgen, die der angegebenen Zeichenfolge s nicht entsprechen. Das Minuszeichen darf nur am Beginn der Zeichenfolge stehen.

Bei der mit *ressource* beschriebenen Dateimenge kann es sich sowohl um bereits existierende als auch um neu zu erstellende Dateien handeln. Beim Neuerstellen einer Datei muss der gewünschte Dateiname dem Wildcardmuster des entsprechenden *mount*-Kommandos entsprechen.

einhängepunkt
: Gibt an, wo die Ressource lokal eingehängt werden soll. Es muss ein absoluter Pfadname angegeben werden. Handelt es sich bei *einhängepunkt* um einen symbolischen Verweis, wird das Dateisystem in das Verzeichnis, auf das sich der symbolische Verweis bezieht, eingehängt und nicht zusätzlich zu dem symbolischen Verweis.

Hinweis

Wenn für das betreffende Dateisystem ein Eintrag in der Datei */etc/vfstab* existiert, kann eine der Optionen *ressource* oder *einhängepunkt* entfallen (Format 2 bzw. 4). Beim Einsatz von bs2fs-Dateisystem ist in diesem Fall Folgendes zu beachten:

– Ist die Option *einhängepunkt* angegeben, dann kann eindeutig ein Eintrag in */etc/vfstab* identifiziert werden, und das entsprechende Dateisystem wird eingehängt.

– Ist nur die Option *ressource* angegeben so können für ein bs2fs-Dateisystem mehrere passende Einträge in der Datei */etc/vfstab* enthalten sein, da dieses Dateisystem parallel an mehreren Stellen eingehängt sein kann. In diesem Falle wird nur das **erste** in der Datei */etc/vfstab* eingetragene bs2fs-Dateisystem eingehängt.

  Als passende Einträge werden nur Einträge mit identischer Wildcard-Zeichenfolge erkannt. Einträge mit unterschiedlicher Wildcard-Zeichenfolge werden auch dann nicht berücksichtigt, wenn sie dieselbe Menge von Dateien definieren.

Datei

/etc/mnttab
Tabelle der eingehängten Dateisysteme

/etc/dfs/fstypes
standardmäßiger verteilter Dateisystem-Typ

/etc/vfstab
Tabelle der automatisch eingehängten Dateisysteme

/etc/mnttab Tabelle der eingehängten Dateisysteme

Die Datei */etc/mnttab* enthält Angaben über alle am lokalen Rechner eingehängten Dateisysteme. Diese Datei enthält Informationen, die durch das Kommando *mount* erzeugt werden.

Jede Zeile enthält folgende Informationen, die durch eine beliebige Anzahl von Leerzeichen und/oder Tabulatoren getrennt sind:

Aufbau

```
ressource     mountp    fstyp    spec-options    time
```

ressource
: absoluter Pfadname des eingehängten Dateisystems bzw. bei bs2fs-Dateisystemen eingehängte BS2000-Dateien in Wildcard-Syntax.

  Für bs2fs-Dateisysteme weicht der Eintrag folgendermaßen von der Angabe beim *mount*-Kommando ab:
  – Er ist vollständig in Großbuchstaben umgewandelt
  – Entwertete Zeichen werden ohne den zugehörigen Entwerter dargestellt

Beispiel:

```
mount -F bs2fs -o ftyp=text :v70a:\$bach.sys\* /home/bach/bs2.1
```

erzeugt folgenden Eintrag in */etc/mnttab*:

```
:V70A:$BACH.SYS* /home/bach/bs2.1 bs2fs ftyp=text,suid,rw,noquota ...
```

mountp
: absoluter Pfadname des Einhängepunkts.

fstype Dateisystem-Typ.

spec-options
: Optionen, wie sie beim *mount*-Kommando angegeben wurden.

time Einhängezeitpunkt, angegeben in Sekunden seit 1.1.1970

Einträge in der Datei */etc/mnttab* werden wieder gelöscht, wenn die Kommandos *umount* oder *umountall* für entsprechende Dateisysteme oder Dateisystemtypen ausgeführt werden.

Beispiel

Geben Sie in der POSIX-Shell ein: `cat /etc/mnttab`

```
/dev/root      /           ufs    rw,suid          802532552
/proc          /proc       proc   rw,              802532553
/dev/fd        /dev/fd     fdfs   rw               802532553
/dev/dsk/3     /var        ufs    suid,rw,noquota  802532558
/dev/dsk/2     /home1      ufs    suid,rw,noquota  802532588
SINTEST1:/nfs  /nfsclient  ufs    rw               802536261
```

/etc/vfstab Tabelle der definierten Dateisysteme

Die Datei */etc/vfstab* beschreibt jedes auf dem lokalen Rechner definierte Dateisystem. Die Datei können Sie mit einem Editor bearbeiten.

Die Dateisysteme, die in der Datei */etc/vfstab* mit *yes* in der Spalte *automnt* eingetragen sind, werden beim POSIX-Start bzw. durch das Kommando *mountall* automatisch eingehängt.

Außerdem werden die Einträge in der Datei benutzt, um bei der Ausführung eines mount-Kommandos ggf. fehlende Angaben für *ressource* oder *einhängepunkt* sowie *mount*-Optionen zu ergänzen.

Für ufs-Dateisysteme, die mit dem POSIX-Installationsprogramm definiert werden, werden automatisch entsprechende Einträge in der Datei */etc/vfstab* erzeugt. Für alle anderen Dateisysteme (z.B. bs2fs- oder nfs-Dateisysteme) sind die Einträge bei Bedarf manuell zu erstellen.

Im Gegensatz zur Datei */etc/mnttab* hat die Ausführung der Kommandos *mount* und *umount* auf die Datei */etc/vfstab* keine Auswirkungen. Entsprechende Einträge bleiben erhalten.

Die Felder in der Tabelle sind durch Tabulatoren und/oder Leerzeichen getrennt. Ein Bindestrich (–) kennzeichnet einen leeren Eintrag im Feld. Die Tabelle enthält folgende Felder:

Aufbau

```
special␣fsckdev␣mountp␣fstype␣fsckpass␣automnt␣mntopts
```

special — beschreibt die einzuhängende Ressource.

Bei (manuellen) Einträgen für bs2fs-Dateisysteme ist Folgendes zu beachten:
- Buchstaben dürfen nur in Großschreibung angegeben werden
- Sonderzeichen dürfen nicht entwertet werden, ebenso ist das Einschließen der Zeichenfolge in Hochkommata nicht zulässig

fsckdev — Name des blockorientierten Geräts bzw. der Ressource des zeichenorientierten Geräts.

mountp — Einhängepunkt: absoluter Pfadname des Verzeichnisses, in dem die Ressource eingehängt werden soll.

fstype — Dateisystem-Typ.

fsckpass — ist die für mehrere *fsck*-Kommandos zu verwendende Durchlaufnummer.

automnt — gibt an, ob die Ressource durch *mountall* automatisch beim Start von POSIX eingehängt werden soll (*yes*) oder nicht (*no*).

mntopt — Liste durch Kommata getrennter Optionen für das Einhängen des Dateisystems. Die Optionen entsprechen den *spez_optionen* des Kommandos *mount*.

Beispiel

Geben Sie in der POSIX-Shell ein: `cat /etc/vfstab`

```
/dev/root              /dev/rroot     /                      ufs    1   yes
     -
/proc                  -              /proc                  proc   -   no
     -
/dev/fd                -              /dev/fd                fdfs   -   no
     -
/dev/dsk/3             /dev/rdsk/3    /var                   ufs    1   yes
     -
172.25.86.64:/home2/froede/SHARE  -   /home/froede/RETSINA   nfs    -   no
     soft
PGOB0004:/home2/froede/SHARE      -   /home/froede/PGOB0004  nfs    -   no
     soft
/dev/dsk/4             /dev/rdsk/4    /home/froede           ufs    1   yes
     -
/dev/dsk/10            /dev/rdsk/10   /home/gast             ufs    1   yes
     -
/dev/dsk/13            /dev/rdsk/13   /mnt/ascii             ufs    1   no
     -
/dev/dsk/8             /dev/rdsk/8    /mnt/dat1              ufs    1   no
     -
/dev/dsk/23            /dev/rdsk/23   /bs2fscont             ufs    1   no
     -
/dev/dsk/24            /dev/rdsk/24   /home/bach/mount3      ufs    1   no
     -
/dev/dsk/25            /dev/rdsk/25   /home/bach/mountxxx    ufs    1   no
     -
/dev/dsk/26            /dev/rdsk/26   /home/bach/mountyyy    ufs    1   no
     -
/dev/dsk/5             /dev/rdsk/5    /home/bach             ufs    1   yes
     -
/dev/dsk/2             /dev/rdsk/2    /home/bach/mount99     ufs    1   yes
     -o
/dev/dsk/6             /dev/rdsk/6    /suderlan              ufs    1   no
     -
:V7OA:$BACH.ASS.*.S    -              /home/bach/bs2.1       bs2fs  1   yes
     ftyp=binary
:V7OA:$BACH.CCC.*.C    -              /home/bs2.2            bs2fs  1   yes
     ftyp=text
:V7OA:$BACH.PLAMLIB*   -              /home/bach/bs2.2       bs2fs  1   yes
     ftyp=textbin
:V7OA:$BACH.SEM*.C     -              /home/bs2000           bs2fs  1   yes
     -
```

Beispiel 1 Auflisten der eingehängten Dateisysteme (Format 1) und Einhängen eines neuen Dateisystems (Format 2). Das Beispiel wird unter der POSIX-Verwalterkennung durchgeführt.

```
# mount
/ on /dev/root read/write/setuid on Thu Jun  5 09:17:49 2009
/proc on /proc read/write on Thu Jun  5 09:17:49 2009
/dev/fd on /dev/fd read/write on Thu Jun  5 09:17:49 2009
/tmp on /dev/dsk/2 setuid/read/write/noquota on Thu Jun  5 09:17:50 2009
/var on /dev/dsk/3 setuid/read/write/noquota on Thu Jun  5 09:17:49 2009
/home on /dev/dsk/4 setuid/read/write/noquota on Thu Jun  5 09:17:51 2009
/home1 on /dev/dsk/5 setuid/read/write/noquota on Thu Jun  5 09:17:51 2009
/home2 on /dev/dsk/6 setuid/read/write/noquota on Thu Jun  5 09:17:51 2009
# mount -p
/dev/root - / ufs - no rw,suid
/proc - /proc proc - no rw
/dev/fd - /dev/fd fdfs - no rw
/dev/dsk/2 - /tmp ufs - no suid,rw,noquota
/dev/dsk/3 - /var ufs - no suid,rw,noquota
/dev/dsk/4 - /home ufs - no suid,rw,noquota
/dev/dsk/5 - /home1 ufs - no suid,rw,noquota
# mount -v
/dev/root on / type ufs read/write/setuid on Thu Jun  5 09:17:49 2009
/proc on /proc type proc read/write on Thu Jun  5 09:17:49 2009
/dev/fd on /dev/fd type fdfs read/write on Thu Jun  5 09:17:49 2009
/dev/dsk/2 on /tmp type ufs setuid/read/write/noquota on Thu Jun  5 09:17:50
2009
/dev/dsk/3 on /var type ufs setuid/read/write/noquota on Thu Jun  5 09:17:49
2009
/dev/dsk/4 on /home type ufs setuid/read/write/noquota on Thu Jun 5 09:17:51
2009
/dev/dsk/5 on /home1 type ufs setuid/read/write/noquota on Thu Jun 5 09:17:51
2009
/dev/dsk/6 on /home2 type ufs setuid/read/write/noquota on Thu Jun 5 09:17:51
2009
# mount -F ufs /dev/dsk/17 /home3
# mount -p
/dev/root - / ufs - no rw,suid
/proc - /proc proc - no rw
/dev/fd - /dev/fd fdfs - no rw
/dev/dsk/2 - /tmp ufs - no suid,rw,noquota
/dev/dsk/3 - /var ufs - no suid,rw,noquota
/dev/dsk/4 - /home ufs - no suid,rw,noquota
/dev/dsk/5 - /home1 ufs - no suid,rw,noquota
/dev/dsk/6 - /home2 ufs - no suid,rw,noquota
/dev/dsk/17 - /home3 ufs - no suid,rw,noquota
```

Beispiel 2 Einhängen des bs2fs-Containers und eines bs2fs-Dateisystems. Das Beispiel wird unter der POSIX-Verwalterkennung durchgeführt.

```
# mount -F ufs -o bs2fscontainer /bs2fscont
# mount -F bs2fs ':V70a:$sysaudit.sys.conslog.2007-06*'  /home/bs2.conslog
# mount
/ on /dev/root read/write/setuid on Tue Nov 27 11:31:04 2007
.
.
.
/bs2fscont on /dev/dsk/23 bs2fscontainer/setuid/read/write/noquota
   on Tue Nov 27 11:35:23 2007
/home/bs2.conslog on :V70A:$SYSAUDIT.SYS.CONSLOG.2007-06* ftyp=text/nosuid
   on Tue Nov 27 13:52:23 2007

#
```

Siehe auch *umount*, *mountall*
mount(), umount() [4]

mountall Mehrere Dateisysteme einhängen (mount file systems)

mountall dient zum Einhängen von Dateisystemen entsprechend einer *dateisystemtabelle* (*/etc/vfstab* ist die Standarddateisystemtabelle). Der Gerätedateiname „-“ liest aus der Standardeingabe. Ist der Bindestrich angegeben, muss die Standardeingabe dasselbe Format haben wie */etc/vfstab*.

Bevor die einzelnen Dateisysteme eingehängt werden, wird mit *fsck* eine Plausibilitätsprüfung durchgeführt, um festzustellen, ob das System einhängbar erscheint (nicht bei Dateisystemen vom Typ *bs2fs* oder *nfs)*. Ist das Dateisystem nicht einhängbar, wird es mit *fsck* korrigiert, bevor versucht wird, es einzuhängen.
Ist **nur** der *dateisystemtyp* angegeben, beschränkt sich die Wirkung von *mountall* auf Dateisysteme des angegebenen Typs.

Die Dateisysteme werden in der Reihenfolge *ufs* - *bs2fs* - *nfs* eingehängt. Somit ist gewährleistet, dass beim Einhängen der bs2fs-Dateisysteme das dafür erforderliche bs2fscontainer-Dateisystem im ufs bereits eingehängt ist.

Syntax **mountall**[␣**-F**␣dateisystemtyp] [␣**-l** | ␣**-r** | ␣**-b**][dateisystemtabelle]

Keine Option angegeben
mountall hängt alle Dateisysteme ein, bei denen das Feld *automnt* in der *dateisystemtabelle* auf *yes* gesetzt ist.

Optionen

-**F**␣dateisystemtyp
Angabe des Dateisystem-Typs, der eingehängt werden soll.

-l Begrenzt den Vorgang auf lokale Dateisysteme (*ufs* und *bs2fs*).

-r Begrenzt den Vorgang auf ferne Dateisystem-Typen (*nfs*).

-b Begrenzt den Vorgang auf bs2fs-Dateisysteme.

dateisystemtabelle
wird *dateisystemtabelle* nicht angegeben, bezieht sich *mountall* auf */etc/vfstab*.

Hinweis Wenn die Option *-F* in Kombination mit einer oder mehrerer der Optionen *-l*, *-r* und *-b* angegeben wird und die Optionen miteinander verträglich sind, so haben die Optionen *-l*, *-r* und *-b* Vorrang. Beispielsweise haben *mountall -F bs2fs -l* und *mountall -F ufs -l* dieselbe Wirkung wie *mountall -l* : Es werden alle lokalen Dateisysteme (also alle ufs- und bs2fs-Dateisysteme) eingehängt. Ebenso führen die Angaben *mountall -F bs2fs* und *mountall -b* zum gleichen Ergebnis: alle bs2fs-Dateisysteme werden gemounted.

Fehler Wenn die Dateisysteme einhängbar und fehlerfrei sind, wird keine Meldung ausgegeben. Fehler- und Warnmeldungen stammen von *fsck* und *mount* bzw. von *mountall* bei falscher Syntaxangabe.

Datei */etc/vfstab*
Standarddateisystemtabelle

Siehe auch *fsck, mount*, *umountall*

mv Dateien versetzen oder umbenennen (move files)

Mit *mv* können Sie eine Datei umbenennen oder im Dateibaum an einen anderen Ort versetzen. Um *mv* ausführen zu können, müssen Sie Schreibrecht für das Dateiverzeichnis haben, in dem sich die Datei befindet bzw. in das sie versetzt werden soll.

mv erzeugt innerhalb eines Dateisystems keine physische Kopie der versetzten oder umbenannten Datei, sondern modifiziert lediglich die Einträge im übergeordneten Dateiverzeichnis. Verweise auf andere Dateien bleiben in diesem Fall erhalten.

Wenn eine Datei allerdings über die Grenzen eines Dateisystems versetzt wird, benutzt *mv* das Kommando *cp*. Die Originaldatei wird dann zunächst kopiert und danach gelöscht. In diesem Fall gehen alle Verweise auf andere Dateien verloren.

Syntax

Format 1: mv[␣**-f**][␣**-i**]␣datei␣dateineu

Format 2: mv[␣**-f**][␣**-i**]␣datei␣...␣dvz

Format 3: mv[␣**-f**][␣**-i**]␣dvz␣dvzneu

Format 1 **Datei umbenennen**

mv[␣**-f**][␣**-i**]␣datei␣dateineu

Keine Option angegeben

Wenn Sie für *dateineu* eine bestehende Datei angeben, für die Sie kein Schreibrecht haben, werden die Zugriffsrechte von *dateineu* ausgegeben und Sie werden gefragt, ob *mv* ausgeführt werden soll. *mv* überschreibt *dateineu* nur dann, wenn Sie die Frage bejahen.

Achtung!
Ist die Standard-Eingabe keine Datensichtstation, unterbleibt die Frage und *dateineu* wird nicht überschrieben.

Optionen

-f Existiert bereits eine Datei mit dem Namen *dateineu*, dann überschreibt *mv* die bestehende Datei, egal, ob Sie für die Datei Schreibrecht besitzen oder nicht.

-i (i - interactive) Wenn Sie für *dateineu* eine bestehende Datei angeben, werden Sie in jedem Fall gefragt, ob *mv* wirklich ausgeführt werden soll.

Eine mehrmalige Angabe von *-f* oder *-i* gilt nicht als Fehler. Die letzte angegebene Option bestimmt das Verhalten von *mv*.

datei

Name der Datei, die Sie umbenennen wollen.

dateineu

Neuer Name der Datei, der sich von *datei* unterscheiden muss. Existiert bereits eine Datei mit dem Namen *dateineu*, wird sie mit dem Inhalt von *datei* überschrieben, wenn Sie für *dateineu* Schreibrecht haben (siehe aber Option *-i*).

Wenn Sie für *dateineu* eine bestehende Datei angeben, für die Sie kein Schreibrecht haben, werden die Zugriffsrechte von *dateineu* ausgegeben und Sie werden gefragt, ob *mv* ausgeführt werden soll. *mv* überschreibt *dateineu* nur dann, wenn Sie die Frage bejahen. Ist die Option *-f* angegeben, unterbleibt diese Frage und *dateineu* wird überschrieben. Ist die Standardeingabe keine Datensichtstation, so unterbleibt die Abfrage und *dateineu* wird nicht überschrieben.

Wenn das übergeordnete Dateiverzeichnis von *dateineu* schreibbar ist, aber das t-Bit (sticky-Bit) gesetzt hat, muss eine der folgenden Bedingungen erfüllt sein, um *dateineu* umzubenennen:
- die Datei muss dem Benutzer gehören
- das Dateiverzeichnis muss dem Benutzer gehören
- der Benutzer muss Schreibberechtigung für die Datei haben
- der Benutzer muss ein privilegierter Benutzer sein

Format 2 **Dateien und Dateiverzeichnisse in ein anderes Dateiverzeichnis versetzen**

mv[␣**-f**][␣**-i**]␣datei␣...␣dvz

option

siehe Format 1

datei

Namen der Dateien oder der Dateiverzeichnisse, die in das Dateiverzeichnis *dvz* übertragen werden sollen. Wenn Sie ein Dateiverzeichnis als Quelle angeben, werden alle darin enthaltenen Dateien und Dateiverzeichnisse ersetzt.

dvz

Name des Dateiverzeichnisses, in das die Dateien oder Dateiverzeichnisse übertragen werden sollen. Sie brauchen für das Ziel-Dateiverzeichnis Schreibrecht.

Wenn *dvz* schreibbar ist, aber das t-Bit (sticky-Bit) gesetzt hat, muss eine der folgenden Bedingungen erfüllt sein, um die Dateien oder Dateiverzeichnisse nach *dvz* zu übertragen:
- die Datei muss dem Benutzer gehören
- das Ziel-Dateiverzeichnis muss dem Benutzer gehören
- der Benutzer muss Schreibberechtigung für die Datei haben
- der Benutzer muss ein privilegierter Benutzer sein

Format 3 **Dateiverzeichnis umbenennen**

mv[␣**-f**][␣**-i**]␣dvz␣dvzneu

option
siehe Format 1

dvz
Name des Dateiverzeichnisses, das Sie umbenennen wollen.

dvzneu
Neuer Name des Dateiverzeichnisses.

dvz und *dvzneu* müssen zum gleichen physikalischen Dateisystem gehören. Die den Dateiverzeichnissen direkt übergeordneten Dateiverzeichnisse .. müssen nicht identisch sein.
Falls es schon ein Dateiverzeichnis mit dem Namen *dvzneu* gibt, versetzt *mv* das Dateiverzeichnis *dvz* in das Dateivezeichnis *dvzneu*.
Gibt es bereits *dvzneu/dvz*, dann versetzt *mv* das Dateiverzeichnis *dvz* nur, wenn *dvzneu/dvz* leer ist.

Fehler `mv: Cannot rename dvz to dvzneu : Permission denied`
Sie besitzen für das Dateiverzeichnis *dvz*, das umbenannt werden soll, kein Schreibrecht.

`mv: Cannot create datei : Permission denied`
Sie besitzen für das Dateiverzeichnis, in das *datei* verlegt werden soll, kein Schreibrecht.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *mv*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für das Verhalten von Bereichen, Äquivalenzklassen und Zeicheneinheiten in erweiterten regulären Ausdrücken für ja/nein-Abfragen fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) sowie das Verhalten von Zeichenklassen in erweiterten regulären Ausdrücken für ja/nein-Abfragen.

LC_MESSAGES
: Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH
: Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die Datei *lieder* im aktuellen Dateiverzeichnis soll in *popsongs* umbenannt und in das Dateiverzeichnis */home/petra/kunst/musik* übertragen werden.

```
$ mv lieder /home/petra/kunst/musik/popsongs
```

Beispiel 2 Die Dateien *efeu*, *papyrus* und *flieder* im aktuellen Dateiverzeichnis sollen ihre Namen behalten und ins Dateiverzeichnis */home/petra/pflanzen* übertragen werden.

```
$ mv efeu papyrus flieder /home/petra/pflanzen
```

Siehe auch *chmod, cp, find, ln, rm*

newgrp Gruppenzugehörigkeit ändern (change to a new group)

Das in die POSIX-Shell *sh* eingebaute Kommando *newgrp* überlagert die aktuelle Shell mit */bin/newgrp*. Das Kommando */bin/newgrp* macht die Nummer der angegebenen Gruppe zu Ihrer aktuellen Gruppennummer und überlagert sich selbst mit einer Shell. Mit END beenden Sie die Shell, in der Sie *newgrp* aufgerufen haben.

Achtung!
Wird *newgrp* mit der Taste DEL in jener Phase abgebrochen, in der */bin/newgrp* die aktuelle Shell überlagert, wird auch die Shell abgebrochen, von der *newgrp* aufgerufen wurde.

Mit *newgrp* können Sie also in eine andere Benutzergruppe wechseln. Das bedeutet:
– Ihre Zugriffsrechte für bestehende Dateien ändern sich entsprechend der neuen Gruppenzugehörigkeit.
– Bei Dateien, die Sie neu anlegen, gelten die Zugriffsrechte für Gruppe ab jetzt der Gruppe, in die Sie gewechselt sind.

Nach dem Wechsel der Gruppe sind in der jetzt aktuellen Shell nur noch die Variablen bekannt, die Sie vorher exportiert haben (siehe *export*). Nicht exportierte Variablen sind entweder nicht definiert oder bekommen von der Shell einen Standard-Wert zugewiesen (siehe *sh*). Auch Shell-Variablen, wie z.B. *PATH* und *HOME*, erhalten Standardwerte, wenn sie nicht vorher vom System oder von Ihnen exportiert worden sind.

Vor dem Aufruf beachten

Die Gruppe, in die Sie wechseln wollen, muss in der Datei */etc/group* eingetragen sein. Andernfalls bricht *newgrp* mit einer Fehlermeldung ab.

Sie können mit *newgrp* in jede Gruppe wechseln, in der Sie Mitglied sind; d.h. Ihre Benutzerkennung ist in der Datei */etc/group* im Eintrag für diese Gruppe enthalten.

Bei den Fehlermeldungen „Sorry“ und/oder „Unknown group“ bricht *newgrp* ab, bei allen anderen Fehlermeldungen beendet sich die Shell.

Syntax

Format 1: newgrp[␣**-l**][␣gruppe]

Format 2: newgrp[␣**-**][␣gruppe]

Beide Formate werden gemeinsam beschrieben, da die Option *-l* in Format 1 äquivalent der Option – in Format 2 ist.

Kein Argument angegeben
Sie wechseln zurück in die Gruppe, deren Gruppen-Nummer (GID) für Ihre Benutzerkennung eingetragen ist.

– bzw. **-l**

Das Kommando *newgrp* überlagert die aktuelle Shell mit einer Login-Shell. Bevor diese Shell ihr Bereitzeichen ausgibt, führt sie die Dateien */etc/profile* und *$HOME/.profile*, falls vorhanden, aus und wechselt in Ihr HOME-Dateiverzeichnis.

Sie arbeiten also in der gleichen Umgebung wie nach der Anmeldung am System, allerdings als Mitglied einer anderen Gruppe, nämlich der, die Sie beim Aufruf von */bin/newgrp* angegeben haben.

– bzw. *-l* nicht angegeben:
newgrp überlagert die aktuelle Shell mit */bin/newgrp*. Das aktuelle Dateiverzeichnis ändert sich nicht, aber der neuen Shell sind nur noch die Variablen bekannt, die Sie vorher exportiert haben. Nicht exportierte Variablen sind entweder nicht definiert oder bekommen von der Shell einen Standard-Wert zugewiesen.

gruppe

Name der Gruppe, in die Sie wechseln wollen. Der Name dieser Gruppe muss in der Datei */etc/group* eingetragen sein. Die zugehörige Gruppennummer (GID) muss bereits für eine Benutzerkennnung eingetragen sein.

Wenn Sie nicht Mitglied der angegebenen Gruppe sind, muss für die Gruppe in der Datei */etc/group* ein Kennwort vereinbart sein. Das Kommando *newgrp* erwartet die Eingabe dieses Kennwortes, bevor der Wechsel in die Gruppe stattfindet.

Wenn für Ihre Benutzerkennung kein Kennwort eingetragen ist, muss für die Gruppe in der Datei */etc/group* ein Kennwort vereinbart sein. Das Kommando *newgrp* erwartet die Eingabe dieses Kennwortes, bevor der Wechsel in die betreffende Gruppe stattfindet.

Wenn Sie wieder in die Benutzergruppe wechseln wollen, die für Sie eingetragen ist, rufen Sie *newgrp* ohne Angabe eines Gruppennamens auf.

gruppe nicht angegeben:
Sie wechseln zurück in die Gruppe, deren Gruppen-Nummer (GID) für Ihre Benutzerkennung eingetragen ist.

Endestatus immer 0

Fehler `Unknown group`
Dieser Name ist nicht in der Datei */etc/group* eingetragen.

`Sorry`
Sie dürfen nicht in diese Gruppe wechseln, weil Sie nicht Mitglied sind und kein Kennwort für diese Gruppe vereinbart ist.

Datei */etc/group*
legt für die eingetragenen Gruppennummern einen Namen fest und bestimmt alle Mitglieder dieser Gruppe.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *newgrp*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Wechseln in die Gruppe mit dem Gruppennamen *consul*:

```
$ newgrp consul
$ >dateineu
$ chmod 640 dateineu
$ ls -lg dateineu
-rw-r----- 1 ROSA   consul       162 Mar 19 18:34 dateineu
```

Die nach dem Gruppenwechsel neu angelegte Datei *dateineu* ist für die Mitglieder der Gruppe *consul* lesbar.

Siehe auch *exec*, *export*

nice Priorität von Kommandos ändern (invoke a utility with an altered system scheduling priority)

Achtung!
Das Kommando *nice* wird nur aus Kompatibiltätsgründen unterstützt. Es hat keinen Einfluss auf die Taskprioritäten des BS2000. Deswegen finden Sie hier nur die Beschreibung der Syntax, aber keine Beschreibung der Optionen, Argumente usw.

Syntax

Format 1: nice[␣**-n**␣zahl]␣kommando[␣argument]

Format 2: nice[␣**-**zahl]␣kommando[␣argument]

nl Textzeilen nummerieren (line numbering filter)

nl liest Zeilen aus einer Datei oder von der Standard-Eingabe und schreibt sie nummeriert auf die Standard-Ausgabe.

nl unterteilt den eingelesenen Text in logische Seiten. Eine logische Seite besteht aus einem Kopfteil, einem Hauptteil und einem Fußteil. Jeder dieser Teile kann leer sein.

Der Beginn des Kopf-, Haupt- und Fußteils einer logischen Seite wird normalerweise durch eine Eingabezeile angezeigt, die ausschließlich eine der folgenden Zeichenketten enthält:

\:\:\: für Beginn des Kopfteils

\:\: für Beginn des Hauptteils

\: für Beginn des Fußteils

Sind im Eingabetext keine Begrenzungszeichenketten enthalten, dann interpretiert *nl* den Text vollständig als Hauptteil einer logischen Seite.

Zu Beginn einer logischen Seite wird der Zeilenzähler zurückgesetzt (Ausnahme: Option *-p*). Für die Zeilennummerierung der drei Teile einer logischen Seite können Sie verschiedene Optionen setzen. So können Sie z.B. festlegen, dass im Kopf- und Fußteil keine Zeilen und im Hauptteil alle leeren Zeilen nummeriert werden sollen.

Syntax **nl**[␣option]...[␣datei]

Keine Option angegeben

nl nummeriert im Hauptteil alle Zeilen, die druckbare Zeichen enthalten; die Zeilen im Kopf- und Fußteil werden nicht nummeriert.
Zu Beginn einer logischen Seite wird der Zeilenzähler auf 1 gesetzt.
nl nummeriert in Einer-Schritten.
Die Zeilennummern haben bis zu 6 Stellen. Sie werden rechtsbündig ausgegeben, führende Nullen werden unterdrückt. Zeilennummer und Text werden durch ein Tabulatorzeichen getrennt.

option

Die Optionen müssen einzeln, d.h. durch Leerzeichen von den anderen Optionen getrennt, eingegeben werden. Der Name der Eingabedatei darf vor, zwischen oder hinter den Optionen stehen. Die Stellung des Dateinamens in der Kommandozeile beeinflusst nicht die Wirkungsweise des Kommandos.

Zeilen auswählen, die nummeriert werden sollen

-b␣typ
(b - body) Mit dieser Option geben Sie an, welche Zeilen im Hauptteil einer logischen Seite nummeriert werden sollen.

typ kann sein: **a**, **n**, **p**regulärer_ausdruck oder **t**.

a (a - all lines) Alle Zeilen werden nummeriert.

n (n - no lines) Keine Zeile wird nummeriert.

pregulärer_ausdruck
Alle Zeilen werden nummeriert, die eine zu *regulärer_ausdruck* passende Zeichenfolge enthalten.
regulärer_ausdruck ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Enthält der reguläre Ausdruck Zeichen, die für die Shell eine Sonderbedeutung haben, dann schließen Sie ihn in Hochkommata ein:
p'regulärer_ausdruck**'**.

t (t - text lines) Alle Zeilen, die druckbare Zeichen enthalten, werden nummeriert.

-b nicht angegeben:
Nur die Zeilen werden nummeriert, die druckbare Zeichen enthalten. (D.h., die Option *-b* mit Argument *t* ist der Standard-Fall.)

-f␣typ
(f - footer) Mit dieser Option geben Sie an, welche Zeilen im Fußteil einer logischen Seite nummeriert werden sollen.

typ kann sein: **a**, **n**, **p**regulärer_ausdruck oder **t**.

-f nicht angegeben:
Die Zeilen im Fußteil werden nicht nummeriert. (D.h., die Option *-f* mit Argument *n* ist der Standard-Fall.)

-h␣typ
(h - header) Mit dieser Option geben Sie an, welche Zeilen im Kopfteil einer logischen Seite nummeriert werden sollen.

typ kann sein: **a**, **n**, **p**regulärer_ausdruck oder **t**.

-h nicht angegeben:
Die Zeilen im Kopfteil werden nicht nummeriert. (D.h., die Option *-h* mit Argument *n* ist der Standard-Fall.)

Zeilenzähler zurücksetzen / nicht zurücksetzen

-p Der Zeilenzähler wird zu Beginn einer logischen Seite nicht zurückgesetzt.

-v␣startnummer
Der Zeilenzähler wird zu Beginn jeder logischen Seite auf den Wert *startnummer* zurückgesetzt.
startnummer ist eine Zahl größer gleich 0.

-v nicht angegeben:
Der Zeilenzähler wird zu Beginn jeder logischen Seite auf den Wert 1 zurückgesetzt.

Schrittweite definieren

-i␣diff
Die Differenz zwischen zwei Zeilennummern ist *diff*.

-i nicht angegeben:
Die Differenz ist 1.

Ausgabeformat festlegen

-n␣format
Format der Zeilennummern.

format kann sein: **ln**, **rn** oder **rz**.

ln	Die Zeilennummern werden linksbündig ausgegeben; führende Nullen werden unterdrückt.
rn	Die Zeilennummern werden rechtsbündig ausgegeben; führende Nullen werden unterdrückt.
rz	Die Zeilennummern werden rechtsbündig ausgegeben; führende Nullen werden nicht unterdrückt.

-n nicht angegeben:
Die Zeilennummern werden rechtsbündig ausgegeben; führende Nullen werden unterdrückt. (D.h. Option *-n* mit Argument *rn* ist der Standard-Fall.)

-s␣trenner
Zeilennummer und Text werden durch die Zeichenkette *trenner* getrennt. *trenner* besteht aus einem oder mehreren Zeichen.

-s nicht angegeben:
Zeilennummer und Text werden durch ein Tabulatorzeichen voneinander getrennt.

-w␣n
Die Zeilennummern haben bis zu *n* Stellen.

Der maximale Wert für *n* ist 100. Wenn Sie einen höheren Wert für *n* angeben, wird für *n* der Wert 100 angenommen.

-w nicht angegeben:
Die Zeilennummern haben bis zu 6 Stellen.

Nummerierung von Leerzeilen festlegen

-l␣n
nl interpretiert *n* aufeinanderfolgende Leerzeilen als eine einzige Leerzeile.

Beispiel

```
$ nl -b a -l 2
```

bewirkt, dass im Hauptteil bei mehreren aufeinanderfolgenden Leerzeilen nur jede zweite Leerzeile nummeriert wird (im Kopf- und Fußteil werden überhaupt keine Zeilen nummeriert).

-l nicht angegeben:
Jede Leerzeile wird als vollständige Zeile interpretiert (n = 1).

Trennzeichen von Kopf-, Haupt- und Fußteil definieren

-d␣x[y]
Die Zeichenkette *xy* kennzeichnet statt \: den Beginn von Kopf-, Haupt- bzw. Fußteil. Wollen Sie für *x* oder *y* einen Gegenschrägstrich \ eingeben, so müssen Sie diesen mit Hochkommata oder einem weiteren Gegenschrägstrich entwerten, z.B. -d'*' oder -d*.

y nicht angegeben:
Die Zeichenkette *x:* kennzeichnet statt \: den Beginn von Kopf-, Haupt- bzw. Fußteil.

datei
Name der Eingabedatei.

datei nicht angegeben:
nl liest von der Standard-Eingabe.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *nl*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für das Verhalten von Bereichen, Äquivalenzklassen und Zeicheneinheiten in regulären Ausdrücken fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) sowie das Verhalten von Zeichenklassen innerhalb von regulären Ausdrücken und um zu entscheiden, ob ein Zeichen in die Zeichenklasse gehört, die durch die Option *-b*, *-f* oder *-h* gewählt wurde.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Datei *gedichte* hat folgenden Inhalt:

```
\:\:\:
Das aesthetische Wiesel
\:\:
     Ein Wiesel
sass auf einem Kiesel
inmitten Bachgeriesel.

     Wisst ihr,
      weshalb?

    Das Mondkalb
   verriet es mir
    im stillen:

   Das raffinier-
      te Tier
tat's um des Reimes willen.
\:
```

```
Christian Morgenstern
\:\:\:
Limerick
\:\:
Eine Frau aus Clausthal-Zellerfeld,
die taeglich in den Keller faellt,
haelt den Rekord
in diesem Sport,
weil sie von Tag zu Tag schneller faellt.
\:
Verfasser unbekannt
```

Einfacher nl-Aufruf

```
$ nl gedichte

       Das aesthetische Wiesel

     1       Ein Wiesel
     2  sass auf einem Kiesel
     3  inmitten Bachgeriesel.

     4       Wisst ihr,
     5        weshalb?

     6      Das Mondkalb
     7     verriet es mir
     8      im stillen:

     9     Das raffinier-
    10        te Tier
    11  tat's um des Reimes willen.

       Christian Morgenstern

       Limerick

     1  Eine Frau aus Clausthal-Zellerfeld,
     2  die taeglich in den Keller faellt,
     3  haelt den Rekord
     4  in diesem Sport,
     5  weil sie von Tag zu Tag schneller faellt.

       Verfasser unbekannt
```

In Zehner-Schritten nummerieren

```
$ nl -v 10 -i 10 gedichte

       Das aesthetische Wiesel

    10       Ein Wiesel
    20  sass auf einem Kiesel
    30  inmitten Bachgeriesel.

    40       Wisst ihr,
    50        weshalb?

    60      Das Mondkalb
    70     verriet es mir
    80      im stillen:

    90     Das raffinier-
   100        te Tier
   110  tat's um des Reimes willen.

       Christian Morgenstern

       Limerick

    10  Eine Frau aus Clausthal-Zellerfeld,
    20  die taeglich in den Keller faellt,
    30  haelt den Rekord
    40  in diesem Sport,
    50  weil sie von Tag zu Tag schneller faellt.

       Verfasser unbekannt
```

Siehe auch *ed*, *pr*

nm Symboltabelle einer Objektdatei ausgeben (write the name list of an object file)

Das Kommando *nm* gibt die Symboltabelle aus, die in der durch *datei* angegebenen Objektdatei, ausführbaren Datei oder Objektdatei-Bibliothek enthalten ist.

Enthält eine gültige Eingabedatei keine Symbolinformationen, wird dies von *nm* ausgegeben.

Syntax **nm**[␣**-APv**][␣**-efox**][␣**-g**|␣**-u**][␣**-t**␣format]␣datei...

Optionen:

-A in jeder Zeile den vollen Pfadnamen bzw. Bibliotheksnamen eines Objekts ausgeben

-e ausschließliche Ausgabe von externen (globale) und statischen Symbolen.

-f erzeugt eine vollständige Ausgabe. Redundante Symbole (wie z.B. .text, .data und .bss), die normalerweise unterdrückt werden, werden mit ausgegeben.

-g nur externe (globale) Symbole ausgeben

-o Wert und Größe eines Symbols in Oktal- statt in Dezimalform ausgeben (gleichbedeutend mit *-t␣o*)

-P Ausgabe in einem portierbaren Format erstellen (siehe Ausgabe).

-t␣format
Ausgabeformat von Wert und Größe eines Symbols festlegen:

- d Wert und Größe werden dezimal ausgegeben (Standard)
- o Wert und Größe werden oktal ausgegeben (entspricht *-o*)
- x Wert und Größe werden hexadezimal ausgegeben (entspricht *-x*)

-u nur undefinierte Symbole ausgeben

-v externe Symbole vor der Ausgabe nach ihrem Wert sortieren (statt alphabetisch)

-x Wert und Größe eines Symbols in Hexadezimal- statt in Dezimalform ausgeben (gleichbedeutend mit *-t␣x*)

datei
Pfadname einer Objektdatei, einer ausführbaren Datei oder einer Objektdatei-Bibliothek.

Ausgabe

Für jedes Symbol werden folgende Informationen ausgegeben:

- Bibliotheksname oder Objektname, wenn *-A* angegeben ist.
- Symbolname
- Symboltyp, der aus einem der folgenden einzelnen Zeichen bestehen kann:
  - A absolutes Symbol global
  - a absolutes Symbol lokal
  - B bss Symbol global
  - b bss Symbol lokal
  - D Datenobjekt-Symbol global
  - d Datenobjekt-Symbol lokal
  - T Textobjekt-Symbol global
  - t Textobjekt-Symbol lokal
  - U undefiniertes Symbol
- Wert des Symbols
- Größe in Bytes, die mit dem Symbol assoziiert ist (wenn vorhanden)

Ist die Option *-P* angegeben, werden die Informationen in einem portierbaren Format ausgegeben. Die drei Formate unterscheiden sich nur, ob für *-t* eine dezimale, oktale oder hexadezimale Darstellung gewählt wurde.
Ist *-t* nicht angegeben, wird implizit *-t␣ x* angenommen.

`"%s%s %s %d %d\n"`*,Bibliotheks-/Objektname, Symbolname, Typ, Wert, Größe*

`"%s%s %s %o %o\n"`*,Bibliotheks-/Objektname, Symbolname, Typ, Wert, Größe*

`"%s%s %s %x %x\n"`*,Bibliotheks-/Objektname, Symbolname, Typ, Wert, Größe*

wobei *Bibliotheks-/Objektname* folgendermaßen formatiert wird:

- ist *-A* nicht angegeben, ist *Bibliotheks-/Objektname* die leere Zeichenkette
- ist *-A* angegeben und ist *datei* keine Bibliothek
  `"%s: "`, *datei*
- ist *-A* angegeben und *datei* kennzeichnet eine Bibliothek. Dann wird für *objektname* die Objektdatei ausgegeben, die in der Bibliothek das Symbol enthält:
  `"%s[%s]: "`, *datei*, *objektdatei*

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *nm*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für Symbolname und Symbolwert fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *ar*
c89 [5]

nohup Kommando ausführen und dabei Signale ignorieren (invoke a utility immune to hangups)

nohup führt ein beliebiges Kommando oder eine Shell-Prozedur aus und ignoriert dabei die Signale SIGHUP (Signalnummer 1) und SIGQUIT (Signalnummer 3). Sie können durch *nohup* verhindern, dass die Ausführung eines Kommandos abgebrochen wird, wenn Sie sich vom System abmelden.

Vor dem Aufruf beachten

Wenn Sie *nohup* auf Pipelines oder Kommandofolgen anwenden wollen, müssen Sie die betreffenden Pipelines oder Kommandolisten in eine Datei schreiben und diese Datei als Shell-Prozedur unter *nohup* aufrufen.

Syntax

nohup␣kommando[␣argument...]

kommando[␣argument...]

Ein beliebiges Kommando oder eine Shell-Prozedur. Sie können *kommando* mit Argumenten aufrufen, indem Sie diese wie üblich hinter *kommando* schreiben.
Die Standard-Ausgabe und Standard-Fehlerausgabe von *kommando* werden in die Datei *nohup.out* im aktuellen Dateiverzeichnis geschrieben, falls sie nicht umgelenkt sind.

Wenn Sie die Datei *nohup.out* im aktuellen Dateiverzeichnis nicht anlegen dürfen (d.h. Sie haben kein Schreibrecht im aktuellen Dateiverzeichnis) oder wenn Sie für die Datei *nohup.out* im aktuellen Dateiverzeichnis kein Schreibrecht haben, wird die Ausgabe in die Datei *$HOME/nohup.out* umgelenkt.

Wenn die Datei *nohup.out* bzw. *$HOME/nohup.out* nicht existiert, wird sie angelegt und erhält dabei ausschließlich Zugriffsrechte für den aufrufenden Benutzer. Andernfalls wird die Ausgabe hinten angehängt.

Endestatus

126 Das angegebene *kommando* existiert, kann aber nicht ausgeführt werden.

127 Das angegebene *kommando* ist nicht auffindbar oder beim Kommando *nohup* ist ein Fehler aufgetreten.

Fehler

`nohup: cannot open/create nohup.out`

Sie dürfen weder im aktuellen Dateiverzeichnis noch im Dateiverzeichnis *$HOME* eine Datei *nohup.out* anlegen bzw. in dort existierende Dateien *nohup.out* schreiben.

Datei

nohup.out

Datei im aktuellen Dateiverzeichnis, die die Standard-Ausgabe und Standard-Fehlerausgabe des Kommandos, das unter *nohup* aufgerufen wurde, enthält.

$HOME/nohup.out
enthält Standard-Ausgabe und Standard-Fehlerausgabe des Kommandos, das unter *nohup* aufgerufen wurde, falls die Datei *nohup.out* im aktuellen Dateiverzeichnis schreibgeschützt ist oder Sie diese Datei im aktuellen Dateiverzeichnis nicht anlegen dürfen (d.h. Sie haben kein Schreibrecht für das aktuelle Dateiverzeichnis).

Variable *HOME*
Der Wert der Umgebungsvariablen *HOME* legt das Dateiverzeichnis fest, in dem die Datei *nohup.out* angelegt wird, falls die Datei *nohup.out* im aktuellen Dateiverzeichnis schreibgeschützt ist oder Sie diese Datei im aktuellen Dateiverzeichnis nicht anlegen dürfen.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *nohup*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

PATH Legt den bei der Kommandosuche verwendeten Suchpfad fest.

Beispiel Sie starten folgenden Auftrag und lassen ihn im Hintergrund ablaufen:

```
$ nohup programm &
sending output to nohup.out
```

Dann melden Sie sich vom System ab. *programm* wird trotzdem nicht abgebrochen. Die Ausgabe wird in die Datei *nohup.out* geschrieben.

Siehe auch *chmod*, *kill*, *nice*, *sh*
signal() [4]

od Inhalt einer Datei oktal ausgeben (dump files in various formats)

od schreibt den Inhalt einer Datei auf die Standard-Ausgabe, wobei Sie für die Ausgabe über Optionen das Ausgabe-Formate bestimmen können.

Die erste Spalte jeder Ausgabe-Zeile gibt die Position des ersten in dieser Zeile enthaltenen Zeichens an. Je nachdem welches Ausgabeformat Sie gewählt haben, ist diese Angabe oktal, dezimal oder hexadezimal.

Syntax

Format 1: od[␣**-v**][␣**-A**␣addr_base][␣**-j**␣skip][␣**-N**␣count][␣**-t**␣type_string] [␣datei...]

Format 2: od[␣**-bcdDfFoOsSvxX**] [␣datei] [␣[+]offset[.][b]]

Format 1 **od**[␣**-v**][␣**-A**␣addr_base][␣**-j**␣skip][␣**-N**␣count][␣**-t**␣type_string][␣datei...]

-v (v - verbose) Alle Daten werden angezeigt.

-v nicht angegeben
Alle Ausgabezeilen, die identisch zu den gerade vorherigen Ausgabezeilen sind, werden ersetzt durch eine Zeile, die nur einen Stern (*) enthält.

-A␣addr_base
kennzeichnet die Eingabe Offset-Basis. *addr_base* ist ein Zeichen. Die Zeichen o, d und x bedeuten, dass die Offset-Basis oktal, dezimal oder hexadezimal geschrieben wird. Das Zeichen n bedeutet, dass kein Offset geschrieben wird.

-j␣skip
(j - jump) Die nächsten *skip* bytes von Beginn der Eingabe werden übersprungen. Ist die Eingabe nicht wenigstens *skip* byte lang, wird eine Fehlermeldung ausgegeben. Standardmäßig wird *skip* als Dezimalzahl interpretiert. Beginnt *skip* mit 0x oder 0X, wird der Offset hexadezimal interpretiert. Beginnt *skip* mit einer führenden Null, wird der Offset oktal interpretiert. Folgt anschließend eines der Zeichen b, k oder m, wird der Offset als ein Vielfaches von 512, 1024 oder 1048576 Bytes interpretiert.

-N␣count
Es werden nur *count* Byte der Eingabe formatiert. Standardmäßig wird *count* als Dezimalzahl interpretiert. Beginnt *count* mit 0x oder 0X, wird der Offset hexadezimal interpretiert. Beginnt *count* mit einer führenden Null, wird der Offset oktal interpretiert. Sind weniger als *count* Byte Eingabe verfügbar, wird keine Fehlermeldung ausgegeben, sondern *od* formatiert die vorhandene Eingabe.

-t␣type_string
kennzeichnet einen oder mehrere Ausgabetypen. *type_string* besteht aus einer Zeichenkette, die die bei der Eingabe verwendeten Typen kennzeichnet. Die Zeichenkette muss aus den typspezifischen Zeichen a (named charcter), c (character), d (decimal), f (floating) , o (oktal), u (unsigned decimal) und x (hexadecimal) bestehen.

datei
Name der Datei, die ausgegeben werden soll.

datei nicht angegeben:
od liest von der Standard-Eingabe.

Format 2 **od**[␣**-bcdDfFoOsSvxX**][␣datei][␣[+]offset[.][b]]

Keine Option angegeben
Je 2 Bytes werden als vorzeichenlose Oktalzahl interpretiert (wie Option *-o*).

option
Wenn Sie mehrere Optionen angeben, um verschiedene Ausgabeformate zu kombinieren, dürfen Sie den Bindestrich - nur einmal angeben und müssen dann die Optionsnamen ohne Leerzeichen hintereinander angeben, z.B. *od -bcs datei.*

-b Jedes einzelne Byte wird als Oktalzahl interpretiert.

-c Jedes Byte wird als Zeichen entsprechend der aktuellen Festlegung durch *LC_CTYPE* interpretiert.

-d Je 2 Bytes werden als vorzeichenlose Dezimalzahl interpretiert.

-D Je 4 Bytes werden als vorzeichenlose Dezimalzahl interpretiert.

-f Je 4 Bytes werden als Gleitkommazahl interpretiert.

-F Je 8 Bytes werden als Zahl mit erweiterter Genauigkeit interpretiert.

-o Je 2 Bytes werden als vorzeichenlose Oktalzahl interpretiert.

-O Je 4 Bytes werden als vorzeichenlose Oktalzahl interpretiert.

-s Je 2 Bytes werden als Dezimalzahl mit Vorzeichen interpretiert.

-S Je 4 Bytes werden als Dezimalzahl mit Vorzeichen interpretiert.

-v (v - verbose) Alle Daten werden angezeigt.

-v nicht angegeben
Alle Ausgabezeilen, die identisch zu den gerade vorherigen Ausgabezeilen sind, werden ersetzt durch eine Zeile, die nur einen Stern (*) enthält.

-x Je 2 Bytes werden als vorzeichenlose Hexadezimalzahl interpretiert.

-X Je 4 Bytes werden als vorzeichenlose Hexadezimalzahl interpretiert.

datei
Name der Datei, die ausgegeben werden soll.

datei nicht angegeben:
od liest von der Standard-Eingabe.

offset
Mit dem Argument *offset* legen Sie fest, ab welcher Stelle in der Datei mit der Ausgabe begonnen werden soll.
Normalerweise wird *offset* als Oktalzahl interpretiert. Wird die Angabe für *offset* mit einem Punkt . abgeschlossen, so wird die angegebene Zahl als Dezimalzahl interpretiert. Wird die Angabe für *offset* mit einem *b* abgeschlossen, so wird die angegebene Zahl als Vielfaches von 512 byte interpretiert.
Wenn Sie für *datei* kein Argument angeben, müssen Sie *offset* ein Pluszeichen + voranstellen, damit *offset* nicht als Dateiname interpretiert wird.

offset nicht angegeben:
Die Ausgabe beginnt am Dateianfang.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *od*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_NUMERIC Legt die gültige Darstellung des Dezimalpunkts fest, wenn Gleichpunktzahlen geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Inhalt der Datei *text* oktal ohne Vorzeichen ausgeben:

```
$ cat text
Noch kann man alles verstehen.
$ od text
0000000 152626 101610 040222 100625 112500 112201 112500 100623
0000020 111605 121100 122605 114642 121605 104205 112513 012400
0000037
```

Beispiel 2 Inhalt der Datei *text* vom sechsten Byte an oktal und als ASCII-Zeichen ausgeben:

```
$ od -bc text 5.
0000005 222 201 225 225 100 224 201 225 100 201 223 223 205 242 100 245
          k   a   n   n       m   a   n       a   l   l   e   s       v
0000021 205 231 242 243 205 210 205 225 113 025
          e   r   s   t   e   h   e   n   .  \n
0000031
```

Siehe auch *Tabellen und Verzeichnisse*, *Zeichensatz EBCDIC*

paste Zeilen zusammenfügen (merge corresponding or subsequent lines of files)

paste fügt jeweils die n-ten Zeilen von mehreren Dateien (Format 1) oder alle Zeilen innerhalb einer Datei (Format 2) zusammen. Das Ergebnis gibt *paste* auf die Standard-Ausgabe aus.

Syntax

Format 1: paste[␣**-d**␣liste]␣datei␣...

Format 2: paste␣**-s**[␣**-d**␣liste]␣datei␣...

Format 1 **Die n-ten Zeilen von mehreren Dateien zusammenfügen**

paste[␣**-d**␣liste]␣datei␣...

paste fügt jeweils die n-ten Zeilen der Eingabedateien zusammen. Jede Datei wird dabei als Spalte einer Tabelle interpretiert; *paste* setzt die Spalten nebeneinander und gibt sie auf die Standard-Ausgabe aus (siehe *Beispiel 1*).

Keine Option angegeben
: Trennzeichen zwischen den ausgegebenen Spalten ist das Tabulatorzeichen.

-d␣liste
: (d - delimiter) Trennzeichen zwischen den ausgegebenen Spalten ist ein Zeichen aus *liste*.
paste verwendet die Zeichen in *liste* der Reihe nach. Ist *paste* beim letzten Zeichen angelangt, so geht es wieder an den Anfang der Liste. Die Zeilen der letzten Eingabedatei werden nicht mit einem Zeichen aus *liste*, sondern mit einem Neue-Zeile-Zeichen abgeschlossen.

  Für *liste* geben Sie eine Folge von beliebigen Zeichen an. Sie können auch folgende Escape-Sequenzen angeben: \n (Neue-Zeile-Zeichen), \t (Tabulatorzeichen), \\ (Gegenschrägstrich) und \0 (leere Zeichenfolge, nicht das Nullzeichen). Enthält *liste* Escape-Sequenzen, Leerzeichen oder Sonderzeichen der Shell, dann müssen Sie *liste* in Anführungszeichen "..." einschließen.

datei
: Name der Eingabedatei.
paste in diesem Format ist nur dann sinnvoll, wenn Sie mehrere Dateien angeben.

  Wenn Sie für *datei* einen Bindestrich - angeben, liest *paste* von der Standard-Eingabe.

Format 2 **Aufeinanderfolgende Zeilen zusammenfügen**

paste␣**-s**[␣**-d**␣liste]␣datei␣...

-s (s - subsequent lines)
paste fügt für jede Eingabedatei die Zeilen zu einer einzigen Zeile zusammen und schreibt diese Zeile auf die Standard-Ausgabe. Innerhalb jeder Ausgabezeile werden die Zeilen der Eingabedatei standardmäßig mit einem Tabulatorzeichen getrennt (siehe Option *-d*). Jede Ausgabezeile wird mit einem Neue-Zeile-Zeichen abgeschlossen.

-d␣liste
(d - delimiter)
An die Nahtstellen zwischen den einzelnen Teilzeilen setzt *paste* nicht ein Tabulatorzeichen, sondern ein Zeichen aus *liste*.
paste verwendet die Zeichen in *liste* der Reihe nach. Ist *paste* beim letzten Zeichen angelangt, so geht es wieder an den Anfang der Liste.

Für *liste* geben Sie eine Folge von beliebigen Zeichen an. Sie können auch folgende Escape-Sequenzen angeben: \n (Neue Zeile), \t (Tabulatorzeichen), \\ (Gegenschrägstrich) und \0 (leere Zeichenfolge, nicht das Nullzeichen). Enthält *liste* Escape-Sequenzen, Leerzeichen oder Sonderzeichen der Shell, dann müssen Sie *liste* in Anführungszeichen einschließen: "*liste*".

datei
Name der Eingabedatei. Sie können mehrere Dateien angeben.
Wenn Sie für *datei* einen Bindestrich - angeben, liest *paste* von der Standard-Eingabe.

Fehler `paste: line too long`
Ausgabezeilen dürfen nicht länger als 511 Zeichen werden.

`paste: too many files – limit 12`
Es dürfen im Format 1 nicht mehr als 12 Eingabedateien angegeben werden.

`paste: cannot open` *datei* `: no such files or directory`
datei ist nicht vorhanden.

`paste: cannot open` *datei* `: Permission denied`
Benutzer hat kein Leserecht.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *paste*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiele zu Format 1

Beispiel 1 Gegenüberstellen von sich entsprechenden Zeilen aus den Dateien *zahlen* und *buchstaben*:

Die Datei *zahlen* enthält die Zahlen von 1 bis 100:

```
  1
  2
  3
  .
  .
100
```

Die Datei *buchstaben* enthält die Kleinbuchstaben von a bis z:

```
a
b
c
.
.
z
```

```
$ paste zahlen buchstaben
  1     a
  2     b
  3     c
  .     .
  .     .
 25     y
 26     z
 27
  .
  .
100
```

Beispiel 2 Das aktuelle Dateiverzeichnis enthält die folgenden Dateien:

```
$ ls
korr
namen
plan
probe
prog.c
tst
verwalt
witze
```

Das folgende Kommando nummeriert diese Dateien (siehe Datei *zahlen* in *Beispiel 1*):

```
$ ls | paste zahlen -
  1     korr
  2     namen
  3     plan
  4     probe
  5     prog.c
  6     tst
  7     verwalt
  8     witze
  9
 10
  .
  .
100
```

Siehe auch *Beispiel 3*

Beispiel 3 Das aktuelle Dateiverzeichnis enthält dieselben Dateien wie in Beispiel 2. Das folgende Kommando gibt den Inhalt des aktuellen Dateiverzeichnisses dreispaltig aus. Bündige Spalten erhalten Sie allerdings nur, wenn die Dateinamen nicht über den nächsten Tabulatorstop hinausreichen.

```
$ ls | paste - - -
korr    namen   plan
probe   prog.c  tst
verwalt witze
```

Wie kommt die Ausgabe zustande? Vergleichen Sie das soeben eingegebene Kommando mit dem Kommando

```
$ paste datei1 datei2 datei3
```

Hier liest *paste* zuerst die ersten Zeilen aus allen drei Dateien und fügt sie zu einer Zeile zusammen. Anschließend liest *paste* die zweiten Zeilen usw.

Beim Kommando `ls | paste - - -` entspricht nun der erste Dateiname, den *paste* von der Standard-Eingabe liest, nämlich *korr*, der ersten Zeile aus *datei1*; der zweite Dateiname *namen* entspricht der ersten Zeile aus *datei2* usw.

Beispiel 4 Das aktuelle Dateiverzeichnis enthält dieselben Dateien wie in Beispiel 2. Sie wollen, wie in Beispiel 3, die Dateinamen dreispaltig ausgeben; zwischen zweiter und dritter Spalte soll jedoch statt eines Tabulatorzeichens ein Doppelpunkt stehen.

```
$ ls | paste -d"\t:" - - -
korr    namen:plan
probe   prog.c:tst
verwalt witze:
```

Beispiel zu Format 2

Beispiel 5 Die Datei *kunden* hat folgenden Inhalt:

```
hinz
schmidt
koeln
kunz
schulz
bremen
nepomuk
meier
plattling
```

```
$ paste -s kunden
hinz    schmidt koeln   kunz    schulz  bremen  nepomuk meier   plattling
```

Das folgende Kommando fügt nur jeweils drei Zeilen der Datei *kunden* zusammen, da als Trennzeichen nach jeder 3. Eingabezeile ein Neue-Zeile-Zeichen angegeben ist:

```
$ paste -s -d"\t\t\n" kunden
hinz    schmidt koeln
kunz    schulz  bremen
nepomuk meier   plattling
```

Siehe auch *cut*, *grep*, *pr*

patch Diff-Liste anwenden (apply changes to files)

patch verwendet eine Korrekturdatei (patch file) mit einer der drei Arten von Deltalisten (normal, mit Kontextangabe oder im *ed*-Format), die vom Kommando *diff* erstellt werden, und wendet diese Deltas auf eine Originaldatei an, wodurch eine korrigierte Version erstellt wird. Standardmäßig wird die Originaldatei durch die korrigierte Version ersetzt. Wenn Sie die Option *-b* angeben, wird die Originaldatei unter demselben Namen mit der Erweiterung „.orig“ gesichert. Mit der Option *-o* können Sie darüber hinaus eine Datei angeben, in die die Ausgabe geschrieben werden soll.

patch versucht beim Start, die Art der Diff-Liste festzustellen, sofern diese nicht bereits über die Option *-c*, *-e* oder *-n* festgelegt wurde. Diff-Listen mit Kontextangabe und normale Diff-Listen werden vom Kommando *patch* direkt angewandt, während mit der Option *-e* erstellte Diff-Listen einfach über eine Pipe an den Editor *ed* weitergeleitet werden.

Enthält die Korrekturdatei zusätzlichen Text vor und/oder nach den Korrekturanweisungen, wird dieser von *patch* übersprungen. Wenn die gesamte Diff-Liste bis zu einer bestimmten Spalte eingerückt ist, wird dies berücksichtigt.

Syntax

patch[␣**-bINR**][␣**-c**|␣**-e**|␣**-n**][␣**-d**␣dir][␣**-D**␣define][␣**-i**␣patchfile][␣**-o**␣outfile]
[␣**-p**␣num][␣**-r**␣rejectfile][file]

Optionen

-b Alle Originaldateien werden mit Suffix *.orig* gesichert, bevor die Korrekturanweisungen angewendet werden. Existiert die Sicherungsdatei bereits, wird sie überschrieben; bei mehreren Korrekturläufen wird nur beim ersten eine Sicherungsdatei angelegt. Wenn Sie zusätzlich die Option *-o* angeben, wird nicht die Originaldatei gesichert, sondern die Ausgabedatei, falls sie bereits existiert.

-c Interpretiert die Korrekturdatei als Diff-Liste mit Kontextangabe (Ausgabe von *diff*, wenn *-c* oder *-C* angegeben wurde).

-d␣dir
patch wechselt vor weiteren Aktionen in das Verzeichnis *dir*.

-D␣define
Markiert Änderungen mit den Anweisungen

```
#ifdef define
...
#endif
```

Beachten Sie, dass sich, anders als beim C-Compiler, zwischen der Option *-D* und dem Argument ein Leerzeichen befinden kann.

-e Interpretiert die Korrekturdatei als ein *ed*-Skript.

-i␣patchfile

patch liest die Korrekturanweisungen aus der Datei *patchfile*. Wenn Sie für *patchfile* einen Bindestrich angeben, liest *patch* von der Standard-Eingabe.
Eine Korrekturdatei enthält ein oder mehrere Korrektur-Skripts, eventuell zusammen mit Zusatzinformationen. Enthält eine Korrekturdatei mehrere Korrektur-Skript, sollte jedes Skript Informationen über Dateinamen enthalten (wie bei *diff -c* entstanden), damit *patch* die betroffenen Dateien automatisch finden kann.

patch wertet folgende Informationen aus:

`Index: pfadname`
pfadname benennt die zu korrigierende Datei

`*** pfadname`
pfadname gibt die „alte“ Datei an, auf deren Grundlage das Korrektur-Skript entstanden ist

`--- pfadname`
pfadname gibt die zu korrigierende Datei an (die Angabe hat Vorrang vor *Index*:)

Jedes Korrektur-Skript enthält Korrekturanweisungen, die einer der drei Arten von Diff-Listen entsprechen.

-i nicht angegeben:
patch liest von der Standard-Eingabe.

-l Eine beliebige Folge von Tabulator- und Leerzeichen in der Korrekturdatei passt zu einer beliebigen Folge von Tabulator- und Leerzeichen in der Eingabedatei. Bei normalen Zeichen muss jedoch weiterhin eine exakte Übereinstimmung vorliegen.

-n Interpretiert die Korrekturdatei als eine normale Diff-Liste.

-N Ignoriert Korrekturanweisungen, die bereits angewandt wurden. Standardmäßig werden solche Korrekturanweisungen zurückgewiesen.

-o␣outfile

Die korrigierte Version wird in *outfile* geschrieben. Mehrere korrigierte Dateien werden dabei aneinandergehängt. Betreffen verschiedene Korrektur-Skripts dieselbe Originaldatei, werden die folgenden Skripts jeweils auf eine Zwischendatei angewendet. Entsprechend werden mehrere Versionen der Originaldatei nach *outfile* geschrieben.

-p␣num

Steuert die Behandlung von in der Korrekturdatei gefundenen Pfadnamen. *num* gibt an, wieviele Schrägstriche vom Anfang des Pfadnamens entfernt werden sollen. (Alle dazwischenliegenden Verzeichnisnamen werden ebenfalls entfernt.) Bei relativen Pfadnamen erfolgt die Suche im aktuellen Verzeichnis oder in dem mit der Option *-d* angegebenen Verzeichnis.

-p nicht angegeben:
Es wird nur der Basisname ohne Pfad verwendet.

Beispiel

Angenommen, der Dateiname in der Korrekturdatei wäre
`/u/howard/src/blurfl/blurfl.c`

Setzen der Option *-p␣0* lässt den gesamten Pfadnamen unverändert. Das Setzen der Option *-p␣1* entfernt den führenden Schrägstrich im Pfadnamen:
`u/howard/src/blurfl/blurfl.c`

Die Angabe der Option *-p␣4* ergibt den folgenden Pfadnamen:
`blurfl/blurfl.c`

Wird die Option *-p* nicht angegeben, wird der Pfadname auf `blurfl.c` gekürzt.

-R *patch* soll den Sinn der Korrekturanweisungen umkehren, bevor sie auf die Originaldatei angewandt werden. Dies ist nötig, wenn die alten und neuen Dateien beim Erstellen der Korrektur vertauscht wurden. *patch* versucht, jeden Korrekturblock umzukehren, bevor er angewandt wird. Zurückgewiesene Korrekturanweisungen werden im umgekehrten Format in die Fehlerdatei geschrieben.
Die Option *-R* kann nicht bei *ed*-Skripts verwendet werden, da die Informationen zur Rekonstruktion der umgekehrten Korrekuranweisungen nicht ausreichen.

Kann der erste Korrekturblocks einer Korrekturdatei nicht angewendet werden, kehrt *patch* den Sinn der Korrekturanweisungen um und versucht, ihn auf diese Weise anzuwenden. Ist dies möglich, werden Sie gefragt, ob die Option *-R* gesetzt werden soll. Ist dies nicht möglich, trägt *patch* den Block in die Fehlerdatei ein und fährt normal fort. (Hinweis: Umgekehrte Korrekturanweisungen können mit dieser Methode nicht erkannt werden, wenn es sich um eine normale Diff-Liste handelt und das erste Kommando eine Einfügung ist, also eigentlich ein Löschvorgang sein sollte.)

-r␣rejectfile
Gibt die Datei an, in der die zurückgewiesenen Korrekturanweisungen gesammelt werden sollen.

-r nicht angegeben:
Die zurückgewiesenen Korrekturen werden in einer Datei gesammelt, die denselben Namen hat wie die Ausgabedatei, jedoch mit dem Suffix *.rej* versehen ist.

file
Pfadname der zu korrigierenden Datei.

file nicht angegeben:
patch versucht, den Dateinamen aus dem Vorspann zu den Korrekturanweisungen zu ermitteln.

Hinweis zur Arbeit mit *patch*

Wenn Sie regelmäßig Korrekturdateien erstellen, empfiehlt es sich, als erste Korrekturanweisung die Änderungsstufe mitzuführen. Wenn Sie eine Zeile „Prereq: “ in die Korrekturdatei stellen, können Sie verhindern, dass Benutzer falsche Korrekturen anwenden, ohne gewarnt zu werden.

Prüfen Sie, ob Sie im Kopf einer Diff-Liste mit Kontextangabe oder in der Zeile „Index:“ die Dateinamen korrekt angegeben haben. Wenn Sie Korrekturen in einem Unterverzeichnis durchführen wollen, teilen Sie dem Korrekturbenutzer mit, dass die Option *-p* entsprechend angegeben werden muss.

Vermeiden Sie möglichst umgestellte Korrekturen.

Stellen Sie zusammengehörende Korrekturen für den Fall eines Fehlers möglichst in separate Dateien.

Werden Fehlerdateien mit zurückgewiesenen Korrekturblöcken erstellt, beendet sich *patch* mit einem Status ungleich null. Wenn Sie eine Reihe von Korrekturen in einer Schleife anwenden, sollten Sie diesen Endestatus prüfen, damit eine spätere Korrektur nicht auf einer nur teilweise korrigierten Datei durchgeführt wird.

Wenn Code dupliziert wurde (zum Beispiel mit „#ifdef OLDCODE ... #else ... #endif“), kann *patch* nicht beide Versionen korrigieren. Wenn das Kommando überhaupt ausgeführt werden kann, korrigiert es möglicherweise die falsche Version, und informiert Sie darüber, dass die Korrekturen erfolgreich abgearbeitet wurden.

Wenn Sie eine bereits angewandte Korrektur nochmals anwenden, nimmt *patch* an, dass es sich um eine umgekehrte Korrektur handelt und bietet die Zurücknahme der Korrektur an.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *patch*:

LANG Gibt einen Standardwert für die Internationalisierungsvariablen an, die nicht gesetzt oder null sind. Ist *LANG* nicht gesetzt oder null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als wäre keine der Variablen definiert.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d.h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die die Interpretation von Byte-Folgen als Zeichen fest (z.B. Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Meldungen fest.

LC_TIME Legt das Format der Dateizeitstempels fest, der von *diff* in einem kontextabhängigen Eingabedatei geschrieben wird.

NLSPATH Legt die Position der Meldungskataloge für die Verarbeitung von *LC_MESSAGES* fest.

Endestatus

0 Erfolgreiche Beendigung.

1 Eine oder mehrere zurückgewiesene Korrekturanweisungen wurden in die Fehlerdatei geschrieben.

>1 Ein Fehler ist aufgetreten.

Datei */tmp/patch**

Siehe auch *diff, ed*

pathchk Pfadnamen überprüfen (check pathnames)

Das Kommando *pathchk* überprüft, ob ein oder mehrere Pfadnamen gültig sind (d.h. sie können zum Zugriff auf eine Datei oder zum Erstellen einer Datei verwendet werden, ohne einen Syntaxfehler zu verursachen) und ob sie portierbar sind (d.h. es ist keine Namensanpassung nötig). Weiterreichende Prüfungen zur Portierbarkeit können mit der Option *-p* durchgeführt werden.

Standardmäßig überprüft das Kommando *pathchk* die Komponenten aller Argumente *pathname* auf der Basis des zugrundeliegenden Dateisystems. In folgenden Fällen wird eine Fehlermeldung zu dem betroffenen Argument *pathname* ausgegeben:
- Es ist länger als die maximal zulässige Länge von Pfadnamen ({PATH_MAX} Byte).
- Es enthält eine Komponente, die länger ist als die maximal zulässige Länge von Dateinamen ({NAME_MAX} Byte) im jeweiligen Verzeichnis.
- Er enthält eine Komponente in einem Verzeichnis, das nicht durchsucht werden kann.
- Er enthält eine Komponente mit Zeichen, die im jeweiligen Verzeichnis ungültig sind.
- Die Namenslänge einer Datei oder Verzeichnisses in einem bs2fs-Dateisystem widerspricht nicht den Regeln eines bs2fs-Dateisystems.

Es ist kein Fehler für *pathchk*, wenn eine oder mehrere Komponenten eines Arguments *pathname* nicht existieren, solange eine Datei mit dem angegebenen Pfadnamen erstellt werden kann, die die oben aufgeführten Überprüfungen erfüllt.

Syntax

pathchk[␣**-p**]␣pathname...

Optionen

-p Führt die Überprüfungen nicht für das zugrundeliegende Dateisystem durch, sondern auf Grundlage allgemeiner Portabilitätsbedingungen. In folgenden Fällen wird eine Fehlermeldung zu dem betroffenen Argument *pathname* ausgegeben:
- Es ist länger als die maximal zulässige Länge für portierbare Pfadnamen ({_POSIX_PATH_MAX} Byte).
- Es enthält eine Komponente, die länger ist als die maximale Länge für portierbare Dateinamen ({_POSIX_NAME_MAX} Byte).
- Er enthält eine Komponente mit Zeichen, die nicht in der portierbaren Zeichenmenge für Dateinamen enthalten sind.

pathname
Der zu überprüfende Pfadname.

Internationale Umgebung

Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *pathchk* aus:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die Interpretation von Byte-Folgen als Zeichen fest (z.B. Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Meldungen fest.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Hinweis

Mit dem Kommando *test* kann überprüft werden, ob ein bestimmter Pfadname eine bereits vorhandene Datei angibt. Es gibt jedoch keine Auskunft darüber, ob eine Komponente des Pfadnamens abgeschnitten wurde (in einem Verzeichnis, für das die Funktion {_POSIX_NO_TRUNC} nicht aktiviert ist). Das Kommando *pathchk* prüft nicht, ob eine bestimmte Datei vorhanden ist. Es prüft lediglich, ob ein bestimmter Pfadname existiert bzw. ob er ohne Namensverkürzung erstellt werden kann. Mit der Option *noclobber* in der Shell (vgl. Beschreibung von *set*) kann eine Datei eindeutig erstellt werden.

Beispiel

Sie können folgendermaßen prüfen, ob alle Pfadnamen in einem importierten Datenaustauscharchiv auf dem aktuellen System zulässig und eindeutig sind:

```
pax -f archive | sed -e \*(hO/ == .*/s///\*(hO | xargs pathchk
if [ $? -eq 0 ]
then
    pax -r -f archive
else
    echo Investigate problems before importing files.
    exit 1
fi
```

Sie können folgendermaßen prüfen, ob alle Dateien in der aktuellen Verzeichnishierarchie auf ein anderes System übertragen werden können, das allgemeine Portabilitätsbedingungen erfüllt und das Kommando *pax* zur Verfügung stellt:

```
find . -print | xargs pathchk -p
if [ $? -eq 0 ]
then
    pax -w -f archive .
else
    echo Portable archive cannot be created.
    exit 1
fi
```

Sie können folgendermaßen überprüfen, ob ein angegebener Pfadname eine lesbare Datei benennt und ob eine Anwendung durch Erweiterung des Dateinamens eine Datei erstellen kann, deren Pfadname nicht automatisch verkürzt wird und die keine bereits vorhandene Datei überschreibt.

```
case $- in
    *C*)    reset="";;
    *)      reset="set +C"
            set -C;;
esac
test -r "$path" && pathchk "$path.out" &&
    rm "$path.out" > "$path.out"
if [ $? -ne 0 ]; then
    printf "%s: %s not found or %s.out fails \
creation checks.\n" $0 "$path" "$path"
    $reset  # reset the noclobber option in case a trap
            # on EXIT depends on it
    exit 1
fi
$reset
PROCESSING < "$path" > "$path.out"
```

Diesem Beispiel liegt folgendes zugrunde:

1. *PROCESSING* stellt den Code dar, den die Anwendung benützt, um *$path* zu verwenden, sobald überprüft worden ist, dass *$path.out* die gewünschten Bedingungen erfüllt.
2. Der Status der Option *noclobber* ist unbekannt, wenn dieser Code aufgerufen wurde, und sollte beim Verlassen wieder auf den Status gesetzt werden, den er beim Aufrufen dieses Codes hatte. (Die Variable *reset* wird in diesem Beispiel zum Wiederherstellen des ursprünglichen Status verwendet.)

3. Beachten Sie die Verwendung der folgenden Konstruktion:

```
rm "$path.out" > "$path.out"
```

   a) Das Kommando *pathchk* hat zu diesem Zeitpunkt bereits geprüft, dass *$path.out* nicht verkürzt wird.

   b) Wenn die Option *noclobber* gesetzt ist, prüft die Shell vor dem Aufruf von *rm*, ob *$path.out* nicht bereits vorhanden ist.

   c) Wenn die Shell *$path.out* erfolgreich erstellt hat, entfernt *rm* diese wieder, so dass die Anwendung die Datei im Schritt *PROCESSING* erneut erstellen kann.

   d) Wenn die Datei beim Aufrufen des Schritts *PROCESSING* bereits vorhanden sein soll, sollte `rm "$path.out" > "$path.out"` durch `> "$path.out"` ersetzt werden. Hierdurch wird sichergestellt, dass die Datei *$path.out* noch nicht vorhanden war, und sie wird gleichzeitig für die Verwendung durch *PROCESSING* angelegt.

Siehe auch *test*

pax Bearbeitung portierbarer Archive (portable archive interchange)

Das Kommando *pax* schreibt Dateien in Archivdateien, liest sie oder erstellt Listen dieser Dateien und kopiert Dateiverzeichnishierarchien. Es werden mehrere Archivformate unterstützt; siehe Option *-x␣format*.

Welche Aktion ausgeführt wird, hängt davon ab, welche der Optionen *-r* und *-w* angegeben wurde. Die vier möglichen Kombinationen von *-r* und *-w* ergeben die vier Betriebsmodi: List-, Lese-, Schreib- und Kopiermodus, entsprechend den vier unter Syntax aufgeführten Formaten.

list Im Listmodus (list mode, d.h. weder *-r* noch *-w* sind angegeben) liest *pax* eine Archivdatei von der Standard-Eingabe und schreibt die Namen der gesicherten Dateien, deren Pfadnamen auf das angegebene Muster *muster* passen, auf die Standard-Ausgabe. Wenn es sich bei einer angegebenen Datei um ein Dateiverzeichnis handelt, wird die davon ausgehende Dateihierarchie ebenfalls ausgegeben.

read Im Lesemodus (read mode, d.h. *-r* ist angegeben, *-w* jedoch nicht) liest *pax* eine Archivdatei von der Standard-Eingabe und extrahiert die Dateien, deren Pfadnamen auf das angegebene Muster *muster* passen. Wenn es sich bei einer extrahierten Datei um ein Dateiverzeichnis handelt, wird die davon ausgehende Dateihierarchie ebenfalls extrahiert. Die extrahierten Dateien werden relativ zu der aktuellen Dateihierarchie erstellt.

write Im Schreibmodus (write mode, d.h. *-w* ist angegeben, *-r* jedoch nicht) schreibt *pax* den Inhalt der *datei*-Operanden in einem Archivformat auf die Standard-Ausgabe. Wenn kein Operand *datei* angegeben ist, wird eine Liste der zu kopierenden Dateien, jeweils eine Datei pro Zeile, von der Standard-Eingabe gelesen. Bei Angabe eines Dateiverzeichnisses werden alle Dateien in der davon ausgehenden Dateihierarchie kopiert.

copy Im Kopiermodus (copy mode, d.h. sowohl *-r* als auch *-w* sind angegeben) kopiert *pax* die *datei*-Operanden in das Zieldateiverzeichnis *dateiverzeichnis*.

Wenn keine *datei*-Operanden angegeben sind, wird eine Liste der zu kopierenden Dateien, jeweils eine Datei pro Zeile, von der Standard-Eingabe gelesen. Bei Angabe eines Dateiverzeichnisses werden alle Dateien in der davon ausgehenden Dateihierarchie kopiert.

Der Kopiervorgang läuft so ab, als würden die kopierten Dateien in eine Archivdatei geschrieben und anschließend extrahiert werden, außer dass hierbei möglicherweise Hard Links zwischen den Originaldateien und den kopierten Dateien bestehen.

Wenn im Lese- oder Kopiermodus zum Extrahieren einer Datei Zwischendateiverzeichnisse erforderlich sind, werden diese von *pax* erstellt.

Wenn nicht mindestens eine Datei mit den angegebenen Operanden *muster* oder *datei* übereinstimmt, schreibt *pax* für jeden Operanden, für den keine Übereinstimmung existiert, eine Diagnosemeldung auf die Standard-Fehlerausgabe und beendet sich mit einem Endestatus ungleich null.

Die unterstützten Archivformate werden beim Lesen von Archiven automatisch erkannt. Das Standardausgabeformat für Archive beim Schreiben (keine Option *-x* angegeben) ist das erweiterte *tar*-Format.

Ein Archiv kann sich über mehrere Dateien erstrecken. Das Kommando *pax* stellt fest, welche Datei als nächste gelesen oder geschrieben werden soll.

Wenn das ausgewählte Archivformat die Angabe von Links zwischen Dateien unterstützt, meldet *pax* einen Fehler, wenn bei der Extrahierung kein Link zwischen diesen Dateien erstellt werden kann.

Syntax

Format 1: pax[␣**-cdnv**][␣**-f**␣archiv][␣**-s**␣anweisung] ...[muster...]

Format 2: pax␣**-r**[␣**-cdiknuv**][␣**-f**␣archiv][␣**-p**␣zeichenkette] ...
[␣**-s**␣anweisung] ... [muster ...]

Format 3: pax␣**-w**[␣**-dituvX**][␣**-b**␣blockgröße][␣**-a**][␣**-f**␣archiv]
[␣**-s**␣anweisung] ... [␣**-x**␣format][datei ...]

Format 4: pax␣**-r**␣**-w**[␣**-diklntuvX**][␣**-p**␣zeichenkette] ... [␣**-s**␣anweisung] ...
[datei ...] dateiverzeichnis

Optionen

-r Liest eine Archivdatei von der Standard-Eingabe. .

-w Schreibt Dateien im angegebenen Archivformat auf die Standard-Ausgabe.

-a Hängt Dateien an das Ende des Archivs an.

-b␣blockgröße
Stellt die Daten in Blöcken in die Archivdatei. Dabei wird die Blockgrösse als positive dezimale Ganzzahl angegeben. Geräte und Archivformate können möglicherweise Beschränkungen für die Blockung festlegen. Bei der Eingabe wird die Blockung automatisch festgestellt. Die Standardblockung beim Erstellen von Archiven hängt vom Archivformat ab (siehe Option *-x* unten). Um ein portables Archiv zu erhalten, darf die Blockgröße nicht größer als 32 KByte sein.

-c Wählt alle Dateien aus, außer den von den Operanden *muster* oder *datei* angegebenen.

-d Bei Dateiverzeichnissen wird die von diesem Dateiverzeichnis ausgehende Dateihierarchie nicht mitkopiert bzw. eingelesen bzw. archiviert.

-f␣archiv
Gibt den Pfadnamen des Eingabearchivs (im Listen- und Lesemodus) oder des Ausgabearchivs (Schreibmodus) an. Dieses Archiv wird anstelle der Standard-Eingabe bzw. Standard-Ausgabe verwendet.

-i Dateien können interaktiv umbenannt werden. Für jede archivierte Datei, deren Name zu einem Operanden *muster* passt, oder für jede Datei, die mit einem Operanden *datei* übereinstimmt, wird eine Eingabeaufforderung nach `/dev/tty` geschrieben. Diese Eingabeaufforderung enthält den Namen der Datei. Von `/dev/tty` wird eine Zeile gelesen. Ist diese Zeile leer, wird die Datei übergangen. Wenn die Zeile aus einem einzelnen Punkt besteht, wird die Datei ohne Namensänderung verarbeitet. Andernfalls wird der Name durch den Inhalt der Zeile ersetzt. Wenn ein Dateiendezeichen beim Lesen einer Antwort auftritt oder wenn `/dev/tty` nicht zum Lesen oder Schreiben geöffnet werden kann, beendet sich das Kommando *pax* umgehend mit einem Endestatus ungleich null.

-k Verhindert das Überschreiben bestehender Dateien.

-l Erstellt Links zwischen Dateien. Im Kopiermodus werden zwischen den Quell- und Zieldateihierarchien, wenn möglich, Hard Links erstellt.

-n Wählt die erste archivierte Datei aus, die zu einem Operanden *muster* passt. Für jeden Operanden *muster* wird immer nur eine passende Datei ausgewählt (Dateihierarchien, die von Dateiverzeichnissen ausgehen, sind von dieser Einschränkung nicht betroffen).

-p␣zeichenkette
Legt Dateimerkmale (Berechtigungen) fest. Das Argument *zeichenkette* beinhaltet die Dateimerkmale, die bei der Extrahierung beibehalten oder ignoriert werden sollen. Die Zeichenkette *zeichenkette* besteht aus den Spezifizierungszeichen a, e, m, o und p. In derselben Zeichenkette können mehrere Merkmale verknüpft werden, und die Option *-p* kann mehrfach angegeben werden.
Die Spezifizierungszeichen haben folgende Bedeutungen:

a Datum und Uhrzeit des letzten Dateizugriffs werden auf das aktuelle Datum gesetzt.

e Benutzer-ID, Gruppen-ID, Zugriffsrechte (siehe *chmod*), Datum und Uhrzeit des letzten Zugriffs, Datum und Uhrzeit der letzten Änderung werden beibehalten.

m Datum und Uhrzeit der letzten Dateiänderung werden nicht beibehalten.

o Benutzer-ID und Gruppen-ID werden beibehalten.

p Zugriffsrechte werden beibehalten.

In der obigen Liste bedeutet „beibehalten“, dass ein im Archiv gespeichertes Attribut abhängig von den Berechtigungen des aufrufenden Prozesses an die extrahierte Datei weitergegeben wird. Andernfalls wird das Attribut wie bei Neuerstellung einer Datei festgelegt.

Wenn weder das Spezifizierungszeichen e noch o angegeben ist oder die Benutzer- und Gruppen-ID nicht beibehalten werden, setzt *pax* die s-Bit für den Dateimodus nicht.

Wenn eines dieser Attribute aus irgendeinem Grund nicht beibehalten werden kann, schreibt *pax* eine Diagnosemeldung auf die Standard-Fehlerausgabe. Wenn ein Attribut nicht beibehalten werden kann, hat dies zwar Auswirkungen auf den Endestatus, die extrahierte Datei wird jedoch nicht gelöscht.

Wenn Spezifizierungszeichen in einem der Argumente *zeichenkette* doppelt vorkommen oder mit anderen Argumenten in Konflikt geraten, erhalten die zuletzt eingegebenen Argumente Vorrang. Wenn beispielsweise *-p␣eme* angegeben wird, werden Datum und Uhrzeit der letzten Dateiänderung beibehalten.

-s␣anweisung
Ändert Dateinamen, die von den Operanden *muster* oder *datei* angegeben wurden, entsprechend der Anweisung *anweisung*. Hierbei wird die Syntax des Kommandos *ed* verwendet. Die Begriffe „Adresse“ und „Zeile“ haben im Kontext des Kommandos *pax* keine Bedeutung. Es wird folgendes Format verwendet:

```
-s /old/new/[gp]
```

Wie auch bei *ed*, ist *old* ein einfacher regulärer Ausdruck, und *new* kann &-Zeichen, \n (wobei \n eine Ziffer ist), Rückverweise und Unterausdrücke enthalten. Die Zeichenkette *old* kann auch Neue-Zeile-Zeichen enthalten.

Als Begrenzungszeichen können alle Zeichen verwendet werden (hier z.B. /). Mehrere Ausdrücke *-s* sind erlaubt. Die Ausdrücke werden in der angegebenen Reihenfolge ausgewertet bis zur ersten erfolgreichen Ersetzung. Das optionale nachgestellte *g* wirkt wie bei *ed* beschrieben. Das optionale nachgestellte *p* bewirkt, dass erfolgreiche Ersetzungen auf die Standard-Fehlerausgabe geschrieben werden. Dateinamen, die durch eine leere Zeichenkette ersetzt werden, werden beim Lesen und Schreiben von Archiven ignoriert.

-t Datum und Uhrzeit des letzten Zugriffs auf archivierte Dateien werden auf den Wert gesetzt, den sie vor dem Zugriff durch *pax* hatten.

-u Ignoriert Dateien, die älter sind (älteres Datum/Uhrzeit der letzten Änderung aufweisen) als eine bereits existierende gleichnamige Datei. Im Lesemodus wird eine archivierte Datei, die denselben Namen wie eine Datei im Dateisystem hat, extrahiert, wenn die archivierte Datei aktueller ist als die im Dateisystem. Im Schreibmodus wird eine Datei nur dann gesichert, wenn es im Archiv keine gleichnamige Datei jüngeren Datums gibt. Im Kopiermodus wird die Datei in der Zielhierarchie durch die Datei in der Quellhierarchie oder durch einen Link auf die Datei in der Quellhierarchie ersetzt, wenn die Datei in der Quellhierarchie aktueller ist.

-v Im Listenmodus wird ein ausführliches Inhaltsverzeichnis auf die Standard-Ausgabe geschrieben. Sonst werden die Pfadnamen der gesicherten Dateien auf die Standard-Fehlerausgabe geschrieben.

-x␣format
Gibt das Archivausgabeformat an. Das Kommando *pax* erkennt die folgenden Formate:

cpio
Erweitertes *cpio*-Austauschformat. Die Standardblockgröße für dieses Format für zeichenorientierte Archivdateien ist 512. Implementierungen von *pax* unterstützen alle Werte für die Blockgröße bis einschließlich 32256, die ein Vielfaches von 512 sind.

ustar
Erweitertes *tar*-Austauschformat. Die Standardblockgröße für dieses Format für zeichenorientierte Archivdateien ist 1024. Implementierungen von *pax* unterstützen alle Werte für die Blockgröße bis einschließlich 32256, die ein Vielfaches von 512 sind.

Bei jedem Versuch, Daten an eine Archivdatei in einem anderen Format als im vorhandenen Archivformat anzuhängen, beendet sich *pax* sofort mit einem Endestatus ungleich null.

-X Beim Durchlauf durch eine Dateihierarchie, die durch einen Pfadnamen angegeben ist, wechselt *pax* nicht in Dateiverzeichnisse, die in einem anderen Dateisystem liegen.

Die Optionen, die mit den Namen von Dateien arbeiten (*-c*, *-i*, *-n*, *-s*, *-u* und *-v*) werden wie im Folgenden beschrieben nacheinander ausgewertet.

Im Lesemodus werden die Dateien aufgrund der vom Benutzer angegebenen Operanden *muster* und entsprechend den Änderungen durch die Optionen *-c*, *-n* und *-u* ausgewählt. Anschließend ändern die Optionen *-s* und *-i* in dieser Reihenfolge die Namen der ausgewählten Dateien. Die Option *-v* gibt die Namen aus, die sich aufgrund dieser Änderungen ergeben.

Im Schreibmodus werden die Dateien aufgrund der vom Benutzer angegebenen Pfadnamen und entsprechend den Änderungen durch die Optionen *-n* und *-u* ausgewählt. Anschließend ändern die Optionen *-s* und *-i* in dieser Reihenfolge die Namen der ausgewählten Dateien. Die Option *-v* gibt die Namen aus, die sich aufgrund dieser Änderungen ergeben.

Wenn sowohl die Option *-u* als auch die Option *-n* angegeben werden, berücksichtigt *pax* eine Datei nur, wenn sie aktueller als die Datei ist, mit der sie verglichen wird.

dateiverzeichnis
Pfadname des Zieldateiverzeichnisses für den Kopiermodus.

datei
Pfadname einer zu kopierenden oder archivierenden Datei.

muster
Muster, das auf einen oder mehrere Pfadnamen von archivierten Dateien passt. Standardmäßig (wenn kein *muster* angegeben ist) werden alle Dateien im Archiv ausgewählt.

Standard-Eingabe (stdin)

Im Schreibmodus wird die Standard-Eingabe nur verwendet, wenn keine *datei*-Operanden angegeben sind. Die Standard-Eingabe muss dann eine Textdatei sein, die eine Liste der Pfadnamen enthält. Die Datei muss einen Pfadnamen pro Zeile beinhalten und es dürfen keine führenden oder nachgestellten Leerzeichen vorkommen.

Im Listen- und Lesemodus muss die Standard-Eingabe eine Archivdatei sein.

Andernfalls wird die Standard-Eingabe nicht verwendet.

Datei

Die vom Argument *archiv* angegebene Eingabedatei bzw. die Standard-Eingabe, wenn das Archiv von ihr gelesen wird, ist eine Datei, die entsprechend einem der bei der Option *-x* aufgeführten Archivformate formatiert ist.

Auf bzw. von *stdin/stdout* werden Eingabeaufforderungen geschrieben und Antworten gelesen.

Standard-Ausgabe (stdout)

Wenn *-f* im Schreibmodus nicht angegeben ist, ist die Standard-Ausgabe das Archiv, das entsprechend einem der bei der Option *-x* aufgeführten Archivformate formatiert ist.

Im Listenmodus wird das Inhaltsverzeichnis der ausgewählten Dateien im folgenden Format auf die Standard-Ausgabe geschrieben: .

```
"%s\n", pathname
```

Ist im Listenmodus die Option *-v* angegeben, wird das Inhaltsverzeichnis der ausgewählten Dateien in folgenden Formaten auf die Standard-Ausgabe geschrieben:

Pfadnamen, die Hard Links zu vorherigen Dateien darstellen, werden im folgenden Format geschrieben: .

```
"%s == %s\n", ls_-l_listing, linkname
```

Alle anderen Pfadnamen werden im folgenden Format geschrieben:

```
 "%s\n", ls_-l_listing
```

Hierbei ist *ls_-l_*listing das Format, das vom Kommando *ls* mit der Option *-l* erzeugt wird.

Standard-Fehlerausgabe (stderr)

Wenn *-v* im Lese-, Schreib- oder Kopiermodus angegeben ist, schreibt *pax* die Pfadnamen, die es verarbeitet, im folgenden Format auf die Standard-Fehlerausgabe:

```
"%s\n", pathname
```

Diese Pfadnamen werden bei Beginn der Verarbeitung der Datei ausgegeben. Das abschließende Neue-Zeile-Zeichen wird ausgegeben, nachdem die Datei gelesen oder geschrieben wurde.

Wenn die Option *-s* angegeben ist und sich ein nachgestelltes *p* in der Ersatzzeichenkette befindet, werden Ersetzungen im folgenden Format auf die Standard-Fehlerausgabe geschrieben:

```
"%s >> %s\n", original_pathname, new_pathname
```

Eventuelle Meldungen über das Format des Archivs werden ebenfalls auf die Standard-Fehlerausgabe geschrieben.

Ausgabedateien

Im Lesemodus haben die extrahierten oder kopierten Dateien den Typ der archivierten Datei.

Im Schreibmodus ist die mit dem Argument *-f* benannte Ausgabedatei entsprechend einem der bei der Option *-x* aufgeführten Archivformate formatiert.

Auswirkung von Fehlern

Wenn *pax* beim Lesen eines Archivs eine Datei oder einen Link nicht erstellen kann, beim Schreiben eines Archivs eine Datei nicht finden kann oder bei gesetzter *-p*-Option die Benutzer-ID, Gruppen-ID bzw. die Zugriffsrechte nicht beibehalten kann, wird eine Diagnosemeldung auf die Standard-Fehlerausgabe geschrieben und ein Endestatus ungleich null zurückgegeben. Die Verarbeitung wird jedoch fortgesetzt. Wenn *pax* keinen Link auf eine Datei erstellen kann, erstellt es standardmäßig keine zweite Kopie der Datei.

Wenn die Extrahierung einer Datei aus einem Archiv vorzeitig durch ein Signal oder einen Fehler beendet wird, hat *pax* möglicherweise nur einen Teil der Datei extrahiert, oder (wenn die Option *-n* nicht angegeben wurde) hat es möglicherweise eine Datei unter dem vom Benutzer angegebenen Namen extrahiert, die jedoch nicht die vom Benutzer gewünschte Datei ist. Darüber hinaus können Zugriffsrechte von extrahierten Dateiverzeichnissen möglicherweise zusätzliche Bit aus der Schutzbit-Maske sowie nicht korrekte Datums- und Uhrzeitangaben der letzten Änderung und des letzten Zugriffs haben.

Internationale Umgebung

Die folgenden Umgebungsvariablen wirken sich auf die Ausführung von *pax* aus:

LANG Gibt einen Standardwert für Internationalisierungsvariablen an, die nicht gesetzt oder null sind. Ist *LANG* nicht gesetzt oder null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich *pax* so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d.h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für das Verhalten von Bereichen, Äquivalenzklassen und Sortierelementen fest, die in Ausdrücken für den Musterabgleich für den Operanden *muster*, im einfachen regulären Ausdruck für die Option *-s* und im erweiterten regulären Ausdruck, der für das Schlüsselwort *yesexpr* in der Kategorie *LC_MESSAGES* definiert wurde, verwendet werden.

LC_CTYPE Legt die internationale Umgebung für die Interpretation von Byte-Folgen als Zeichen fest (zum Beispiel Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien), für das Verhalten von Zeichenklassen, die im erweiterten regulären Ausdruck verwendet werden, der für das Schlüsselwort *yesexpr* in der Kategorie *LC_MESSAGES* definiert wurde, sowie für den Musterabgleich.

LC_MESSAGES
Legt die internationale Umgebung für die Verarbeitung von ja/nein-Antworten fest, sowie Format und Sprache von Diagnosemeldungen, die *pax* ausgibt.

LC_TIME Legt das Format und den Inhalt von Datums- und Uhrzeitangaben fest, wenn die Option *-v* angegeben ist.

NLSPATH Legt die Position der Meldungskataloge für die Verarbeitung von *LC_MESSAGES* fest.

Hinweis Dieser Hinweis gilt nur für die Anwender, die im SINIX Archive mit den SINIX-Kommandos tar und cpio erzeugt haben.

Die Option *-p* wurde eingeführt, um die Unterschiede zwischen den herkömmlichen Implementierungen *tar* und *cpio* anzugleichen. Diese beiden Kommandos verwenden insbesondere *-m* auf unterschiedliche Art und Weise. Die Option *-p* bietet darüber hinaus erweiterte Möglichkeiten für die einheitliche Adressierung künftiger Dateiattribute, beispielsweise für erweiterte Sicherungssysteme oder Hochleistungsarchive. Von den möglichen Kombinationen werden meistens nur zwei Modi verwendet:

-p␣e „Alles beibehalten“ wird vom herkömmlichen Superuser verwendet, der über alle entsprechenden Berechtigungen verfügt, um alle Dateiattribute so beizubehalten, wie sie im Archiv aufgezeichnet wurden. Das Flag e ist die Summe aus o und p.

-p␣p „Beibehalten“ der Zugriffsrechte. Wird vom Benutzer mit regulären Berechtigungen verwendet, der die Dateiattribute außer dem Eigentümer beibehalten möchte. Datum- und Uhrzeitangaben der Datei werden standardmäßig beibehalten. Mit zwei anderen Flags können diese jedoch deaktiviert und Datum- und Uhrzeit der Extrahierung verwendet werden.

Für Teile der beschriebenen Funktionalität sind entspechende Berechtigungen für den Aufrufer von *pax* erforderlich, insbesondere beim Erstellen von block- oder zeichenorientierten Gerätedateien, beim Wiederherstellen von Datum und Uhrzeit des Dateizugriffs (Option *-t*), sofern der Benutzer nicht Eigentümer der Dateien ist, oder für das Beibehalten des Dateieigentümers, der Gruppe und des Modus (Option *-p*).

Beispiel

Das folgende Kommando legt ein Archiv mit dem Namen *archiv* an, das die Dateien *datei1* bis *datein* sowie das Dateiverzeichnis *dir1* mit allen seinen Unterverzeichnissen enthält:

```
$ pax –w –f archiv datei1...datein dir1
```

Die folgenden Kommandos kopieren die Dateiverzeichnishierarchie *olddir* in *newdir*:

```
mkdir newdir
pax –rw olddir newdir
```

Das folgende Kommando liest das Archiv *a.pax*, wobei alle von `/usr` ausgehenden Dateiverzeichnisse/Dateien des Archivs relativ zum aktuellen Dateiverzeichnis extrahiert werden.

```
pax –r –s ',/old/*usr//*,,' –f a.pax
```

pdbl Benutzerspezifischen Programm-Cache einrichten und verwalten (set up and manage user-specific program cache)

Dieses Kommando kann von jedem Benutzer aufgerufen werden. Es können benutzerspezifische Programm-Caches gehalten werden, die vom jeweiligen Nutzer selbst zu verwalten sind. Der Scope eines benutzerspezifischen Programm-Cache ist wahlweise

SESSIONWIDE alle Prozesse einer Sitzung sind angeschlossen

USERWIDE alle Prozesse einer User-ID sind angeschlossen

Syntax

pdbl{␣**-s**[␣sid]|␣**-u**}␣**-i**

pdbl{␣**-s**[␣sid]|␣**-u**}␣**-e**␣größe

pdbl{␣**-s**[␣sid]|␣**-u**}{␣**-a**|␣**-d**}

pdbl{␣**-s**[␣sid]|␣**-u**}␣**-D**

pdbl{␣**-s**[␣sid]|␣**-u**}␣**-b**␣pfad

pdbl{␣**-s**[␣sid]|␣**-u**}␣**-l**[␣element]

pdbl{␣**-s**[␣sid]|␣**-u**}␣**-r**␣element

pdbl␣**-h**

Optionen

-s sid
Der Programm-Cache einer Sitzung (Scope *SESSIONWIDE*) wird ausgewählt. *sid* ist die ID der gewünschten Sitzung. Wird *sid* nicht angegeben, dann wird automatisch die aktuelle Sitzung genommen.

Wird die Option *-s* gewählt, dann beziehen sich alle folgenden Optionen auf den Programm-Cache der gewählten bzw. aktuellen Sitzung.

-u Der Programm-Cache der Benutzer-ID (Scope *USERWIDE*) wird ausgewählt.

Wird die Option *-u* gewählt, dann beziehen sich alle folgenden Optionen auf den Programm-Cache des aktuellen Benutzers.

-i Der Status des Programm-Cache und statistische Daten u.a. über Größe und Belegung werden in folgendem Format auf der Standard-Ausgabe ausgegeben:

```
Cache name            CREATED: datum zeit   STATE: status
                      SIZE: größe MB      ENTRIES: einträge
                      FREE PAGES: seiten
```

name — Der Name des Programm-Cache setzt sich zusammen aus den Buchstaben *DBL*, dem Scope (*S* für *SESSIONWIDE* oder *U* für *USERWIDE*) und der entsprechenden ID der Sitzung bzw. des Benutzers. So hat z. B. der Programm-Cache der Sitzung *504* den Namen *DBLS504*.

datum — Datum der Einrichtung des Programm-Cache.

zeit — Uhrzeit der Einrichtung des Programm-Cache.

status — Aktueller Status des Programm-Cache (*active*, *inactive* oder *in delete*).

größe — Gesamte Cache-Größe in Megabyte.

einträge — Aktuelle Anzahl der gespeicherten Core-Images.

seiten — Anzahl der noch verfügbaren Speicherseiten im Cache. Im ungünstigsten Fall ist für ein weiteres Core-Image eine verfügbaren Speicherseite weniger vorhanden, da die Erweiterung des Cache-Katalogs eine Seite beansprucht.

-e größe

Der Programm-Cache wird in der angegebenen Größe (in Megabyte) eingerichtet und aktiviert. Die maximale Größe des Cache wird nicht von *pdbl*, sondern von system- und taskspezifischen Einstellungen bestimmt. Die Größe des Cache kann das ADDRESS-SPACE-LIMIT der Benutzerkennung nicht überschreiten.

-a Der Programm-Cache wird aktiviert und ab sofort bei Ladevorgängen berücksichtigt.

-d Der Programm-Cache wird deaktiviert und ab sofort bei Ladevorgängen nicht mehr berücksichtigt. Sind keine aktuell gespeicherten Core-Images mehr vorhanden, so wird der Programm-Cache aufgelöst (analog Option *-D*). Sind noch aktuell gespeicherte Core-Images vorhanden, bleibt der Program-Cache im Status *inactive* erhalten.

-D Der Programm-Cache wird aufgelöst und ab sofort bei Ladevorgängen nicht mehr berücksichtigt.

Ist der Programm-Cache wegen bereits eingeleiteter Ladevorgänge gesperrt, bleibt er im Status *in delete* erhalten, bis alle laufenden Ladevorgänge abgeschlossen sind.

-b pfad

Das Core-Image eines durch seinen Pfadnamen identifizierten Programms wird im Programm-Cache gespeichert. Das unter *pfad* angegebene Programm muss ausführbar sein.

-l Es wird eine Liste aller aktuell im Programm-Cache gespeicherten Core-Images in folgendem Format auf der Standard-Ausgabe ausgegeben:

```
element größe datum zeit bibliothek
```

element — Name des Programmelements in der PLAM-Bibliothek oder der einfache Dateiname des Programms im UFS.

größe Anzahl der Speicherseiten, die das Core-Image belegt.

datum Datum des letzten Zugriffs auf das Core-Image.

zeit Uhrzeit des letzten Zugriffs auf das Core-Image.

status Aktueller Status des Programm-Cache (*active*, *inactive* oder *in delete*).

bibliothek Name der PLAM-Bibliothek aus der geladen wurde oder Pfadname des LLM im UFS.

-I element

Es werden Detailinformationen über das Core-Image *element* im Programm-Cache in folgendem Format auf der Standard-Ausgabe ausgegeben:

```
element          CREATED : cdatum czeit     ACCESS: adatum azeit
                 START AT: stadresse        CACHESIZE: cgröße kB
                                            USECOUNT: anzahl
                 ------------------------------------------
                 SLICES  : sl LOADADDR:     SIZE:
                              loadresse     sgröße kB
                              . . . . .     . . . . .
                 ------------------------------------------
info
```

Erläuterung der Ausgabe

element Name des Programmelements in der PLAM-Bibliothek oder der einfache Dateiname des Programms im UFS.

cdatum Datum der Erstellung des Core-Image.

czeit Uhrzeit der Erstellung des Core-Image.

adatum Datum des letzten Zugriffs auf das Core-Image.

azeit Uhrzeit des letzten Zugriffs auf das Core-Image.

stadresse Startadresse des Core-Image beim Ablauf.

cgröße Anzahl der Kilobytes, die das Core-Image belegt.

anzahl Anzahl der Ladezugriffe auf das Core-Image.

sl Anzahl der Slices.

loadresse Ladeadresse der Slices.

sgröße Anzahl der Kilobytes, die die Slices belegen.

info Information über die Herkunft des Core-Image.

-r element
Aus dem Programm-Cache wird das Core-Image *element* gelöscht. Wird als Element ein * angegeben, so werden alle Core-Images des Programm-Cache gelöscht.

-h Es wird eine Übersicht über alle Optionen und Parameter ausgegeben.

Beispiel

```
# pdbl -u -e 16         # Programm-Cache einrichten

# pdbl -u -i            # Status ausgeben
Cache DBLU101           CREATED: 01/27/09 16:04:01       STATE: active
                        SIZE: 16 MB     ENTRIES: 0
                        FREE PAGES: 4095

# cd /usr/demo/bin

# ls -l hello           # Zeige LLM in UFS
-rwxr-xr-x   1 ROOT    SYSROOT   364544 Feb 20 11:09 hello

# pdbl -u -b hello      # Core-Image erzeugen und speichern

# pdbl -u -l            # Core-Images im Programm-Cache ausgeben
hello           57 Jan 27 16:05:37 /usr/demo/bin/hello
```

ping Senden von Echo-Request-Paketen an Netzwerkkomponenten (send echo requests to network hosts)

Mit *ping* kann die Erreichbarkeit von Netzwerkkomponenten getestet werden.

Das Kommando verwendet das ICMP- bzw. ICMPv6-Protokoll. Es sendet Echo-Request-Pakete an bestimmte Netzwerkkomponenten, um zu ermitteln, ob diese über das Netzwerk erreichbar sind oder nicht. Diese Netzwerkkomponenten antworten, indem sie Echo-Reply-Pakete zurücksenden.

Syntax

ping [**-nv**] host [timeout]

ping -s[**nv**] host [packetsize [packetcount]]

ping -l host

host Angabe der Netzwerkkomponente als (FQDN-)Name, IPv4-Adresse oder IPv6-Adresse.

-n Statt des Hostnamens wird die Hostadresse angezeigt.

-v Verbose-Modus; in bestimmten Fehlerfällen werden zusätzliche Meldungen ausgegeben.

-s Statistikanzeige (1 Echo-Request pro Sekunde); folgende Informationen werden angezeigt:

- Zeit bis zur Antwort der Netzwerkkomponente,
- Größe der einzelnen Datenpakete,
- Anzahl der gesendeten, erhaltenen und verlorenen Datenpakete,
- die kürzeste, die längste und die durchschnittliche Antwortzeit.

-l Alle IPv4- und IPv6-Adressen des Hosts ausgeben (lookup).

packetsize
: Größe des zu sendenden Paketes (Standard: 56 Bytes, Minimum: 24 Bytes, Maximum: 1400 Bytes); auf diese Größe werden 8 Bytes für den Header aufgeschlagen.

packetcount
: Anzahl der zu sendenden Pakete. Standardmäßg (*packetcount* = 0) werden unendlich viele Pakete gesendet, bis das *ping*-Kommando durch das Signal `SIGINT` unterbrochen wird.

timeout
: legt fest, wie lange *ping* neue Anfragen senden bzw. auf eine Antwort warten soll (Standard: 20 Sekunden).

Beispiel 1 Erfolgreicher *ping*-Aufruf:

```
$ ping linux6
linux6 is alive
$
```

Beispiel 2 Erfolgreiche *ping*-Aufrufe zu einer IPv4-Adresse und zu einer IPv6-Adresse mit der Option *-s* (Paketgröße 72 Bytes, 4 Pakete):

```
$ ping -s linux4 72 4
PING linux4: 72 data bytes
80 bytes from linux4 (172.17.29.15): icmp_seq=1 time=448.067 ms
80 bytes from linux4 (172.17.29.15): icmp_seq=2 time=33.339 ms
80 bytes from linux4 (172.17.29.15): icmp_seq=3 time=2.368 ms
80 bytes from linux4 (172.17.29.15): icmp_seq=4 time=2.834 ms

----linux4 PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 2.368/121.652/448.067
$


$ ping -s linux6 72 4
PING linux6: 72 data bytes
80 bytes from linux6 (3ffe:1:1001:3000:230:5ff:fefb:54e7): icmp_seq=1
time=3.690 ms
80 bytes from linux6 (3ffe:1:1001:3000:230:5ff:fefb:54e7): icmp_seq=2
time=3.222 ms
80 bytes from linux6 (3ffe:1:1001:3000:230:5ff:fefb:54e7): icmp_seq=3
time=3.285 ms
80 bytes from linux6 (3ffe:1:1001:3000:230:5ff:fefb:54e7): icmp_seq=4
time=3.197 ms

----linux6 PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 3.197/3.348/3.690
$
```

Beispiel 3 *ping*-Aufruf zu einem Host, der keine Antwort liefert

```
$ ping test
no answer from test.region.example.net
used IPv4 address:      172.25.92.53
$
```

Beispiel 4 *ping*-Aufruf zu einem Host, der nicht existiert

```
$ ping inexist.region.example.net
ping: unknown host inexist.region.example.net
$
```

Beispiel 5 *ping*-Aufruf zu einem Host, der nicht antwortet. Falls vorhanden werden alternative IPv4/IPv6-Adressen angezeigt, unter denen der Host möglicherweise erreichbar ist:

```
$ ping www.ct.de
ping: sendto: No route to host
used IPv6 address:      2a02:2e0:3fe:100::8
alternate addresses:
        193.99.144.80        (IPv4)
$
```

Beispiel 6 *ping*-Aufruf zur Ermittlung der IP-Adressen eines Hosts

```
$ ping -l www.ct.de
hostname: www.ct.de
        IPv6 address:   2a02:2e0:3fe:100::8
        IPv4 address:   193.99.144.80
$
```

pkginfo Informationen über Software-Pakete im POSIX anzeigen (show information on software packages)

Das Kommando *pkginfo* zeigt Informationen über Software-Pakete an, die im POSIX installiert sind. Ein im POSIX installiertes Software-Paket wird beschrieben durch:

- Name des Software-Produkts
- Paket (Package) aus dem Software-Produkt (optional)
- Version des Software-Produkts
- Pfad, unter dem das Software-Paket installiert ist (Standard: /)
- BS2000-Biliothek (SINLIB), aus der das Software-Paket installiert wurde
- Datum der (letzten) Installation.

Syntax

pkginfo[␣**-l**|**-q**][[␣**-v**␣version][␣**-P**␣package][␣**-I**␣ipath]␣product]

Keine Option angegeben
: Es wird eine Übersicht über alle installierten Software-Pakete ausgegeben, siehe Beispiel 1.

Optionen

-l (l - long) detallierte Ausgaben

-q (q - quiet) keine Ausgaben, es wird nur der Endestatus gesetzt

-v version
: nur Produkt(e) der angegebenen Version werden angezeigt

-P package
: nur das angegebene Paket (Package) eines Produkts wird angezeigt

-I ipath
: (I - Installation) nur Produkte mit dem angegebenen Installationspfad werden angezeigt

Endestatus

0 installiertes Produkt bzw. installierte Produkte entsprechend den Angaben gefunden

1 kein installiertes Produkt entsprechend den Angaben gefunden

>1 Fehlerfälle

Fehler

Die Datei */var/sadm/pkg/instlog* kann nicht gelesen werden oder entspricht nicht dem erwarteten Format.

Vom Laufzeitsystem (CRTE) gemeldete Fehler.

Datei

/var/sadm/pkg/instlog - Log-Datei der POSIX-Package-Installationen

Hinweis Das Kommando *pkginfo* gibt die Informationen immer auf Englisch aus, unabhängig von der eingestellten Sprache. Damit werden Probleme vermieden, die sonst bei der Analyse der Ausgaben in Shell-Skripts auftreten könnten.

Die Namen von Software-Produkt und Package können in Groß- oder Kleinbuchstaben angegeben werden.

Beispiel 1 Übersicht aller installierten Software-Pakete ausgeben:

```
$ pkginfo
PRODUCT                        PACKAGE         VERSION  INSTALLATION-PATH
POSIX-BC                       -               080      /
POSIX-SH                       -               070      /
NFS                            -               030      /
POSIX-NSL                      -               070      /
POSIX-SOCKETS                  -               070      /
CRTE                           -               028      /
IMON-BAS                       -               031      /
POSIX-HEADER                   -               018      /
TCP-IP-SV                      PRNGD           031      /opt/TCP-IP-SV/prngd
TCP-IP-SV                      OPENSSH         031      /opt/TCP-IP-
SV/openssh
SYMAPI                         -               066      /opt/emc
SBA-BS2                        -               062      /
SCCA-BS2                       -               020      /opt/emc/sccabs2
$
```

Beispiel 2 Details über alle Pakete eines Software-Produkts ausgeben:

```
$ pkginfo -l tcp-ip-sv
PRODUCT NAME               : TCP-IP-SV
  PRODUCT PACKAGE          : PRNGD
  PRODUCT VERSION          : 031
  INSTALLATION PATH        : /opt/TCP-IP-SV/prngd
  INSTALLATION LIBRARY     : $TSOS.SINLIB.TCP-IP-SV.031.PRNGD
  INSTALLATION DATE        : Fri Oct 10 08:08:56 2008
PRODUCT NAME               : TCP-IP-SV
  PRODUCT PACKAGE          : OPENSSH
  PRODUCT VERSION          : 031
  INSTALLATION PATH        : /opt/TCP-IP-SV/openssh
  INSTALLATION LIBRARY     : $TSOS.SINLIB.TCP-IP-SV.031.OPENSSH
  INSTALLATION DATE        : Fri Oct 10 08:06:08 2008
$
```

Beispiel 3 Details über ein bestimmtes Paket eines Software-Produkts ausgeben:

```
$ pkginfo -l -P openssh tcp-ip-sv
PRODUCT NAME              : TCP-IP-SV
  PRODUCT PACKAGE         : OPENSSH
  PRODUCT VERSION         : 031
  INSTALLATION PATH       : /opt/TCP-IP-SV/openssh
  INSTALLATION LIBRARY    : $TSOS.SINLIB.TCP-IP-SV.031.OPENSSH
  INSTALLATION DATE       : Fri Oct 10 08:06:08 2008
$
```

Beispiel 4 Installation bestimmter Versionen eines Software-Produkts prüfen:

```
$ pkginfo -q -v 027 crte && echo "INSTALLED." || echo "NOT INSTALLED."
NOT INSTALLED.
$ pkginfo -q -v 028 crte && echo "INSTALLED." || echo "NOT INSTALLED."
INSTALLED.
$
```

posdbl Globalen Programm-Cache einrichten und verwalten (set up and manage global program cache)

Dieses Kommando kann nur vom Superuser aufgerufen werden. Gehalten wird ein globaler Programm-Cache in skalierbarer Größe zur Speicherung ablauffähiger Core-Images von POSIX-Programmen. Diese Core-Images werden entweder implizit bei Aufruf eines POSIX-TOOLS aus der Shell-Library oder explizit mit diesem Kommando gespeichert. Zum Laden eines gespeicherten Programms steht der globale Programm-Cache allen Benutzern zur Verfügung.

Syntax

posdbl{␣**-s**|␣**-h**|␣**-S**|␣**-D**|␣**-n**}

posdbl{␣**-e**|␣**-d**}{**loader**|**linker**|**both**}

posdbl␣**-b**␣path

posdbl␣**-l**[␣element]

posdbl␣**-r**␣element

posdbl␣**-L**

posdbl␣**-A** library

posdbl␣**-R** library

Optionen

-s Der Status des globalen Programm-Cache sowie des impliziten Link-Vorgangs und statistische Daten u.a. über Größe und Belegung werden in folgendem Format auf der Standard-Ausgabe ausgegeben:

```
POSIX-DBL              linker status       loader status
Cache POSIX@DBL        CREATED: datum zeit
                       SIZE: größe MB    ENTRIES: einträge
                       FREE PAGES: seiten
```

Bedeutung der Ausgabe

status	Aktueller Status des impliziten Linkvorgangs und des Ladevorgangs (*ON*, *OFF*).
datum	Datum der Einrichtung des Programm-Cache.
zeit	Uhrzeit der Einrichtung des Programm-Cache.
größe	Gesamte Cache-Größe in Megabyte.

einträge Aktuelle Anzahl der gespeicherten Core-Images.

seiten Anzahl der noch verfügbaren Speicherseiten im Cache. Im ungüstigsten Fall ist für ein weiteres Core-Image eine verfügbaren Speicherseite weniger vorhanden, da die Erweiterung des Cache-Katalogs eine Seite beansprucht.

-h Es wird eine Übersicht über alle Optionen und Parameter ausgegeben.

-S Es wird ein Skript nach stdout ausgegeben, mit dem der aktuelle Inhalt des Programm-Caches wiederhergestellt werden kann.

-D Der Programm-Cache wird gelöscht.

-n Es wird ein neuer, leerer Programm-Cache erzeugt.

Beim Erzeugen eines neuen Programm-Caches werden die beiden Funktionen Ladevorgang (*loader*) und Linkvorgang (*linker*) (siehe Optionen *-e* / *-d*) nicht eingeschaltet. Das Laden von Programmen aus dem Cache und das automatische Caching sind also nicht aktiviert.

-e / -d

Der Ladevorgang (*loader*), der implizite Linkvorgang (*linker*) oder beide Vorgänge (*both*) werden aktiviert (Option *-e*) bzw. deaktiviert (Option *-d*).

Der implizite Linkvorgang (*linker*) wirkt für alle Bibliotheken, für die das automatische Caching aktiviert ist (siehe Optionen *-A*, *-R* und *-L*).

-b pfad

Das Core-Image eines durch seinen Pfadnamen identifizierten Programms wird im Programm-Cache gespeichert. Das unter *pfad* angegebene Programm muss ausführbar sein.

-l Es wird eine Liste aller aktuell im Programm-Cache gespeicherten Core-Images in folgendem Format auf der Standard-Ausgabe ausgegeben:

```
element größe datum zeit bibliothek
```

element Name des Programmelements in der PLAM-Bibliothek oder der einfache Dateiname des Programms im UFS. Wurde das Core-Image mit dem Kommandoaufruf

posdbl -b *pfad*

gespeichert, so wird dem Namen des Programmelements ein Plus-Zeichen (+) vorangestellt.

größe Anzahl der Speicherseiten, die das Core-Image belegt.

datum Datum des letzten Zugriffs auf das Core-Image.

zeit Uhrzeit des letzten Zugriffs auf das Core-Image.

status Aktueller Status des Programm-Cache (*active*, *inactive* oder *in delete*).

bibliothek Name der PLAM-Bibliothek aus der geladen wurde oder Pfadname des LLM im UFS.

-I element

Es werden Detailinformationen über das Core-Image *element* im Programm-Cache in folgendem Format auf der Standard-Ausgabe ausgegeben:

```
element          CREATED : cdatum czeit     ACCESS: adatum azeit
                 START AT: stadresse        CACHESIZE: cgröße kB
                                            USERCOUNT: anzahl
                 ------------------------------------------------
                 SLICES  : sl LOADADDR:     SIZE:
                              loadresse     sgröße kB
                              . . . . .     . . . . .
                 ------------------------------------------------
info
```

Bedeutung der Ausgabe

element Name des Programmelements in der PLAM-Bibliothek oder der einfache Dateiname des Programms im UFS.

cdatum Datum der Erstellung des Core-Image.

czeit Uhrzeit der Erstellung des Core-Image.

adatum Datum des letzten Zugriffs auf das Core-Image.

azeit Uhrzeit des letzten Zugriffs auf das Core-Image.

stadresse Startadresse des Core-Image beim Ablauf.

cgröße Anzahl der Speicherseiten, die das Core-Image belegt.

anzahl Anzahl der Ladezugriffe auf das Core-Image.

sl Anzahl der Slices.

loadresse Ladeadresse der Slices.

sgröße Anzahl der Speicherseiten, die die Slices belegen.

info Information über die Herkunft des Core-Image.

Es wird zunächst versucht ein Element in der angegebenen Schreibweise zu finden. Ist das nicht erfolgreich, wird noch einmal ohne Berücksichtigung von Groß-/Kleinschreibung gesucht.

-r element

Aus dem Programm-Cache wird das Core-Image *element* gelöscht. Wird als Element ein * angegeben, so werden alle Core-Images des Programm-Cache gelöscht.

Es wird zunächst versucht, ein Element in der angegebenen Schreibweise zu finden. Ist das nicht erfolgreich, wird noch einmal ohne Berücksichtigung von Groß-/Kleinschreibung gesucht.

-A library

Standardmäßig werden beim impliziten Linkvorgang nur die Bibliotheken SINLIB.POSIX-BC.*vvv*.SHELL und SINLIB.POSIX-SH.*vvv* berücksichtigt. Mit der Option *-A* können weitere Bibliotheken in die Liste der zu berücksichtigenden Bibliotheken aufgenommen werden.

Automatisch in den Programm-Cache vorgeladen werden nur Programme, die im POSIX-Dateisystem nicht als LLM, sondern als Verweis auf ein Element in einer PLAM-Bibliothek installiert sind. Dies ist z.B. bei allen mit POSIX-BC ausgelieferten Kommandos und Tools der Fall. Einige wohldefinierte Programme (z.B. Dämonen und die Kommandos *mount*, *umount* und *share*) werden nicht automatisch in den Cache geladen.

Da die CAT-ID nicht beachtet wird, können nur Bibliotheken hinzugefügt werden, die auf dem Default-Pubset vorhanden sind.

Bei der Angabe des Bibliotheksnamen wird die Groß-/Kleinschreibung nicht berücksichtigt.

```
posdbl -A \$TSOS.SINLIB.POSIX-BC.090.ROOT
```
Das Zeichen "$" muss enwertet werden.

```
posdbl -A SINLIB.POSIX-BC.090.ROOT
```
Ist die Benutzerkennung der Bibliothek nicht angegeben, wird die Benutzerkennung des Aufrufers eingesetzt, also z.B.:
$SYSROOT.SINLIB.POSIX-BC.090.ROOT

```
posdbl -A :DAT0:\$TSOS:SINLIB.POSIX-BC.090.ROOT
```
Die Angabe der Katalogkennung :DAT0: hat keine Auswirkung und wird ignoriert. Daher können nur Bibliotheken hinzugefügt werden, die sich auf dem Default-Pubset der Benutzerkennung befinden.

-R library

Die Bibliothek *library* wird aus der Bibliotheken-Liste entfernt. Alle Programme aus *library* werden nun beim Ausführen nicht mehr automatisch in den Programm-Cache geladen, sondern müssen bei Bedarf mit der Option *-b* explizit hinzugefügt werden.

Es gelten dieselben Hinweise wie bei der Option *-A*.

-L

Es wird eine Liste von Bibliotheken angezeigt, deren Programme beim Aufrufen automatisch in den Programm-Cache geladen werden.

Beispiel

```
# posdbl -s              # Status ausgeben
POSIX-DBL:               linker ON       loader ON
Cache POSIX@DBL          CREATED: 07/18/02 13:06:11
                         SIZE: 24 MB     ENTRIES: 9
                         FREE PAGES: 2688

# posdbl -d linker       # Impliziten Ladevorgang deaktivieren
POSIX-DBL:               linker OFF      loader ON

# posdbl -l              # Core-Images im Programm-Cache ausgeben
 SH              202 Feb 19 11:05:14  $TSOS.SINLIB.POSIX-BC.090.SHELL
 RM               38 Feb 19 11:02:33  $TSOS.SINLIB.POSIX-BC.090.SHELL
 LS               40 Feb 19 10:56:15  $TSOS.SINLIB.POSIX-BC.090.SHELL
. . .
. . .

# cd /usr/demo/bin

# ls -l hello            # Zeige LLM in UFS
-rwxr-xr-x   1 ROOT    SYSROOT   364544 Feb 20 11:09 hello

# posdbl -b hello        # Core-Image erzeugen und speichern

# posdbl -l              # Core-Images im Programm-Cache ausgeben
 SH              202 Feb 19 11:05:14  $TSOS.SINLIB.POSIX-BC.090.SHELL
 RM               38 Feb 19 11:02:33  $TSOS.SINLIB.POSIX-BC.090.SHELL
 LS               40 Feb 19 10:56:15  $TSOS.SINLIB.POSIX-BC.090.SHELL
. . .
. . .
+hello            22 Feb 20 11:10:55 /usr/demo/bin/hello
```

pr Dateien formatieren und auf die Standard-Ausgabe ausgeben (print files)

Mit *pr* können Sie Dateien formatieren und auf die Standard-Ausgabe ausgeben.

Die Ausgabe erfolgt entweder einspaltig (Standard) oder mehrspaltig. Mehrere Spalten können Sie entweder über die Option - (Bindestrich) oder über die Option *-m* definieren.

Syntax **pr**[␣option][␣datei]...

Keine Option angegeben
Die Dateien werden in Seiten aufgeteilt, die durch Sequenzen von Zeilenvorschub-Zeichen getrennt werden. Die Seitenlänge beträgt 66 Zeilen inklusive 10 Zeilen für Kopf- und Fußzeilen.
Die Kopfzeilen bestehen aus zwei Leerzeilen, einer Textzeile mit Seitennummer, Datum und Uhrzeit der letzten Änderung und Dateinamen sowie nochmals zwei Leerzeilen.
Die Fußzeilen bestehen aus fünf Leerzeilen.

Die Dateien werden einspaltig ausgegeben. Überlange Zeilen werden umgebrochen.

Wenn die Standard-Ausgabe auf ein Terminal geleitet ist, werden Fehlermeldungen erst nach der Ausgabe aller angegebenen Dateien ausgegeben.

Übersicht über die Optionen

Die folgende Übersicht sagt Ihnen, mit welcher Option Sie die Ausgabe in welcher Weise beeinflussen:

+ Anfangsseite festlegen

- Text in Spalten aufteilen

-a Auffüllreihenfolge bei Spalten festlegen

-w Seitenbreite bei mehrspaltiger Ausgabe festlegen

-s Abschneiden von Zeilen bei Spalten verhindern

-m Dateien in Spalten nebeneinander ausgeben

-d Zeilenabstand verdoppeln

-e Tabulatorzeichen in Leerzeichen umwandeln

-i Leerzeichen in Tabulatorzeichen umwandeln

-n Zeilen nummerieren

-o Text nach rechts einrücken

-l Seitenlänge ändern

-h Dateiname in der Kopfzeile ändern

-p Datei seitenweise auf den Bildschirm ausgeben

-f Ausgabeseiten durch ein Formularvorschub-Zeichen voneinander trennen

-F Ausgabeseiten durch ein Formularvorschub-Zeichen voneinander trennen

-r Fehlermeldungen unterdrücken

-t Kopf- und Fußzeilen unterdrücken

Beschreibung der Optionen in alphabetischer Reihenfolge

+seitennummer
Beginnt mit der Ausgabe ab Seite *seitennummer*.

+ nicht angegeben:
Die Ausgabe beginnt mit der ersten Seite.

-anzahl_spalten
Gibt die Datei in *anzahl_spalten* Spalten aus. Die Ausgabe erscheint so, als ob *-e* und *-i* mit ihren Standardwerten gesetzt wären.

Diese Option kann nicht mit *-m* kombiniert werden.

Die Spalten einer Seite füllt *pr* nacheinander von oben nach unten.
Mit *-a* können Sie diese Einstellung ändern.

Die Breite einer Seite ist bei mehrspaltiger Ausgabe 72 Zeichen.
Mit *-w* können Sie diese Einstellung ändern.

Die Breite einer Spalte errechnet *pr*, indem es die Seitenbreite durch die Spaltenanzahl dividiert. Ist eine Zeile zu lang, dann schneidet *pr* sie rechts ab. Mit *-s* können Sie das Abschneiden verhindern.

- nicht angegeben:
Die Ausgabe einer Datei ist einspaltig.

-a (a - across) Füllt die Spalten einer Seite von links nach rechts.

Die Anzahl der Spalten muss größer als 1 sein. Diese Anzahl können Sie mit *-anzahl_spalten* oder *-w* bestimmen.

Ist eine Zeile zu lang, um in eine Spalte zu passen, so schneidet *pr* die Zeile rechts ab. Mit *-s* können Sie das Abschneiden verhindern.

Die Option *-a* kann nicht mit *-m* verwendet werden.

-a nicht angegeben:
pr füllt die Spalten von oben nach unten.

-d (d - double-space) Gibt nach jeder Zeile eine Leerzeile aus. Eine solche Leerzeile wird gelöscht, wenn sie die erste Zeile einer Seite ist.

-e[tab_zeichen][abstand]
(e - expand) Ersetzt jedes Tabulatorzeichen im Eingabetext durch entsprechend viele Leerzeichen.

tab_zeichen
Zeichen, das *pr* als Tabulatorzeichen interpretiert. Sie können ein beliebiges nicht-numerisches Zeichen angeben.

tab_zeichen nicht angegeben:
pr interpretiert das Horizontal-Tabulatorzeichen als Tabulatorzeichen (siehe Abschnitt „Zeichensatz ASCII (ISO 646)“ auf Seite 951).

abstand Abstand zwischen zwei Tabulatorpositionen. Die erste Tabulatorposition einer Zeile ist immer die erste Spalte. Wenn Sie 0 für *abstand* angeben, wird der Standardwert 8 gesetzt.

abstand nicht angegeben:
Der Abstand zwischen zwei Tabulatorpositionen beträgt 8 Zeichen.

-f (f - form feed) Trennt die Ausgabeseiten durch ein einzelnes Formularvorschub-Zeichen.

Falls *pr* auf einen Bildschirm ausgibt, hält es vor der ersten Seite mit einem akustischen Signal an. Sie starten die Ausgabe mit der Taste ↵.

-f nicht angegeben:
Die Seiten werden durch eine Sequenz von Zeilenvorschub-Zeichen getrennt.

-F (F - form feed) Trennt die Ausgabeseiten durch ein einzelnes Formularvorschub-Zeichen.

-F nicht angegeben:
Die Seiten werden durch eine Sequenz von Zeilenvorschub-Zeichen getrennt.

Achtung:
In früheren Versionen hatte die Option *-F* eine andere Bedeutung. Diese wird von *pr* jetzt standardmäßig unterstützt.

-h␣kopfzeile
(h - header) Schreibt in die Kopfzeile anstelle des Dateinamens Ihren Text für *kopfzeile*. Diese Option wird ignoriert, wenn Sie gleichzeitig *-t* oder *-l* mit einer Seitenlänge kleiner oder gleich 10 angeben.

-i[tab_zeichen][abstand]
(i - insert) Fügt anstelle von Leerzeichen im Eingabetext das Tabulatorzeichen *tab_zeichen* im Ausgabetext ein.

tab_zeichen

Zeichen, das *pr* als Tabulatorzeichen setzt. Sie können ein beliebiges nicht-numerisches Zeichen angeben.

tab_zeichen nicht angegeben:
pr setzt das Horizontal-Tabulatorzeichen als Tabulatorzeichen (siehe Abschnitt „Zeichensatz ASCII (ISO 646)“ auf Seite 951).

abstand Abstand zwischen zwei Tabulatorpositionen. Die erste Tabulatorposition einer Zeile ist immer die erste Spalte. Wenn Sie 0 für *abstand* angeben, wird der Standardwert 8 gesetzt.

abstand nicht angegeben:
Der Abstand zwischen zwei Tabulatorpositionen beträgt 8 Zeichen.

-l␣seitenlaenge
(l - length) Setzt die Seitenlänge jeder Ausgabeseite fest.

Die Seitenlänge beinhaltet die insgesamt 10 Kopf- und Fußzeilen. Wenn Sie also für *seitenlaenge* eine Zahl kleiner oder gleich 10 angeben, werden Kopf- und Fußzeilen nicht ausgegeben (siehe auch *-t*).

-l nicht angegeben:
Eine Seite hat 66 Zeilen.

-m (m - merge) Gibt die angegebenen Dateien gleichzeitig in Spalten nebeneinander aus, mit je einer Datei pro Spalte. Wenn Sie *-m* benutzen, können Sie maximal neun Dateien für *datei* angeben.

-m kann nicht mit *-anzahl_spalten* kombiniert werden.
Ansonsten gelten dieselben Regeln, wie bei *-anzahl_spalten*.

Die Option *-m* kann nicht mit *-a* verwendet werden.

-m nicht angegeben:
pr gibt mehrere Dateien hintereinander aus.

-n[trennzeichen][nummernlaenge]
(n - number) Nummeriert die Zeilen. Bei mehrspaltiger Ausgabe werden die Zeilen jeder einzelnen Spalte nummeriert. Die Zeilennummer besetzt jeweils die ersten *nummernlaenge*+1 Stellen pro Zeile bzw. pro Spaltenzeile.

trennzeichen

Zeichen, das Zeilennummer und Beginn der Zeile trennt. Sie können ein beliebiges nicht-numerisches Zeichen angeben.

trennzeichen nicht angegeben:
Als Trennzeichen dient das Horizontal-Tabulatorzeichen (siehe Abschnitt „Zeichensatz ASCII (ISO 646)“ auf Seite 951).

nummernlaenge
Anzahl der Stellen für eine Zeilennummer.

nummernlaenge nicht angegeben:
Die Anzahl der Stellen für eine Zeilennummer ist 5.

-o␣anzahl_stellen
(o - offset) Verschiebt jede Ausgabezeile um *anzahl_stellen* Stellen nach rechts.

-p (p - pause) Falls *pr* auf einen Bildschirm ausgibt, hält es vor jeder neuen Seite mit einem akustischen Signal an. *pr* zeigt die Seite an, wenn Sie auf die Taste ↵ drücken.

-r (r - report) Unterdrückt eine Fehlermeldung, wenn *pr* nicht auf eine Datei zugreifen kann.

-r nicht angegeben:
pr gibt eine Fehlermeldung am Ende der Gesamtausgabe aus, wenn es nicht auf eine Datei zugreifen kann.

-s[spaltentrenner]
(s - separate) Trennt Spalten durch ein einzelnes Zeichen *spaltentrenner* anstatt durch ein Tabulatorzeichen. Wenn Sie nicht gleichzeitig *-w* angeben, so verhindert *-s* außerdem das Abschneiden überlanger Zeilen (bis 512 Zeichen) bei spaltenweiser Ausgabe.

spaltentrenner: Zeichen, das zwei Spalten trennt.

spaltentrenner nicht angegeben:
Das Trennzeichen ist das Horizontal-Tabulatorzeichen (siehe Abschnitt „Zeichensatz ASCII (ISO 646)“ auf Seite 951).

-t (t - trailer) Unterdrückt die Kopf- und Fußzeilen. Die Ausgabe wird nach der letzten Seite jeder Datei beendet, ohne den Rest der Seite mit Leerzeilen aufzufüllen.

-w␣seitenbreite
(w - width) Legt bei spaltenweiser Ausgabe die Breite einer Seite fest.
Spaltenweise Ausgabe bestimmen Sie mit *-anzahl_spalten* oder *-m*.

seitenbreite: Anzahl aller Zeichen in einer Zeile.

-w nicht angegeben:
Die Seitenbreite bei spaltenweiser Ausgabe beträgt 72 Zeichen.

datei
Name der Datei, die Sie formatieren und ausgeben möchten. Sie können mehrere Dateien angeben. Wenn Sie mehrere Dateien angeben, gibt *pr* diese hintereinander aus. Geben Sie für *datei* einen Bindestrich an, so liest *pr* von der Standard-Eingabe.

Wenn Sie *-m* für mehrspaltige Ausgabe benutzen, können Sie maximal neun Dateien angeben.

datei nicht angegeben: *pr* liest von der Standard-Eingabe.

Datei */dev/tty**
*/dev/term/tty**

Gerätedateien der einzelnen Terminals.

Wenn die Standard-Ausgabe auf ein Terminal */dev/tty** geleitet wird, werden andere Ausgaben auf dieses Terminal so lange verzögert, bis die Standard-Ausgabe beendet ist. Damit wird verhindert, dass Fehlermeldungen mit der Ausgabe vermischt werden.

Variable *TZ*
bestimmt die Zeitzone für die Kopfzeilen.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *pr*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien) und bestimmt, welche Zeichen als nicht-druckbar gelten. Nicht-druckbare Zeichen werden zwar auf die Standardausgabe ausgegeben, sie werden jedoch bei der Berechnung der Zeilenlänge und Spaltenbreite nicht mitgezählt.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_TIME bestimmt, in welchem Format Datums- und Zeitangaben auf den Seitenköpfen ausgegeben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Sie wollen hintereinander *datei1* und *datei2* dreispaltig ausdrucken (*-3*). Damit Sie vom Druckertyp unabhängig sind, soll *pr* am Ende einer Seite das Zeichen für den Formularvorschub benutzen (*-f*):

```
$ pr -3f datei1 datei2 | lp
```

Beispiel 2 Sie wollen den Tabulatorabstand 8 in *datei1* auf 6 in *datei2* ändern. Dazu wandeln Sie zunächst die Tabulatorzeichen aus *datei1* in Leerzeichen um (*-e*; Tabulatorabstand 8); anschließend wandeln Sie die Leerzeichen wieder in Tabulatorzeichen um (*-i6*; Tabulatorabstand 6); das Ergebnis schreiben Sie in *datei2*; mit *-t* unterdrücken Sie bei beiden *pr*-Kommandos den Kopf- und den Fußteil:

```
$ pr -et datei1 | pr -i6t > datei2
```

Beispiel 3 In der Datei *monate* steht in jeder Zeile ein Monatsname. Diese Datei soll *pr* dreispaltig (*-3*) mit einer zweistelligen Nummerierung ausgeben (*-n2*). Die Spalten sollen von links nach rechts gefüllt werden (*-a*):

```
$ pr -3n2a monate


Jan 27 16:21 2009  monate Seite 1


 1      Januar          5      Mai             9      September
 2      Februar         6      Juni           10      Oktober
 3      Maerz           7      Juli           11      November
 4      April           8      August         12      Dezember
```

Siehe auch *cat*, *fold*, *more*

print Ausgabemechanismus ähnlich *echo* (write arguments to standard output)

print ist der Ausgabemechanismus der POSIX-Shell.

Syntax **print**[␣**-Rnprsu**[dateikennzahl]][␣argument]...

Ohne Option oder mit einer der Optionen - oder -- schreibt *print* auf die Standard-Ausgabe, wie es beim Kommando *echo* beschrieben ist.

-R oder **-r**
Im Raw-Modus werden alle Steuerzeichen von *echo* ignoriert. Die Option *-R* schreibt alle folgenden Optionen und Argumente mit Ausnahme der Option *-n*.

-n An die Ausgabe wird kein Neue-Zeile-Zeichen angefügt.

-p Statt auf die Standard-Ausgabe werden die Argumente auf die Pipeline zu dem durch *|&* erzeugten Prozess geschrieben.

-s Die Argumente werden in die *History*-Datei und nicht auf die Standard-Ausgabe geschrieben.

-udateikennzahl
Die einstellige *dateikennzahl* wird anstelle der Standard-Ausgabe zum Schreiben der Argumente verwendet.

argument
siehe *echo*

Beispiel 1 Verschiedene Ausgabemöglichkeiten der Zeichenkette *abcdef*.

```
$ print '"abc\tdef'"
abc     def
$ print -r '"abc\tdef"'
abc\tdef
$ print -R '"abc\tdef"'
abc\tdef
$ print -n abc; print def
abcdef
```

Beispiel 2 Ausgabe in eine Datei über deren Dateikennzahl.

```
$ exec 4>print.out
$ prinf -u4 abc
$ cat print.out
abc
```

Weitere Beispiele siehe *echo*.

Siehe auch *echo, read*

printf Formatierte Ausgabe (formatted output)

printf gibt die von Ihnen angegebenen Argumente in formatierter Form aus. *printf* unterstützt alle Formatangaben für Zeichenketten wie bei der C-Funktion *printf()*.

Syntax **printf**␣format[␣arg]...

format
Zeichenfolge, die drei verschiedene Objekttypen enthalten kann:
- Zeichen, die unverändert ausgegeben werden sollen.
- Escape-Folgen für Sonderzeichen, die in der Ausgabe in entsprechende Zeichen umgewandelt werden, z. B. wird \n in ein Neue-Zeile-Zeichen umgewandelt.
- Formatelemente, von denen jedes eines der angegebenen Argumente *arg* bearbeitet.

arg
Zeichenfolge, die im durch *format* bestimmten Format auf der Standard-Ausgabe ausgegeben wird. Sind weniger Argumente vorhanden als *format* verlangt, wird für die fehlenden Argumente 0 oder die leere Zeichenkette eingesetzt. Sind mehr Argumente vorhanden als *format* verlangt, wird *format* mehrfach verwendet (außer bei Angabe von *arg_nr$*; in diesem Fall werden die überzähligen Argumente ignoriert).

arg nicht angegeben:
Das Ergebnis ist nicht definiert.

Sonderzeichen

Folgende Sonderzeichen werden von *printf* interpretiert:

\\	Backslash (zur Unterscheidung von Oktalzahlen)
\a	Warnung, Klingel *)
\b	Backspace, Rücksetzzeichen *)
\f	Form Feed, Seitenvorschub
\n	Newline, Neue-Zeile-Zeichen
\r	Carriage Return, Wagenrücklauf
\t	Tabulator
\v	Vertikal-Tabulator *)
oktal	Oktalzahl, wobei *oktal* aus einer, zwei oder drei Ziffern besteht

*) Die so markierten Sonderzeichen werden nur auf Zeichenterminals unterstützt (also bei Zugang zur POSIX-Shell über rlogin)

Formatelemente

Ein Formatelement besteht aus:

```
%[arg_nr$][feldbreite][.genauigkeit]konvertierungszeichen
```

% steht immer am Anfang des Formatelements. Soll das *%*-Zeichen nicht Bestandteil des Formatelements sein, sondern ein auszugebendes Zeichen, muss es durch ein zusätzliches *%*-Zeichen entwertet werden (*%%*).

arg_nr**$**
: dezimale Ganzzahl, mit der Sie die Position des Arguments angeben, das bearbeitet werden soll. Der Zahl muss ein *$*-Zeichen folgen.

  %arg_nr$ und *%* sollten nicht vermischt verwendet werden.

  arg_nr nicht angegeben:
  Das Argument, das auf das zuletzt umgewandelte Argument folgt, wird bearbeitet.

  Wenn Sie für ein Argument *arg_nr$* verwenden, sollten Sie *arg_nr$* auch für alle anderen Argumente verwenden.

feldbreite
: dezimale Ganzzahl, mit der Sie die minimale Feldbreite festlegen. Wenn die umzuwandelnde Zeichenkette aus weniger Zeichen besteht als durch *feldbreite* bestimmt, wird sie rechtsbündig ausgegeben. Wollen Sie eine linksbündige Ausgabe, so müssen Sie vor der dezimalen Ganzzahl einen Bindestrich - angeben. Beginnt *feldbreite* mit einer Null, so wird das Feld bei rechtsbündiger Ausgabe statt mit Leerzeichen mit Nullen aufgefüllt. Statt einer Zahl können Sie bei *feldbreite* auch einen Stern * einsetzen. In diesem Fall bestimmt dann ein ganzzahliges Argument die Feldbreite. Dieses Argument muss vor der umzuwandelnden Zeichenkette stehen. Die Angabe eines Sterns * funktioniert jedoch nur dann, wenn *arg_nr$* nicht verwendet wird.
  Ist die Zeichenkette länger als die Feldbreite, wird das Feld automatisch erweitert.

.genauigkeit
: dezimale Ganzzahl, mit der Sie angeben, wieviele Zeichen der umzuwandelnden Zeichenkette maximal ausgegeben werden sollen. Vor dieser Zahl muss ein Punkt stehen. Ist die Zahl Null, wird nichts ausgegeben. Negative Zahlen werden wie positive Zahlen behandelt. Die Ausgabe richtet sich immer nach dieser Zahl, auch wenn Sie für *feldbreite* eine andere Zahl eingegeben haben.
  Statt einer Zahl können Sie bei *genauigkeit* auch einen Stern * einsetzen. In diesem Fall bestimmt dann ein ganzzahliges Argument die Genauigkeit. Dieses Argument muss vor der umzuwandelnden Zeichenkette stehen. Die Angabe eines Sterns * funktioniert jedoch nur dann, wenn *arg_nr$* nicht verwendet wird.

konvertierungszeichen

Folgende *konvertierungszeichen* können Sie bei *printf* verwenden:

b Zeichenkette mit Sonderzeichen

c Einzelnes Zeichen

d Ganzzahl dezimal

e Gleitkommazahl in Exponent-Darstellung, z.B. 5.234e+2

E Gleitkommazahl in Exponent-Darstellung, z.B. 5.234E+2

f Gleitkommazahl, z.B. 52.34

g %e oder %f, je nachdem, welche Darstellung kürzer ist

G %E oder %f, je nachdem, welche Darstellung kürzer ist

o Ganzzahl oktal (Basis 8)

s Zeichenkette

u Ganzzahl dezimal ohne Vorzeichen

x Ganzzahl hexadezimal (Basis 16)

Bei *s* werden alle Zeichen der Zeichenkette ausgegeben, bis die bei *genauigkeit* angegebene Zeichenzahl erreicht wird. Ist *genauigkeit* nicht angegeben, wird die gesamte Zeichenkette ausgegeben.

Bei *b* kann die Zeichenkette in *arg* Sonderzeichen enthalten. *printf* unterstützt hier alle vom Kommando *echo* interpretierten Escape-Sequenzen, d.h. die oben angegebenen Escape-Sequenzen mit der Ausnahme, dass Oktalzahlen als *\0oktal* angegeben werden, und zusätzlich *\c*. *\c* bewirkt, dass *printf* die Ausgabe an dieser Stelle abbricht und nicht mit einem Neue-Zeile-Zeichen abschließt.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *printf*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_NUMERIC Legt die internationale Umgebung für die numerische Ausgabe fest. Diese Variable beeinflusst das Format der Zahlen, die bei den Konvertierungszeichen *e*, *E*, *f*, *g* und *G* verwendet werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Sie wollen auf dem Bildschirm „Guten Morgen Heike“ ausgeben.

```
$ printf '%s %s %s\n' Guten Morgen Heike
```

Folgendes Kommando erzielt die gleiche Wirkung:

```
$ printf '%2$s %3s %1$s\n' Heike Guten Morgen
```

Beispiel 2 Sie wollen die ersten 6 Zeichen Ihres HOME-Dateiverzeichnisses /usr/huber mit einer entsprechenden Meldung ausgeben.

```
$ printf 'Die ersten 6 Zeichen von %s sind %.6s.\n' $HOME $HOME
Die ersten 6 Zeichen von /usr/huber sind /usr/h.
```

Beispiel 3 Sie wollen eine 4stellige Zahl in ein 8stelliges Feld rechtsbündig, rechtsbündig mit führenden Nullen und anschließend linksbündig ausgeben.

```
$ printf '%8s\n' 1860
    1860

$ printf '%08s\n' 1860
00001860

$ printf '%-8s\n' 1860
1860␣␣␣␣
```

Siehe auch *awk, bc, echo*
printf() [4]

ps Prozessdaten abfragen (report process status)

ps gibt Informationen über Prozesse aus. Dabei wird eine Momentaufnahme über den Zustand des Systems bzw. der Prozesse ausgegeben, der nach dem Bruchteil einer Sekunde schon wieder überholt sein kann und daher zum Zeitpunkt der Ausgabe nicht mehr den wahren Zustand widerspiegelt.

Syntax

ps[␣**-aAcdeflj**][␣**-g**␣grplist][␣**-G**␣grplist][␣**-n**␣namelist][␣**-o**␣format]...
[␣**-p**␣proclist][␣**-s**␣sesslist][␣**-t**␣termlist][␣**-T**][␣**-u**␣userlist][␣**-U**␣userlist]

Keine Option angegeben

ps gibt Informationen über Prozesse aus, die mit der kontrollierenden Datensichtstation verbunden sind. Die Ausgabe besteht aus einer kurzen Auflistung

- der Prozessnummer PID
- der Nummer der Datensichtstation TTY
- der gesamten Ausführzeit TIME
- des Kommandonamens COMD

Die Bedeutung der Ausgabespalten ist im Abschnitt *Ausgabe* genauer erläutert.

option

-a Es werden Informationen über die mit einer Datensichtstation verbundenen Prozesse ausgegeben, wobei jedoch Prozessgruppenführer nicht berücksichtigt werden.

-A Es werden Informationen über die eigenen Prozesse ausgegeben. Diese Option ist äquivalent zur Option *-e*.
Der POSIX-Verwalter erhält Informationen über alle Prozesse.

-c Es werden zusätzlich Informationen über die Klasse und die Priorität der Prozesse ausgegeben.

-d Neben Informationen über die mit einer Datensichtstation verbundenen Prozesse, werden auch Informationen über Prozesse ausgegeben, die nicht mit einer Datensichtstation verbunden sind. Prozessgruppenführer werden jedoch nicht berücksichtigt.

-e Es werden Informationen über die eigenen Prozesse ausgegeben. Diese Option ist äquivalent zur Option *-A*.

BS2000:
Nur der POSIX-Verwalter erhält Informationen über alle Prozesse.
Sollen auch andere Benutzer Informationen über alle Prozesse erhalten, muss der POSIX-Verwalter für die Datei */sbin/ps* nach der Intallation von POSIX-BC das s-Bit setzen (Kommando: *chmod +s /sbin/ps*).

-f (f - full list) Der Umfang an Informationen zu den einzelnen Prozessen wird erweitert. Welche Ausgabespalten zur Ausgabe der Option *-f* gehören, ist im Abschnitt *Ausgabe* näher erläutert.
Ist *-f* angegeben, so gibt *ps* den Kommandonamen und seine Argumente aus. Die Argumente eines Prozesses werden jedoch nur angezeigt, wenn der Prozess demjenigen gehört, der *ps* aufgerufen hat bzw. wenn *ps* vom POSIX-Verwalter aufgerufen wird. Enthält der Kommandoname eines Prozesses nichtdruckbare Zeichen, so wird der Kommandoname in eckige Klammern *[ ]* eingeschlossen. Wird die Option *-f* ohne weitere Optionen angegeben, so beziehen sich die Informationen auf Prozesse, die mit der kontrollierenden Datensichtstation verbunden sind.

-g␣grplist
Es werden nur Informationen über Prozesse ausgegeben, deren Prozessgruppenführer in *grplist* angegeben werden.

grplist
grplist ist eine Liste der Prozessnummern von Prozessgruppenführern. *grplist* hat folgendes Format:

Die Nummern werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen "...", wobei die Nummern dann durch Komma und/oder Leerzeichen getrennt werden können.

-G␣grplist
Es werden nur Informationen über Prozesse ausgegeben, deren reale Prozessgruppenführer in *grplist* angegeben werden.

grplist
grplist ist eine Liste der Prozessnummern von Prozessgruppenführern. *grplist* hat folgendes Format:

Die Nummern werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen "...", wobei die Nummern dann durch Komma und/oder Leerzeichen getrennt werden können.

-j Es werden zusätzlich die Kennummer der Sitzung und Kennummer der Prozessgruppe ausgegeben.

-l (l - long list) Es werden umfassende Informationen zu den einzelnen Prozessen ausgegeben. Welche Ausgabespalten zur Ausgabe der Option *-l* gehören, ist im Abschnitt *Ausgabe* näher erläutert. Wird die Option *-l* ohne weitere Optionen angegeben, so beziehen sich die Informationen auf Prozesse, die mit der kontrollierenden Datensichtstation verbunden sind.

-n␣namelist
Es wird die bei *namelist* angegebene Systemdatei anstelle der Standarddatei verwendet.

-o␣format
Gibt die Informationen gemäß den bei *format* angegebenen Definitionen aus (siehe Abschnitt „Benutzerformatierte Ausgabe“ auf Seite 682).

-p␣proclist
Es werden nur Informationen zu Prozessen ausgegeben, deren Prozessnummern in *proclist* angegeben sind.

proclist
proclist ist eine Liste von Prozessnummern.

Die Nummern werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen "...", wobei die Nummern dann durch Komma und/oder Leerzeichen getrennt werden können.

-s␣sesslist
Es werden nur Informationen über Prozesse ausgegeben, die zu einer in *sesslist* angegeben Sitzung gehören.

sesslist
sesslist ist eine Liste von Sitzungs-Kennummern. *sesslist* hat folgendes Format:

Die Nummern werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen "...", wobei die Nummern dann durch Komma und/oder Leerzeichen getrennt werden können.

-t␣termlist
Es werden nur Informationen über Prozesse ausgegeben, die mit den in *termlist* genannten Datensichtstationen verbunden sind.

termlist
termlist ist eine Liste von Datensichtstationen. Die Datensichtstationen können auf zwei Arten angegeben werden: entweder mit ihrem Gerätenamen (z.B. *term/tty004*) oder, falls der Gerätename aus einer Zusammensetzung mit *tty* besteht, nur mit der Nummer (z.B. *004*).

Die Angaben zu den Datensichtstationen werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen "...", wobei die Einträge dann durch Komma und/oder Leerzeichen getrennt werden können.

-T Es wird zusätzlich die BS2000-TSN der Prozesse ausgegeben. *-T* kann zusammen mit anderen Optionen angegeben werden, *-T* wirkt bei allen Optionen außer der Option *-o*.

-u␣userlist
Es werden nur Informationen über Prozesse ausgegeben, deren Prozesseigentümer in *userlist* angegeben sind.

userlist
userlist ist eine Liste von Benutzernummern bzw. Benutzerkennungen.

Die Angaben in *userlist* werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen "...", wobei die Einträge dann durch Komma und/oder Leerzeichen getrennt werden können.

-U␣userlist
Es werden nur Informationen über Prozesse ausgegeben, deren reale Prozesseigentümer in *userlist* angegeben sind.

userlist
userlist ist eine Liste von Benutzernummern bzw. Benutzerkennungen.

Die Angaben in *userlist* werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen "...", wobei die Einträge dann durch Komma und/oder Leerzeichen getrennt werden können.

Standard-Ausgabe (stdout)

Im Folgenden werden die Überschriften und die Bedeutung der Spalten in der Ausgabe von *ps* erläutert, wenn nicht die Option *-o* gesetzt ist. Die Buchstaben in Klammern bezeichnen die Option, bei der die entsprechende Spalte in der Ausgabe erscheint. Die Angabe *alle* bedeutet, dass die Spalte bei allen Optionen erscheint. Beachten Sie bitte, dass Sie mit den Optionen *-c, -j, -f* und *-l* nur bestimmen, welche Informationen Sie zu einem Prozess erhalten, nicht aber zu welchem Prozess.

F (*l*)
Flags des Prozesses (hexadezimal und additiv). Die Flags sind maschinenabhängig und werden deshalb hier nicht angegeben.

S (*l*)
Status des Prozesses.

0: Prozess wird gerade ausgeführt

S: Sleeping:
Prozess schläft, wartet auf ein Ereignis

R: Runnable:
Prozess ist ablaufbereit

I: Idle:
Prozess ist im Entstehen

Z: Zombie-Status:
Prozess ist beendet, der Endestatus wurde aber vom Vaterprozess noch nicht durch einen *wait()*-Systemaufruf abgefragt

T: Traced:
Ein überwachter Prozess ist gestoppt

X: SXBRK-Status:
Prozess wartet auf mehr Speicherplatz

UID (*f, l*)
(UID - user ID) Benutzernummer des Prozesseigentümers. Wenn die Option *-f* gesetzt ist, wird statt der UID die Benutzerkennung ausgegeben.

Es werden nur die ersten 7 Zeichen der Benutzerkennung ausgegeben.

PID (*alle*)
(PID - process ID) Prozessnummer. Jeder Prozess erhält zum Zeitpunkt seiner Erzeugung eine eindeutig bestimmte Prozessnummer. Unter dieser Nummer kann z.B. der Prozess mit *kill* beendet werden.

TSN (*T*)
(TSN - task sequence number) BS2000-TSN des Prozesses.

PPID (*f, l*)
(PPID - parent process ID) Prozessnummer des Vaterprozesses.

PGID (*j*)
(PGID - process group ID) Kennummer der Prozessgruppe.

SID (*j*)
(SID - session ID) Kennummer der Sitzung.

C (*f, l*)
Prozessor-Auslastung für das Scheduling.

CLS (*c*)
Scheduling-Gruppe (Prozesse, die im Zeitscheibenverfahren verarbeitet werden).

PRI (*l,c*)
Prioritätswert des Prozesses. Höhere Zahlen bedeuten normalerweise niedrigere Priorität. Wurde jedoch *-c* angegeben, so bedeuten höhere Zahlen höhere Priorität.

NI (*l*)
Wert, mit dem die Prozess-Priorität verändert wurde (siehe *nice*). Nur Prozesse der Klasse *time-sharing* haben einen solchen Wert.

ADDR (*l*)
Adresse (physikalische Seiten-Frame-Nummer) des Benutzerbereichs, falls resident, andernfalls die Plattenadresse des ausgelagerten Prozesses.

SZ (*l*)
Die Größe des Speicherabbildes (core-image) des Prozesses in Blocks.

WCHAN (*l*)
Adresse des Ereignisses, auf das der Prozess wartet. Wenn die Spalte leer ist, läuft der Prozess.

STIME (*f*)
Startzeit des Prozesses. Innerhalb der ersten 24 Stunden wird die Uhrzeit ausgegeben, danach das Datum.

TTY (*alle*)
Die kontrollierende Datensichtstation des Prozesses. Falls der Prozess nicht von einer Datensichtstation kontrolliert wird, wird ein Fragezeichen ? ausgegeben.

TIME (*alle*)
Gesamte Ausführungszeit des Prozesses in Minuten und Sekunden.

COMD (*alle*)
Der Name des Kommandos. Wenn die Option *-f* gesetzt ist, wird der vollständige Kommandoname ausgegeben.

Prozesse, die bereits beendet sind, deren Endestatus aber von von ihrem Vaterprozess noch nicht durch einen *wait()*-Systemaufruf abgefragt wurde, werden als <*defunct*> gekennzeichnet.

Wenn *termlist*, *proclist*, *userlist* oder *grplist* nicht angegeben wurde, überprüft *ps* die Standard-Ein-/Ausgabe sowie die Standard-Fehlerausgabe in dieser Reihenfolge, um die kontrollierende Datensichtstation zu ermitteln. Es werden dann Informationen zu allen Prozessen ausgegeben, die von dieser Datensichtstation kontrolliert werden. Falls diese drei Kanäle umgeleitet wurden, findet *ps* keine kontrollierende Datensichtstation und gibt daher auch keine Information aus.

Benutzerformatierte Ausgabe

Mit der Option *-o␣format* bestimmen Sie sich ihre eigene Ausgabe (siehe auch Beispiel 2). *format* ist eine Liste von Variablen, die als einzelnes Argument, oder mit Komma oder Blank voneinander getrennt, eingegeben wird.

Jede Variable hat einen Standardheader. Der Standardheader kann umbenannt werden, d.h. einen neuen Headertext durch Anfügen von Gleichheitszeichen „=“ und den neuen Namen erhalten.

Die in *format* angegebenen Variablen werden nebeneinander auf die Standardausgabe geschrieben. Die einzelnen Feldbreiten sind dabei abhängig von der Länge der Standardheader. Ist der Text eines Headers null, z.B. -o user=, wird mindestens die Breite des Standardheaders verwendet.

Folgende Variablen können in *format* angegeben werden, den Variablen wird der in Klammern angegebene Standardheader zugeordnet, wenn nicht anders über *-o* angegeben.:

ruser (RUSER)
: die reelle Benutzerkennung des Prozesses. Wenn sie ermittelt werden kann und die Feldbreite es erlaubt, wird sie textartig dargestellt, sonst dezimal.

user (USER)
: die effektive Benutzerkennung des Prozesses. Wenn sie ermittelt werden kann und die Feldbreite es erlaubt, wird sie textartig dargestellt, sonst dezimal.

rgroup (RGROUP)
: die reelle Gruppenkennung des Prozesses. Wenn sie ermittelt werden kann und die Feldbreite es erlaubt, wird sie textartig dargestellt, sonst dezimal.

group (GROUP)
: die effektive Gruppenkennung des Prozesses. Wenn sie ermittelt werden kann und die Feldbreite es erlaubt, wird sie textartig dargestellt, sonst dezimal.

pid (PID)
: der Dezimalwert der Prozessnummer (process id)

ppid (PPID)
: der Dezimalwert der Vaterprozessnummer (parent process id)

pgid (PGID)
: der Dezimalwert der Prozessgruppennummer (process group id)

pcpu (%CPU)
: der Prozentwert der verbrauchten CPU-Zeit

vsz (VSZ)
: der Dezimalwert des Speicherbedarfs des Prozesses, in Kilobyte

nice (NI)
: der Dezimalwert der System Scheduling Priorität

etime (ELAPSED)
: Zeit, seit der der Prozess läuft, angegeben in [[dd-]hh:]mm:ss

time (TIME)
: Zeit, den den gesammelten CPU-Verbrauch für den Prozess angibt

tty (TT)
: Name des Terminals, auf dem der Prozess läuft

comm (COMMAND)
: Name des Kommandos, das ausgeführt wird, in textartiger Form (es können auch Leerzeichen enthalten sein)

args (COMMAND)
Kommando mit allen Argumenten, in textartiger Form (es können auch Leerzeichen enthalten sein)

tsn (TSN)
BS2000-TSN des Prozesses

Variable *COLUMNS*
Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Definition der Breite des Editierfensters beim Editiermodus der POSIX-Shell und für die Ausgabe der *select*-Liste verwendet. Diese Variable ist sinnvoll, wenn der Zugang zur POSIX-Shell über *rlogin* erfolgt.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *ps*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE beeinflusst die Sortierreihenfolge.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Meldungen fest,

LC_TIME bestimmt das Format der Zeitangaben bei Verwendung der Option *-f*.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Durch einen Aufruf von *ps* mit der Option *-l* wird eine ausführliche Liste über alle laufenden Prozesse auf der kontrollierenden Datensichtstation ausgegeben:

```
$ ps -l
 F S   UID   PID  PPID  C PRI NI     ADDR     SZ    WCHAN TTY      TIME COMD
10 S   110  1455  1453  0  30 20 c0fec9d8     23 d114f200 tty004  0:02 sh
10 O   110  1862  1455 12  50 20 c0fec870     60          tty004  0:00 ps
18 S   110  1858  1455 10  20 20 c0fecaf8     55 d115f280 tty004  1:03 find
```

Jetzt soll das *find*-Kommando durch einen Aufruf von *kill* mit Angabe der in der Spalte PID angezeigten Prozessnummer abgebrochen werden. Ein erneuter Aufruf von *ps* zeigt anschließend, dass der betreffende Prozess nicht mehr existiert.

```
$ kill 1858
$ ps
   PID TTY      TIME COMD
  1455 tty004   0:02 sh
  1873 tty004   0:00 ps
1858 Terminated
```

Beispiel 2 Benutzerformatierte Ausgabe des ps-Kommandos: Ausgabe von USER, PID, PPID (soll den neuen Namen FATHER erhalten) und Ausgabe der Argumente.

```
$ ps -o user,pid,ppid=FATHER -o args
   USER   PID     FATHER     COMMAND
 HELENE    62          0      [sh]
 HELENE   122         62      [ps]
```

Beispiel 3 Es soll die nur BS2000-TSN der aktuellen Shell (ohne Überschrift) ausgegeben werden.

```
$ ps -o tsn= -p $$
7R6C
```

Zum Vergleich:

```
$ ps -T
   PID   TSN TTY      TIME CMD
   180  7R69 term/002  0:00 ps
   148  7R6C term/002  0:11 sh
```

Siehe auch *kill*, *nice*

pwd Pfadnamen des aktuellen Dateiverzeichnisses ausgeben (return working directory name)

Das in die POSIX-Shell *sh* eingebaute Kommando *pwd* schreibt den absoluten Pfadnamen des aktuellen Dateiverzeichnisses auf die Standard-Ausgabe.

Syntax **pwd**

Wenn die Fehlermeldungen

```
Cannot open ...
```

oder

```
Read error in ...
```

auftreten, liegt ein Fehler im Dateisystem vor. Wenden Sie sich an den POSIX-Verwalter.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *pwd*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Sie wollen Ihr aktuelles Dateiverzeichnis als HOME-Dateiverzeichnis definieren:

```
$ pwd
/usr/art/cobol/prg
$ HOME=`pwd`
$ echo $HOME
/usr/art/cobol/prg
```

Siehe auch *cd*

rcp Datei von oder zu einem fernen Rechner kopieren (remote file copy)

Mit *rcp* können Sie Dateien und Dateiverzeichnisse
– vom lokalen zu einem fernen Rechner
– von einem fernen zum lokalen Rechner
– zwischen zwei fernen Rechnern
kopieren.

Das Kommando wird z.B. verwendet, wenn ein Benutzer mit Dateien, die sich auf einem fernen Rechner befinden, weiterarbeiten möchte. Das Kommando *rcp* kann symmetrisch von BS2000-POSIX und fernen UNIX-Systemen verwendet werden.

Format 1 des Kommandos kopiert eine Datei von einem Rechner in eine Datei eines anderen Rechners.

Format 2 des *rcp*-Kommandos kopiert mehrere Dateien oder ein Dateiverzeichnis eines anderen Rechners.

Das Kommando *rcp* kann nur von Benutzern angewendet werden, die auf dem fernen Rechner zugriffsberechtigt sind. Dazu muss der Name des lokalen Rechners oder ein Pluszeichen + sowie die Benutzerkennung in der Datei *.rhosts* eingetragen sind, die sich im *HOME*-Dateiverzeichnis der Kennung am fernen Rechner befindet. Die Benutzerkennung muss eine Standardabrechnungsnummer für den *rlogin*-Zugang haben. Diese kann mit den Kommandos ADD-USER, MODIFY-USER-ATTRIBUTES oder ADD-POSIX-USER zugewiesen werden.

Die in UNIX-Systemen gebräuchlichen Dateien */etc/hosts* und */etc/hosts.equiv* werden in POSIX nicht verwendet. Die Zuordnung der Rechnernamen zu IP-Adressen erfolgt über BCAM bzw. DNS.

Syntax

Format 1: rcp[␣**-p**][␣**-b**]␣datei1␣datei2

Format 2: rcp[␣**-pr**][␣**-b**]␣datei␣...␣dvz

Format 1 **Einzelne Dateien kopieren**

rcp[␣**-p**][␣**-b**]␣datei1␣datei2

Keine Option angegeben
Jede Kopie erhält als Änderungs- und Zugriffszeit die aktuelle Zeit. Die Zugriffsrechte richten sich nach *umask*.

-p Jede Kopie erhält dieselbe Änderungszeit, Zugriffszeit und dieselben Zugriffsrechte wie die Originaldatei.

-b Die Datei wird binär übertragen.

datei1

ist die Datei, die kopiert werden soll. Sie kann am lokalen oder an einem fernen Rechner gespeichert sein.

Ist die Datei im lokalen Rechner gespeichert, kann sie mit *pfad* angegeben werden. *pfad* ist der Pfadname von *datei1*. Ist *pfad* kein absoluter Pfadname, wird er als relativer Pfad zum aktuellen Dateiverzeichnis interpretiert.

Ist die Datei in einem fernen Rechner gespeichert, muss *datei1* in folgender Form angegeben werden:

`login@ferner_rechner:pfad` oder

`ferner_rechner:pfad` oder

`login@ferner_rechner.domäne:pfad`

ferner_rechner

ist der Name des fernen Rechners oder seine IP-Adresse in IPv4- oder IPv6-Punktnotation:

```
IPv4: n.n.n.n          n = 0..255 (dezimal)
IPv6:x:x:x:x:x:x:x:x   x = 0..FFFF (hexadezimal)
```

Für die IPv6-Punkt-Notation sind die in RFC 2373 beschriebenen alternativen Darstellungsweisen zulässig.
Wird für den Namen des fernen Rechners seine Adresse in IPv6-Punktnotation angegeben, muss diese in eckige Klammern eingeschlossen werden, z.B.:
rcp test hansi@[::ffff:AC19:7D37]:hansi_test

Sonst würde der der erste Doppelpunkt in der IPv6-Adresse als Trennzeichen zwischen Rechnernamen und Pfadnamen fehlinterpretiert.

login

ist die Benutzerkennung am fernen Rechner.

pfad

ist der Pfadname von *datei1*.

domäne

ist der Name der DNS-Domäne, der der ferne Rechner angehört.

Pfadnamen können die üblichen Metazeichen (z. B. *, ?) zur Dateinamenerzeugung enthalten. Diese werden auf dem lokalen Rechner ausgewertet. pfad darf keinen Doppelpunkt ":" enthalten. Die Zeichenkette für *login@ferner_rechner* darf keinen Schrägstrich "/" enthalten.

datei2
ist die Datei, in die kopiert werden soll. Sie kann am lokalen oder an einem fernen Rechner liegen. Ist sie bereits vorhanden, wird sie überschrieben.

Das Format ist das gleiche wie bei *datei1*.

Format 2 **Mehrere Dateien oder Dateiverzeichnisse kopieren**

rcp[␣**-pr**][␣**-b**]␣datei␣...␣dvz

Keine Option angegeben
Jede Kopie erhält als Änderungs- und Zugriffszeit die aktuelle Zeit. Die Zugriffsrechte richten sich nach *umask*.

-p Jede Kopie erhält dieselbe Änderungszeit, Zugriffszeit und Zugriffsrechte wie die Originaldatei.

-r Es werden rekursiv alle Unterverzeichnisse von *dvz* kopiert. In diesem Fall muss *datei* ein Dateiverzeichnis sein.

-b Die Datei wird binär übertragen.

datei
kann sein
– eine Datei
– mehrere Dateien
– ein Dateiverzeichnis (bei *-r*)
am lokalen oder an einem fernen Rechner.

Am lokalen Rechner gilt folgende Form:

```
dateiname ...
```

oder

```
dateiverzeichnis
```

dateiname ist jeweils der Name oder der Pfadname einer oder mehrerer Dateien.
dateiverzeichnis ist jeweils der Name oder der Pfadname des Dateiverzeichnisses.

Liegt *datei* am fernen Rechner, gilt

`login@ferner_rechner:pfad` oder

`ferner_rechner:pfad` oder

`login@ferner_rechner.domäne:pfad`

ferner_rechner
ist der Name des fernen Rechners.

login
ist die Benutzerkennung am fernen Rechner.

pfad
ist der Pfadname von *datei1*.

domäne
ist der Name der DNS-Domäne, der der ferne Rechner angehört.

Pfadnamen können die üblichen Metazeichen (z. B. *, ?) zur Dateinamenerzeugung enthalten. Diese werden auf dem lokalen Rechner ausgewertet.

Beispiel

```
$ rcp ferner_rechner:pfad1/* pfad2/dvz
```

kopiert alle Dateien, die zum ersten Pfadnamen passen, in das angegebene Dateiverzeichnis.

dvz
ist das Dateiverzeichnis, in das kopiert werden soll. Es kann am lokalen oder an einem fernen Rechner liegen. Sind Dateien gleichen Namens bereits vorhanden, werden sie überschrieben.

Das Format für *dvz* am lokalen Rechner ist:

```
dateiverzeichnis
```

Das Format für *dvz* am fernen Rechner ist:

```
login@ferner_rechner:dateiverzeichnis
```

oder

```
login@ferner_rechner.domäne:dateiverzeichnis
```

Fehler *ferner_rechner*: unknown host
Der Rechner ist BCAM nicht oder nur unter einem anderen Alias-Namen bekannt.

ferner_rechner: Connection timed out
Der ferne Rechner hat sich nicht innerhalb einer bestimmten Zeit gemeldet.

Datei *$HOME/.rhosts*
Liste der Rechnernamen mit Benutzerkennungen, die sich unter Ihrer Kennung anmelden dürfen.

Beispiel 1 Der POSIX-Benutzer *hansi* möchte aus seinem aktuellen Dateiverzeichnis die Datei *test* an den Benutzer *hansi* am fernen Solaris-Rechner *rainbow* kopieren. Die Datei soll am Rechner *rainbow* den Namen *hansi_test* bekommen. Die Zugriffsrechte erlauben den Kopiervorgang.

```
$ rcp test hansi@rainbow:hansi_test
```

Beispiel 2 Der POSIX-Benutzer *hansi* kopiert die Dateiverzeichnisse *test* und *uebung* von der Solaris-Benutzerkennung *hansi* am Rechner *rainbow* in das POSIX-Dateiverzeichnis */home/hansi/sinix_copy*.

```
$ rcp -r hansi@rainbow:test hansi@rainbow:uebung /home/hansi/sinix_copy
```

Siehe auch *rsh*

read Argumente von der Standard-Eingabe lesen und Shell-Variablen zuweisen (read a line from standard input)

Das in die POSIX-Shell *sh* eingebaute Kommando *read* liest eine Zeile von der Standard-Eingabe. Die gelesenen Argumente der Eingabezeile weist *read* den beim Aufruf angegebenen Shell-Variablen der Reihe nach als Wert zu.

Als Argument-Trennzeichen erkennt *read* nur die Zeichen, die der Shell-Variablen *IFS* zugewiesen sind. Standardmäßig sind das Leer-, Tabulatorzeichen und Neue-Zeile-Zeichen.

Wenn *read* in einer Shell-Prozedur steht und die Standard-Eingabe nicht umgelenkt ist, hält *read* die Prozedur an und liest Ihre Eingaben von der Standard-Eingabe. Sobald Sie ein Neue-Zeile-Zeichen eingeben, wird die Prozedur fortgesetzt (siehe auch Beispiele auf Seite 694).

Syntax

read␣[-option][␣name**?**abfrage][␣name]...

option

-p Statt von der Standard-Eingabe wird von der Pipeline zu dem, durch |*&* erzeugten Prozess gelesen. Bei Dateiende von der Pipeline wird so bereinigt, dass durch |*&* ein neuer Prozess erzeugt werden kann.

-r Im Raw-Modus hat der Gegenschrägstrich \ am Zeilenende keine Sonderfunktion, d.h. die Zeile wird nicht fortgesetzt.

-s Die Eingabe wird als Kommando in die *History*-Datei geschrieben.

-udateikennzahl
Die einstellige *dateikennzahl* wird statt der Standard-Eingabe zum Lesen verwendet. Die Dateikennzahl kann mit dem eingebauten Kommando *exec* geöffnet werden. Der Standardwert für *dateikennzahl* ist 0.

name
Name der Shell-Variablen, der das entsprechende Argument aus der Eingabe-Zeile zugewiesen wird: Dem ersten Namen wird das erste Argument zugewiesen, dem zweiten Namen das zweite, usw., wobei dem letzten Namen der Rest der Eingabezeile zugewiesen wird.

Die Namen von Shell-Variablen müssen mit einem Buchstaben oder einem Unterstrich _ beginnen und dürfen nur Buchstaben, Ziffern und _ enthalten.

Überzählige Argumente in der Eingabe-Zeile werden der letzten Variablen des Kommandos *read* zugewiesen.
Überzähligen Variablen des Kommandos *read* wird die leere Zeichenkette zugewiesen.

Enthält das erste Argument ein *?*, wird der Rest des Wortes als ein Bereitzeichen auf die Standard-Fehlerausgabe geschrieben.

name nicht angegeben:
Es wird *REPLY* für *name* verwendet.

Endestatus

0 *read* wurde erfolgreich ausgeführt

>0 *read* hat keine Eingabe erhalten, also nur Datei-Ende (EOF) gelesen oder Fehler.

Fehler

`sh: text: not an identifier`
Diese Fehlermeldung kann folgende Ursachen haben:
– Sie haben beim Aufruf keinen Variablen-Namen angegeben, oder
– der angegebene Name enthält unerlaubte Zeichen.

`read: missing arguments`
Sie haben *read* ohne Argumente aufgerufen.

Variable

IFS
Argument-Trennzeichen. Standardmäßig sind der Variablen *IFS* Leerzeichen, Tabulatorzeichen und Neue-Zeile-Zeichen zugewiesen.

PS2
Bereitzeichen, das die POSIX-Shell auf Standard-Fehlerausgabe schreibt, wenn eine Zeile mit einem abschließenden Gegenschrägstrich \ gelesen wird und die Option *-r* nicht angegeben war oder wenn ein Here-Dokument nicht nach einem Neue-Zeile-Zeichen beendet wird.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *read*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Das Kommando *read* wird in der Shell-Prozedur *readtest* aufgerufen. Diese Shell-Prozedur hat folgenden Inhalt:

```
: Aufruf mit sh readtest, wird fuer Eingabe angehalten
echo Bitte geben Sie Kunden-Namen ein:
read kunde1 kunde2 kunde3
if [ -z "$kunde1" ]
then exit 5
else echo Kunde1: $kunde1
     echo Kunde2: $kunde2
     echo Kunde3: $kunde3
fi
```

Die Shell-Prozedur *readtest* wird aufgerufen:

```
$ sh readtest
Bitte geben Sie Kunden-Namen ein:
Mayr Brandl Aulich Weigl
Kunde1: Mayr
Kunde2: Brandl
Kunde3: Aulich Weigl
```

Nach dem Aufruf gibt die Shell-Prozedur die Meldung des Kommandos *echo* aus und startet *read*. Die Prozedur hält an, und *read* liest die eingegebenen Namen der Kunden.
Das Neue-Zeile-Zeichen beendet für *read* die Eingabe-Zeile. Der dritten Variablen *kunde3* weist *read* zwei Namen zu, da die Eingabe-Zeile vier Argumente enthielt.

Beispiel 2 Das Kommando *read* liest die erste Zeile aus einer Datei:

```
$ read zeile < /etc/group
$ echo $zeile
root::0:root
```

In diesem Fall liest *read* auch bei wiederholtem Aufruf immer wieder die erste Zeile der Datei */etc/group*.

Beispiel 3 In der folgenden Shell-Prozedur soll das Kommando *read* nacheinander die Zeilen einer Datei lesen:

```
: Aufruf mit sh lies
exec </etc/group
for i in 1 2 3 4 5 6 7
do
  read satz$i
  eval echo satz$i: \$satz$i
done
```

Mit dem eingebauten *sh*-Kommando *exec* wird in der Shell-Prozedur *lies* die Standard-Eingabe für das nachfolgende Kommando *read* umgelenkt auf die Datei */etc/group*.

Wegen der *for*-Schleife wird *read* in der Prozedur siebenmal aufgerufen. Jeder Aufruf positioniert den Lesezeiger in */etc/group* auf die nächste Zeile.

Deshalb gibt *echo* die ersten sieben Zeilen der Datei */etc/group* nacheinander aus:

```
$ sh lies
satz1: root::0:root
satz2: daemon::1:daemon
satz3: sys::2:sys:
satz4: bin::3:bin,admin
satz5: uucp::4:
satz6: ces::5:
satz7: other::10:gast,mgast,tele
```

Die Angabe \$satz$i kann die Shell nur dann richtig auswerten, wenn sie die *echo*-Kommandozeile zweimal interpretiert; deshalb der Aufruf mit *eval*. Beim erstenmal interpretiert die Shell nur $i, denn das erste Dollarzeichen $ ist durch den Gegenschrägstrich \ entwertet. Beim zweitenmal interpretiert die Shell $satz[1-7].

Siehe auch *exec*

readonly Shell-Variablen schützen (set read-only attributes for variables)

Das in die POSIX-Shell *sh* eingebaute Kommando *readonly* schützt die angegebene Shell-Variable vor Änderungen, d.h. in der aktuellen Shell können Sie dieser Variablen keinen anderen Wert zuweisen. Wenn Sie es doch versuchen, erhalten Sie eine Fehlermeldung.

Dieser Schutz gilt nur in der aktuellen Shell. In einer Subshell dieser Shell ist die Variable nicht mehr geschützt. Das gleiche gilt, wenn Sie die aktuelle Shell beenden. In der aktuellen Shell können Sie jedoch diesen Schutz nicht rückgängig machen.

Wenn Sie *readonly* ohne Argumente aufrufen, schreibt *readonly* die Namen der Shell-Variablen, die Sie in der aktuellen Shell geschützt haben, auf die Standard-Ausgabe.

Syntax

Format 1: readonly[␣name[=wert]]...

Format 2: readonly␣-p

Format 1 **readonly**[␣name[=wert]]...

name
: Name der Shell-Variablen, die vor Veränderung geschützt werden soll. Sie können beliebig viele Shell-Variablen angeben, jeweils getrennt durch ein Leerzeichen.

  name nicht angegeben:
  readonly schreibt die Namen aller Shell-Variablen, die in der aktuellen Shell geschützt sind, auf die Standard-Ausgabe. Die Ausgabe hat folgende Form: `name=wert`

Format 2 **readonly␣-p**

readonly schreibt die Namen aller Shell-Variablen, die in der aktuellen Shell geschützt sind, auf die Standard-Ausgabe. Die Ausgabe hat folgende Form:

```
readonly name=wert
 :
```

Fehler *name*`: is read only`
Diese Fehlermeldung erhalten Sie, wenn Sie versuchen, einer geschützten Shell-Variablen einen Wert zuzuweisen.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *readonly*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Shell-Variable *HOME* in der aktuellen Shell schützen:

```
$ readonly HOME
$ readonly
HOME=/USER1
$ sh
$ readonly
$ END bzw. @@dd
$ readonly
HOME=/USER1
```

In der Subshell ist die Variable *HOME* nicht geschützt.

Siehe auch *export, env, set, typeset*

renice Priorität laufender Prozesse ändern (set system scheduling priorities of running processes)

Achtung!
Das Kommando *renice* wird nur aus Kompatibiltätsgründen unterstützt. Es hat keinen Einfluss auf die Taskprioritäten des BS2000. Deswegen finden Sie hier nur die Beschreibung der Syntax, aber keine Beschreibung der Optionen, Argumente usw.

Syntax

Format 1: renice[␣**-n**␣zahl][␣**-g**|␣**-p**|␣**-u**]␣ID...

Format 2: renice␣priorität␣[**-p**]␣pid...[␣**-g**␣gid...][␣**-p**␣pid...][␣**-u**␣benutzer]

Format 3: renice␣priorität␣**-g**␣gid...[␣**-g**␣gid**...**][␣**-p**␣pid...][␣**-u**␣benutzer]

Format 4: renice␣priorität␣**-u**␣benutzer...[␣**-g**␣gid...][␣**-p**␣pid...][␣**-u**␣benutzer]

Siehe auch *nice*

rm Dateien löschen (remove directory entries)

rm löscht den Eintrag einer oder mehrerer Dateien im Dateiverzeichnis. Sie können Einträge aber nur löschen, wenn Sie für das Dateiverzeichnis, in dem die Datei steht, das Schreibrecht haben.

Syntax **rm**[␣option]␣datei...

Keine Option angegeben

Wenn Sie für *datei* das Schreibrecht haben, löscht *rm* den Eintrag ohne Warnung! Wenn Sie für *datei* kein Schreibrecht haben und die Standard-Eingabe eine Datensichtstation ist, werden die Zugriffsrechte ausgegeben und *rm* fragt, ob *datei* gelöscht werden soll. Nur wenn Sie bestätigen, wird der Eintrag gelöscht; anderenfalls bleibt er erhalten. Wenn die Standard-Eingabe keine Datensichtstation ist, wird der Eintrag ohne Rückfrage gelöscht.

Optionen

-f Die Einträge werden ohne Rückfrage gelöscht. Wenn Sie kein Schreibrecht für das Dateiverzeichnis haben, werden Dateien nicht gelöscht.

-i *rm* fragt für jede Datei und, wenn die Option *-r* gesetzt ist, für jedes Dateiverzeichnis, ob sie bzw. es gelöscht werden soll.
Die Abfrage erfolgt auch, wenn die Option *-f* gesetzt ist oder wenn die Standard-Eingabe keine Datensichtstation ist.

-r Sie können für *datei* ein Dateiverzeichnis angeben. *rm* gibt dann nicht, wie normalerweise, eine Fehlermeldung aus. *rm* löscht rekursiv den gesamten Inhalt des angegebenen Dateiverzeichnisses ebenso wie das Dateiverzeichnis selbst.
Das übergeordnete Dateiverzeichnis (..) kann nicht gelöscht werden. Symbolische Verweise werden bei dieser Option nicht weiter verfolgt.
Löschen eines nicht leeren und schreibgeschützten Dateiverzeichnisses ist nicht möglich (auch nicht mit der Option *-f*). In diesem Fall wird eine Fehlermeldung ausgegeben.

-R analog zu *-r*.

datei

Name der Datei, die gelöscht werden soll. Wenn Sie die Option *-r* angegeben haben, kann *datei* auch ein Dateiverzeichnis sein. Sie können mehrere Dateien bzw. Dateiverzeichnisse angeben.

Wenn Sie eine Datei angeben, auf die es mehrere Verweise gibt, wird lediglich ein Verweis gelöscht, die Datei selbst bleibt vorhanden (der Verweiszähler wird um 1 herabgesetzt). Nur wenn ein Eintrag der letzte Verweis auf eine Datei war, wird die Datei gelöscht.

Für das Dateiverzeichnis, in dem die Datei steht, brauchen Sie das Schreibrecht. Für die Datei selbst brauchen Sie jedoch weder das Lese- noch das Schreibrecht, um sie zu löschen.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *rm*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für das Verhalten von Bereichen, Äquivalenzklassen und Zeicheneinheiten in erweiterten regulären Ausdrücken für ja/nein-Abfragen fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) sowie das Verhalten von Zeichenklassen innerhalb von erweiterten regulären Ausdrücken für ja/nein-Abfragen.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Löschen aller Dateien, die auf *.prog* enden, mit Abfrage:

```
$ rm -i *.prog
ablauf.prog:? (y/n) y
code.prog:? (y/n) yuppie
eingabe.prog:? (y/n) n
zufall.prog:? (y/n) nein
a.prog:? (y/n) morgen
$
```

Die Verweise auf die Dateien *ablauf.prog* und *code.prog* werden gelöscht, die übrigen bleiben bestehen.

Beispiel 2 Löschen des Dateiverzeichnisses *norm* mit allen Dateien und Unterverzeichnissen.

```
$ rm -r norm
```

Siehe auch *rmdir*

rmdir Dateiverzeichnisse löschen (remove directories)

rmdir löscht eines oder mehrere leere Dateiverzeichnisse. Dateiverzeichnisse mit Inhalt können mit *rmdir* nicht gelöscht werden. Um ein Dateiverzeichnis mit Inhalt zu löschen, steht Ihnen das Kommando *rm* mit der Option *-r* zur Verfügung.

Syntax

rmdir[␣**-p**]␣dateiverzeichnis␣...

Keine Option angegeben
rmdir löscht die angegebenen Dateiverzeichnisse.

option

-p (p - parents) Das angebene Dateiverzeichnis wird gelöscht sowie alle übergeordneten Dateiverzeichnisse, die dadurch leer werden.

dateiverzeichnis
Name des zu löschenden Dateiverzeichnisses.
Sie können mehrere Dateiverzeichnisse angeben.

Fehler

`rmdir: dv1: Directory not empty`
Sie haben versucht, mit *rmdir* ein Dateiverzeichnis *dv1* mit Inhalt zu löschen.
Um ein Dateiverzeichnis mit Inhalt zu löschen, steht Ihnen das Kommando *rm* mit der Option *-r* zur Verfügung.

`rmdir: dv1: Directory does not exist`
Das Dateiverzeichnis *dv1* existiert nicht.

`rmdir: ../.: Can't remove current directory or ..`
`rmdir: ../dv1: Can't remove current directory or ..`
Das aktuelle bzw. übergeordnete Dateiverzeichnis kann nicht gelöscht werden. Wechseln Sie in das übergeordnete Verzeichnis.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *rmdir*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Löschen der Dateiverzeichnisse *pro* und *proz*.

Ihr aktuelles Dateiverzeichnis hat folgenden Inhalt:

```
drwxr-xr-x  11 RENATE   other       5720   Nov 18 14:16 ./
drwxr-xr-x  13 SYSROOT  SYSROOT     3380   Nov 04 11:48 ../
-rw-------   1 RENATE   other         79   Jul 19 14:21 .profile
-rwx------   1 RENATE   other        125   Mai 25 10:29 anfang
drwx------   2 RENATE   other         32   Okt 11 15:36 pro/
drwx--x--x   2 RENATE   other         32   Nov 07 10:43 proz/
```

Die Dateiverzeichnisse *pro* und *proz* sind leer, sie können also mit *rmdir* gelöscht werden.

```
$ rmdir pro proz
$ ls -lpa
drwxr-xr-x  11 RENATE   other       5720   Nov 18 14:16 ./
drwxr-xr-x  13 SYSROOT  SYSROOT     3380   Nov 04 11:48 ../
-rw-------   1 RENATE   other         79   Jul 19 14:21 .profile
-rwx------   1 RENATE   other        125   Mai 25 10:29 anfang
```

Siehe auch *rm*

rmpart Partition entfernen (remove partition)

Das Kommando *rmpart* entfernt die Partition *unitnumber* aus der Partitionstabelle. Die damit verbundene BS2000-Datei wird nicht gelöscht.

Syntax

rmpart[␣**-V**]␣unitnumber

Achtung!
Die Funktionalität des Kommandos *rmpart* wird bereits durch das POSIX-Installationsprogramm abgedeckt (siehe POSIX- Handbuch „Grundlagen für Anwender und Systemverwalter“ [1]) . Deswegen wird es hier nicht mehr ausführlich beschrieben.

rsh Shell-Kommando am fernen Rechner ausführen (remote shell)

Mit *rsh* können Sie ein Kommando an einem fernen UNIX-System ausführen.

Das Kommando *rsh* wenden Sie z.B. an, wenn
– Sie sich über den Inhalt eines Dateiverzeichnisses am fernen UNIX-System informieren wollen
– Sie eine Datei am fernen UNIX-System ausdrucken wollen

Das Kommando kann nur angewendet werden, wenn der Benutzer auf dem fernen UNIX-System zugriffsberechtigt ist. Dazu muss der Name des lokalen Rechners oder ein Pluszeichen + sowie die Benutzerkennung in der Datei *.rhosts* eingetragen sind, die sich im *HOME*-Dateiverzeichnis der Kennung am fernen UNIX-System befindet. Die Benutzerkennung muss eine Standardabrechnungsnummer für den *rlogin*-Zugang haben. Diese kann mit den Kommandos ADD-USER, MODIFY-USER-ATTRIBUTES oder ADD-POSIX-USER zugewiesen werden.

Das Kommando *rsh* kann symmetrisch von BS2000-POSIX und fernen UNIX-Systemen verwendet werden.

Die in UNIX-Systemen gebräuchlichen Dateien */etc/hosts* und */etc/hosts.equiv* werden in POSIX nicht verwendet. Die Zuordnung der Rechnernamen zu IP-Adressen erfolgt über BCAM bzw. DNS.

Das Kommando kann nur dann ausgeführt werden, wenn am am fernen Rechner der Dämon *rshd* aktiv ist. Der Dämon *rshd* wird von *inetd* gestartet.

Der Kommandointerpreter, der zur Ausführung des Kommandos auf dem fernen Rechner aufgerufen wird, wird durch den Eintrag des Benutzers in der Datei */etc/passwd* bestimmt.

Wird *rsh* von einer Datei ausgeführt, die nicht *rsh* heißt, dann verwendet *rsh* diesen Namen als Namen des fernen Rechners. Wenn Sie also unter dem Namen des fernen Rechners einen symbolischen Verweis auf *rsh* einrichten, können Sie *rsh* einfach durch Angabe dieses Rechnernamens aufrufen.

Stop-Signale unterbrechen nur den *rsh*-Prozess auf dem lokalen Rechner.

Die Umgebungsvariablen des lokalen Rechners werden nicht an den Kommandointerpreter auf des fernen Rechners weitergegeben.

Syntax

rsh␣host[␣**-n**][␣**-l**␣login][␣**-x**][␣kommando][␣option]

host

BCAM- oder DNS-Name des fernen Rechners, an dem ein Shell-Kommando ausgeführt wird. Der Name des fernen Rechners muss im lokalen BCAM oder im DNS-Server bekannt sein.

Anstelle des Namen des fernen Rechners kann auch seine IP-Adresse in IPv4- oder IPv6-Punkt-Notation angegeben werden. Die IP-Adresse muss BCAM bekannt sein.

IPv4: n.n.n.n
n = 0..255 (dezimal)

IPv6: x:x:x:x:x:x:x:x
x = 0..FFFF (hexadezimal)

Für die IPv6-Punkt-Notation sind die in RFC 2373 beschriebenen alternativen Darstellungsweisen zulässig.

-n wird angegeben, wenn das Kommando *rsh* seine Standard-Eingabe nicht auf das Kommando am fernen Rechner lenken soll. Die Standard-Eingabe von *rsh* wird nach */dev/null* umgeleitet.

Das ist z.B. dann sinnvoll, wenn die Ausgabe des Kommandos *rsh* über eine Pipe an ein Programm weitergegeben wird, das selbst von der Standard-Eingabe liest (siehe Beispiel 1).

-n wird aber auch benötigt, wenn ein *rsh*-Kommando in den Hintergrund gestartet wird.

Der Aufruf von

```
rsh host dd if=/dev/nrmt0 bs=20b | tar xvpBf -
```

führt z.B. dazu, dass sich *tar* vor *rsh* beendet. *rsh* versucht dann, in die unterbrochene Pipe zu schreiben und konkurriert mit dem Kommandointerpreter um die Standard-Eingabe, anstatt sich zu beenden. Mit der Option *-n* können Sie dies vermeiden.

Dieses Problem tritt nur dann auf, wenn *rsh* am Anfang einer Pipe steht und nicht von der Standard-Eingabe liest. Wenn *rsh* tatsächlich von der Standard-Eingabe lesen soll, dürfen Sie die Option *-n* nicht verwenden.

Rufen Sie z. B. das Kommando

```
tar cf - . | rsh host dd of=/dev/rmt0 obs=20b
```

mit der Option *-n* auf, dann liest *rsh* fälschlicherweise von */dev/null* statt von der Pipe.

-l␣login
Die Benutzerkennung, mit der sich der Benutzer am fernen Rechner anmeldet. Sie muss angegeben werden, wenn Sie am fernen Rechner unter einer anders lautenden Benutzerkennung arbeiten wollen als am lokalen Rechner.

-x
Das *rsh*-Kommando liefert den Ende-Status des fernen Kommandos zurück, falls der *rsh*-Server am fernen Rechner dies unterstützt (BS2000-POSIX, UNIX-System).

kommando
Das Kommando, das am fernen Rechner ausgeführt werden soll.

kommando nicht angegeben:
rsh benutzt *rlogin*, um Sie am fernen Rechner anzumelden.

option
Die Argumente, die mit dem Kommando am fernen Rechner verwendet werden.

Arbeitsweise

Das Kommando *rsh* überträgt
- seine Standard-Eingabe dem Kommando am fernen Rechner
- die Standard-Ausgabe des Kommandos am fernen Rechner auf die Standard-Ausgabe des Kommandos *rsh*
- die Standard-Fehlerausgabe des Kommandos am fernen Rechner auf die Standard-Fehlerausgabe des Kommandos *rsh*

Die Signale *interrupt*, *quit* und *terminate* werden an das Kommando am fernen Rechner weitergegeben.

Das aktuelle Dateiverzeichnis des gestarteten Kommandos wird auf das *HOME*-Dateiverzeichnis des Benutzers am fernen Rechner gesetzt.

Nicht entwertete Meta-Zeichen (z.B. <, >, &) werden am lokalen Rechner ausgewertet. Entwertete Meta-Zeichen werden am fernen Rechner interpretiert.

Das Kommando *rsh* ist beendet, wenn sich das Kommando am fernen Rechner beendet.

Bildschirmorientierte Programme, wie z.B. ced können nicht mit dem Kommando *rsh* aktiviert werden. Dafür ist das Kommando *rlogin* zu verwenden.

Beispiel 1 Der POSIX-Benutzer *hansi* möchte sich den Inhalt des Dateiverzeichnisses */home/hansi/test* auf dem fernen Rechner *rainbow* anschauen. Die Ausgabe am Bildschirm soll überlaufgesteuert sein. Die Benutzerkennung *hansi* ist auch am fernen Rechner *rainbow* vorhanden.

```
$ rsh rainbow -n ls -l /home/hansi/test | more
total 167232
drwxrwxr-x   2 hansi 99          1024 Mar 18 09:24 ablage
drwxr-xr-x   2 hansi 99          1024 Apr 14 14:15 de
drwxr-xr-x   2 hansi 99          1024 Apr 22 08:19 en
 ...
--More--
```

Beispiel 2 Der POSIX-Benutzer *hansi* möchte den Inhalt der fernen Datei */home/hansi/hallo_1* am fernen Rechner *rainbow* an die lokale Datei *test_1* anhängen. Die Benutzerkennung *hansi* ist auch am fernen Rechner *rainbow* vorhanden.

```
$ rsh rainbow cat /home/hansi/hallo_1 >> test_1
```

Beispiel 3 Der POSIX-Benutzer *hansi* möchte die ferne Datei */home/hansi/reise_1* an die ferne Datei */home/hans/holland* anhängen. Beide Dateien befinden sich im Rechner *rainbow*. Die Benutzerkennung *hansi* ist auch auf dem Rechner *rainbow* vorhanden.

```
$ rsh rainbow cat /home/hansi/reise_1 ">>" /home/hansi/holland
```

Beispiel 4 Der POSIX-Benutzer *hansi* möchte die ferne Datei */home/emil/test* im Rechner *rainbow* am Rechner *rainbow* ausdrucken. Der POSIX-Benutzer *hansi* ist am Rechner *rainbow* in die Datei */home/emil/.rhosts* eingetragen.

```
$ rsh rainbow -l emil lpr /home/emil/test
```

Beispiel 5 Ein POSIX-Benutzer möchte die lokale Datei */home/hansi/dat* am fernen Rechner *rainbow* ausdrucken.

```
$ rsh rainbow lpr < /home/hansi/dat
```

Datei */$HOME/.rhosts*
Liste der Rechnernamen mit Benutzerkennungen, die sich unter Ihrer Kennung anmelden dürfen.

Siehe auch *rcp*

sed Editor im Prozedurbetrieb (stream editor)

sed ist ein nicht interaktiver, zeilenorientierter Editor mit ähnlichem Funktionsumfang wie der auch zeilenorientierte, aber interaktive *ed*.

sed eignet sich besonders, um

– mehrere globale Editierfunktionen effizient in einem Schritt durchzuführen,

– die Ausgabe eines Kommandos über eine Pipeline auf einfache Weise zu editieren,

– eine Folge von Editierkommandos anzuwenden, die für die interaktive Eingabe zu kompliziert ist.

sed liest Dateien sequentiell, bearbeitet die eingelesenen Zeilen entsprechend den *sed*-Kommandos, die Sie in ein *sed*-Skript geschrieben haben, und gibt die bearbeiteten Zeilen auf die Standard-Ausgabe aus. *sed* liest das *sed*-Skript entweder von der Kommandoaufrufzeile oder aus einer Datei. Die Eingabedateien bleiben unverändert. Wenn Sie die Änderungen sichern möchten, lenken Sie die Standard-Ausgabe in eine Datei um.

Syntax

Format 1: sed[␣**-n**]␣skript[␣datei...]

Format 2: sed [␣**-n**][␣**-e**␣skript]...[␣**-f**␣skriptdatei]...[␣datei...]

Format 1 **sed**[␣**-n**]␣skript[␣datei...]

-n unterdrückt die standardmäßig erfolgende Ausgabe jeder bearbeiteten Eingabezeile auf die Standard-Ausgabe (siehe *Beispiel 6*).

-n nicht angegeben:
sed gibt jede bearbeitete Eingabezeile auf die Standard-Ausgabe aus. Ob die Zeile bei der Bearbeitung verändert wurde oder nicht, hängt von den Kommandos im *sed*-Skript ab. Die Zeilen werden also auch ausgegeben, wenn im *sed*-Skript kein Ausgabekommando, wie z.B. *p* steht. Eingabezeilen, die mit einem Ausgabekommando, wie z.B. *p* bearbeitet werden, werden zweimal nacheinander ausgegeben. Die erste Ausgabe erfolgt als standardmäßige Ausgabe aller bearbeiteten Zeilen. Die zweite Ausgabe erfolgt wegen der speziellen Bearbeitung, die in diesem Fall die Ausgabe bewirkt.

skript
sed liest das *sed*-Skript *skript*, mit dem die Eingabedatei bearbeitet werden soll, von der Kommandoaufrufzeile. Wenn *skript* Leerzeichen, Neue-Zeile-Zeichen oder Shell-Sonderzeichen enthält, müssen Sie das Kommando in Hochkommas einschließen: '*skript*'

datei
Name der Datei, deren Inhalt *sed* bearbeiten soll. Sie können nur Text-Dateien mit *sed* bearbeiten.

datei nicht angegeben:
sed liest von der Standard-Eingabe.

Format 2 **sed** [␣**-n**][␣**-e**␣skript]...[␣**-f**␣skriptdatei]...[␣datei...]

-n unterdrückt die standardmäßig erfolgende Ausgabe jeder bearbeiteten Eingabezeile auf die Standard-Ausgabe (siehe *Beispiel 6*).

-n nicht angegeben:
sed gibt jede bearbeitete Eingabezeile auf die Standard-Ausgabe aus. Ob die Zeile bei der Bearbeitung verändert wurde oder nicht, hängt von den Kommandos im *sed*-Skript ab. Die Zeilen werden also auch ausgegeben, wenn im *sed*-Skript kein Ausgabekommando, wie z.B. *p* steht. Eingabezeilen, die mit einem Ausgabekommando, wie z.B. *p* bearbeitet werden, werden zweimal nacheinander ausgegeben. Die erste Ausgabe erfolgt als standardmäßige Ausgabe aller bearbeiteten Zeilen. Die zweite Ausgabe erfolgt wegen der speziellen Bearbeitung, die in diesem Fall die Ausgabe bewirkt.

-e␣skript
sed liest das *sed*-Skript *skript*, mit dem die Eingabedatei bearbeitet werden soll, von der Kommandoaufrufzeile. Wenn *skript* Leerzeichen, Neue-Zeile-Zeichen oder Shell-Sonderzeichen enthält, müssen Sie das Kommando in Hochkommas einschließen: '*skript*'

Sie können *-e skript* mehrmals und auch zusammen mit *-f skriptdatei* angeben. *sed* liest dann die *sed*-Kommandos aus allen angegebenen *sed*-Skripts.

Enthält die Kommandoaufrufzeile nur einmal die Option *-e* und nicht die Option *-f*, können Sie *skript* auch ohne *-e* angeben.

-f␣skriptdatei
sed liest das *sed*-Skript, mit dem die Eingabedatei bearbeitet werden soll, aus der Datei *skriptdatei*.

Sie können *-f skriptdatei* mehrmals und auch zusammen mit *-e skript* angeben. *sed* liest dann die *sed*-Kommandos aus allen angegebenen *sed*-Skripts.

datei
Name der Datei, deren Inhalt *sed* bearbeiten soll. Sie können nur Text-Dateien mit *sed* bearbeiten.

datei nicht angegeben:
sed liest von der Standard-Eingabe.

Arbeitsweise

sed arbeitet jeweils mit Kopien der Eingabezeilen. Sie werden nacheinander in den sogenannten Musterspeicher kopiert.

Normalerweise bearbeitet *sed* die Eingabezeilen in folgendem Zyklus:

1. Schritt: Die nächste (am Dateianfang die erste) Eingabezeile wird in den Musterspeicher kopiert.
2. Schritt: Alle *sed*-Kommandos im *sed*-Skript, welche die zuletzt in den Musterspeicher kopierte Zeile adressieren, werden nacheinander auf den Inhalt des Musterspeichers angewandt. Abhängig von den *sed*-Kommandos wird dabei der Inhalt des Musterspeichers verändert oder auch nicht.
3. Schritt: Der bearbeitete Inhalt des Musterspeichers wird auf die Standard-Ausgabe ausgegeben, und der Musterspeicher wird gelöscht.

Dieser Zyklus wird solange durchlaufen, bis die Eingabe erschöpft ist.

Der Musterspeicher kann manchmal auch mehrere Zeilen enthalten (siehe *sed*-Kommandos *g*, *G*, *N*). Als Adresse des Musterspeichers gilt jedoch immer die Adresse der zuletzt in den Musterspeicher kopierten Zeile.

Einige *sed*-Kommandos (*g*, *G*, *h*, *H*) benutzen zusätzlich einen sogenannten Haltespeicher, in dem der Inhalt des Musterspeichers vollständig oder teilweise zur späteren Wiederverwendung gespeichert wird.

Format eines sed-Skripts

Ein *sed*-Skript besteht aus Kommandozeilen der Form:

```
[bereich]sed-Kommando[parameter]...
```

oder

```
[bereich]{sed-Kommando[parameter]...
          sed-Kommando[parameter]...
              .
              .
              .
          }
```

Die geschweiften Klammern müssen Sie nur angeben, wenn Sie *bereich* angeben.
Die schließende geschweifte Klammer } muss an einem Zeilenanfang stehen, d.h. es dürfen ihr nur Leerzeichen oder Tabulatorzeichen vorangehen.

Zwischen *bereich* und *sed-Kommando* darf kein Leerzeichen stehen.

Mit *bereich* wählen Sie bestimmte Eingabezeilen aus. Wenn *bereich* den Musterspeicher, d.h. die zuletzt in den Musterspeicher kopierte Eingabezeile adressiert, wird das zugehörige *sed*-Kommando bzw. die *sed*-Kommandoliste auf den Inhalt des Musterspeichers angewandt.

Sie können für *bereich* eine Adresse oder zwei durch ein Komma getrennte Adressen angeben.

bereich = adresse
: Jede zuletzt in den Musterspeicher kopierte Eingabezeile, die zu *adresse* passt, ist adressiert.

bereich = adresse1,adresse2
: Der Bereich von der durch *adresse1* adressierten Eingabezeile bis zu der durch *adresse2* adressierten Eingabezeile, einschließlich der Intervallgrenzen, ist adressiert. Adressiert *adresse2* eine Zeile, die in der Eingabedatei vor der mit *adresse1* adressierten Zeile liegt, gilt nur *adresse1*.

bereich nicht angegeben:
Jede zuletzt in den Musterspeicher kopierte Eingabezeile ist adressiert.

Adressen

$ letzte Zeile der Datei

n *n*-te Eingabezeile, wobei *n* eine positive ganze Zahl ist. Die Nummern ergeben sich, indem alle Eingabezeilen fortlaufend durchnummeriert werden.

/muster/
: Eingabezeile, die eine zu *muster* passende Zeichenkette enthält. Wenn muster selbst einen Schrägstrich enthält, muss er mit einem Gegenschrägstrich \ davor entwertet werden. muster ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*) mit folgenden Änderungen:

  Die Angabe *?regulärer ausdruck?*, wobei *?* ein beliebiges Zeichen ist, gleichbedeutend mit */regulärer ausdruck/*. Dabei steht ein mit einem Gegenschrägstrich \ entwertetes ? für sich selbst, begrenzt also nicht den regulären Ausdruck. Zum Beispiel steht die Adresse *\xabc\xdefx* für die Zeichenkette: *abcxdef*

  Die Escape-Sequenz \n passt zum Neue-Zeile-Zeichen im Musterspeicher.

  Ein Punkt passt zu jedem Zeichen außer dem letzten Neue-Zeile-Zeichen im Musterspeicher.

sed-Kommandos

Im Folgenden sind die *sed*-Kommandos in alphabetischer Reihenfolge beschrieben.
Die eckigen Klammern [ ] sind dabei nicht einzugeben! Sie zeigen lediglich an, dass die dazwischen eingeschlossene Adressangabe fakultativ ist.

Das Argument *text* besteht aus einer oder mehreren Zeilen, von denen alle bis auf die letzte mit einem Gegenschrägstrich \ enden müssen, um das anschließende Neue-Zeile-Zeichen zu entwerten.

[adresse]**a**\
text
(a - append) Im Anschluss an die Ausgabe des Inhalts des Musterspeichers wird *text* ausgegeben.

[bereich]**b**[marke]
(b - branch) Im *sed*-Skript wird zu dem *sed*-Kommando *:marke* verzweigt.

marke nicht angegeben:
Es wird ans Ende des *sed*-Skripts verzweigt.

[bereich]**c**\
text
(c - change) Der adressierte Bereich wird gelöscht, wenn er im Musterspeicher ist, *text* wird ausgegeben und der nächste Zyklus wird gestartet.

[bereich]**d**
(d - delete) Der Inhalt des Musterspeichers wird gelöscht und der nächste Zyklus wird gestartet. Der 3. Schritt, die Ausgabe des Inhalts des Musterspeichers, entfällt.

[bereich]**D**
(d - delete) Der Inhalt des Musterspeichers wird vom Anfang bis zum ersten Neue-Zeile-Zeichen gelöscht und der nächste Zyklus wird gestartet.

[bereich]**g**
Der Inhalt des Haltespeichers ersetzt den Inhalt des Musterspeichers.

[bereich]**G**
Der Inhalt des Haltespeichers wird an den Musterspeicher angefügt.

[bereich]**h**
Der Inhalt des Musterspeichers ersetzt den Inhalt des Haltespeichers.

[bereich]**H**
Der Inhalt des Musterspeichers wird an den Haltespeicher angefügt.

[adresse]**i**\
text
(i - insert) *text* wird vor dem Inhalt des Musterspeichers ausgegeben.

[bereich]**l**

(l - list) Der Inhalt des Musterspeichers wird auf die Standard-Ausgabe ausgegeben, wobei nicht-druckbare Zeichen durch Ersatz-Zeichen (z.B. Tabulatorzeichen durch das Größerzeichen>) oder als Oktalzahlen (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*) in der Form \nn dargestellt werden. Überlange Zeilen mit mehr als 71 Zeichen werden auf zwei oder mehr Zeilen unterteilt. Ein Gegenschrägstrich \ am Ende einer Zeile zeigt an, dass die Textzeile in der nächsten Bildschirmzeile fortgesetzt wird.

[bereich]**n**

(n - next) Der Inhalt des Musterspeichers wird auf die Standard-Ausgabe ausgegeben und durch die nächste Eingabezeile ersetzt. Die Adresse der eigegebenen Zeile wird zur Adresse des Musterspeichers.

[bereich]**N**

Die nächste Eingabezeile wird einschließlich des Neue-Zeile-Zeichens an den Inhalt des Musterspeichers angefügt. Die Adresse der letzten angefügten Zeile wird zur Adresse des Musterspeichers.

[bereich]**p**

(p - print) Der Inhalt des Musterspeichers wird auf die Standard-Ausgabe ausgegeben. Nicht druckbare Zeichen werden nicht dargestellt.

[bereich]**P**

(p - print) Der Anfang des Musterspeichers wird bis zum ersten Neue-Zeile-Zeichen einschließlich auf die Standard-Ausgabe ausgegeben. Nicht druckbare Zeichen werden nicht dargestellt.

[adresse]**q**

(q - quit) *sed* wird beendet. Haben Sie mehrere *sed*-Skripts angegeben, wird *sed* beim ersten *q* beendet, das in irgendeinem der Skripts steht.

[bereich]**r**␣rdatei

(r - read) Der Inhalt der Datei *rdatei* wird gelesen und vor dem Kopieren der nächsten Eingabezeile in den Musterspeicher auf die Standard-Ausgabe ausgegeben. *rdatei* muss, durch genau ein Leerzeichen vom *sed*-Kommando *r* getrennt, am Ende der Kommandozeile stehen.

[bereich]**s/**rA**/**ersetzungszeichenkette**/**[parameter]

(s - substitute) Zeichenketten im Musterspeicher, zu denen der reguläre Ausdruck *rA* passt, werden durch *ersetzungszeichenkette* ersetzt. Für *rA* können Sie einfache reguläre Ausdrücke angeben (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Als Trennzeichen können Sie außer dem Schrägstrich / jedes beliebige Zeichen verwenden. Nähere Informationen siehe *ed*-Kommando *s*.

parameter

n (n - number) Nur die *n*-te zu *rA* passende Zeichenkette in einer Zeile wird durch *ersetzungszeichenkette* ersetzt. *n* ist eine ganze Zahl von 1 bis 512.

g (g - global) Alle zu *rA* passenden Zeichenketten in einer Zeile werden durch *ersetzungszeichenkette* ersetzt.

p (p - print) Der Inhalt des Musterspeichers wird auf die Standard-Ausgabe ausgegeben, falls eine Ersetzung durchgeführt wurde, auch wenn *sed* mit Option *-n* aufgerufen wurde.

w␣wdatei (w - write) Der Inhalt des Musterspeichers wird in die Datei *wdatei* geschrieben, falls eine Ersetzung durchgeführt wurde. Der Inhalt einer bereits vor dem *sed*-Aufruf existierenden Datei mit dem Namen *wdatei* wird überschrieben. Wenn innerhalb eines *sed*-Skripts mehrere *w*-Kommandos in dieselbe Datei *wdatei* schreiben, wird der Inhalt des Musterspeichers jeweils an den Inhalt von *wdatei* angehängt. *wdatei* muss, durch genau ein Leerzeichen vom *sed*-Kommando *w* getrennt, am Ende der Kommandozeile stehen. In einem *sed*-Aufruf dürfen Sie maximal 10 verschiedene Dateien für *wdatei* verwenden.

Achtung!
Wenn Sie für *wdatei* den Namen Ihrer Eingabedatei angeben, zerstören Sie diese!

parameter nicht angegeben:
Nur die erste zu *rA* passende Zeichenkette einer Zeile wird durch *ersetzungszeichenkette* ersetzt.

[bereich]**t**␣marke
(t -test) Im *sed*-Skript wird zu dem *sed*-Kommando *:marke* verzweigt, falls seit dem letzten Kopieren einer Eingabezeile in den Musterspeicher oder dem letzten Aufrufen eines *t*-Kommandos eine Ersetzung durchgeführt wurde.

marke nicht angegeben:
Im *sed*-Skript wird ans Ende gesprungen.

[bereich]**w**␣wdatei
(w - write) Der Inhalt des Musterspeichers wird in die Datei *wdatei* geschrieben. Der Inhalt einer bereits vor dem *sed*-Aufruf existierenden Datei mit dem Namen *wdatei* wird überschrieben. Wenn innerhalb eines *sed*-Skripts mehrere *w*-Kommandos in dieselbe Datei *wdatei* schreiben, wird der Inhalt des Musterspeichers jeweils an den Inhalt von *wdatei* angehängt. *wdatei* muss, durch genau ein Leerzeichen vom *sed*-Kommando *w* getrennt, am Ende der Kommandozeile stehen. In einem *sed*-Aufruf dürfen Sie maximal 10 verschiedene Dateien für *wdatei* verwenden.

Achtung!
Wenn Sie für *wdatei* den Namen Ihrer Eingabedatei angeben, zerstören Sie diese!

[bereich]**x**
(x - exchange) Der Inhalt des Muster- und Haltespeichers wird ausgetauscht.

[bereich]**y**/zeichenkette1/zeichenkette2/

Jedes Vorkommen eines Zeichens aus *zeichenkette1* wird durch das entsprechende Zeichen aus *zeichenkette2* ersetzt. *zeichenkette1* und *zeichenkette2* müssen gleich lang sein und explizit angegeben werden. Reguläre Ausdrücke können nicht verwendet werden.

[bereich]**!**kommando
[bereich]**!**{kommandoliste}

kommando ist ein *sed*-Kommando oder eine in geschweiften Klammern {...} eingeschlossene *sed*-Kommandoliste und wird auf alle Zeilen angewandt, die *nicht* durch *bereich* adressiert sind.

:marke

Dieses Kommando setzt im Skript die *marke*, die von den Kommandos *b* und *t* angesprungen werden kann. Für *marke* können Sie bis zu 8 Zeichen angeben.

[adresse]**=**

Die aktuelle Zeilennummer wird auf die Standard-Ausgabe in einer eigenen Zeile ausgegeben.

[bereich]**{**sed-Kommando
sed-Kommando
.
.
.
}

Die in geschweiften Klammern {...} eingeschlossenen *sed*-Kommandos werden nacheinander ausgeführt, wenn *bereich* den aktuellen Musterspeicher adressiert.
Der öffnenden Klammer { können Leerzeichen vorausgehen, gefolgt von weiteren Leerzeichen und Tabulatoren.
Den Kommandos können Leerzeichen und Tabulatoren vorausgehen.
Die schließende Klammer } muss auf ein Neue-Zeile-Zeichen folgen. Es dürfen ihr nur Leerzeichen vorausgehen.

↵ Das Neue-Zeile-Zeichen gilt als leeres Kommando und wird ignoriert. Sie können so Ihre sed-Skripts durch Leerzeilen übersichtlich gliedern.

\# Steht in der ersten Zeile einer Skriptdatei als erstes Zeichen ein #, wird der darauffolgende Zeileninhalt als Kommentar interpretiert.

#n Steht in der ersten Zeile einer Skriptdatei als erstes Zeichen ein #*n*, wird die standardmäßige Ausgabe des Musterspeichers unterdrückt (entspricht der Option -n). Der auf #*n* folgende Zeileninhalt wird als Kommentar gewertet und nicht als *sed*-Kommando interpretiert.

Fehler `sed: command garbled: ...`

Das *sed*-Skript enthält einen Syntaxfehler. Nach dem Doppelpunkt wird die Stelle aus dem Skript ausgegeben, an der sich *sed* beendet hat.

`Cant't open` *datei*
Sie haben eine Eingabedatei angegeben, die nicht existiert oder für die Sie kein Leserecht besitzen.

`Unrecognized command: ...`
Das *sed*-Skript enthält ein unbekanntes Kommando.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *sed*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für die Bedeutung von Zeichenbereichen, Äquivalenzklassen und Zeicheneinheiten in geklammerten regulären Ausdrücken fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten). *LC_CTYPE* bestimmt zusätzlich, welche Zeichen bei Verwendung des Kommandos *l* als nichtdruckbar betrachtet werden.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 In alle Leerzeilen einer Datei die Zeichenkette XXXXX schreiben und die Ausgabe in eine andere Datei umlenken:

```
$ sed '/^$/s/^/XXXXX/' datei > dateineu
```

Mit /^$/ sind alle Leerzeilen adressiert, d.h. Zeilen, in denen zwischen Zeilenanfang und Zeilenende nichts, auch kein Leerzeichen, steht. *sed* sucht nach dem Zeilenanfang ^ und ersetzt ihn durch die Zeichenkette XXXXX.

Beispiel 2 Alle Zeilen einer Datei, die mit einer Ziffer beginnen, um 4 Leerzeichen einrücken und die Ausgabe in eine andere Datei umlenken:

```
$ sed '/^[0-9]/s/^/␣␣␣␣/' datei > dateineu
```

Mit /^[0-9]/ sind alle Zeilen adressiert, in denen am Zeilenanfang eine Ziffer von 0 bis 9 einschließlich steht. *sed* sucht nach dem Zeilenanfang ^ und ersetzt ihn durch 4 Leerzeichen ␣␣␣␣, d.h. verschiebt den bisherigen Zeileninhalt um 4 Positionen nach rechts und füllt Position 1 bis 4 mit Leerzeichen.

Beispiel 3 Alle Zeilen einer Datei ausgeben, die keine Leerzeilen sind:

```
$ sed '/^$/d' datei
```

Alle Leerzeilen werden mit /^$/ adressiert und mit *d* gelöscht.

Beispiel 4 Alle Zeilen einer Datei mit jeweils einer Leerzeile dazwischen ausgeben:

```
$ sed 's/$/\↵
> /' datei
```

Da keine Adresse angegeben ist, sucht *sed* in *jeder* Zeile nach dem Zeilenende $ und ersetzt es durch ein Neue-Zeile-Zeichen ↵, d.h. es wird ein zusätzliches Neue-Zeile-Zeichen eingefügt.

Beispiel 5 Ausgeben der zweiten und dritten Spalte einer Datei, deren Spalten jeweils durch einen Doppelpunkt voneinander getrennt sind. Die dritte Spalte soll vor der zweiten ausgegeben werden:

```
$ sed 's/[^:]*:\([^:]*\):\([^:]*\):.*/\2:\1/' datei
```

Erläuterung:

s/ / /
: In jeder Zeile wird die Zeichenkette zwischen dem ersten und zweiten Schrägstrich gesucht und ersetzt durch die Zeichenkette zwischen dem zweiten und dritten Schrägstrich.

[^:]*
: Beliebig viele (*) Zeichen außer Doppelpunkt ([^:])

: : Doppelpunkt

\(\)
: Klammerung eines Teilausdrucks, der in der Ersetzungszeichenkette zwischen dem zweiten und dritten Schrägstrich wiederverwendet werden soll

\2 : Die Ersetzungszeichenkette soll mit dem in der zweiten Klammer \(...\) stehenden Teilausdruck beginnen.

: : Zwischen dem ersten und zweiten Teilausdruck in der Ersetzungszeichenkette soll ein Doppelpunkt stehen.

\1 : Die Ersetzungszeichenkette soll mit dem in der ersten Klammer \(...\) stehenden Teilausdruck enden.

Beispiel 6 Aus der Datei *pers* alle Berufe herausfiltern und mit einer neuen Überschrift in die Datei *berufe* schreiben. Voraussetzung für das folgende Beispiel ist, dass die erste Zeile der Eingabedatei keinen Suchbegriff, in diesem Fall „Beruf:“ enthält.

Die Datei *pers* hat folgenden Aufbau:

```
Name: Hans Mueller
Familienstand: geschieden
Beruf: Journalist

Name: Karin Becker
Familienstand: verheiratet
Beruf: Programmiererin
usw.
```

sed-Programm:

```
$ sed -n '1{s/^.*/Berufe:/
           h
          }

         /^Beruf:/{s/^Beruf: *\(.*\)/\1/
                   H
          }

         ${g
           p
          }' pers > berufe
```

```
$ cat berufe
Berufe:
Journalist
Programmiererin
usw.
$
```

Erläuterung:
Zeile 1 im Musterspeicher wird ersetzt durch die Zeichenkette *Berufe:* und dann mit *h* in den Haltespeicher geschrieben.
Jede Zeile im Musterspeicher, die mit *Beruf:* beginnt, wird ersetzt durch die Zeichenkette, die jeweils darauf folgt und mit *H* an den Haltespeicher angefügt.
Die letzte Zeile der Datei im Musterspeicher ($) wird mit *g* ersetzt durch den gesamten Inhalt des Haltespeichers. Mit *p* wird der Inhalt des Musterspeichers ausgegeben.

Die Option *-n* sorgt dafür, dass die in den Musterspeicher kopierten Eingabezeilen nach ihrer Bearbeitung nicht automatisch auf die Standard-Ausgabe geschrieben werden. Nur der mit dem *sed*-Kommando *p* bearbeitete Inhalt des Musterspeichers wird ausgegeben.

Siehe auch *awk*, *ed*, *grep*
Tabellen und Verzeichnisse, Reguläre Ausdrücke

set Shell-Optionen oder Stellungsparameter setzen (set or unset options and positional parameters)

Das in die POSIX-Shell *sh* eingebaute Kommando *set* hat drei Funktionen:

- Ist kein Operand angegeben, gibt *set* die Namen und Werte aller Shell-Variablen, die für die aktuelle Shell definiert sind, auf die Standard-Ausgabe aus.
  Das Kommando *env* dagegen gibt die Namen und Werte aller Shell-Variablen auf die Standard-Ausgabe aus, die an jedes aufgerufene Kommando und jede Subshell weitergereicht werden. Variablen wie *IFS, MAILCHECK, PATH, PS1* und *PS2* sind in dieser Ausgabe nur enthalten, wenn sie mit *export* exportiert wurden.
- Entsprechend den angegebenen Optionen steuert *set* den Ablauf der aktuellen Shell. Die gleichen Optionen können Sie auch bei *sh* angeben, allerdings steuert *sh* den Ablauf der aufgerufenen Subshell.
- Mit Argumenten, die statt einer Option oder nach den Optionen angegeben sind, führt *set* folgendes aus:
  – die Stellungsparameter *$1, $2,* ... und *$9* versorgt *set* mit den ersten neun angegebenen Argumenten,
  – die Shell-Parameter *$** und *$@* versorgt *set* mit allen angegebenen Argumenten,
  – den Shell-Parameter *$#* versorgt *set* mit der Anzahl der angegebenen Argumente.

  Sind weniger als neun Argumente angegeben, werden die restlichen Stellungsparameter mit der leeren Zeichenkette versorgt.
  Mit *shift* können Sie das zehnte und weitere Aufruf-Argumente direkt ansprechen.

Wenn Sie mit *set* den Ablauf einer Shell-Prozedur beeinflussen wollen, müssen Sie dieses Kommando in die Prozedur schreiben, da Shell-Prozeduren immer von einer Subshell ausgeführt werden.

Syntax

Format 1: set␣[**+|-AabCefhkmnuvx**][␣**-t**][**-o**␣option...][␣argument...]

Format 2: set␣**--**␣[␣argument...]

Format 3: set␣**-**␣[␣argument...]

Format 1

Kein Operand angegeben
set gibt die Namen und Werte aller Shell-Variablen, die für die aktuelle Shell definiert sind, auf die Standard-Ausgabe aus.

Keine Option angegeben
set versorgt die Stellungsparameter *$1, $2,* ... und *$9* mit den ersten neun angegebenen Argumenten. Die Shell-Parameter $* und $@ versorgt *set* mit allen angegebenen Argumenten und den Shell-Parameter $# mit der Anzahl der angegebenen Argumente. Sind

weniger als neun Argumente angegeben, werden die restlichen Stellungsparameter mit der leeren Zeichenkette versorgt.
Mit *shift* können Sie das zehnte und weitere Aufruf-Argumente direkt ansprechen.

option
steuert den Ablauf der aktuellen Shell.

Wenn Sie mehrere Optionen angeben wollen, müssen Sie die entsprechenden Buchstaben ohne Leerzeichen aneinanderhängen. Der Bindestrich - darf nur vor dem ersten Buchstaben stehen
Die Option -- können Sie nicht mit anderen Optionen kombinieren.

Mit der Eingabe *echo $-* stellen Sie fest, welche Optionen bereits mit *set* oder *sh* gesetzt sind. Gesetzte Optionen können bis auf *-n* und *-t* wieder zurückgesetzt werden, ohne dass die aktuelle Shell beendet werden muss.

Folgende Optionen stehen zur Verfügung:

-A␣name
Wertzuweisung für Felder:
Löscht den Wert der Variablen *name* und weist ihr dann sequentiell aus der *argument*-Liste Werte zu.

+A Verwenden Sie das Pluszeichen *+A*, dann werden die Werte vor der Zuweisung nicht gelöscht.

-a (a - automatic) In der aktuellen Shell werden ab sofort alle Shell-Variablen automatisch exportiert, die Sie neu definieren oder denen Sie einen anderen Wert zuweisen.

-a nicht angegeben und noch nicht gesetzt:
Neu definierte oder geänderte Shell-Variablen werden in der aktuellen Shell nicht automatisch exportiert. Nur wenn Sie eine Shell-Variable mit dem eingebauten *sh*-Kommando *export* exportieren, ist sie in einer Subshell bekannt.

+a *-a* wird zurückgesetzt. Alle Shell-Variablen, die während gesetzter Option *-a* definiert wurden, werden auch weiterhin exportiert.

-b (b - background) Es werden alle beendeten Hintergrundjobs angezeigt. Folgende Meldung wird auf Standardfehlerausgabe geschrieben:

"[%d]%c %s%s\n", <job-number>,<current>,<status>,<job-name>

<current> Das Zeichen „+“ kennzeichnet den Job, der als Standard für das *fg* oder *bg* Kommando verwendet wird. Der Job kann auch in Form job_id %+ oder %% gekennzeichnet werden.
Das Zeichen „-“ kennzeichnet den Job, der Standard wird, wenn der aktuelle Standardjob beendet wird. Der Job kann auch in der Form job_id %- gekennzeichnet werden.

Alle anderen Jobs werden durch ein Leerzeichen dargestellt. Es kann immer nur ein Job mit „+“ und nur ein Job mit „-“ gekennzeichnet sein.

<job-number> Zahl, um die Prozessgruppe für die Kommandos *wait*, *fg*, *bg* und *kill* zu kennzeichnen. Werden diese Kommandos benutzt, wird der Jobnummer ein „%“ vorangestellt.

+b *-b* wird zurückgesetzt.

-C Verhindert das Überschreiben von Dateien bei Ausgabeumlenkung mit „>“. Wird die Ausgabe mit „>|“ umgelenkt, wird die *noclobber* Option für die einzelne Datei überlagert.

+C *-C* wird zurückgesetzt.

-e (e - exit) Die aktuelle Shell wird sofort beendet, falls ein Kommando den Ende-Status ≠ 0 zurückgibt.

-e nicht angegeben und noch nicht gesetzt:
Eine Dialog-Shell beendet sich, wenn Sie die Taste [END] drücken. Eine Shell, die eine Shell-Prozedur ausführt, beendet sich nur dann vorzeitig, wenn sie einen Syntax-Fehler entdeckt. Sonst beendet sie sich, wenn die Shell-Prozedur abgearbeitet ist, unabhängig vom Endestatus der einzelnen Kommandos.

+e *-e* wird zurückgesetzt.

-f (f - file name) In der aktuellen Shell verlieren ab sofort die Zeichen Stern * und Fragezeichen ? und die eckigen Klammern [...] ihre Sonderbedeutung. Das bedeutet, dass die entsprechenden Zeichenketten nicht durch passende Dateinamen ersetzt werden.

-f nicht angegeben und noch nicht gesetzt:
Entsprechende Zeichenketten werden durch passende Dateinamen ersetzt.

+f *-f* wird zurückgesetzt.

-h (h - hash) Die aktuelle Shell ändert ihr Verhalten bei der Definition von Shell-Funktionen wie folgt: Wenn Sie eine Shell-Funktion definieren, trägt die Shell bereits alle innerhalb der Funktion verwendeten Kommandos in die Hash-Tabelle ein (siehe *hash*).

-h nicht angegeben und noch nicht gesetzt:
Die innerhalb einer Shell-Funktion verwendeten Kommandos werden erst dann in die Hash-Tabelle eingetragen, wenn die Shell-Funktion ausgeführt wird.

+h *-h* wird zurückgesetzt.

-k (k - keyword) Variablen-Zuweisungen, also Angaben der Form *name=wert*, können Sie beim Aufruf eines Kommandos an beliebiger Stelle in der Kommando-Zeile angeben. Die Shell führt die Variablen-Zuweisung aus und trägt den entsprechenden Schlüsselwortparameter in die Ablauf-Umgebung dieses Kommandos ein (siehe *Beispiel 1*).

-k nicht angegeben und noch nicht gesetzt:
Variablen-Zuweisungen müssen vor dem Kommando-Namen stehen. Die Shell trägt die entsprechenden Schlüsselwortparameter in die Ablauf-Umgebung dieses Kommandos ein.

+k *-k* wird zurückgesetzt.

-m Hintergrundkommandos werden in einer eigenen Prozessgruppe ausgeführt. Die Beendigung wird in einer Zeile berichtet. Der Endestatus von Hintergrundaufträgen wird durch eine Beendigungsmeldung quittiert. Bei einer interaktiven POSIX-Shell wird diese Option automatisch eingeschaltet.

+m *-m* wird zurückgesetzt.

-n (n - no execution) Die aktuelle Shell liest und interpretiert alle Kommandos, führt sie aber nicht aus; d.h. der letzte Schritt bei der Bearbeitung einer Kommando-Zeile entfällt (siehe Abschnitt „Kommandos durch die POSIX-Shell verarbeiten“ auf Seite 42).
Mit *-n* können Sie feststellen, ob eine Shell-Prozedur Syntax-Fehler enthält. Die Option *-n* wird von einer interaktiven Shell ignoriert.

-n nicht angegeben und noch nicht gesetzt:
Alle Kommandos werden gelesen, interpretiert und ausgeführt.

+n *-n* wird zurückgesetzt (außer in einer interaktiven Shell).

-o ␣argument
argument kann einer der folgenden Optionsnamen sein:

allexport	entspricht *-a*
errexit	entspricht *-e*
bgnice	Alle Hintergrundaufträge werden mit niedrigerer Priorität ausgeführt. Dies ist die standardmäßige Voreinstellung.
ignoreeof	Die POSIX-Shell wird nicht durch Dateiende beendet. Benutzen Sie dazu das eingebaute Kommando.
keywords	entspricht *-k*
markdirs	Alle Dateiverzeichnisse, die bei der Dateinamen-Erzeugung entstehen, erhalten eine Schrägstrich */* am Ende angefügt.
monitor	entspricht *-m*
noclobber	Bei der Umlenkung der Ausgabe mit > werden keine existierenden Dateien überschrieben.
noexec	entspricht *-n*
noglob	entspricht *-f*
nolog	Funktionsdefinitionen werden nicht in der *History*-Datei gespeichert.

nounset	entspricht *-u*
privileged	entspricht *-p*
verbose	entspricht *-v*
trackall	entspricht *-h*
vi	ist abhängig vom Wert der Variablen TERM. Ist TERM=BLOCK oder ist TERM nicht gesetzt, dann wird ein BS2000-Blockterminal angenommen und das Argument *vi* wird nicht unterstützt. Jeder andere Wert für TERM bedeutet ein Zeichenterminal. Das Argument *vi* versetzt Sie in den Eingabemodus eines *vi*-ähnlichen Zeilen-Editors, bis die ESC drücken. Sie befinden sich dann im Kommandomodus. Ein Neue-Zeile-Zeichen sendet die Zeile.
viraw	ist abhängig vom Wert der Variablen TERM. Ist TERM=BLOCK oder ist TERM nicht gesetzt, dann wird ein BS2000-Blockterminal angenommen und das Argument vi wird nicht unterstützt. Jeder andere Wert für TERM bedeutet ein Zeichenterminal. Mit dem Argument *viraw* wird bei eingeschaltetem *vi*-Modus jedes Zeichen bearbeitet, sobald es getippt wird.
xtrace	entspricht *-x*

-t (t - terminate) Die aktuelle Shell beendet sich, wenn die aktuelle Kommando-Zeile ausgeführt ist. Das ist die Zeile, die das Kommando *set -t* enthält.

-t nicht angegeben:
Eine Dialog-Shell beendet sich, wenn Sie die Taste END drücken. Eine Shell, die eine Shell-Prozedur ausführt, beendet sich nur dann vorzeitig, wenn sie einen Syntax-Fehler entdeckt. Sonst beendet sie sich, wenn die Shell-Prozedur abgearbeitet ist.
Ist in der aktuellen Shell die Option *-e* gesetzt, beendet sie sich, sobald ein Kommando den Endestatus ≠ 0 zurückgibt.

-u (u - unset) Die aktuelle Shell gibt ab sofort eine Fehlermeldung aus, wenn sie in einer Kommando-Zeile auf eine Shell-Variable trifft, die nicht definiert ist. Das entsprechende Kommando wird nicht ausgeführt.
Shell-Prozeduren werden abgebrochen, sobald die Shell auf eine nicht definierte Shell-Variable trifft.

-u nicht angegeben und noch nicht gesetzt:
Die Shell gibt keine Fehlermeldung aus, wenn eine verwendete Shell-Variable nicht definiert ist. Standardmäßig wird die leere Zeichenkette als Wert eingesetzt.

+u *-u* wird zurückgesetzt.

-v (v - verbose) Die aktuelle Shell schreibt ab sofort jede Eingabe ohne Änderungen auf die Standard-Fehlerausgabe. Erst dann wird das entsprechende Kommando ausgeführt.
Zusammen mit der Option *-x* könnnen Sie so Shell-Prozeduren testen.

-v nicht angegeben und noch nicht gesetzt:
Die aktuelle Shell protokolliert die Eingabe nicht.

+v *-v* wird zurückgesetzt.

-x (x - execute) Die aktuelle Shell interpretiert die Eingabe und schreibt sie auf die Standard-Fehlerausgabe. Jedes so bearbeitete Kommando kennzeichnet die Shell mit einem Pluszeichen + am Zeilenanfang. Besteht ein eingegebenes Kommando nur aus Variablen-Zuweisungen, so wird es ohne + auf die Standard-Fehlerausgabe geschrieben. Anschließend wird das entsprechende Kommando ausgeführt.
Im Gegensatz zur Ausgabe bei Option *-v* erkennen Sie an dieser Ausgabe, wie die Shell die angegebenen Sonderzeichen und Shell-Variablen ersetzt hat.

Shell-Prozeduren können Sie also wie folgt testen:
- Tragen Sie *set -xv* als erstes Kommando in die entsprechende Shell-Prozedur ein oder
- führen Sie die entsprechende Shell-Prozedur aus mit dem Kommando *sh -xv prozedur*.

-x nicht angegeben und noch nicht gesetzt:
Die aktuelle Shell protokolliert die bearbeitete Eingabe nicht.

+x *-x* wird zurückgesetzt.

argument
beliebige Zeichenkette, die jeweils durch Leer- oder Tabulatorzeichen begrenzt wird. Das letzte Argument wird durch ein Kommando-Trennzeichen abgeschlossen.

Sie können beliebig viele Argumente angeben, jeweils getrennt durch mindestens ein Tabulator- bzw. ein Leerzeichen.

set versorgt die Stellungsparameter *$1, $2,* ... und *$9* mit den ersten neun angegebenen Argumenten. Die Shell-Parameter *$** und *$@* versorgt *set* mit allen angegebenen Argumenten und den Shell-Parameter *$#* mit der Anzahl der angegebenen Argumente. Sind weniger als neun Argumente angegeben, werden die restlichen Stellungsparameter mit der leeren Zeichenkette versorgt.
Mit *shift* können Sie das zehnte und weitere Aufruf-Argumente direkt ansprechen.

argument nicht angegeben:
Die Stellungsparameter sowie die Shell-Parameter *$@, $** und *$#* ändern sich nicht.

Format 2 **set --** [␣argument...]

- - (Minuszeichen) kann nicht mit anderen Optionen kombiniert werden.

Das erste Argument, das Sie nach *--* angeben, interpretiert *set* nicht als Option, auch wenn es mit einem Bindestrich - beginnt. Auf diese Weise können Sie mit *set* den Stellungsparameter $1 mit einer Zeichenkette versorgen, die mit einem Bindestrich beginnt (siehe dazu *Beispiel 3*).

Mit *set --* ändern Sie die aktuell gesetzten Optionen nicht.

argument

beliebige Zeichenkette, die jeweils durch Leer- oder Tabulatorzeichen begrenzt wird. Das letzte Argument wird durch ein Kommando-Trennzeichen abgeschlossen.

Sie können beliebig viele Argumente angeben, jeweils getrennt durch mindestens ein Tabulator- bzw. ein Leerzeichen.

set versorgt die Stellungsparameter *$1, $2,* ... und *$9* mit den ersten neun angegebenen Argumenten. Die Shell-Parameter *$** und *$@* versorgt *set* mit allen angegebenen Argumenten und den Shell-Parameter *$#* mit der Anzahl der angegebenen Argumente. Sind weniger als neun Argumente angegeben, werden die restlichen Stellungsparameter mit der leeren Zeichenkette versorgt.
Mit *shift* können Sie das zehnte und weitere Aufruf-Argumente direkt ansprechen.

argument nicht angegeben:
Die Stellungsparameter sowie die Shell-Parameter *$@, $** und *$#* ändern sich nicht.

Format 3 **set -** [␣argument...]

- Die Optionen *-x* und *-v* werden zurückgesetzt.

argument

set versorgt die Stellungsparameter *$1, $2,* ... und *$9* mit den ersten neun angegebenen Argumenten. Sind weniger als neun Argumente angegeben, werden die restlichen Stellungsparameter mit der leeren Zeichenkette versorgt.
Mit *shift* können Sie das zehnte und weitere Aufruf-Argumente direkt ansprechen.

argument nicht angegeben
Die Stellungsparameter ändern sich nicht.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *set*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die Shell-Prozedur *archiv* hat folgenden Inhalt:

```
set | grep kunde
echo $kunde
```

In der aktuellen Shell wird die Option *-k* gesetzt und dann die Shell-Prozedur wie folgt ausgeführt:

```
$ set -k
$ sh archiv kunde=Meier
kunde=Meier
Meier
```

Nach *set -k* kann die Shell-Variable *kunde* in der Kommandozeile nach dem Prozedur-Namen definiert werden. Diese Variable ist aber nur innerhalb der Prozedur bekannt; im Beispiel greifen die Kommandos *echo* und *set* darauf zu.

Nach Ablauf der Shell-Prozedur ist auch die Subshell beendet, die diese Prozedur ausgeführt hat. Die jetzt gültige Shell kennt die Variable *kunde* nicht. Das folgende Kommando gibt deshalb nichts aus:

```
$ set | grep kunde
```

Jetzt wird die Option *-k* zurückgesetzt. Wenn die Shell-Prozedur wieder genauso aufgerufen wird wie zuvor, ist die Shell-Variable *kunde* innerhalb der Prozedur nicht definiert. Das Kommando *echo* gibt nur eine Leerzeile aus.
In diesem Fall muss also die Variablen-Definition in der Kommandozeile vor dem Aufruf der Prozedur stehen.

```
$ set +k
$ sh archiv kunde=Meier
$ kunde=Meier sh archiv
kunde=Meier
Meier
```

Beispiel 2 Die Shell-Prozedur *evaltest* hat folgenden Inhalt:

```
set –xv
eval echo \$$#
echo $0
```

Mit *set* werden die Optionen *-x* und *-v* in der Subshell gesetzt, die diese Shell-Prozedur ausführt. Die Shell-Prozedur wird aufgerufen:

```
$ sh evaltest heute ist freitag
eval echo \$$#
+ eval echo $3
+ echo freitag
freitag
echo $0
+ echo evaltest
evaltest
```

Die Option *-v* bewirkt, dass jedes Kommando vor der Bearbeitung unverändert ausgegeben wird.

Die Option *-x* bewirkt, dass die Shell jede Kommando-Zeile interpretiert und mit einem + versehen ausgibt. Die Ausgabe zeigt, dass die Aufruf-Argumente des Kommandos *eval* von der Shell zweimal interpretiert werden.

Beispiel 3 Die folgende Shell-Prozedur zählt die Argumente, die von der Standard-Eingabe gelesen werden. Allerdings ist sie etwas langsamer als das Kommando *wc -w*:

```
1  : sh zaehl zaehlt alle Argumente auf Standard-Eingabe
2  zahl=0
3  while read zeile
4  do
5     set -- $zeile
6     zahl=`expr $zahl + $#`
7  done
8  echo $zahl
```

Zeile 3:
Das eingebaute *sh*-Kommando *read* liest eine Zeile von der Standard-Eingabe und weist sie der Variablen *zeile* zu.

Zeile 5:
Das eingebaute *sh*-Kommando *set* weist die Argumente, aus denen sich der Wert der Variablen *zeile* zusammensetzt, den Stellungsparametern *$1, $2*, ... zu; *$#* ist die Anzahl der Argumente, aus denen sich *$zeile* zusammensetzt.

Die Option -- hat in diesem Beispiel zwei Funktionen:
– Wenn *$zeile* die leere Zeichenkette ist, gibt *set* trotzdem nicht die aktuelle Umgebung aus.
– Wenn *$zeile* mit einem Bindestrich beginnt, interpretiert *set* das erste Argument aus *$zeile* nicht als Option.

Zeile 6:
Für jede gelesene Zeile addiert *expr* die Anzahl Argumente in dieser Zeile, also $# zum aktuellen Wert der Variablen *zahl*. Anschließend enthält *zahl* den neuen Wert.

Die Kommandos in den Zeilen 5 und 6 werden so oft wiederholt, bis *read* die letzte Eingabe-Zeile gelesen hat. Dann gibt *echo* den Inhalt der Variablen *zahl* aus; das ist die Anzahl aller eingegebenen Argumente.

Siehe auch *env, getopts, typeset*

sh Kommandointerpreter und Programmiersprache der POSIX-Shell (shell, the standard command language interpreter)

Kommandos werden in der Login-Shell nacheinander aus der Datei */etc/profile* und dann, falls eine der Dateien existiert, entweder aus *.profile* des aktuellen Verzeichnisses oder aus *$HOME/.profile* gelesen. Danach werden noch die Kommandos aus der Datei gelesen, deren Name durch den Wert der Shell-Variable *ENV* nach Parametersubstitution gegeben wird.

Syntax **sh**[␣option...][␣datei][␣argument]...

Keine Option angegeben
Eine Subshell mit der Option *-s* wird aufgerufen. Der zuvor aktuelle Shell-Prozess wird zum Vater der neuen Shell. Die Subshell können Sie terminalabhängig entweder mit END oder @@d oder mit dem Kommando *exit* beenden

option
sh interpretiert beim Aufruf die nachfolgenden Optionen und zusätzlich die Optionen, die beim eingebauten Kommando *set* der POSIX-Shell beschrieben sind:

-c␣kommando_zeichenkette
Die Kommandos werden aus *kommando_zeichenkette* gelesen.

-s Ist Option *-s* angegeben oder fehlen *datei* und *argument*, dann liest die POSIX-Shell die Kommandos von der Standard-Eingabe und schreibt die Ausgaben auf die Standard-Ausgabe. Nur Diagnosemeldungen werden auf die Standard-Fehlerausgabe geschrieben.

-i Ist Option *-i* angegeben oder sind Standard-Eingabe und -Ausgabe mit einer Datensichtstation verbunden, dann wird eine interaktive POSIX-Shell aufgerufen. In diesem Fall wird zum einen TERM ignoriert, so dass *kill 0* keine interaktive Shell beendet, und zum anderen INTR abgefangen und ignoriert, damit *wait* unterbrechbar ist. Auf jeden Fall wird QUIT von der POSIX-Shell ignoriert.

-r Mit Option *-r* wird eine eingeschränkte POSIX-Subshell gestartet. In dieser Shell gelten folgende Einschränkungen:
- Das eingebaute *sh*-Kommando *cd* wird abgewiesen, d.h. Sie können Ihr aktuelles Dateiverzeichnis nicht verlassen
- Sie können den Wert der Variablen PATH nicht ändern
- Kommandos werden abgewiesen, wenn der Name der zugehörigen Kommandodatei einen Schrägstrich / enthält. Sie können nur solche Kommandos ausführen, die in Ihrem aktuellen Dateiverzeichnis stehen oder in den Dateiverzeichnissen, deren Pfade der Shell-Variablen PATH zugewiesen sind.
- Kommandos werden abgewiesen, wenn der Aufruf die Zeichen > oder >> enthält. Die Ausgabe der Kommandos kann also nicht mit > oder >> in eine Datei umgelenkt werden.

Die eingeschränkte Subshell können Sie terminalabhängig entweder mit END oder @@d oder mit dem Kommando *exit* beenden

datei
Wurde die Option *-s* nicht angegeben, dafür aber *datei*, dann wird entsprechend dem Suchpfad nach der Shell-Prozedur *datei* gesucht. Für die Prozedur muss das Leserecht gesetzt sein. Falls das s-Bit für Eigentümer oder Gruppe gesetzt ist, so wird dieses ignoriert. Kommandos werden wie im Folgenden beschrieben gelesen.

argument
Als Argumente können Sie Kommandos angeben. Der Kommandoname wird als nulltes Argument übergeben. Die Kommandos sind unten beschrieben.
Wird die POSIX-Shell vom Systemaufruf *exec* aufgerufen und ist das erste Zeichen des nullten Arguments ein Bindestrich -, dann wird die Shell als *login*-Shell behandelt.

Endestatus Normalerweise gibt die POSIX-Shell den Endestatus des letzten ausgeführten Kommandos zurück (siehe *exit*).
Von der POSIX-Shell erkannte Fehler, wie z.B. Syntaxfehler, führen zu einem Endestatus ungleich 0. Wird die POSIX-Shell nicht-interaktiv benutzt, dann wird die Bearbeitung der Prozedur-Datei abgebrochen. Von der POSIX-Shell erkannte Laufzeit-Fehler werden durch Ausgabe des Kommandonamens und der Fehlerbedingung berichtet. Ist die Zeilennummer der Zeile mit dem fehlerhaften Kommando größer eins, dann wird nach dem Kommando- oder Funktionsnamen noch die Zeilennummer in eckigen Klammern *[...]* angegeben.

Datei *\/etc/profile*
\/etc/suid_profile
$HOME/.profile
*\/tmp/sh**
\/dev/null

Variable Folgende Variablen wirken sich auf *sh* aus:
FCEDIT, HISTFILE, HISTSIZE, HOME, IFS, MAIL, MAIL, MAILCHECK, MAILPATH und *PATH*
(Beschreibung siehe Abschnitt „POSIX-Shell-Variablen und Parameter-Ersetzung“ auf Seite 50ff).

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *sh*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für die Bedeutung von Zeichenbereichen, Äquivalenzklassen und Zeicheneinheiten in Mustervergleichen fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien). Die Variable bestimmt außerdem, welche Zeichen als Buchstaben definiert sind sowie das Verhalten von Zeichenklassen im Mustervergleich.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *cat, cd, chmod, cut, echo, env, paste, stty, test, umask, vi*
dup(), exec(), fork(), getrlimit(), ioctl(), lseek(), pipe(), signal(), umask(), ulimit(), wait(), rand() [4]

shift Die Werte der Stellungsparameter nach links verschieben (shift positional parameters)

Das in die POSIX-Shell *sh* eingebaute Kommando *shift* verschiebt die Werte der Stellungsparameter nach links. Wenn Sie *shift* ohne Argument aufrufen, sieht diese Verschiebung so aus:

- Der Shell-Parameter *$0* bleibt unverändert.
- Der ursprüngliche Wert von *$1* fällt heraus, auf diesen Wert können Sie nicht mehr zugreifen.
- Stattdessen erhält *$1* den Wert von *$2*, *$2* den Wert von *$3*, ..., *$8* den Wert von *$9*.
- Der Stellungsparameter *$9* wird mit dem zehnten Aufruf-Argument versorgt.
- *$#* wird um 1 erniedrigt.
- *$** und *$@* enthalten alle Aufruf-Argumente, beginnend mit dem neuen Wert von *$1*. Der ursprüngliche Wert von *$1* ist herausgefallen.

Die Shell bietet standardmäßig nur die Möglichkeit, mit den Stellungsparametern die ersten 9 Aufruf-Argumente des Kommandos *set* bzw. einer Shell-Prozedur direkt anzusprechen. Mit *shift* können Sie diese Einschränkung umgehen, denn es verschiebt die Werte entsprechend weit nach links.

Syntax **shift**[␣zahl]

zahl
: Eine positive ganze Zahl; *shift* wird *zahl*-mal ausgeführt. Der Stellungsparameter *$1* erhält als Wert das (*zahl*+1)te Aufruf-Argument, usw.

  Wenn Sie *zahl* zu groß wählen, gibt *shift* eine Fehlermeldung aus, sobald für *$1* kein Aufruf-Argument mehr übrig bleibt; d.h. *$#* ist gleich 0.

  zahl nicht angegeben:
  shift wird einmal ausgeführt.

  Wenn *$1* bereits das letzte Aufruf-Argument enthält, so erhalten Sie beim nächsten Aufruf von *shift* eine Fehlermeldung.

Fehler `sh: shift: bad number`
Diese Fehlermeldung erhalten Sie, wenn *zahl* >*$#* ist. Dem Stellungsparameter *$1* konnte also kein Wert zugewiesen werden.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *shift*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwen-

det. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Innerhalb einer Shell-Prozedur soll das zehnte Aufruf-Argument angesprochen werden:

```
:
arg1=$1
shift
arg10=$9
:
```

Die Variable *arg1* enthält den ursprünglichen Wert von *$1*. So ist dieser Wert auch nach *shift* noch verfügbar.

Beispiel 2 Der folgende Bildschirmdialog soll zeigen, wie sich bei *shift* der Inhalt der Shell-Parameter *$1, $** und *$#* ändert:

```
$ set 1 2 3 4 5 6 7 8 9 10 11 12
$ echo $1
1
$ echo $#
12
$ echo $*
1 2 3 4 5 6 7 8 9 10 11 12
$ shift 6
$ echo $1
7
$ echo $*
7 8 9 10 11 12
$ echo $#
6
```

Siehe auch *set*

show_pubset_export Vom Export eines Pubsets betroffene Dateisysteme anzeigen

Das Kommando liefert dem Systemadministrator die Information, welche Dateisysteme in POSIX vom Export eines Pubsets betroffen sind und somit vor der Ausführung des Kommando EXPORT-PUBSET ausgehängt werden müssen.

Diese Information ist insbesondere beim Einsatz von bs2fs-Dateisystemen hilfreich. Dabei genügt es nämlich nicht, wie bei ufs- und NFS-Dateisystemen, zu prüfen, ob der Einhängepunkt eines Dateisystems auf dem betroffenen Pubset liegt.

Bei bs2fs-Dateisystemen ist zusätzlich auch die Lage der mittels bs2fs eingehängten BS2000-Dateien relevant. Außerdem spielt der Einhängepunkt des bs2fs-Containers eine Rolle. Liegt dieser auf dem zu exportierenden Pubset, so sind alle eingehängten bs2fs-Dateisysteme vom Export betroffen, unabhängig von ihrer Lage.

Abhängig vom Dateisystemtyp ist ein Dateisystem dann vom EXPORT-PUBSET betroffen, wenn die in der folgenden Liste aufgeführten Objekte auf dem zu exportierenden Pubset liegen:

ufs:
: Einhängepunkt oder Behälterdatei

nfs:
: Einhängepunkt

bs2fs:
: Einhängepunkt oder eingehängte BS2000-Dateien oder Einhängepunkt bzw. Behälterdatei des bs2fs-Containers (vgl. ufs)

Syntax

show_pubset_export cat-id

cat-id
: Katalog-Kennung des Pubsets, das geprüft werden soll (ohne einschließende Doppelpunkte „:“). Die Angabe kann in Groß- oder Kleinschreibung oder gemischt erfolgen, die Prüfung wird mit der in Großbuchstaben umgewandelten *cat-id* durchgeführt.

Datei

Die folgenden Dateien werden zur Ermittlung der betroffenen Dateisysteme nach der angegebenen Katalog-Kennung durchsucht:

/etc/mnttab
Tabelle aller eingehängten Dateisysteme

/etc/partitions
Tabelle aller möglichen Partitions
Falls in dieser Datei die Angabe der Katalogkennung fehlt, so wird die Default-Id über das BS2000 ermittelt und für die Prüfung verwendet.

SYSSSI.POSIX-BC.<version>
SYSSSI-Datei von POSIX aus dem BS2000
Aus dieser Datei wird der BS2000-Dateiname der Behälterdatei des root-Dateisystems ermittelt (Parameter ROOTFSNAME), da dieser Name nicht in der Datei */etc/partitions* eingetragen ist.

Beispiel

Die vom Export des Pubsets DATA betroffenen Dateisysteme werden ermittelt. Die Behälterdatei des unter dem Home-Verzeichnis */home/bach* eingehängten ufs-Dateisystems liegt auf dem Pubset DATA.

```
# show_export DaTa

the nfs filesystems on pubset DATA

nfs filesystem PGTR0157:/home4 mounted on /home/bach/nfs4
nfs filesystem PGTR0157:/home5 mounted on /home/bach/nfs5

the bs2fs filesystems on pubset DATA

bs2fs filesystem :DATA:$BACH.ASS.* mounted on /home/bach/bs2/mount.ass
bs2fs filesystem :V70A:$BACH.CCC.* mounted on /home/bach/bs2/mount.ccc
bs2fs filesystem :DATA:$BACH.PLAM* mounted on /home/bach/bs2/mount.plam

the ufs filesystems on pubset DATA

ufs filesystem /dev/dsk/4 mounted on /home/froede
ufs filesystem /dev/dsk/5 mounted on /home/bach
#
```

sleep Prozesse zeitweise stillegen (suspend execution for an interval)

Das Kommando *sleep* verzögert die Ausführung des Prozesses, der es aufgerufen hat, um eine frei wählbare Zeitspanne.
sleep benutzt man vor allem in Shell-Prozeduren, um die Ausführung des nächsten Kommandos zu verzögern.

Syntax

sleep␣zeit

zeit
Zeit in Sekunden, um die die Ausführung des Prozesses verzögert werden soll.
Sie müssen für *zeit* eine nicht-negative Dezimalzahl angeben.

Fehler

`sleep: bad character in argument`
Sie haben für *zeit* eine negative Dezimalzahl oder einen nichtnumerischen Ausdruck angegeben.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *sleep*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 In diesem Beispiel benutzen Sie *sleep* von der Kommandoebene aus. Sie starten einen Prozess im Hintergrund, der Sie in 10 Minuten (600 Sekunden) daran erinnert, einen Telefonanruf zu erledigen:

```
$ (sleep 600; echo 'Herrn Meier anrufen!')&
696
```

Beispiel 2 In diesem Beispiel benutzen Sie *sleep* innerhalb einer Shell-Prozedur namens *immer*. *immer* ruft alle 2 Minuten (120 Sekunden) das Programm *aufraeumen* auf:

```
$ cat immer
while true
   do
      aufraeumen
      sleep 120
   done
```

Wenn Sie die Prozedur *immer* im Hintergrund ablaufen lassen, können Sie sie nur mit dem Kommando *kill*, aber nicht mit der Taste DEL abbrechen.

Siehe auch *alarm()*, *sleep()* [4]

sort Dateien sortieren und/oder mischen (sort, merge or sequence check text files)

sort sortiert die Zeilen der Eingabedatei und schreibt das Ergebnis auf die Standard-Ausgabe.

Geben Sie mehrere Eingabedateien an, sortiert und mischt *sort* die Dateien in einem Arbeitsgang, d.h. der Inhalt aller Eingabedateien wird sortiert ausgegeben.

Sie können entweder nach der ganzen Zeile oder nach bestimmten Zeilenausschnitten sortieren. Ein solcher Zeilenausschnitt heißt Sortierfeld. Wenn Sie nach der ganzen Zeile sortieren wollen, geben Sie keine Sortierfelder an. Wenn Sie nur nach bestimmten Ausschnitten der Zeilen sortieren wollen, geben Sie ein oder mehrere Sortierfelder an. Ein Sortierfeld geben Sie anhand der Felder einer Zeile durch Positionsangaben in der Form *+pos1 -pos1* an (siehe *Festlegen bestimmter Sortierfelder*). Beachten Sie den Unterschied zwischen einem Feld und einem Sortierfeld!

sort teilt die Zeilen einer Datei in Felder ein. Ein Feld ist eine Zeichenkette, die durch ein Feldtrennzeichen oder Neue-Zeile-Zeichen abgegrenzt wird. Standardmäßig sind Leerzeichen und Tabulatorzeichen Feldtrennzeichen. Bei einer Folge von einem oder mehreren Standard-Feldtrennzeichen gehören alle Standard-Feldtrennzeichen zum folgenden Feld. Führende Leerzeichen in einer Zeile gehören somit standardmäßig zum ersten Feld.

Syntax

Format 1: sort[␣**-m**][␣**-o**␣ausgabe_datei][␣**-bdfiMnru**][␣**-t**␣*Z*eichen]
[**-k**␣keydef...][␣−**z**␣recsz][␣**-y**[kmem]][␣**-T**␣dateiverzeichnis][␣datei...]

Format 2: sort␣**-c**[␣**-bdfiMnru**][␣**-t**␣*Z*eichen]
[**-k**␣keydef...][␣−**z**␣recsz][␣**-y**[kmem]][␣**-T**␣dateiverzeichnis][␣datei...]

Format 3: sort[␣**-m**][␣**-o**␣ausgabe_datei][␣**-bdfiMnru**][␣**-t**␣*Z*eichen]
[␣**+**pos1[␣**-**pos2]][␣−**z**␣recsz][␣**-y**[kmem]][␣**-T**␣dateiverzeichnis][␣datei...]

Format 4: sort␣**-c**[␣**-bdfiMnru**][␣**-t**␣*Z*eichen]
[␣**+**pos1[␣**-**pos2]][␣−**z**␣recsz][␣**-y**[kmem]][␣**-T**␣dateiverzeichnis][␣datei...]

Die Formate werden gemeinsam beschrieben, da
- die Optionen *-m* und *-o␣ausgabe_datei* in Format 1 und 3 gegen die Option *-c* in Format 2 und 4 ausgetauscht sind
- die Option *-k␣keydef* in Format 1 und 2 gegen die Option *+pos1[␣-pos2]* in Format 3 und 4 ausgetauscht sind.

Keine Option angegeben
sort sortiert die Eingabezeilen lexikographisch, wobei als Einzelzeichen jeweils ein Byte verwendet wird. Für die Bytes gilt die Sortierreihenfolge der Maschine.

Optionen, die das Verhalten von sort ändern

Wollen Sie mehrere dieser Optionen verwenden, müssen Sie diese jeweils einzeln mit Bindestrich und durch Leerzeichen voneinander getrennt angeben.

-c (c - check) *sort* prüft nur, ob die Eingabedatei bereits entsprechend den gültigen Sortierkriterien sortiert ist. Wenn ja, wird nichts ausgegeben. Wenn nein, wird die erste Zeile ausgegeben, die den Sortierkriterien nicht entspricht.

Zusammen mit Option *-c* dürfen Sie nur eine Datei angeben!

-m (m - merge) *sort* mischt Eingabedateien zusammen, die bereits sortiert sind.

-o␣ausgabe_datei
(o - output) Für *ausgabe_datei* geben Sie den Namen einer Datei an, in die *sort* den sortierten Inhalt der Eingabedatei schreiben soll. Sie können für *ausgabe_datei* auch eine Eingabedatei angeben. Deren ursprünglicher, nicht sortierter Inhalt wird dann allerdings überschrieben.

-o ausgabe_datei nicht angegeben:
sort schreibt auf die Standard-Ausgabe.

-T␣dateiverzeichnis
Temporärdateien werden in *dateiverzeichnis* angelegt.

-u (u - unique) Identische Zeilen werden nur einmal ausgegeben. Als identische Zeilen zählen Zeilen mit identischen Sortierfeldern.

-y[kmem]
Mit der Option *-y* legen Sie fest, mit welcher Speicherplatzgröße *sort* zu sortieren anfängt. Die Ausführungsgeschwindigkeit von *sort* hängt in hohem Maße von diesem zu Beginn bereitgestellten Speicherplatz ab. Denn es ist eine Verschwendung von Speicherplatz bzw. von Rechenzeit, eine kleine Datei in einem großen Speicher bzw. eine große Datei in einem kleinen Speicher zu sortieren.

kmem
Größe des Speichers in Kbyte, mit dem *sort* zu sortieren beginnt. Geben Sie für *kmem* einen Wert an, der über dem oberen Grenzwert von 1 Mbyte bzw. unter dem unteren Grenzwert von 16 Kbyte liegt, wird der entsprechende Grenzwert verwendet. So startet *sort* z.B. mit dem minimalen Speicherplatz, wenn Sie für *kmem* den Wert 0 angeben: *-y0*

kmem nicht angegeben:
sort startet mit dem maximalen Speicherplatz.

-y[kmem] nicht angegeben:
sort startet mit einer Standard-Speichergröße von 32 Kbyte und benützt mehr Speicher, falls mehr benötigt wird.

-z␣recsz

Mit der Option *-z* stellen Sie für die Mischphase Puffer von ausreichender Größe bereit. Dies ist nur notwendig, wenn Option *-c* oder *-m* gesetzt ist, d.h. wenn nicht sortiert wird:

Wenn sortiert wird, speichert *sort* die Länge der längsten in der Sortierphase gelesenen Zeile und stellt dadurch für die Mischphase ausreichend große Puffer bereit.

Wenn nicht sortiert wird, verwendet *sort* für die Puffergröße normalerweise einen Standard-Wert. Zeilen, die länger sind als die reservierte Puffergröße, führen zur abnormalen Beendigung von *sort*. Dies können Sie verhindern, indem Sie die Länge der längsten zu mischenden Zeile bzw. einen noch größeren Wert in byte für *recsz* angeben.

Optionen zum Ändern der Sortierkriterien

Die folgenden Optionen können Sie auf zwei Arten angeben:

– entweder vor der ersten Positionsangabe *+pos1*:

  Sie gelten dann global für alle mit *+pos1* angegebenen Sortierfelder.

  Bei mehreren Optionen werden wie üblich alle einzeln mit Bindestrich und durch Leerzeichen voneinander getrennt oder nur die erste mit Bindestrich und die übrigen direkt ohne Bindestrich und Leerzeichen angehängt angegeben.

– oder nach einer Positionsangabe *+pos1* oder *-pos2*:

  Sie heben dann für das mit dieser Positionsangabe angesprochene Sortierfeld die globalen Einstellungen auf. D.h. für dieses Sortierfeld gilt die Änderung des Sortierkriteriums gemäß dieser Option.

  Diese Optionen werden ohne Bindestrich und ohne Leerzeichen direkt an *+pos1* oder *-pos2* angehängt.

-b *sort* ignoriert führende Feldtrennzeichen bei der Ermittlung von Anfang und Ende eines Sortierfeldes. Die Option *-b* ist jedoch nur wirksam, wenn nach Sortierfeldern und nicht nach der ganzen Zeile sortiert wird.

-d *sort* sortiert lexikalisch, d.h. nur Zeichen werden berücksichtigt, für die die C-Funktionen *isalnum()* oder *isspace()* das Ergebnis „wahr“ zurückliefern. Das sind Zeichen, die in der aktuell gültigen Umgebung als alphanumerische Zeichen oder als Zeichen definiert sind, oder die einen Zwischenraum produzieren, z.B. Leer- oder Tabulatorzeichen.

-f *sort* behandelt Groß- und Kleinbuchstaben gleich. Vor dem Sortierverfahren werden Kleinbuchstaben durch Großbuchstaben ersetzt.

-i Bei nicht numerischen Vergleichen der Sortierfelder werden alle Zeichen, für die die C-Funktion *isprint()* das Ergebnis „falsch“ zurückliefert, nicht berücksichtigt. Dies sind Zeichen, die in der aktuell gültigen Umgebung als nicht druckbar definiert sind. Liegt für

die Sortierreihenfolge z.B. die ASCII-Tabelle zugrunde, werden die Zeichen 001-037 (oktal) einschließlich und das Zeichen 0177 (oktal) nicht berücksichtigt (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*).

-M Die ersten drei Zeichen des Sortierfeldes werden in Großbuchstaben verwandelt, als Monatsnamen betrachtet und entsprechend der Reihenfolge der Monate sortiert. Die Option *-M* impliziert die Option *-b*.

-n (n - number) *sort* sortiert nach Zahlenwerten. Ein Zahlenwert muss am Anfang des Sortierfeldes stehen und kann bestehen aus: Leerzeichen, Minuszeichen, Ziffern 0-9, Dezimalpunkt. Mit der Option *-n* wird automatisch die Option *-b* gesetzt, d.h. führende Leerzeichen werden ignoriert.

-r (r - reverse) *sort* sortiert in umgekehrter Reihenfolge.

Option zum Ändern der Feldtrennzeichen in den Eingabezeilen

Diese Option müssen Sie einzeln mit Bindestrich angeben.

-t␣zeichen
sort behandelt das Zeichen, das Sie für *zeichen* angeben, als Feldtrennzeichen. Im Unterschied zu den Standard-Feldtrennzeichen gehört *zeichen* selbst nicht zu einem Feld. Es kann aber Teil eines Sortierfeldes sein, z.B. wenn das Sortierfeld vom ersten bis dritten jeweils durch *zeichen* getrennten Feld reicht. Jedes Trennzeichen *zeichen* ist signifikant, d.h. *zeichenzeichen* begrenzt ein leeres Feld.

-t␣zeichen nicht angegeben:
Es gelten die Standard-Feldtrennzeichen: Leer- und Tabulatorzeichen. Eine Folge von einem oder mehreren Standard-Feldtrennzeichen gehört zum nachfolgenden Feld.

Festlegen bestimmter Sortierfelder

Beachten Sie bei der Angabe von Sortierfeldern, dass Buchstabenfolgen, die in der aktuell gültigen Umgebung als Zeicheneinheit definiert sind, als ein Buchstabe zählen. In einer spanischen Umgebung wäre z.B. *ch* eine Zeicheneinheit.

-k␣keydef
Festlegen der Sortierfelder. *keydef* ist als Sortierfeld in der folgenden Form definiert:

```
Anfang_des_Sortierfeldes[typ][,Ende_des_Sortierfeldes[typ]]
```

wobei Anfang_des_Sortierfeldes *+pos1* entspricht und Ende_des_Sortierfeldes *-pos2* (siehe nachfolgende Beschreibung). *typ* entspricht einer der Optionen *b, d, f, i, n* oder *r*.

+pos1[␣**-**pos2]
Mit *+pos1 -pos2* bestimmen Sie anhand der Felder der Eingabezeilen Anfang und Ende eines Sortierfeldes.

+pos1 legt die Position des ersten Zeichens *im* Sortierfeld,
-pos2 legt die Position des ersten Zeichens *nach* dem Sortierfeld fest. *+pos1* muss vor *-pos2* stehen.

-pos2 nicht angegeben:
Das Sortierfeld geht von *+pos1* bis zum Zeilenende.

Die Argumente *pos1* und *pos2* haben das Format:

m[.n]

m und *n* sind ganze Zahlen, die folgende Bedeutung haben:

m *m* Felder der Zeile überspringen. Angesprochen ist also Feld *m+1*.

.n *n* Zeichen einschließlich Feldtrennzeichen nach dem letzten Zeichen von Feld *m* überspringen. Angesprochen ist also Zeichen *n+1* im Feld *m+1*. Ist Option *-b* angegeben, zählen Feldtrennzeichen am Feldanfang nicht mit, *+m.nb* spricht also das *n+1*te Nicht-Trennzeichen nach dem Feld *m* an.

.n nicht angegeben:
Dies ist mit .0 gleichbedeutend und spricht also das erste Zeichen nach dem Feld *m* an. Ist Option *-b* angegeben, zählen Feldtrennzeichen am Feldanfang nicht mit, *+m.0b* spricht also das erste Nicht-Trennzeichen nach dem Feld *m* an.

Beispiel

Das Sortierfeld beginnt im zweiten Feld beim vierten Zeichen und endet mit diesem Feld. Sie geben das Sortierfeld so an:

```
sort +1.3 -2
```

Erläuterung:

```
        Ende        Ende    Ende
        Feld1       Feld2   Feld3
           |           |       |
   030-456537 A.Meierlein München
              |       |
              Sortierfeld
```

+1.3
Feld 1 und 3 Zeichen überspringen:
das 4. Zeichen nach Feld 1 ist das 1. Zeichen im Sortierfeld: M

-2
Feld 2 und 0 Zeichen überspringen:
das 1. Zeichen nach Feld 2 ist das 1. Zeichen nach dem Sortierfeld: Leerzeichen.
Das Zeichen davor ist also das letzte Zeichen im Sortierfeld: n

Beachten Sie, dass Standard-Feldtrennzeichen, anders als ein mit Option *-t* definiertes Feldtrennzeichen, zum nachfolgenden Feld gehören. Das 1. Zeichen von Feld 2 ist somit das Leerzeichen, das 2. Zeichen das A usw.

Gibt es mehrere Sortierfelder, so sortiert *sort* zunächst nach dem ersten, bei Gleichheit im ersten Sortierfeld nach dem nächsten usw.

datei

Name der Datei, die Sie sortieren möchten.
Sie können mehrere Dateien angeben. Alle angegebenen Dateien werden sortiert und zusammengemischt, so dass die Eingabezeilen aus allen Dateien sortiert auf die Standard-Ausgabe ausgegeben werden. In der Eingabedatei zählen alle Buchstabenfolgen als ein Buchstabe, die in der aktuell gültigen Umgebung als Zeicheneinheit definiert sind. In einer spanischen Umgebung wäre z.B. *ch* eine Zeicheneinheit. Fehlt in der letzten Zeile einer Datei *datei* ein Neue-Zeile-Zeichen, so fügt *sort* es ein, gibt eine Warnung aus und setzt die Bearbeitung fort.

Zusammen mit Option *-c* dürfen Sie nur eine Datei angeben!

Wenn Sie für *datei* einen Bindestrich - angeben, liest *sort* von der Standard-Eingabe.

datei nicht angegeben:
sort liest von der Standard-Eingabe.

Endestatus Folgende Endewerte können auftreten:

0 alle Eingabedateien wurden erfolgreich bearbeitet. War *-c* gesetzt, dann wurde die Eingabedatei korrekt sortiert.

1 War -c gesetzt, wurde die Eingabedatei nicht wie angegeben sortiert. War sowohl -c als auch -u gesetzt, wurden zwei identische Eingabezeilen mit gleichem Sortierfeld gefunden.

>1 Ein Fehler ist aufgetreten.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *sort*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE bestimmt die gültige Sortierrreihenfolge, nach der *sort* die Eingabe sortiert.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) sowie für die Optionen *-b*, *-d*, *-f* und *-i* die Zeichenklassifizierung.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_NUMERIC bestimmt für die Option *-n* die gültige Darstellung des Dezimalpunkts.

LC_TIME bestimmt für die Option *-M* die aktuell gültigen Monatsnamen und ihre Abkürzung und beeinflusst ihre Sortierreihenfolge.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Der Inhalt von *eingabe_datei* soll nach dem zweiten Feld sortiert werden.

```
$ sort +1 -2 eingabe_datei
```

Beispiel 2 Der Inhalt von *eingabe_datei1* und *eingabe_datei2* soll in umgekehrter Reihenfolge nach dem zweiten Zeichen im zweiten Feld (= 1. Nicht-Leerzeichen bei vorausgesetzter Feldtrennung durch jeweils 1 Leerzeichen) sortiert werden; die Ausgabe soll in *ausgabe_datei* geschrieben werden.

```
$ sort -r -o ausgabe_datei +1.0 -1.2 eingabe_datei1 eingabe_datei2
```

Beispiel 3 Der Inhalt von *eingabe_datei1* und *eingabe_datei2* soll in umgekehrter Reihenfolge nach dem ersten Nicht-Leerzeichen im zweiten Feld sortiert werden.

```
$ sort -r -o ausgabe_datei +1.0b -1.1b eingabe_datei1 eingabe_datei2
```

Beispiel 4 Die bereits sortierte Datei *eingabe_datei* soll ausgegeben werden. Dabei soll von mehreren Zeilen, bei denen das dritte Feld übereinstimmt, jeweils nur die erste Zeile ausgegeben werden.

```
$ sort -u +2 -3 eingabe_datei
```

Siehe auch *comm*, *join*, *uniq*

split Datei auf mehrere Dateien verteilen (split files into pieces)

split teilt Dateien in kleinere Abschnitte auf. Die Abschnitte schreibt *split* in einzelne Ausgabedateien. Die ursprüngliche Datei bleibt erhalten. Die Ausgabedateien werden automatisch „durchnummeriert“; dazu verwendet *split* ein Suffix aus zwei Kleinbuchstaben (aa, ab ... zz) aus der aktuellen internationalen Umgebung. Die letzte Datei enthält den Rest der Eingabedatei und kann weniger Zeilen enthalten als vorgegeben.

Werden mehr Ausgabedateien benötigt als durch die verwendete Suffixlänge möglich, schreibt *split* die letzte Datei nicht (da diese mehr Zeilen enthalten würde, als vorgegeben) und beendet sich mit einem Endestatus >0. Die bereits angelegten Dateien werden nicht gelöscht.

Syntax

Format 1: split␣**-b**␣byte[␣**-a**␣zahl][␣datei[␣name]]

Format 2: split[␣**-l**␣zeilen][␣**-a**␣zahl][␣datei[␣name]]

Format 3: split[␣zeilen][␣**-a**␣zahl][␣datei[␣name]]

Keine Option angegeben
: Die Ausgabedateien heißen `xaa`, `xab` usw. bis `xzz` in lexikographischer Reihenfolge. In diesem Fall legt *split* höchstens 676 Ausgabedateien an.

Format 1 **split**␣**-b**␣byte[␣**-a**␣zahl][␣datei[␣name]]

-a␣zahl
: Das Suffix für die Ausgabedatei besteht aus *zahl* Buchstaben. Z.B. erzeugt *-a*␣*4* die Ausgabedateien `xaaaa`, `xaaab` usw. bis `xzzzz`.
Die maximale Anzahl theoretisch möglicher Dateinamen beträgt also 26^{zahl} für 0<*zahl*<8. Ist *zahl*>7, wird *UINT_MAX* angenommen.

-a nicht angegeben:
Das Suffix besteht aus 2 Buchstaben.

-b␣byte
: *split* teilt die Eingabedatei in Abschnitte der Größe *byte*. *byte* können Sie wie folgt angeben:

n	als Anzahl Byte
nk	als Vielfaches von 1024 Byte
nm	als Vielfaches von 1048576 Byte

datei
: Name der Eingabedatei, die aufgeteilt werden soll.
Wenn Sie für *datei* einen Bindestrich - angeben, liest *split* von der Standard-Eingabe.

datei nicht angegeben:
split liest von der Standard-Eingabe.

name
Name der Ausgabedateien: Die erste Ausgabedatei erhält den Namen *name***aa**, die zweite den Namen *name***ab** usw. bis *name***zz**.
name muss deshalb um zwei Zeichen kürzer sein (bzw. um *zahl* Zeichen bei Angabe von *-a*), als die im jeweiligen Dateisystem zugelassene maximale Dateinamenslänge ({NAME_MAX} Bytes).

Wenn Sie einen Wert für *name* angeben, dann müssen Sie auch einen Wert für *datei* angeben.

Format 2 **split**[␣**-l**␣zeilen][␣**-a**␣zahl][␣datei[␣name]]

Format 3 **split**[␣zeilen][␣**-a**␣zahl][␣datei[␣name]]

-l␣zeilen
split teilt die Eingabedatei in Abschnitte zu je *zeilen* Zeilen auf.
Dies entspricht der alten Option ␣*zeilen*, die weiterhin unterstützt wird.

-l nicht angegeben:
split teilt die Eingabedatei in Abschnitte zu je 1000 Zeilen auf.

-a␣zahl
Das Suffix für die Ausgabedatei besteht aus *zahl* Buchstaben. Z.B. erzeugt *-a*␣*4* die Ausgabedateien xaaaa, xaaab usw. bis xzzzz.
Die maximal Anzahl theoretisch möglicher Dateinamen beträgt also 26^{zahl} für 0<*zahl*<8.
Ist *zahl*>7, wird *UINT_MAX* angenommen.

-a nicht angegeben:
Das Suffix besteht aus 2 Buchstaben.

datei
Name der Eingabedatei, die aufgeteilt werden soll.
Wenn Sie für *datei* einen Bindestrich - angeben, liest *split* von der Standard-Eingabe.

datei nicht angegeben:
split liest von der Standard-Eingabe.

name
Name der Ausgabedateien: Die erste Ausgabedatei erhält den Namen *name***aa**, die zweite den Namen *name***ab** usw. bis *name***zz**.
name muss deshalb um zwei Zeichen kürzer sein (bzw. um *zahl* Zeichen bei Angabe von *-a*), als die im jeweiligen Dateisystem zugelassene maximale Dateinamenslänge ({NAME_MAX} Bytes).

Wenn Sie einen Wert für *name* angeben, dann müssen Sie auch einen Wert für *datei* angeben.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *split*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Der Inhalt der Datei *beispiel* soll zu je 20 Zeilen auf verschiedene Dateien verteilt werden:

```
$ split -l 20 beispiel
$ ls
beispiel
xaa
xab
xac
xad
```

Beispiel 2 Je zwei Zeilen der Standard-Eingabe sind in Dateien namens *aus* zu schreiben. Da Sie den Namen der Ausgabedateien mit *aus* explizit angeben, darf der Bindestrich für Standard-Eingabe nicht fehlen!

```
$ split -l 2 - aus
Was wahr ist, war immer wahr
und wird immer wahr bleiben.
Was aber nicht wahr ist, war nie wirklich
und wird nie wirklich werden.
END

$ ls
ausaa
ausab
```

Siehe auch *csplit*

start_bs2fsd Kopierdämonen starten

Mit *start_bs2fsd* kann der Systemadministrator zusätzliche Kopierdämonen *bs2fsd* starten.

Syntax **start_bs2fsd** [number]

Keine Option angegeben
start_bs2fsd informiert über die Anzahl der aktuell laufenden Kopierdämonen.

number
Gibt an wie viele Kopierdämonen zusätzlich zu den bereits laufenden Kopierdämonen gestartet werden sollen. Es können maximal 8 Kopierdämonen laufen.

Beispiel Prüfung, wie viele Dämonen aktuell laufen

```
# start_bs2fsd
/sbin/start_bs2fsd: 2 bs2fs daemons are running
```

Start eines zusätzlichen Dämonen und erneute Prüfung

```
# start_bs2fsd 1
/sbin/start_bs2fsd: start additional daemon 1 of 1
# start_bs2fsd
/sbin/start_bs2fsd: 3 bs2fs daemons are running
```

strings Druckbare Zeichenketten in Objekt- oder Binärdateien suchen (find printable strings in files)

strings durchsucht Binärdateien nach Zeichenketten und gibt diese auf die Standard-Ausgabe aus. Als Zeichenkette gilt standardmäßig jede Folge von vier oder mehr druckbaren ASCII-Zeichen, die mit einem Neue-Zeile-Zeichen oder mit einem Null-Byte enden (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*). *strings* können Sie z.B. benutzen, um unbekannte Objektdateien zu identifizieren.

Syntax

Format 1: strings[␣**-a**][␣**-o**][␣**-t**␣format][␣**-n**␣anzahl][␣datei...]

Format 2: strings[␣**-**][␣**-o**][␣**-t**␣format][␣**-**anzahl][␣datei...]

-a *strings* sucht in der ganzen Datei nach druckbaren Zeichenketten.

Dies entspricht der alten Option -, die weiterhin unterstützt wird.

-a nicht angegeben:
strings sucht nur im initialisierten Datenbereich von Objektdateien nach druckbaren Zeichenketten.

-n␣anzahl
Als Zeichenkette gilt jede Folge von *anzahl* oder mehr druckbaren Zeichen, die mit einem Neue-Zeile-Zeichen oder einem Null-Byte enden.

Dies entspricht der alten Option *-anzahl*, die weiterhin unterstützt wird.

-n nicht angegeben:
Als Zeichenkette gilt jede Folge von vier oder mehr druckbaren Zeichen, die mit einem Neue-Zeile-Zeichen oder einem Null-Byte enden.

-o Vor jeder Zeichenkette wird ihre Position in der Datei ausgegeben.

-t␣format
Vor jeder Zeichenkette wird ihre Position in der Datei ausgegeben. Das *format* der Positionsangabe legen Sie wie folgt fest:

- d Positionsangabe dezimal
- o Positionsangabe oktal
- x Positionsangabe hexadezimal

datei
Name der Datei, die *strings* nach druckbaren Zeichenketten durchsuchen soll.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *strings*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) und erkennt druckbare Zeichenketten.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Suchen aller druckbaren Zeichenketten in der ausführbaren Binärdatei *a.out*:

```
$ strings a.out
L
LLM-UFS
~                       2009-01-27027 16:24:55
39UD2009-01-27027 16:24:55        39UD2009-01-27027 16:24:55-
2009-01-27027 16:24:552009-01-27027 16:24:55
:
halli hallo
~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~
~~~~
C(BS2000) COMPILER: V03.2B00,COMPILATION DATE: 2009-01-27,REP AREA:
:
```

Dies kann die Ausgabe für eine Datei sein, deren Quellcode wie folgt aussah:

```
#include <stdio.h>

main()
{
printf("halli hallo\n");
}
```

Siehe auch *nm, od*

stty Eigenschaften einer Datensichtstation ausgeben oder ändern (set terminal type)

Mit *stty* können Sie Eigenschaften einer Datensichtstation ändern oder abfragen.

Abhängig von der Implementierung werden einige Optionen von *stty* zwar akzeptiert, vom Treiber jedoch nicht interpretiert, so dass diese Optionen ohne Auswirkung bleiben. Dazu gehören u.a. *parext, iexten, stflush*.

Das Kommando *stty* ist nur sinnvoll, wenn der Zugang zur POSIX-Shell über rlogin erfolgte. Bei BS2000-Blockterminals hat das Kommando *stty* keine Wirkung; es können nur die Eigenschaften von Zeichenterminals geändert werden.

Syntax **stty**[␣option]...

Keine Option angegeben
stty gibt einige Eigenschaften der Datensichtstation aus, welche die Standard-Eingabe von *stty* ist.

Optionen, die den Umfang der Ausgabe festlegen

-a (a - all) *stty* gibt alle momentan gültigen Einstellungen der Datensichtstation aus.

-g *stty* gibt die Einstellungen in einem Format aus, so dass Sie diese als Eingabe für ein weiteres *stty*-Kommando verwenden können.

Optionen zum Einstellen der Ein-/Ausgabe-Eigenschaften

Im Folgenden sind die Optionen zum Einstellen der Ein- und Ausgabe-Eigenschaften der Datensichtstation, die die Standard-Eingabe für *stty* ist, beschrieben. Die Ein-/Ausgabe-Optionen sind in fünf Gruppen zusammengefasst:

- Datenübertragung steuern
- Dateneingabe steuern
- Datenausgabe steuern
- Lokale Merkmale
- Zuweisungen an Steuerzeichen

Zusätzlich gibt es eine Gruppe von Optionen, die durch Kombinationen von Optionen dieser fünf Gruppen entstehen. Das Kommando *stty* überprüft Kombinationen von Optionen nicht auf ihre Plausibilität, auch wenn viele Kombinationen nicht sinnvoll sind.

Erklärungen für die in Klammern angegebenen Optionen sind ebenfalls in Klammern gesetzt.

Datenübertragung steuern

Da am BS2000 keine seriellen Leitungen unterstützt werden, haben die Optionen, die die Datenübertragung steuern, keinen Einfluss. Aus Vollständigkeitsgründen finden Sie aber nachfolgend eine Übersicht der Optionen:

parenb
Paritätsbits erzeugen

parext
erweiterte Erzeugung und Überprüfung des Paritätsbits

parodd
ungerade Zeichenparität

csize
Zeichengröße festlegen: *cs5, cs6, cs7* oder *cs8*

ispeed zahl
Datenübertragungsrate für Eingang auf *zahl* Baud setzen

ospeed zahl
Datenübertagungsrate für Ausgang auf *zahl* Baud setzen

zahl
Datenübertragungsrate auf *zahl* Baud setzen

hupcl
Leitung nach dem letzten Aufruf von *close()* abbauen

hup
wie *hupcl*

cstopb
pro Zeichen zwei Stop-Bits verwenden

cread
Datenempfang freigeben

clocal
Modem-Steuersignale werden nicht unterstützt

Dateneingabe steuern

Folgende Optionen können Sie zur Steuerung der Dateneingabe verwenden:

ignbrk (**-ignbrk**)
(ignbrk - ignore break) Break wird bei der Eingabe ignoriert (nicht ignoriert).

brkint (**-brkint**)
(brkint - interrupt on break) Bei der Eingabe von Break wird das Signal SIGINT gesendet (nicht gesendet).

ignpar (**-ignpar**)
(ignpar - ignore parity errors) Paritätsfehler werden ignoriert (nicht ignoriert).

parmrk (**-parmrk**)
(parmrk - mark parity errors) Zeichen mit Paritätsfehler werden markiert (nicht markiert).

inpck (**-inpck**)

(inpck - input parity check) Die Paritätsprüfung bei der Eingabe wird aktiviert (nicht aktiviert).

istrip (**-istrip**)
Eingabe-Zeichen werden auf 7 bit maskiert (nicht auf 7 bit maskiert).

inlcr (**-inlcr**)
(inlcr - input newline to carriage return) Neue-Zeile-Zeichen (new-line) in der Eingabe werden in Wagenrücklauf-Zeichen (carriage return) verwandelt (nicht in Wagenrücklauf-Zeichen verwandelt).

igncr (**-igncr**)
(igncr - ignore carriage return) Wagenrücklauf-Zeichen (carriage return) in der Eingabe werden ignoriert (nicht ignoriert).

icrnl (**-icrnl**)
(icrnl - input carriage return to newline) Wagenrücklauf-Zeichen (carriage return) in der Eingabe werden in Neue-Zeile-Zeichen (new-line) verwandelt (nicht in Neue-Zeile-Zeichen verwandelt).

iuclc (**-iuclc**)
(iuclc - input upper case to lower case) Großbuchstaben in der Eingabe werden in die entsprechenden Kleinbuchstaben verwandelt (nicht verwandelt).

ixon (**-ixon**)
Die Ausgabe arbeitet mit (ohne) START/STOP Kontrolle. STOP ist das ASCII-Zeichen DC3, START ist das ASCII-Zeichen DC1.

ixany (**-ixany**)
Die Ausgabe wird durch ein beliebiges Zeichen fortgesetzt (wird nur durch das ASCII-Zeichen DC1 START fortgesetzt).

ixoff (**-ixoff**)
Das System sendet ein (kein) START-Zeichen, wenn die Eingabe-Warteschlange fast leer ist und ein (kein) STOP-Zeichen, wenn die Eingabe-Warteschlange fast voll ist.

Datenausgabe steuern

Folgende Optionen können Sie zur Steuerung der Datenausgabe verwenden:

opost (**-opost**)
(opost - postprocess output) Die Ausgabe wird entsprechend der restlichen Optionen nachbearbeitet (nicht nachbearbeitet: in diesem Fall werden alle anderen Optionen zur Steuerung der Daten-Ausgabe ignoriert).

olcuc (**-olcuc**)
(olcuc - output lower case to upper case) Kleinbuchstaben in der Ausgabe werden in die entsprechenden Großbuchstaben umgewandelt (nicht umgewandelt).

ocrnl (**-ocrnl**)
(ocrnl - output carriage return to newline) Bei der Ausgabe werden Wagenrücklauf-Zeichen (carriage return) in Neue-Zeile-Zeichen (new line) umgewandelt (nicht umgewandelt).

onocr (**-onocr**)
In Spalte 0 werden keine Wagenrücklauf-Zeichen (carriage return) ausgegeben (in Spalte 0 werden Wagenrücklauf-Zeichen ausgegeben).

onlret (**-onlret**)
Ein Neue-Zeile-Zeichen (new line) in der Ausgabe führt auch die Funktion eines Wagenrücklauf-Zeichens (carriage return) aus (führt nur die Funktion eines Neue-Zeile-Zeichens aus).

ofill (**-ofill**)
Für Verzögerungen bei der Datenübertragung werden Füllzeichen gesendet (werden keine Füllzeichen gesendet, sondern die Übertragung wird zeitlich verzögert).

ofdel (**-ofdel**)
Als Füllzeichen werden Abbruch-Zeichen DEL 0x7f verwendet
(werden Null-Zeichen NUL 0x0 verwendet).

cr0 cr1 cr2 cr3
(cr - carriage return) Dauer der Verzögerung bei der Ausgabe von Wagenrücklauf-Zeichen wählen.

nl0 nl1
(nl - newline) Dauer der Verzögerung bei der Ausgabe von Neue-Zeile-Zeichen (new line) wählen.

tab0 tab1 tab2 tab3
(tab - tabulator) Dauer der Verzögerung bei der Ausgabe von Tabulatorzeichen wählen.

bs0 bs1
(bs - backspace) Dauer der Verzögerung bei der Ausgabe von Rücksetz-Zeichen (backspace) wählen.

ff0 ff1
(ff - form feed) Dauer der Verzögerung bei der Ausgabe von Seitenvorschub-Zeichen (form feed) wählen.

vt0 vt1
(vt- vertical tabulator) Dauer der Verzögerung bei der Ausgabe von Vertikaltabulator-Zeichen wählen.

Lokale Merkmale

isig (**-isig**)
Jedes Zeichen der Eingabe wird auf die Steuerzeichen intr, quit und swtch (siehe Abschnitt „Zuweisungen an Steuerzeichen“ auf Seite 756) abgeprüft (nicht abgeprüft). Bei Übereinstimmung wird die zum Steuerzeichen gehörende Funktion ausgeführt (nicht ausgeführt).

icanon (**-icanon**)
(icanon - input canonical) Der kanonische Eingabe-Modus wird eingeschaltet (ausgeschaltet), d.h., die Sonderfunktionen der Zeichen erase und kill werden bei der Eingabe dieser Zeichen ausgeführt (werden nicht ausgeführt).

xcase (**-xcase**)
Kanonische Klein-/Großbuchstaben-Darstellung einschalten (nicht einschalten).

echo (**-echo**)
Jedes eingegebene Zeichen wird angezeigt (nicht angezeigt).

echoe (**-echoe**)
(echoe - echo erase) Erase-Zeichen werden als die Zeichenkette Rücksetz-Zeichen - Leerzeichen - Rücksetz-Zeichen (backspace-space-backspace) angezeigt (werden nicht als eine solche Zeichenkette angezeigt).

Wenn die Option *echoe* gesetzt ist, wird auf vielen Datensichtstationen das mit erase überschriebene Zeichen gelöscht. Bei entwerteten Zeichen, Tabulatorzeichen und Rücksetz-Zeichen kann es jedoch zu Problemen kommen, da die jeweils aktuelle Spaltenposition nicht beibehalten wird.

echok (**-echok**)
(echok - echo kill) Nach einem kill-Zeichen wird zusätzlich ein Neue-Zeile-Zeichen ausgegeben (nicht ausgegeben).

echonl (**-echonl**)
(echonl - echo newline) Neue-Zeile-Zeichen ausgeben (nicht ausgeben).

noflsh (**-noflsh**)
(noflsh - no flush) Die Eingabe- und die Ausgabe-Warteschlange werden nicht gelöscht (werden gelöscht), sobald ein intr-, quit- oder swtch-Zeichen erkannt wurde.

tostop (**-tostop**)
Wenn ein Hintergrundprozess auf die Datensichtstation schreibt, wird das Signal SIGTTOU geschickt (nicht geschickt).

iexten (**-iexten**)
Spezielle Kontrollzeichen, die momentan nicht von *icanon, isig, ixon* oder *ixoff* kontrolliert werden, sind eingeschaltet (sind ausgeschaltet).

Zuweisungen an Steuerzeichen

steuerzeichen c
Das Zeichen *c* wird als Steuerzeichen *steuerzeichen* interpretiert. *steuerzeichen* kann dabei sein:
ctab, discard, dsusp, eof, eol, eol2, erase, intr, kill, lnext, quit, reprint, start, stop, susp, swtch, werase, min oder *time* (*min* und *time* werden bei *-icanon* benutzt, ctab bei *-stappl*). Wenn dem Zeichen *c* das Zeichen ^ vorangestellt ist, so wird es als CTRL-Zeichen interpretiert (z.B. entspricht ^D einem CTRL D). ^? wird als DEL interpretiert, ^- als nicht definiertes Zeichen.

min zahl/**time** zahl
min bzw. *time* wird der Wert *zahl* zugewiesen. *min* und *time* werden bei *-icanon* benutzt.

line i
Das Leitungsprotokoll *i* (0 < *i* < 127) wird verwendet.

Kombinierte Modi

Folgende kombinierte Modi zur Steuerung der Datensichtstation sind möglich:

evenp oder **parity**
parenb und *cs7* werden gesetzt.

oddp
parenb, *cs7* und *parodd* werden gesetzt.

-parity oder **-evenp**
-parenb und *cs8* werden gesetztz .

-oddp
-parenb, *-parodd* und *cs8* werden gesetzt.

raw (**-raw** oder **cooked**)
Raw-Modus einschalten (ausschalten). Raw-Modus heißt: *cs8* ist gesetzt, die Steuerzeichen erase, kill, intr, quit, swtch und eof haben keine Bedeutung, es findet keine Nachbearbeitung der Ausgabe statt und es wird kein Paritätsbit gesetzt.

nl (**-nl**)
Setzen von *-icrnl* (*icrnl*) und *-onlcr* (*ocrnl*). Zusätzlich wird *-inlcr*, *-igncr*, *-ocrnl* und *-onlret*) gesetzt.

lcase (**-lcase**)
Setzen von *xcase*, *iuclc* und *olcuc* (*-xcase*, *-iuclc* und *-olcuc*).

LCASE (**-LCASE**)
Wie *lcase* (*-lcase*).

tabs (**-tabs** oder **tab8**)
Tabulatorzeichen werden unverändert ausgegeben (in Leerzeichen expandiert).

ek Die Steuerzeichen erase und kill werden auf die voreingestellten Werte zurückgesetzt. Die Voreinstellungen sind systemabhängig.

sane
Alle Eigenschaften der Datensichtstation werden auf sinnvolle Werte gesetzt.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *stty*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) sowie die druckbaren Zeichen.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Aktuelle Einstellung abfragen (bei Zugang zur POSIX-Shell über rlogin):

```
$ stty -a
speed 38400 baud;
intr = DEL; quit = ^\; erase = ^h; kill = ^x;
eof = ^d; eol = <undef>; eol2 = <undef>; swtch = <undef>;
start = ^q; stop = ^s; susp = ^z; dsusp = <undef>;
rprnt = ^r; flush = ^o; werase = ^w; lnext = ^v;
-parenb -parodd cs8 -cstopb hupcl cread -clocal -loblk -parext
-ignbrk brkint ignpar -parmrk -inpck -istrip -inlcr -igncr icrnl -iuclc
ixon -ixany -ixoff -imaxbel
isig icanon -xcase echo echoe echok -echonl -noflsh
-tostop -echoctl -echoprt -echoke -defecho -flusho -pendin -iexten
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel -tabs
```

Beispiel 2 Falls Sie möglicherweise unbeabsichtigt Aktionen durchgeführt haben, die bewirkten, dass am Bildschirm keine Eingabe mehr sichtbar ist, können Sie die Eingabe folgendermaßen wieder sichtbar machen:

```
$ stty echo
```

Diese Eingabe ist am Bildschirm allerdings nicht sichtbar.

Beispiel 3 Rücksetzen der Eigenschaften der Datensichtstation auf sinnvolle Werte:

```
$ stty sane
```

Beispiel 4 Sie können die Eigenschaften Ihrer Datensichtstation auch mit Hilfe der Option *-g* zurücksetzen:

```
$ STATUS=`stty -g `
```

Damit definieren Sie eine Variable, die die aktuell gültigen Einstellungen der Datensichtstation enthält (und die möglicherweise anders als die Einstellungen von *stty sane* sind). Nun können Sie die Einstellungen Ihrer Datensichtstation verändern. Um die Einstellungen wieder zurückzusetzen, geben sie ein:

```
$ stty $STATUS
```

Siehe auch *ioctl()* [4]

su Benutzerkennung wechseln (substitute user ID)

su startet eine neue Shell unter einer anderen Benutzerkennung.

Syntax **su** [**-**] [user]

- Die neue Shell wird als Login-Shell gestartet, d.h. die Skripts */etc/profile* und *$HOME/.profile* werden ausgeführt.

user Name der Benutzerkennung für die neue Shell; Standardwert: TSOS.

Hinweis *su* kann nur dann erfolgreich ausgeführt werden, wenn für die angegebene Benutzerkennung eine Standard-Abrechnungsnummer für rlogin-Tasks festgelegt ist (siehe Operand POSIX-RLOGIN-DEFAULT im Kommando MODIFY-USER-ATTRIBUTES).

Datei */var/adm/sulog*
Datei zur Protokollierung aller erfolgreichen und fehlgeschlagenen Aufrufe.

Beispiel 1 Erfolgreicher Aufruf von *su*

```
$ su
Password (TSOS):
# id
uid=0(TSOS) gid=0(SYSROOT) groups=0(SYSROOT)
# exit
$
```

Eintrag in der sulog-Datei:

```
   SU 2011/05/10 13:56:26 + pts/0      FROEDE-TSOS
```

Beispiel 2 Abgewiesener Aufruf von *su*

```
$ su - sysaudit
Password (SYSAUDIT):
Sorry
$ id
uid=109(FROEDE) gid=2001(OS315) groups=2001(OS315),7777(dialout)
$
```

Eintrag in der sulog-Datei:

```
   SU 2011/05/10 13:56:23 - pts/0      FROEDE-SYSAUDIT
```

sum Prüfsumme einer Datei berechnen (print checksum and block count of a file)

sum berechnet für eine Datei eine Prüfsumme und gibt diese auf die Standard-Ausgabe aus. Außerdem gibt *sum* aus, wieviel Speicherplatz (Einheit: 512 byte) die Datei belegt. *sum* können Sie z.B. verwenden, um nach einer Dateiübertragung zu prüfen, ob die Datei unverändert und vollständig angekommen ist.

Syntax **sum**[␣**-r**][␣datei]...

-r *sum* verwendet zur Berechnung der Prüfsumme einen anderen Algorithmus und gibt auch eine andere Prüfsumme aus.

datei

Name der Datei, für die die Prüfsumme berechnet werden soll. Sie können einfache Dateien oder Dateiverzeichnisse angeben. Pro Aufruf sind mehrere Angaben möglich.

datei nicht angegeben:
sum liest von der Standard-Eingabe.

Achtung!
Die Algorithmen zur Berechnung der Prüfsumme sind unter Umständen nicht portabel, d.h. auf unterschiedlichen Systemen gibt *sum* für die gleiche Datei möglicherweise unterschiedliche Prüfsummen aus.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *sum*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Prüfsumme der Datei *monate* nach dem ersten Algorithmus:

```
$ sum monate
6658 1 monate
```

Beispiel 2 Prüfsumme der Datei *monate* nach dem zweiten Algorithmus:

```
$ sum -r monate
62412     1 monate
```

Siehe auch *cksum, wc*

sync Systempuffer zurückschreiben (flush system buffers)

sync bewirkt, dass alle bisher noch nicht auf Festplatte geschriebenen Systempuffer herausgeschrieben werden. Damit werden alle Dateiänderungen gesichert.

Ein Systempuffer ist ein Puffer, der dem Benutzer nicht zugänglich ist und nur der Leistungssteigerung dient.

Syntax **sync**

Anwendung

Durch *sync* werden nur lokale Puffer auf lokale Festplatten geschrieben. Die Pufferung auf fernen Rechnern kann durch *sync* nicht beeinflusst werden.

tabs Tabulatorstops setzen (set terminal tabs)

Dieses Kommando kann nur von Benutzern verwendet werden, die sich über *rlogin* Zugang zur POSIX-Shell verschafft haben.

tabs setzt Tabulatorstops auf der Datensichtstation. Vorangegangene Einstellungen werden vorher gelöscht. Für das Kommando *tabs* muss die Datensichtstation über Hardware-Tabulatoren verfügen, die extern durch Escape-Sequenzen eingestellt werden können.
Die niedrigste Spaltennummer ist 1. Spalte 1 bedeutet bei *tabs* immer die erste Spalte ganz links auf dem Bildschirm. Dies gilt auch für Geräte, deren Spaltenmarkierer mit 0 beginnen.

Syntax **tabs**[␣**-T**␣typ][␣**+m**n][␣tabspec]

-T␣typ
Datensichtstations-Typ, der zur Randeinstellung bekannt sein muss. *typ* ist ein in */usr/share/lib/terminfo* enthaltener Name (siehe *term* [13]).

-T␣typ nicht angegeben:
tabs verwendet den Wert der Umgebungsvariablen *TERM*. Ist *TERM* in der Umgebung (siehe *environ* [13]) nicht definiert, verwendet *tabs* einen Typ, dessen Eigenschaften für viele Datensichtstationen zutreffen.

+mn
Setzt den linken Rand neu. Alle Tabulatoren werden um *n* Stellen nach rechts verschoben, die Spalte *n+1* wird zum linken Rand. Wird für *n* kein Wert angegeben, wird der Wert 10 angenommen. Bei TermiNet sollte der erste angegebene Wert in der Tabulatorliste 1 sein, sonst wird der Rand noch weiter nach rechts verlagert. Normalerweise wird der äußerste linke Rand mit *+m0* gesetzt. Der Rand wird bei den meisten Datensichtstationen nur nach rechts verschoben, wenn *+m* explizit angegeben ist.

tabspec
Für die Festlegung von Tabulatoren sind vier verschiedene Tabulatorangaben zulässig:

-code	in bestimmtem Format festgelegte Tabulatorpositionen
-n	Tabulatorpositionen in regelmäßigen Abständen
--datei	Tabulatorpositionen wie in *datei* festgelegt
n1[,n2...]	frei wählbare Tabulatorpositionen

-code Sie bestimmen die Folge von Tabulatoren in einer Zeile mit einer der folgenden Angaben:

-a Tabulatorstops in den Spalten 1,10,16,36,72.
Dies entspricht dem ersten Format des Assembler, IBM S/370.

-a2 Tabulatorstops in den Spalten 1,10,16,40,72.
Dies entspricht dem zweiten Format des Assembler, IBM S/370.

-c Tabulatorstops in den Spalten 1,8,12,16,20,55.
Dies entspricht dem üblichen COBOL-Format.

-c2 Tabulatorstops in den Spalten 1,6,10,14,49.
Dies entspricht dem kompakten COBOL-Format. Dabei werden die Spalten 1-6 ausgelassen. Das erste eingegebene Zeichen entspricht also der Bildschirmspalte 7. Durch Eingeben eines Leerzeichens wird auf Spalte 8 positioniert, mit einem Tabulatorzeichen auf Spalte 12. Dateien mit dieser Tabulatoreinstellung sollten folgende Formatangabe enthalten (siehe *fspec* [13]):
`<:t -c2 m6 s66 d:>`

-c3 Tabulatorstop in den Spalten 1,6,10,14,18,22,26,30,34,38,42,46,50, 54,58,62,67.
Dies entspricht dem kompakten COBOL-Format mit zusätzlichen Tabulatoren. Dabei werden die Spalten 1-6 ausgelassen. Das erste eingegebene Zeichen entspricht also der Bildschirmspalte 7. Durch Eingeben eines Leerzeichens wird auf Spalte 8 positioniert, mit einem Tabulatorzeichen auf Spalte 12. Dateien mit dieser Tabulatoreinstellung sollten folgende Formatangabe enthalten (siehe *fspec* [13]):
`<:t -c3 m6 s66 d:>`.
Dieses Format wird für COBOL empfohlen.

-f Tabulatorstops in den Spalten 1,7,11,15,19,23.
Dies entspricht dem FORTRAN-Format.

-p Tabulatorstops in den Spalten 1,5,9,13,17,21,25,29,33,37,41,45,49, 53,57,61.
Dies entspricht dem PL/I-Format.

-s Tabulatorstops in den Spalten 1,10,55.
Dies entspricht dem SNOBOL-Format.

-u Tabulatorstops in den Spalten 1,12,20,44.
Dies entspricht dem UNIVAC 1100 Assembler-Format.

-n Die angegebene Zahl bewirkt, dass Tabulatorstops in den Spalten 1+*n*,1+2*n*,1+3*n*... gesetzt werden. Bei Eingabe einer 8 erhalten Sie die Standard-Tabulatoreinstellung des UNIX-Systems. Bei Eingabe einer Null werden alle Tabulatorstops gelöscht.

--datei *tabs* liest die erste Zeile der Datei und sucht nach einer Formatangabe (siehe *fspec* [13]). Ist eine solche vorhanden, setzt *tabs* die Tabulatorstops dieser Angabe entsprechend. Ist keine Angabe vorhanden, wird die Standardeinstellung *-8* vorgenommen. Dieses Format ist sinnvoll, wenn sicher-

gestellt werden soll, dass eine mit Tabulatoren versehene Datei mit den richtigen Tabulatoreinstellungen ausgegeben wird und wenn *tabs* in Zusammenhang mit *pr* benutzt werden soll (siehe *pr*).

n1[,n2,...] Sie können die Spalten für die Tabulatorstops beliebig wählen, indem Sie eine Liste von bis zu 64 Zahlen in aufsteigender Reihenfolge angeben. Geht einer Zahl ein Pluszeichen voraus, so wird sie zum Wert der vorhergehenden Zahl dazugezählt. Dies gilt nicht für die erste Zahl. Das Format *1,10,20,30* können Sie z.B. auch durch *1,10,+10,+10* angeben.

Die Zahlen werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen, wobei die Zahlen durch Komma und/oder Leerzeichen getrennt werden können. *n1[,n2,...]* muss als letztes Argument in der Kommandozeile angegeben werden.

tabspec nicht angegeben:
tabs nimmt die Einstellung *-8* vor, die den Standard-Tabulatoren des UNIX-Systems entspricht.

Hinweis Das Löschen von Tabulatorstops und die Einstellung des linken Randes ist bei verschiedenen Datensichtstationen nicht einheitlich.
tabs kann nur 20 Tabulatorstops löschen, aber 64 setzen.

Fehler `illegal tabs`
Beim Wählen beliebiger Tabulatorstops wurde die aufsteigende Reihenfolge nicht eingehalten oder es wurde beim Wählen beliebiger Tabulatorstops eine Null angegeben.

`illegal increment`
Beim Wählen beliebiger Tabulatorstops wurde eine Null angegeben oder es fehlt ein Inkrement.

`unknown tab code`
Der für *tabspec* angegebene Code kann nicht gefunden werden.

`can't open`
Die für *tabspec* angegebene Datei kann nicht geöffnet werden.

`file indirection`
Die Formatangabe der für *tabspec* angegebenen Datei zeigt noch auf eine andere Datei. Verweise dieser Art sind nicht zugelassen.

`tabs cannot be set for this terminal`
Das Kommando wurde für einen unbekannten Terminaltyp, z.B. für ein Blockterminal, aufgerufen.

Datei */usr/share/lib/terminfo/?/**
Dateien, die der Benennung und Beschreibung von Terminals dienen.

Variable *TERM*
Typ der Datensichtstation

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *tabs*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Sie wollen Tabulatorstops gemäß dem üblichen COBOL-Format setzen.

```
$ tabs -c
```

Beispiel 2 Sie wollen in jeder 6. Spalte einen Tabulator-Stop setzen, d.h. in den Spalten 1,7,13,19,...

```
$ tabs -6
```

Beispiel 3 Sie wollen in den Spalten 1, 8 und 36 einen Tabulator-Stop setzen.

```
$ tabs 1,8,36
```

Die gleiche Wirkung erzielen Sie mit

```
$ tabs 1,+7,+28
```

Beispiel 4 Sie wollen Tabulatorstops gemäß der ersten Zeile (Formatangabe) Ihrer Datei *$HOME/tabspec.list/dat1* setzen.

```
$ tabs --$HOME/tabspec.list/dat1
```

Siehe auch *pr*, *tput*
environ, fspec, *terminfo*, *term* [13]

tail Den letzten Teil einer Datei ausgeben (deliver the last part of a file)

tail gibt den Inhalt der angegebenen Datei ab einer festgelegten Stelle auf die Standard-Ausgabe aus. Wenn Sie beim Aufruf von *tail* keine Datei angeben, wird der Inhalt der Standard-Eingabe ab einer festgelegten Stelle auf die Standard-Ausgabe ausgegeben. Wird *tail* auf zeichenorientierte Gerätedateien angewendet, so kann es zu Problemen kommen.

Syntax

Format 1: tail[␣**-f**][␣**-c**␣stelle | ␣**-n**␣stelle][␣datei]

Format 2: tail[␣**-f**][␣**+**[stelle]|␣**-**[stelle][**l**|**b**|**c**]][␣datei]

Format 1 **tail**[␣**-f**][␣**-c**␣stelle | ␣**-n**␣stelle][␣datei]

-f (f - follow) Wenn die Eingabedatei keine Pipe ist, dann beendet sich das Programm nach der Ausgabe nicht, sondern tritt in eine Endlosschleife ein. Es schläft jeweils mindestens eine Sekunde lang und versucht, dann weitere Datensätze der Eingabedatei zu lesen und auszugeben. Auf diese Weise können Sie z.B. das Wachstum einer Datei überwachen, in die von einem anderen Prozess geschrieben wird.
-f wird ignoriert, wenn keine *datei* angegeben und die Standardeingabe eine Pipe ist.

-c␣stelle
(c - character) Die Ausgabe von *tail* beginnt an dem durch *stelle* festgelegten Zeichen. Mit *stelle* legen Sie die Stelle in der Eingabedatei fest, an der die Ausgabe beginnt. Sie können angeben:

[[**+**|**-**]zahl] Für *zahl* geben Sie eine ganze Dezimalzahl an. Wenn Sie *+zahl* angeben, wird vom Dateianfang aus gezählt. Wenn Sie *-zahl* angeben, wird vom Dateiende aus gezählt.

Achtung!
Abschnitte, die vom Dateiende aus gezählt werden, werden von *tail* in einem Puffer zwischengespeichert. Die Größe des Puffers ist auf 4 Kbyte beschränkt.

+|- nicht angegeben:
tail zählt vom Dateiende aus.

zahl nicht angegeben:
Standard-Wert ist 10.

stelle nicht angegeben:
Standard-Wert ist -10, d.h. 10 Einheiten vom Dateiende aus gezählt.

-n␣stelle
(n - number) Die Option ist äquivalent zu *-c stelle*, mit der Ausnahme, dass die Ausgabe zeilenweise gezählt wird und nicht byteweise.

datei
Name der Eingabedatei, die *tail* ausgeben soll.

datei nicht angegeben:
tail liest von der Standard-Eingabe.

Format 2 **tail**[␣**-f**][␣**+**[stelle]|␣**-**[stelle][**l**|**b**|**c**]][␣datei]

-f (f - follow) Wenn die Eingabedatei keine Pipe ist, dann beendet sich das Programm nach der Ausgabe nicht, sondern tritt in eine Endlosschleife ein. Es schläft jeweils mindestens eine Sekunde lang und versucht, dann weitere Datensätze der Eingabedatei zu lesen und auszugeben. Auf diese Weise können Sie z.B. das Wachstum einer Datei überwachen, in die von einem anderen Prozess geschrieben wird.

stelle
Mit *stelle* legen Sie die Stelle in der Eingabedatei fest, an der die Ausgabe beginnt. Für *stelle* geben Sie eine ganze Dezimalzahl an. Wenn Sie +[*stelle*] angeben, wird vom Dateianfang aus gezählt. Wenn Sie -[*stelle*] angeben, wird vom Dateiende aus gezählt.

Achtung!
Abschnitte, die vom Dateiende aus gezählt werden, werden von *tail* in einem Puffer zwischengespeichert. Die Größe des Puffers ist auf 4 Kbyte beschränkt.

stelle nicht angegeben:
Standard-Wert ist 10, d.h. 10 Einheiten ab Dateianfang oder vom Dateiende aus gezählt.

Zwischen +[*stelle*] bzw. -[*stelle*] und der Einheit (**l**,**b** oder **c**) darf **kein** Leerzeichen stehen.

l (l - line) Die Ausgabe von *tail* beginnt an der durch *stelle* festgelegten Zeile.

b (b - block) Die Ausgabe von *tail* beginnt an dem durch *stelle* festgelegten Block. Ein Block ist eine Einheit von 512 byte.

c (c - character) Die Ausgabe von *tail* beginnt an dem durch *stelle* festgelegten Zeichen.

datei
Name der Eingabedatei, die *tail* ausgeben soll.

datei nicht angegeben:
tail liest von der Standard-Eingabe.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *tail*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 In der Datei *monate* steht jeweils in einer Zeile ein Monatsname. Die beiden *tail*-Aufrufe

```
$ tail -n 5 monate

$ tail -n +8 monate
```

haben die gleiche Ausgabe zur Folge:

```
August
September
Oktober
November
Dezember
```

Beispiel 2 Um die Wirkung der Option *-f* zu verdeutlichen, können Sie z.B. eine Shell-Prozedur im Hintergrund ablaufen lassen, die in einer Endlosschleife in eine Datei schreibt.

Die Shell-Prozedur steht in der Datei *unendlich* und hat folgenden Inhalt:

```
while true
do {
   date >>anna
   sleep 5
   }
done
```

Sie lassen nun diese Prozedur im Hintergrund ablaufen und rufen dann *tail* mit der Option *-f* auf die Datei *anna* angewendet auf:

```
$ unendlich&
656
$ tail –n –5 –f anna
Diese fuenf
Zeilen
standen vorher
schon in
der Datei
FRI JAN 23 11:40:35 MEZ 2009
FRI JAN 23 11:40:40 MEZ 2009
.
.
.
```

tail gibt zunächst die letzten fünf Zeilen der Datei *anna* aus und dann das, was die Prozedur *unendlich* in diese Datei schreibt. Sie können den Prozess, der durch den *tail -f*-Aufruf erzeugt wurde, bei Zeichenterminals mit der Taste DEL (oder bei Blockterminals mit der Tastenfolge @@c) abbrechen.

Siehe auch *cat*, *head*, *more*

talk Dialog mit anderem Benutzer führen (talk to another user)

Dieses Kommando kann nur von Benutzern verwendet werden, die sich über *rlogin* Zugang zur POSIX-Shell verschafft haben.

Mit *talk* kommunizieren Sie mit einem anderen Benutzer, der an einer Datensichtstation am gleichen oder an einem anderen Rechner arbeitet. *talk* sendet Eingabezeilen von Ihrer Datensichtstation zu der Ihres Kommunikationspartners.

Syntax

talk␣benutzerkennung[␣tty-name]

benutzerkennung
: Benutzerkennung des Benutzers, mit dem Sie kommunizieren möchten. Wenn dieser Benutzer gleichzeitig mehrfach am System angemeldet ist, können Sie mit dem Argument *tty-name* die Datensichtstation auswählen, an der Sie ihn ansprechen möchten.

  Wenn der Benutzer an einem anderen Rechner arbeitet geben Sie an:

  benutzerkennung@rechnername

tty-name
: Name der Datensichtstation, an der der Benutzer arbeitet, mit dem Sie kommunizieren wollen. Sie brauchen *tty-name* nur angeben, wenn der gewünschte Kommunikationspartner *benutzerkennung* mehrfach am System angemeldet ist. Von welchen Datensichtstationen aus sich *benutzerkennung* am System angemeldet hat, können Sie mit dem Kommando *who* abfragen.

Arbeitsweise

Bei der Kommunikation mit *talk* gibt es einen Sender und einen Empfänger. Als Sender wird derjenige bezeichnet, der zuerst *talk* aufruft. Der Benutzer mit der dabei angegebenen Benutzerkennung ist der Empfänger.

Wenn der Sender *talk* aufruft, erscheint am Bildschirm des Empfängers:

```
Message from TalkDaemon@empfänger_rechneratempfangszeitpunkt talk: connection
requested by sender@sender_rechner talk: respond with: talk
sender@sender_rechner
```

und am Bildschirm des Senders:

```
Waiting for your party to respond
```

vorausgesetzt, die Benutzerkennung des Empfängers ist definiert, er ist aktuell am System angemeldet und die Verbindung konnte erfolgreich hergestellt werden.

Wenn der Empfänger das Kommunikationsangebot annehmen will, muss er an dieser Stelle eingeben:

```
talk sender@sender_rechner
```

Anschließend erscheinen auf den Bildschirmen von Sender und Empfänger jeweils zwei Fenster, das obere zum Schreiben, das untere zum Lesen von Nachrichten. Beide Kommunikationspartner können gleichzeitig Nachrichten schreiben und lesen.

Wenn der Empfänger keine Kommunikation aufnehmen will, muss er die Taste DEL drücken. Anschließend erscheint das Bereitzeichen der Shell und er kann normal weiterarbeiten.

Während die Kommunikationsverbindung besteht, werden Eingaben wie folgt behandelt:

– druckbare Zeichen werden an den Kommunikationspartner weitergeleitet

– das Klingelzeichen wird an den Kommunikationspartner weitergeleitet

– mit CTRL L können Sie den Bildschirm neu aufbauen

– beenden können Sie die Kommunikation mit der Taste DEL. Es wird dann die Meldung ausgegeben

  ```
  Connection closing. Exiting
  ```

  und am unteren Bildschirmrand erscheint das Bereitzeichen der Shell und dahinter die Schreibmarke.

Mit dem Kommando *mesg* können Sie anderen Benutzern das Recht erteilen (*mesg y*) oder verweigern (*mesg n*), mit *talk* eine Kommunikationsverbindung zu Ihnen aufzubauen. Standardmäßig steht *mesg* auf *y*. Einige Kommandos, z.B. *pr*, lassen während ihres Laufs keine Nachrichten zu, um einem ungeordneten Bildschirmaufbau zu verhindern.

Fehler

`No connection yet`
Die Verbindung zum Empfänger konnte noch nicht hergestellt werden. Am besten warten Sie ein wenig und versuchen es nochmal, wenn nach einigen Sekunden nicht die Meldung erscheint *Waiting for your party to respond*.

`Your party is not logged on`
Sie haben als Empfänger eine Benutzerkennung angegeben, die nicht definiert ist oder die aktuell nicht am System angemeldet ist.

`Your party is refusing messages`
Die Benutzerkennung, die Sie als Empfänger angegeben haben, verweigert das Recht, an ihren Bildschirm Nachrichten zu senden (siehe *mesg*).

Datei *`/etc/hosts`*
Diese Datei wird benötigt, um den Rechner des Empfängers zu finden.

`/var/adm/utmp`
Diese Datei wird benötigt, um die Datensichtstation des Empfängers zu finden. In ihr sind alle angemeldeten Benutzer registriert.

Variable *TERM*
Typ der Datensichtstation, die der Aufrufer benutzt

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *talk*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Verbindungsaufnahme zum Benutzer *werner*, der am gleichen Rechner arbeitet:

```
$ talk werner
  ...
```

Beispiel 2 Verbindungsaufnahme zum Benutzer *heinrich*, der am Rechner *mephisto* an der Datensichtstation *tty003* arbeitet:

```
$ talk heinrich@mephisto tty003
   ...
```

Siehe auch *mailx, mesg, pr, who, write*

tar Archivieren von Dateien (file archiver)

Das Kommando *tar* bearbeitet Archivdateien. Archivdateien können nur im POSIX-Dateisystem - nicht auf Magnetbändern und -kassetten - abgelegt werden.

Die Aktionen werden durch den Operanden *hauptoption* beschrieben.

Das Kommando *tar* wurde aus dem XPG4-Standard zurückgezogen. Es wird durch das Kommando *pax* abgelöst.

In Zukunft sollte nur noch das Kommando *pax* verwendet werden, da *tar* nur noch aus Kompatibilitätsgründen unterstützt wird.

Syntax **tar**␣hauptoption[zusatzoption...][␣datei]...

Hauptoptionen

hauptoption besteht aus einem Funktionszeichen, das mit oder ohne Bindestrich angegeben werden kann.
Unmittelbar anschließend können ein oder mehrere Zusatzoptionen (siehe Seite 775) angegeben werden.

hauptoption kann eines der folgenden Zeichen sein:

[-]c (c- create) Ein neues Archiv wird generiert. Die angegeben *datei* oder Dateien werden an den Beginn des Archivs geschrieben.

[-]r (r - replace) die angegebene *datei* oder Dateien werden an das Ende des Archivs angefügt.

[-]u (u - update) die angegebene *datei* oder Dateien werden an das Ende des Archivs angefügt, wenn sie noch nicht im Archiv vorhanden sind, die Zeit der letzten Änderung im Archiv früher liegt als die Zeit der letzten Änderung der zu archivierenden Datei.

[-]t (t - table) *tar* gibt ein Inhaltverzeichnis des Archivs aus.

Die Namen der angegebenen *datei* oder Dateien werden auf die Standard-Ausgabe geschrieben, falls die Datei im Archiv vorhanden ist.

Sind keine Dateien angegeben, werden die Namen aller im Archiv befindlichen Dateien auf die Standard-Ausgabe ausgegeben.

[**-**]**x** (x - extract) Die angegebene *datei* oder Dateien werden aus dem Archiv gelesen.

Steht die Datei mit relativen Pfadnamen im Archiv, so wird sie in das aktuelle Dateiverzeichnis kopiert.

Steht die Datei mit absoluten Pfadnamen im Archiv, wird sie in das entsprechende Dateiverzeichnis kopiert, falls dieses vorhanden ist.

Ist die angegebene Datei ein Dateiverzeichnis, so werden alle darin enthaltenen Unterverzeichnisse (rekursiv) kopiert.

Existiert die angegebene Datei noch nicht in dem Dateiverzeichnis, in das sie kopiert werden soll, so wird die Datei mit folgenden Attributen angelegt:
- Benutzernummer des Eigentümers, Gruppennummer und Zeit der letzten Änderung werden von der archivierten Datei übernommen.
- Die Zugriffsrechte werden so gesetzt, wie sie mit *umask* definiert sind. Das s-Bit wird nur berücksichtigt, wenn *tar* von einem Benutzer mit Systemverwalter-Rechten aufgerufen wird.

Existiert die Datei bereits, so werden die Zugriffsrechte nicht verändert.

Befinden sich mehrere Dateien mit gleichem Namen im Archiv, so überschreibt die letzte die vorangegangenen.

Zusatzoptionen

Die folgenden Zeichen können als *zusatzoption* unmittelbar an die *hauptoption* angefügt werden. Dabei ist folgendes zu beachten:
- Geben Sie zuerst alle Zusatzoptionen ohne Leerzeichen ein.
- Geben Sie dann die Argumente getrennt durch Leerzeichen an. Die Reihenfolge der Argumente wird bestimmt durch die Reihenfolge, in der die zugehörigen Zusatzoptionen eingegeben wurden.

+v (v - verbose) Der Name jeder bearbeiteten Datei wird auf die Standard-Fehlerausgabe geschrieben. Dem Namen wird ein Zeichen vorangestellt, der die Art der Bearbeitung anzeigt:

- a für die Hauptoptionen c, r, und u, bzw.
- x für die Hauptoptionen x.

Anschließend an den Dateinamen wird die Größe der Datei in Blöcken ausgegeben.

Wird die *zusatzoption* **v** bei der *hauptoption* [**-**]**t** verwendet, so werden die Zugriffsrechte sowie Eigentümer und Gruppe der Datei ausgegeben.

w (w -what) *tar* erwartet für jede Datei eine Bestätigung, bevor die der angegebenen *hauptoption* entsprechende Bearbeitung durchgeführt wird. Sie erhalten die folgende Ausgabe:

`hauptoption dateiname:`

tar erwartet nach dem Doppelpunkt Ihre Eingabe. Nur wenn Sie bestätigen (sprachabhängig mit `j` oder `y`), wird die Aktion durchgeführt.

Ist **w** nicht angegeben, so werden die Bearbeitungen ohne Rückfrage durchgeführt.

f (f - file) Der erste angegebene *datei*-Operand (bzw. der zweite, falls **b** bereits angegeben ist) wird als Name des zu bearbeitenden Archivs interpretiert.

Ist der angegebene Name '**-**' (Bindestrich), so wird das Archiv auf die Standard-Ausgabe geschrieben bzw. von der Standard-Eingabe gelesen. Dadurch kann *tar* auf beiden Enden einer Kommando-Pipe verwendet werden:

- Falls **f** mit den Hauptoptionen **-c**, **-r** oder **-u** angegeben ist und der zugehörige *datei*-Operand '**-**' ist, so wird das Archiv auf die Standard-Ausgabe geschrieben.
- Falls **f** im Zusammenhang mit der *hauptoption* **-t** oder **-x** angebenen ist und der zugehörige *datei*-Operand '**-**' (Bindestrich) ist, so wird das Archiv von der Standard-Eingabe gelesen.

Ist **f** nicht angegeben, so wird die Standard-Archivdatei verwendet, die mit der Shell-Variablen *TAPE* festgelegt ist.

h *tar* verfolgt symbolische Verweise.

Ist **h** nicht angegeben, so bearbeitet *tar* nur den symbolischen Verweis, nicht aber die zugehörige Datei.

l (l - link) *tar* meldet, wenn ein Verweis auf andere Dateien nicht aufgelöst werden kann.

Ist **l** nicht angegeben, so wird keine Meldung ausgegeben.

m (m - modification date/time) *tar* setzt beim Kopieren der angegebenen Datei aus dem Archiv die Zeit der letzten Änderung für die Kopie auf das aktuelle Datum und Uhrzeit.

Ist **m** nicht angegeben, so wird das im Archiv gespeicherte Datum und Uhrzeit übernommen.

o (o - owner) Die aus dem Archiv kopierte Datei erhält als Eigentümer den Benutzer, der *tar* aufgerufen hat.
Die Gruppe dieses Benutzers wird der Datei ebenfalls zugewiesen.

Ist **o** nicht angegeben, so werden die im Archiv für die Datei gespeicherten Eigentümer und Gruppe übernommen.

datei

Pfadname einer Datei oder eines Dateiverzeichnisses, die/das in ein Archiv geschrieben werden soll (*hauptoption* **-c**,**-r** oder **-u**), aus einem Archiv gelesen werden soll (*hauptoption* **-x**) oder aufgelistet werden soll (*hauptoption* **-t**).

Ist *datei* der Name eines Dateiverzeichnisses, so werden die entsprechenden Aktionen auf alle Dateien und Unterverzeichnisse (rekursiv) dieses Verzeichnisses angewandt.

Ist die *zusatzoption* **f** angebenen, so wird der erste *datei*-Operand als Archivname interpretiert.

Variable *TAPE*

bestimmt den Namen des Archivs, der verwendet wird, falls die *zusatzoption* **f** nicht angegeben ist.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *tar*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_COLLATE Beeinflusst die Sortierreihenfolge

LC_MESSAGES

Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Archivdatei erzeugen:
Alle im aktuellen Dateiverzeichnis (.) und seinen Unterverzeichnissen enthaltenen Dateien sollen in die Archivdatei *$HOME/archive/example1.TAR* geschrieben werden.

```
$ tar cvf $HOME/archive/example1.TAR .
a ./.sh_history 8 Blocks
a ./.profile 1 Blocks
a ./test.proc 1 Blocks
a ./pos_examp/file1 1 Blocks
a ./pos_examp/file2 1 Blocks
a ./pos_examp/file3 8 Blocks
tar: ./example1.TAR : same as archive file
```

Beispiel 2 Inhaltsverzeichnis einer Archivdatei auf Standard-Ausgabe ausgeben:

```
$ tar tvf $HOME/archive/example1.TAR
Tar: blocksize = 20
USTAR format archive
rw------- 632/-993   3866 May 26 18:12 2008, ./.sh_history
rwxr-xr-x 632/-993     48 Mar 20 12:40 2008, ./.profile
rw-r--r-- 632/-993     29 Apr 23 13:31 2008, ./test.proc
rw-r--r-- 632/-993     29 May 26 18:05 2008, ./pos_examp/file1
rw-r--r-- 632/-993     52 May 26 18:07 2008, ./pos_examp/file2
rw------- 632/-993   3602 May 26 18:07 2008, ./pos_examp/file3
```

Beispiel 3 Dateien aus einer fremden Archivdatei in das aktuelle Dateiverzeichnis einlesen:
Die Dateien in der Archivdatei haben eine andere Benutzer- und Gruppennummer als der Benutzer, der das *tar*-Kommando aufruft. Der Benutzer möchte erreichen, dass bei den eingelesenen Dateien seine Benutzernummer als Eigentümer und seine Gruppennummer eingetragen wird:

```
$ id ──────────────────────────────────────────────────────────────────── (1)
uid=3010(VSC0) gid=3030(VSCGO)
$ tar tvf /home/vsx/vsx1/km/example.TAR ──────────────────────────────── (2)
Tar: blocksize = 20
USTAR format archive
rw-r--r-- 3001/3002    883 May 28 13:29 2008, ./pos_examp/file1
rw-r--r-- 3001/3002   1766 May 28 13:30 2008, ./pos_examp/file2
rw-r--r-- 3001/3002   2649 May 28 13:30 2008, ./pos_examp/file3
$ tar xvfo  /home/vsx/vsx1/km/example.TAR ─────────────────────────────── (3)
Tar: blocksize = 20
USTAR format archive
tar: . : exists - no update
x ./pos_examp/file1, 883 bytes, 2 tape blocks
x ./pos_examp/file2, 1766 bytes, 4 tape blocks
x ./pos_examp/file3, 2649 bytes, 6 tape blocks
$ ls -lR ──────────────────────────────────────────────────────────────── (4)
total 8
drwxr-xr-x   2 VSC0     VSCGO       2048 May 28 13:42
pos_examp
./pos_examp:
total 24
-rw-r--r--   1 VSC0     VSCGO        883 May 28 13:29 file1
-rw-r--r--   1 VSC0     VSCGO       1766 May 28 13:30 file2
-rw-r--r--   1 VSC0     VSCGO       2649 May 28 13:30 file3
```

(1) Der Benutzer VSC0 lässt sich seine Benutzer- und Gruppennummer anzeigen.

(2) Das Inhaltsverzeichnis der fremden Archivdatei *example.TAR* wird ausgegeben. Dabei ist erkennbar, dass die Dateien und Verzeichnisse eine andere Benutzer- und Gruppennummer haben als die des Benutzers VSC0.

(3) Mit der Option *-o* im *tar*-Kommando legt der Benutzer VSC0 fest, dass beim Entpacken der Archivdatei in das aktuelle Dateiverzeichnis seine Benutzer- und Gruppennummer übernommen werden.

(4) Zur Überprüfung, ob die Benutzer- und Gruppennummer tatsächlich übernommen wurde, verwendet der Benutzer VSC0 das *ls*-Kommando.

Hinweis **Austausch von *tar*-Archiven zwischen UNIX-Systemen und BS2000**

In UNIX-Systemen erstellte *tar*-Archive liegen im ASCII-Format vor. Da POSIX die Dateien jedoch standardmäßig im EBCDIC-Format behandelt und eine normale ASCII-EBCDIC-Konvertierung der Archivdateien nicht möglich ist, kann der Austausch von Archivdateien nur über ASCII-Dateisysteme erfolgen. Im Folgenden sind die möglichen Vorgehensweisen kurz erläutert:

Variante 1 (mit NFS)

Das *tar*-Archiv ist auf dem UNIX-System gespeichert und in POSIX zugreifbar (montiert über NFS). Wenn die Shell-Variable *IO_CONVERSION* mit *YES* belegt ist, kann das *tar*-Archiv mit automatischer Konvertierung in das POSIX-Dateisystem entpackt werden.

Variante 2 (ohne NFS)

In POSIX ist ein ASCII-Dateisystem anzulegen (siehe POSIX-Handbuch „Grundlagen für Anwender und Systemverwalter“ [1], Beschreibung zum *POSIX filesystem marker = n* bei *Install POSIX subsystem*). Die Dateien in einem solchen Dateisystem werden bei Nutzung der Shell-Variablen *IO_CONVERSION=YES* als ASCII-Dateien behandelt.
Das *tar*-Archiv muss binär vom UNIX-System in ein solches POSIX-Dateisystem übertragen werden. Anschließend kann bei automatischer Konvertierung in ein POSIX-EBCDIC-Dateisystem entpackt werden.

Siehe auch *ar*, *cpio*, *pax*

tee Pipes zusammenfügen und Eingabe kopieren (duplicate standard input)

tee überträgt Daten von der Standard-Eingabe auf die Standard-Ausgabe und kopiert die Daten gleichzeitig in die angegebene Datei.

Syntax **tee**[␣**-ai**][␣datei...]

-a (a - append) *tee* fügt seine Ausgabe an den alten Inhalt von *datei* an, wenn *datei* beim Aufruf von *tee* schon existiert.

-a nicht angegeben:
tee überschreibt mit seiner Ausgabe den alten Inhalt von *datei*, wenn *datei* beim Aufruf von *tee* schon existiert.

-i (i - ignore) Das Signal SIGINT (siehe *kill*) wird ignoriert.

datei
Name der Datei, in die *tee* seine Ausgabe schreibt. Wenn *datei* beim Aufruf von *tee* noch nicht existiert, wird sie angelegt. Wenn *datei* bereits existiert und Sie *tee* ohne Option *-a* aufrufen, wird der alte Inhalt von *datei* überschrieben. Wenn Sie mehrere Dateien angeben, schreibt *tee* seine vollständige Ausgabe in jede dieser Dateien.

datei nicht angegeben:
tee schreibt seine Ausgabe nur auf die Standard-Ausgabe.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *tee*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Das Kommando *tee* in einer Pipe

```
$ (date; who) | tee sicher| wc -l
      3
$ cat sicher
Mon Mar 9 16:19:41 MEZ 2009
QM212JNA   term/003   Mar 8 14:06:28
user2      pts/0      Mar 9 16:02:54
```

Die Kommandos *date* und *who* geben das aktuelle Datum und die aktuell am System angemeldeten Benutzer auf die Standard-Ausgabe aus. Diese Ausgabe wird dem Kommando *tee* durch das Pipe-Zeichen | als Standard-Eingabe zur Verfügung gestellt. *tee* liest die Eingabe und schreibt sie in die Datei *sicher* und auf die Standard-Ausgabe. Die Standard-Ausgabe von *tee* wird durch das nächste Pipe-Zeichen zur Standard-Eingabe des Kommandos *wc -l*. *wc -l* gibt die Anzahl der eingelesenen Zeilen auf die Standard-Ausgabe aus:
3 (2 angemeldete Benutzer und 1 Zeile für das Datum).

test Bedingungen prüfen (evaluate expression)

Das in die POSIX-Shell *sh* eingebaute Kommando *test* prüft, ob Bedingungen erfüllt sind. Bedingungen können sein:
– Eigenschaften von Dateien,
– Eigenschaften und Vergleiche von Zeichenketten und
– algebraische Vergleiche ganzer Zahlen.

Sie können Bedingungen auch verneinen; mehrere Bedingungen können Sie miteinander verknüpfen.

Abhängig vom Endestatus können Sie unterschiedliche Kommandos ausführen, Schleifen abbrechen usw.

Für das eingebaute *sh*-Kommando *test* gibt es zwei Schreibweisen (siehe Syntax).
Die Wirkung ist dieselbe.

Syntax

test[␣ausdruck]
[[␣ausdruck␣**]]**

[␣ausdruck␣]
Die eckigen Klammern müssen Sie angeben, ebenso das Leerzeichen ␣ vor bzw. nach *ausdruck*. Auch in diesem Fall führt die Shell das eingebaute *sh*-Kommando *test* aus.

ausdruck
eine Bedingung oder mehrere Bedingungen, die miteinander verknüpft sein können (siehe Abschnitt „Bedingungen verknüpfen“ auf Seite 787).

test prüft die folgenden Bedingungen:

Eigenschaften von Dateien

-r␣datei
(r - read) wahr, wenn *datei* existiert und Sie Leserecht haben.

-w␣datei
(w - write) wahr, wenn *datei* existiert und Sie Schreibrecht haben.

-x␣datei
(x - execute) wahr, wenn *datei* existiert und Sie Ausführrecht haben.

-f␣datei
(f - file) wahr, wenn *datei* existiert und eine einfache Datei ist.

-d␣datei
(d - directory) wahr, wenn *datei* existiert und ein Dateiverzeichnis ist.

-e␣datei
wahr, wenn *datei* existiert.

-h␣datei
wahr, wenn *datei* existiert und ein symbolischer Verweis ist. Normalerweise verfolgen alle anderen Bedingungen symbolische Verweise.

-c␣datei
(c - character device) wahr, wenn *datei* existiert und eine zeichenorientierte Gerätedatei ist.

-b␣datei
(b - block device) wahr, wenn *datei* existiert und eine blockorientierte Gerätedatei ist.

-p␣datei
(p - pipe) wahr, wenn *datei* existiert und eine benannte Pipe (FIFO) ist.

-u␣datei
(u - set user ID) wahr, wenn *datei* existiert und für den Eigentümer das s-Bit gesetzt ist.

-g␣datei
(g - set group ID) wahr, wenn *datei* existiert und für die Gruppe das s-Bit gesetzt ist.

-k␣datei
(k - stic**k**y bit) wahr, wenn *datei* existiert und das Sticky-Bit gesetzt ist.

-s␣datei
(s - size) wahr, wenn *datei* existiert und nicht leer ist.

-t[␣dateikennzahl]
(t - terminal) wahr, wenn die angegebene *dateikennzahl* geöffnet und einer Datensichtstation zugeordnet ist.

dateikennzahl nicht angegeben:
Standardmäßig wird 1 als Dateikennzahl verwendet.

datei
Name der Datei oder des Dateiverzeichnisses, deren Eigenschaften geprüft werden sollen. Sie können auch relative oder absolute Pfadnamen angeben.
Wenn Sie im Dateinamen Sonderzeichen der Shell verwenden, prüft *test* nur die erste zu diesem Namen passende Datei.

Wenn Sie für *datei* die leere Zeichenkette angeben, also nur zwei Anführungszeichen "" oder zwei Hochkommas ”, setzt *test* den Namen Ihres aktuellen Dateiverzeichnisses ein.
Wenn Sie *datei* nicht angeben, gibt *test* eine Fehlermeldung aus und bricht mit Endestatus 1 ab.

Sie können für *datei* auch einen Shell-Parameter angeben. Diesen sollten Sie immer in Anführungszeichen "..." einschließen. Wenn die entsprechende Shell-Variable nicht definiert ist, erhält *test* als Argument die leere Zeichenkette.
Die Anführungszeichen garantieren also, dass *test* beim Ersetzen eines Shell-Parameters immer ein Argument erhält.

Eigenschaften und Vergleiche von Zeichenketten

Als Zeichenkette können Sie eine beliebige Folge von Zeichen angeben. Enthält die Zeichenkette Leer- oder Tabulatorzeichen, müssen Sie diese entwerten. Wenn die Shell die Zeichenkette nicht interpretieren soll, entwerten Sie die entsprechenden Sonderzeichen mit einem vorangestellten Gegenschrägstrich \ oder schließen die ganze Zeichenkette in Anführungszeichen oder Hochkommas ein.

Die leere Zeichenkette geben Sie an durch zwei aufeinanderfolgende Anführungszeichen oder Hochkommas. Wenn Sie keine Zeichenkette angeben, gibt *test* eine Fehlermeldung aus und bricht mit Endestatus 1 ab.

Sie können als Zeichenkette auch einen Shell-Parameter angeben. Diesen sollten Sie immer in Anführungszeichen einschließen. Wenn die entsprechende Shell-Variable nicht definiert ist, erhält *test* als Argument die leere Zeichenkette. Die Anführungszeichen garantieren also, dass *test* beim Ersetzen eines Shell-Parameters immer ein Argument erhält.

[**-n**␣]zeichenkette
: (n - non zero) wahr, wenn die angegebene *zeichenkette* nicht die leere Zeichenkette ist, also eine Länge größer 0 hat.
So können Sie testen, ob einer Shell-Variablen ein Wert zugewiesen ist. Den entsprechenden Shell-Parameter schließen Sie in Anführungszeichen ein.

  -n nicht angegeben:
  Die Angabe ohne *-n* hat die gleiche Bedeutung. Mit *-n* ist die Bedingung leichter lesbar.

-z␣zeichenkette
: (z - zero) wahr, wenn die angegebene *zeichenkette* die leere Zeichenkette ist, also die Länge 0 hat.
So können Sie feststellen, ob einer Shell-Variablen kein Wert zugewiesen ist. Den entsprechenden Shell-Parameter schließen Sie in Anführungszeichen ein.

zeichenkette1␣**=**␣zeichenkette2
: wahr, wenn die beiden Zeichenketten identisch sind. Vor und nach dem Gleichheitszeichen muss jeweils ein Leerzeichen stehen, da *test* dieses Zeichen als eigenständiges Argument erwartet. Wenn Sie Shell-Parameter vergleichen, sollten Sie diese in Anführungszeichen einschließen.

zeichenkette1␣**!=**␣zeichenkette2
: wahr, wenn die beiden Zeichenketten verschieden sind. Vor und nach dem Ungleichheitszeichen muss jeweils ein Leerzeichen stehen, da *test* dieses Zeichen als eigenständiges Argument erwartet. Wenn Sie Shell-Parameter vergleichen, schließen Sie diese in Anführungszeichen ein.

zeichenkette
: wahr, wenn *zeichenkette* nicht die leere Zeichenkette ist.

Algebraische Vergleiche ganzer Zahlen

Ganze Zahlen können Sie direkt oder als Werte von Shell-Variablen angeben. Die angegebenen Zahlenwerte können beliebig groß sein. Einer Shell-Variablen können Sie ebenfalls beliebig große Zahlenwerte zuweisen.
Wenn Sie als Zahl einen Shell-Parameter angeben, sollten Sie diesen immer in Anführungszeichen ".." einschließen. Wenn die entsprechende Shell-Variable nicht definiert ist, erhält *test* als Argument die leere Zeichenkette. Die Anführungszeichen garantieren also, dass *test* beim Ersetzen eines Shell-Parameters immer ein Argument erhält.

zahl1␣op␣zahl2
: *test* vergleicht die beiden ganzen Zahlen *zahl1* und *zahl2* algebraisch entsprechend der Angabe für *op*.
Die Vergleichsoperatoren erwartet *test* als eigenständige Argumente. Deshalb müssen sie zwischen zwei Leerzeichen stehen.

op kann sein:

-eq (eq - equal) wahr, wenn die beiden Zahlen gleich sind.

-ne (ne - not equal) wahr, wenn die beiden Zahlen ungleich sind.

-ge (ge - greater than or equal) wahr, wenn *zahl1* größer oder gleich *zahl2* ist.

-gt (gt - greater than) wahr, wenn *zahl1* größer ist als *zahl2*.

-le (le - less than or equal) wahr, wenn *zahl1* kleiner oder gleich *zahl2* ist.

-lt (lt - less than) wahr, wenn *zahl1* kleiner ist als *zahl2*.

Bedingungen verneinen

!␣bedingung
: wahr, wenn die angegebene Bedingung nicht erfüllt ist. Nach dem Ausrufezeichen muss ein Leerzeichen stehen.

Beispiel

```
$ ! -r datei
```

Wenn Sie die angegebene Datei nicht lesen dürfen, liefert *test* als Endestatus den Wert 0 (wahr) zurück.

Bedingungen verknüpfen

Mehrere Bedingungen können Sie miteinander zu einem *ausdruck* verknüpfen. Die Bedingung selbst kann auch verneint sein.
Die entsprechenden Verknüpfungsoperatoren erwartet *test* als eigenständige Argumente. Deshalb müssen sie zwischen zwei Leerzeichen stehen.

Das Kommando *find* durchsucht Dateiverzeichnisse nach Dateien, die vorgegebene Bedingungen erfüllen. Diese Bedingungen werden ähnlich verknüpft wie bei *test*.

Bedingungen können Sie wie folgt miteinander verknüpfen:

bedingung␣**-a**␣bedingung
: (a - and) wahr, wenn jede der so aneinandergereihten Bedingungen erfüllt ist, also logisches UND. Vor und nach dem Operator *-a* muss jeweils ein Leerzeichen stehen.

bedingung␣**-o**␣bedingung
: (o - or) wahr, wenn mindestens eine der Bedingungen erfüllt ist, also logisches ODER. Vor und nach dem Operator *-o* muss jeweils ein Leerzeichen stehen.

\(␣ausdruck␣**\)**
: *ausdruck* steht hier für zwei oder mehr Bedingungen, die beliebig miteinander verknüpft sind. Die runden Klammern fassen diese Bedingungen so zusammen, dass eine vor der Klammer angegebene Verknüpfung sich auf den Klammer-Inhalt und nicht nur auf die direkt nachfolgende Bedingung bezieht.
Da die runden Klammern für die Shell eine Sonderbedeutung haben, müssen sie mit \ entwertet werden. Vor und nach *ausdruck* muss jeweils ein Leerzeichen stehen.

Beispiel

```
$ ! \( -r datei -o -w datei \)
```

ODER

Der Ausdruck in den runden Klammern ist wahr, wenn Sie für die angegebene Datei Lese- oder Schreibrecht haben. Das Ausrufezeichen verneint den Klammerinhalt. Deshalb liefert *test* als Endestatus den Wert 0 (wahr) zurück, wenn Sie für die angegebene Datei weder Lese- noch Schreibrecht haben.

Priorität der Verknüpfungen

Die Verknüpfungen haben für *test* folgende Priorität:
Klammern vor Verneinung vor UND vor ODER

Im vorigen Beispiel wird also zuerst der Inhalt der Klammern ausgewertet und anschließend verneint.

Endestatus

0 der angegebene Ausdruck ist wahr

1 der angegebene Ausdruck ist syntaktisch richtig, aber die Auswertung des Ausdrucks liefert den Wert falsch oder Sie haben keinen Ausdruck angegeben.

>1 Fehler (z.B. syntaktisch falscher Ausdruck)

Fehler `test: argument expected`
Diese Fehlermeldung erhalten Sie, wenn Sie eine Bedingung unvollständig angegeben haben, wenn also eine Datei bzw. eine Zeichenkette bzw. eine Zahl in der Angabe fehlen. Schließen Sie deshalb die Shell-Parameter immer in Anführungszeichen ein. Andernfalls fehlt ein Argument, wenn die entsprechende Shell-Variable nicht definiert ist.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *test*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die folgende Shell-Prozedur prüft, ob der angegebene Stellungsparameter der Name einer Datei oder eines Dateiverzeichnisses ist.

```
if test -f "$1"                 # oder: if [ -f "$1" ]
   then
   echo $1 ist eine Datei
   elif test -d "$1"            # oder: elif [ -d "$1" ]
      then
      echo $1 ist ein Dateiverzeichnis
fi
```

Der Stellungsparameter *$1* ist in Anführungszeichen eingeschlossen. Deshalb setzt *test* dafür das aktuelle Dateiverzeichnis ein, falls Sie beim Aufruf der Shell-Prozedur kein weiteres Argument angeben.
Ohne Anführungszeichen würde *test* in diesem Fall mit einer Fehlermeldung abbrechen.

Beispiel 2 Die folgende Prozedur prüft mit dem Operator *-gt*, ob die beim Aufruf zuerst angegebene Datei, also *$1*, mehr Zeilen enthält als die danach angegebene Datei, also *$2*:

```
if [ ` wc -l "$1"`  -gt ` wc -l "$2"`  ]
     then echo $1 enthaelt mehr Zeilen als $2
fi
```

Das Kommando *wc -l* zählt die Zeilen der beiden Dateien und gibt die jeweiligen Zeilenzahlen aus. Die Shell ersetzt die Angabe in den Gegenhochkommas durch entsprechende Zeilenzahl.

Beispiel 3 Es soll festgestellt werden, ob der Shell-Variablen *TAPE* ein Wert zugewiesen ist. Dazu gibt es mehrere Möglichkeiten:

Möglichkeit A

```
if [ ! -z "$TAPE" ]
    then echo Der Variablen TAPE ist ein Wert zugewiesen
    else ....
fi
```

Möglichkeit B

```
if [ -n "$TAPE" ]
    then echo Der Variablen TAPE ist ein Wert zugewiesen
    else ....
fi
```

Die Angabe *-n* kann auch entfallen.

Der Shell-Parameter *$TAPE* steht in Anführungszeichen, damit *test* keine Fehlermeldung ausgibt, falls der Variablen kein Wert zugewiesen ist.

Siehe auch *find*

time Laufzeit eines Kommandos messen (time a simple command)

Mit *time* können Sie die Laufzeit eines Programms oder einer Shell-Prozedur messen. Das Programm oder die Shell-Prozedur wird ausgeführt, anschließend schreibt *time* folgende Rechenzeiten auf die Standard-Fehlerausgabe: real, user, sys.

– *real* ist die Laufzeit des gestarteten Prozesses und seiner Sohnprozesse, d.h. die Zeit zwischen Programmaufruf und -abschluss.

– *user* ist die Zeit, in der sich der Prozess oder einer seiner Sohnprozesse im Benutzermodus befunden hat. Im Benutzermodus befindet sich ein Prozess, wenn er Maschinenbefehle des eigenen Textsegments ausführt.

– *sys* ist die Zeit, in der sich der Prozess oder einer seiner Sohnprozesse im Systemmodus befunden hat. Im Systemmodus befindet sich ein Prozess, wenn er Maschinenbefehle aus Systemaufrufen ausführt.

Die Ausgabe hat das Format hh:mm:ss.zz, wobei *hh* für Stunden, *mm* für Minuten, *ss* für Sekunden und *zz* für Hundertstel Sekunden steht.

Syntax **time**[␣**-p**]␣prog[␣arg...]

-p schreibt die Mess-Ergebnisse auf die Standardfehlerausgabe.

prog
Name des Programms (bzw. der Shell-Prozedur), dessen (bzw. deren) Laufzeit Sie messen möchten.

arg
Argument, das Sie an *prog* genauso übergeben können wie beim Aufruf von *prog* ohne *time*.

Wenn Sie *time* auf einem Multiprozessor-System aufrufen, dann ist die Summe aus der Zeit im Benutzermodus und der Zeit im Systemmodus möglicherweise größer als die reale Laufzeit. Die Angabe einer scheinbaren CPU-Auslastung von mehr als 100% ist also die Folge der Verteilung der Sohnprozesse auf verschiedene Prozessoren.

Endestatus entspricht dem Endestatus von *prog*.

1-125 Fehler in *time*

126 *prog* kann nicht ausgeführt werden.

127 *prog* wurde nicht gefunden

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *time*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

PATH Legt den bei der Kommandosuche verwendeten Suchpfad fest.

Beispiel Die Ausführungszeit des *ls*-Kommandos soll gemessen werden. Die Standard-Ausgabe von *ls* wird in die Datei *liste* umgeleitet.

```
$ time ls -l >liste
real    0m0.04s
user    0m0.57s
sys     0m0.08s
```

Siehe auch *times*
time(), times() [4]

times Gesamtlaufzeit der bisher gestarteten Prozesse ausgeben (write process times)

Das in die POSIX-Shell *sh* eingebaute Kommando *times* gibt aus, wieviel Zeit die Prozesse, die die aktuelle Shell bisher gestartet hat, insgesamt verbraucht haben. Es werden auch die Zeiten der Sohnprozesse ausgegeben.

Die Ausgabe ist aufgeschlüsselt nach Benutzerzeit und Systemzeit der Shell (1. Zeile) und Benutzerzeit und Systemzeit des Sohnprozesses (2. Zeile), angegeben in Minuten (m) und Sekunden (s).

Die Benutzerzeit ist die Zeit, die die Prozesse in der Benutzer-Phase verbraucht haben. Die Systemzeit ist die Zeit, die die Prozesse in der System-Phase verbraucht haben.

Wenn Sie wissen wollen, wieviel Zeit ein bestimmtes Kommando verbraucht, verwenden Sie *time*.

Syntax **times**

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *times*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Gesamt-Laufzeit aller Prozesse abfragen, die die aktuelle Shell bisher gestartet hat:

```
$ times
0m8.68s 0m2.5s
0m22.74s 0m10.14s
```

Die Prozesse haben in der Shell-Benutzer-Phase 8.68 Sekunden und in der Shell-System-Phase 2.5 Sekunden verbraucht. In der Sohn-Benutzer-Phase wurden bisher 22.74 Sekunden und in der Sohn-System-Phase 10.14 Sekunden verbraucht.

Siehe auch *time*
times() [4]

touch Änderungs- und Zugriffszeiten aktualisieren (change file access and modification times)

touch setzt den Zeitpunkt der letzten Änderung bzw. des letzten Zugriffs für Dateien auf das aktuelle oder ein gewünschtes Datum.

Syntax

Format 1: touch[␣**-acm**][**-r**␣refdatei |␣**-t**␣zeit]␣datei␣...

Format 2: touch[␣**-acm**][␣MMDDhhmm[yy]]␣datei␣...

Format 1 **touch**[␣**-acm**][**-r**␣refdatei|␣**-t**␣zeit]␣datei␣...

Keine Option angegeben

touch setzt den Zeitpunkt der letzten Änderung und den Zeitpunkt des letzten Zugriffs für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum. Existiert eine Datei noch nicht, so legt *touch* sie an. *touch* ohne Option wirkt also wie *touch -am*.

option

-a (a - access time) *touch* setzt den Zeitpunkt des letzten Zugriffs für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum.

Weder *-a* noch *-m* angegeben:
touch setzt den Zeitpunkt der letzten Änderung und den Zeitpunkt des letzten Zugriffs für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum.

-c *touch* legt Dateien, die nicht existieren, nicht an. Es wird hierfür keine Meldung ausgegeben.

-m (m - modification time) *touch* setzt den Zeitpunkt der letzten Änderung für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum.

Weder *-a* noch *-m* angegeben:
touch setzt den Zeitpunkt der letzten Änderung und den Zeitpunkt des letzten Zugriffs für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum.

-r␣refdatei

touch benutzt anstelle des aktuellen Zeitpunkts das entsprechende Datum und Zeit von *refdatei*.

-t␣zeit

touch benutzt anstelle der aktuellen Zeit den bei *zeit* angegebenen Zeitpunkt. *zeit* wird als Dezimalzahl folgendermaßen angegeben:
[[CC]YY]MMDDhhmm[.SS]

CC: die ersten beiden Stellen der Jahresangabe (Jahrtausend). Die Angabe des Jahrtausends ist optional. Wird kein Jahrtausend angegeben, wird das aktuelle Jahrtausend verwendet.

YY: die letzten beiden Stellen der Jahresangabe (Jahrhundert). Die Angabe des Jahrhunderts ist optional. Wird kein Jahrhundert angegeben, wird das aktuelle Jahrhundert verwendet.
Wird zwar das Jahrhundert angegeben, aber nicht das Jahrtausend *CC*, dann wird *CC* wie folgt hergeleitet:

bei 69≤YY<99 wird CC zu 19
bei 00≤YY≤38 wird CC zu 20

MM: Monat (01-12)

DD: Tag (01-31)

hh: Stunden (00-23)

mm: Minuten (00-59)

SS: Sekunden (00-61). Die Angabe von Sekunden ist optional.
Wird *SS* nicht angegeben, wird der Wert 00 verwendet.
Sollen Sekunden angegeben werden, kann *SS* im Bereich 00-61 liegen (anstelle des üblichen Bereichs 00-59). Die Angaben 60 und 61 sind als Reservesekunden zu interpretieren.

Das größte bei *zeit* angebbare Datum ist:

```
20380119031407
CCYYMMDDhhmmss
```

datei
Name der Eingabedatei. *touch* bearbeitet alle Arten von Dateien, auch Dateiverzeichnisse. Pro Aufruf können Sie mehrere Dateinamen angeben.
Dateinamen, die nur aus Ziffern bestehen, können zu Problemen führen, da *touch* sie möglicherweise als Datumsangabe interpretiert.

Format 2 **touch**[␣**-acm**][␣MMDDhhmm[yy]]␣datei␣...

Keine Option angegeben
touch setzt den Zeitpunkt der letzten Änderung und den Zeitpunkt des letzten Zugriffs für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum. Existiert eine Datei noch nicht, so legt *touch* sie an. *touch* ohne Option wirkt also wie *touch -am*.

option

-a (a - access time) *touch* setzt den Zeitpunkt des letzten Zugriffs für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum.

Weder *-a* noch *-m* angegeben:
touch setzt den Zeitpunkt der letzten Änderung und den Zeitpunkt des letzten Zugriffs für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum.

-c *touch* legt Dateien, die nicht existieren, nicht an. Es wird hierfür keine Meldung ausgegeben.

-m (m - modification time) *touch* setzt den Zeitpunkt der letzten Änderung für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum.

Weder *-a* noch *-m* angegeben:
touch setzt den Zeitpunkt der letzten Änderung und den Zeitpunkt des letzten Zugriffs für die angegebenen Dateien auf das angegebene bzw. aktuelle Datum.

MMDDhhmm[yy]
touch setzt den Zeitpunkt der letzten Änderung bzw. des letzten Zugriffs auf das angegebene Datum. Die Datumsangabe besteht aus einer acht- bzw. zehnstelligen Zahl: Monat (MM) - Tag (DD) - Stunden (hh) - Minuten (mm) - Jahr (yy)

Wird zwar das Jahrhundert angegeben, aber nicht das Jahrtausend *CC*, dann wird *CC* wie folgt hergeleitet:

bei 69≤YY≤99 wird CC zu 19
bei 00≤YY≤38 wird CC zu 20

Das größte bei *zeit* angebbare Datum ist:

```
0119031438
MMDDhhmmYY
```

yy nicht angegeben:
touch geht vom aktuellen Jahr aus.

MMDDhhmm[yy] nicht angegeben:
touch setzt den Zeitpunkt der letzten Änderung bzw. des letzten Zugriffs auf das aktuelle Datum.

datei
Name der Eingabedatei. *touch* bearbeitet alle Arten von Dateien, auch Dateiverzeichnisse. Pro Aufruf können Sie mehrere Dateinamen angeben.
Dateinamen, die nur aus Ziffern bestehen, können zu Problemen führen, da *touch* sie möglicherweise als Datumsangabe interpretiert.

Fehler `date: bad date conversion`
Sie haben ein unzulässiges Datum angegeben, z.B. 13010000.

Variable *TZ*
bestimmt die Zeitzone, die bei *-t␣zeit* angegeben wird.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *touch*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Der Zeitpunkt des letzten Zugriffs und der letzten Änderung soll für *datei* auf das aktuelle Datum gesetzt werden. Wenn diese Datei nicht existiert, soll sie auch nicht angelegt werden.

```
$ touch -c datei
```

Mit *ls -l* können Sie sich den Zeitpunkt der letzten Änderung ausgeben lassen; *ls -lu* gibt den Zeitpunkt des letzten Zugriffs aus.

Beispiel 2 Der Zeitpunkt des letzten Zugriffs auf *datei* soll auf den 26.8., 9 Uhr, gesetzt werden:

```
$ touch -a 08260900 datei
$ ls -lu datei
-rw-r--r--   1 BERTA   qm231       736  Aug 26 09:00 datei
```

Siehe auch *date*, *ls*
utime() [4]

tput Datensichtstation initialisieren oder Datenbank terminfo abfragen (change terminal characteristics)

Dieses Kommando ist nur für die Benutzer sinnvoll, die über *rlogin* Zugang zur POSIX-Shell erhalten haben.

Mit *tput* können Sie
- eine Eigenschaft einer Datensichtstation ausgeben (Format 1)
- mehrere Eigenschaften einer Datensichtstation ausgeben (Format 2)
- eine Datensichtstation initialisieren (Format 3)
- eine Datensichtstation zurücksetzen (Format 4)
- den ausführlichen Namen einer Datensichtstation ausgeben (Format 5)
- den Bildschirm löschen (Format 6).

Syntax

Format 1: tput[␣**-T**␣typ]␣capname[␣parameter...]

Format 2: tput␣**-S**

Format 3: tput[␣**-T**␣typ]␣**init**

Format 4: tput[␣**-T**␣typ]␣**reset**

Format 5: tput[␣**-T**␣typ]␣**longname**

Format 6: tput[␣**-T**␣typ]␣**clear**

Format 1 **Eine Eigenschaft einer Datensichtstation ausgeben**

tput[␣**-T**␣typ]␣capname[␣parameter...]

Jeweils eine Eigenschaft einer Datensichtstation wird ausgegeben. Diese Eigenschaften sind in der Datenbank *terminfo* festgelegt (siehe „Reliant UNIX Referenzhandbuch für Systemverwalter“ [13], *terminfo()*).

-T␣typ
Für *typ* geben Sie den Typ der Datensichtstation an, dessen Eigenschaften Sie abfragen wollen. Wenn Sie die Option *-T␣typ* verwenden, bleibt der Wert der Shell-Variablen *LINES* und *COLUMNS* unverändert, sowie die Größe des Fensters.

-T␣typ nicht angegeben:
Für *typ* wird der Wert der Umgebungsvariablen *TERM* eingesetzt.

capname
(capability name) *capname* ist die Kurzbezeichnung einer Eigenschaft der Datensichtstation, wie sie in der Quelldatei für *terminfo* steht. Um ausgeben zu lassen, ob Ihre Datensichtstation über eine bestimmte Eigenschaft verfügt, müssen Sie die zugehörige Kurzbezeichnung *capname* angeben (siehe „Reliant UNIX Referenzhandbuch für Systemverwalter“ [13], *terminfo()*).

Je nach Typ der Eigenschaft gibt *tput* folgendes aus:

- Eigenschaft vom Typ Boolscher Wert: *tput* liefert nur einen Endestatus zurück und zwar 0 für *wahr*, wenn die Datensichtstation die entsprechende Eigenschaft hat, und 1 für *falsch*, wenn die Datensichtstation die Eigenschaft nicht hat.
  In der POSIX-Shell fragen Sie den Endestatus mit *echo $?* ab.
- Eigenschaft vom Typ Zeichenkette: *tput* gibt die entsprechende Zeichenkette aus.
- Eigenschaft vom Typ Numerische Angabe: *tput* gibt eine ganze Zahl aus.

Wenn der durch *capname* festgelegten Eigenschaft für den angegebenen Datensichtstationstyp in *terminfo* kein Wert zugewiesen ist, gibt *tput* -1 aus.

parameter
Wenn die Eigenschaft *capname* eine Zeichenkette ist, die Parameter benötigt, dann geben Sie diese als *parameter* an. *capname* und *parameter* werden als zusammengesetzte Zeichenkette an *tput* übergeben. Ein rein numerischer Parameter wird als Zahl übergeben.

Format 2 **Mehrere Eigenschaften einer Datensichtstation ausgeben**

tput␣-S

Mit diesem Aufruf von *tput* können mehrere Eigenschaften der aktuellen Datensichtstation ausgegeben werden. Sie übergeben die Eigenschaften nicht von der Kommandozeile, sondern von der Standard-Eingabe (siehe *Beispiel 4*). Es ist nur jeweils eine Eigenschaft der Datensichtstation pro Zeile zulässig. Wenn Sie mit der Option *-T␣typ* arbeiten, bleibt der Wert der Shell-Variablen *LINES* und *COLUMNS* unverändert. Die Werte 0 und 1 als Endestatus ändern sich (siehe Abschnitt *Fehler*).

capname
(capability name) *capname* ist die Kurzbezeichnung einer Eigenschaft der Datensichtstation, wie sie in der Quelldatei für *terminfo* steht. Um ausgeben zu lassen, ob Ihre Datensichtstation über eine bestimmte Eigenschaft verfügt, müssen Sie die zugehörige Kurzbezeichnung *capname* angeben (siehe „Reliant UNIX Referenzhandbuch für Systemverwalter“ [13], *terminfo()*). Je nach Typ der Eigenschaft gibt *tput* Folgendes aus:

- Eigenschaft vom Typ Boolscher Wert: *tput* liefert nur einen Endestatus zurück und zwar 0 für *wahr*, wenn die Datensichtstation die entsprechende Eigenschaft hat, und 1 für *falsch*, wenn die Datensichtstation die Eigenschaft nicht hat. In der POSIX-Shell fragen Sie den Endestatus mit *echo $?* ab.
- Eigenschaft vom Typ Zeichenkette: *tput* gibt die entsprechende Zeichenkette aus.
- Eigenschaft vom Typ Numerische Angabe: *tput* gibt eine ganze Zahl aus.

– Es kann eine einfache Eingabeumlenkung gemacht werden:
`tput -S <`*dateiname*
Jede Zeile von *dateiname* enthält einen Eintrag in der Form *capname[␣parameter]*.
Wird nur *tput -S* angegeben, muss die Eingabe mit DEL oder @@d beendet werden.

Wenn der durch *capname* festgelegten Eigenschaft für den angegebenen Datensichtstationstyp in *terminfo* kein Wert zugewiesen ist, gibt *tput* -1 aus.

parameter
Wenn die Eigenschaft *capname* eine Zeichenkette ist, die Parameter benötigt, dann geben Sie diese als *parameter* an. *capname* und *parameter* werden als zusammengesetzte Zeichenkette an *tput* übergeben. Ein rein numerischer Parameter wird als Zahl übergeben.

Format 3 **Datensichtstation initialisieren**

tput[␣**-T**␣typ]␣**init**

Wenn die Datenbank *terminfo* einen Eintrag für die aktuell benutzte Datensichtstation enthält, initialisiert *tput* diese entsprechend dem Typ, der bei *-T␣typ* angegeben ist. Im Einzelnen führt *tput* folgende Aktionen durch:
– Die Zeichenketten für die Initialisierung der Datensichtstation (*capname is1*, *is2*, *is3*, *if*, *iprog*) werden ausgegeben, sofern sie vorhanden sind.
– Alle eingetragenen Verzögerungen (z.B. Neue-Zeile-Zeichen) werden im Treiber für die Datensichtstation eingestellt.
– Tabulatorzeichen werden je nach Eintrag unverändert ausgegeben oder zu Leerzeichen expandiert.
– Falls Tabulatorzeichen unverändert ausgegeben werden, werden Standardtabulatorsprünge eingestellt (alle acht Zeichen).

Falls der *terminfo*-Eintrag die Informationen für eine dieser Aktionen nicht enthält, entfällt diese Aktion ohne Fehlermeldung.

-T␣typ
Für *typ* geben Sie den Typ der Datensichtstation an, mit dessen Eigenschaften Sie Ihre Datensichtstation initialisieren wollen.

-T␣typ nicht angegeben:
Für *typ* wird der Wert der Umgebungsvariablen *TERM* eingesetzt.

Format 4 **Datensichtstation zurücksetzen**

tput[␣**-T**␣typ]␣**reset**

Statt der Zeichenketten für die Initialisierung werden die Zeichenketten für das Zurücksetzen der Datensichtstation ausgegeben (*capname rs1*, *rs2*, *rs3*, *rf*). Falls dafür keine Einträge vorhanden sind, jedoch für die Initialisierung, werden die Zeichenketten für die Initialisierung ausgegeben. Ansonsten verhält sich *reset* wie *init*.

-T␣typ
Für *typ* geben Sie den Typ der Datensichtstation an, mit dessen Eigenschaften Sie Ihre Datensichtstation zurücksetzen wollen.

-T␣typ nicht angegeben:
Für *typ* wird der Wert der Umgebungsvariablen *TERM* eingesetzt.

Format 5 **Ausführlichen Namen einer Datensichtstation ausgeben**

tput[␣**-T**␣typ]␣**longname**

Wenn die Datenbasis *terminfo* existiert und einen Eintrag für die benutzte Datensichtstation (vgl. Format 1, Option *-T␣typ*) enthält, wird die ausführliche Bezeichnung der Datensichtstation ausgegeben. Diese ausführliche Bezeichnung ist der letzte Name in der ersten Zeile der Beschreibung der Datensichtstation in der Datenbank *terminfo* (siehe „Reliant UNIX Referenzhandbuch für Systemverwalter“ [13], *terminfo()*).

Format 6 **Bildschirm löschen**

tput[␣**-T**␣typ]␣**clear**

Statt der Zeichenketten für die Initialisierung werden die Zeichenketten für das Löschen des Bildschirms ausgegeben.

-T␣typ
Für *typ* geben Sie den Typ der Datensichtstation an, dessen Bildschirm Sie löschen wollen.

-T␣typ nicht angegeben:
Für *typ* wird der Wert der Umgebungsvariablen *TERM* eingesetzt.

Endestatus *capname* vom Typ Boolescher Wert und die Option *-S* nicht gesetzt:

0 wenn der angegebene Datensichtstationstyp die Eigenschaft hat,

1 wenn der angegebene Datensichtstationstyp die Eigenschaft nicht hat.

capname vom Typ Zeichenkette und die Option *-S* nicht gesetzt:

0 wenn die Eigenschaft *capname* für diesen Datensichtstationstyp definiert ist.

1 wenn die Eigenschaft *capname* für diesen Datensichtstationstyp nicht definiert ist (auf die Standard-Ausgabe wird nichts ausgegeben).

capname vom Typ Boolescher Wert oder Zeichenkette und Option *-S*:

0 wenn alle Zeilen erfolgreich abgearbeitet werden konnten.

1 Endestatus 1 kann niemals vorkommen, da nicht angegeben werden kann, welche Zeile nicht abgearbeitet werden konnte.

capname vom Typ ganze Zahl:

0 Endestatus immer 0. Anhand der Standard-Ausgabe kann entschieden werden, ob *capname* für den angegebenen Datensichtstationstyp definiert ist oder nicht.
Wird -1 auf die Standard-Ausgabe ausgegeben, bedeutet das, dass *capname* nicht definiert ist.

Fehlersituationen werden durch einen Endestatus 2, 3 oder 4 angezeigt.

Fehler Je nach Endestatus gibt *tput* folgende Fehlermeldungen aus:

`tput: unknown terminal type`
Unbekannter Datensichtstationstyp *type* oder keine *terminfo* Datenbasis vorhanden, Endestatus 3

`tput: unknown terminfo capability capname`
Unbekannte Datensichtstationseigenschaft *capname*, Endestatus 4.

Variable *TERM*
Standard-Wert für den Typ der Datensichtstation, der eingesetzt wird, wenn Sie die Option *-T␣typ* nicht angeben.

Datei

*/usr/share/lib/terminfo/?/**
Datenbank mit Binärversionen der Beschreibungen für die einzelnen Datensichtstationstypen.

/usr/include/curses.h
Header-Datei von *curses()*

/usr/include/term.h
Header-Datei von *terminfo()*

*/usr/lib/tabset/**
Information über die Behandlung von Tabulatorzeichen (siehe auch „Reliant UNIX Referenzhandbuch für Systemverwalter“ [13], *terminfo()*, Abschnitt *Tabulatoren und Initialisierung*)

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *tput*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Bildschirm initialisieren

```
$ tput init
```

Eine Steuerzeichen-Folge, die den Bildschirm initialisiert, wird an die aktuelle Datensichtstation geschickt. Es wird die Steuerzeichenfolge genommen, die für den durch die Umgebungsvariable *TERM* festgelegten Datensichtstationstyp gilt.

Beispiel 2 Anzahl der Spalten der aktuellen Datensichtstation ausgeben

```
$ tput cols
80
```

Beispiel 3 Ist die aktuelle Datensichtstation ein Hardcopy-Terminal?

```
$ tput hc
$ echo $?
1
```

hc ist vom Typ Boolescher Wert, *tput* liefert daher nur einen Endestatus zurück. Diesen Endestatus fragen Sie mit *echo $?* ab. Der Endestatus ist 1, die aktuelle Datensichtstation ist also kein Hardcopy-Terminal.

Beispiel 4 Mehrere Eigenschaften der Datensichtstation mit einem *tput*-Aufruf ausgeben

```
$ tput -S <<here
> clear
> cup 10 10
> bold
> here
```

Der Bildschirm wird gelöscht, die Schreibmarke auf Position 10, 10 gebracht und verstärkte Darstellung eingeschaltet. *here* in der letzten Zeile beendet die Liste.

Beispiel 5 Die Datei *input* hat den Inhalt:

```
cols
lines
```

Wird diese Datei als Input z.B. für die Terminaltypen 97801 und 97808 verwendet, dann ergibt der *tput*-Aufruf folgendes:

```
$ tput -S <input
89
24
$
```

Siehe auch *stty*, *tabs*

tr Zeichen ersetzen oder löschen (translate characters)

tr liest einen Eingabetext von der Standard-Eingabe, ersetzt (Format 1 und 2) oder löscht (Format 3 und 4) einzelne Zeichen und schreibt das Ergebnis auf die Standard-Ausgabe.

Syntax

Format 1: tr[␣**-cs**]␣zeichenkette1␣zeichenkette2

Format 2: tr␣**-s**[␣**-c**]␣zeichenkette1

Format 3: tr␣**-d**[␣**-c**]␣zeichenkette1

Format 4: tr␣**-ds**[␣**-c**]␣zeichenkette1␣zeichenkette2

Wenn Sie bei einem Aufruf mehrere Optionen angeben, dann können Sie diese in allen Formaten mit einem führenden Bindestrich und ohne Leerzeichen aneinanderreihen, z.B. *-cs* oder *-dc*.

Zeichen ersetzen

Format 1 **tr**[␣**-cs**]␣zeichenkette1␣zeichenkette2

Format 2 **tr**␣**-s**[␣**-c**]␣zeichenkette1

tr ersetzt im Eingabetext jedes Zeichen, das in *zeichenkette1* vorkommt, durch das entsprechende Zeichen in *zeichenkette2*: Das i-te Zeichen in *zeichenkette1* wird im Eingabetext ersetzt durch das i-te Zeichen in *zeichenkette2*. Ist *zeichenkette2* kürzer als *zeichenkette1*, dann werden die Zeichen aus *zeichenkette1*, zu denen es kein entsprechendes Zeichen in *zeichenkette2* gibt, nicht ersetzt (siehe Beispiel 1 auf Seite 810).

-c (c - complement) *zeichenkette1* wird bezüglich des aktuell gültigen Zeichensatzes (mit oktalen Werten 001 bis 377) komplementiert. Die komplementierte *zeichenkette1* enthält dann alle Zeichen des aktuell gültigen Zeichensatzes außer den in der ursprünglichen *zeichenkette1* angegebenen Zeichen.
Anschließend ersetzt *tr* im Eingabetext das i-te Zeichen in der komplementierten *zeichenkette1* durch das i-te Zeichen in *zeichenkette2*.

-s (s - squeeze) Nach der Ersetzung verkürzt *tr* in der Ausgabe jede Folge von gleichen Zeichen, die in *zeichenkette2* vorkommen, zu einem Zeichen (siehe Beispiel 2 auf Seite 811).

zeichenkette1␣[zeichenkette2]
In *zeichenkette1* geben Sie die Zeichen an, die ersetzt werden sollen. *zeichenkette2* ist die Ersetzungszeichenkette.

In beiden Zeichenketten müssen Sie die Zeichen ohne Leer- oder sonstige Trennzeichen aneinanderreihen.

Enthält eine Zeichenkette Zeichen, die für die Shell eine besondere Bedeutung haben, müssen Sie die Zeichenkette in Hochkommas '...' einschließen oder die Zeichen mit einem vorangestellten Gegenschrägstrich \ entwerten.

Die Zeichenketten können folgende Angaben enthalten:

zeichen beliebiges druckbares Zeichen.

\oktalzahl wobei *oktalzahl* eine ein-, zwei- oder dreistellige Oktalzahl ist. Den Gegenschrägstrich müssen Sie entwerten, damit die Ziffern als Oktalzahl erkannt werden.

tr bearbeitet auch das Zeichen NUL (\000). Achtung: In früheren Versionen wurde das Zeichen NUL als Eingabezeichen stets gelöscht.

sonderzeichen
als Escape-Sequenzen wie beim Kommando *printf*. Folgende Escape-Sequenzen können Sie angeben:

\\ Backslash (zur Unterscheidung von Oktalzahlen)

\a Warnung, Klingel *)

\b Backspace, Rücksetzzeichen *)

\f Form Feed, Seitenvorschub

\n New Line, Zeilenumbruch

\r Carriage Return, Wagenrücklauf

\t Tabulator

\v Vertikal-Tabulator *)

*) Die so markierten Sonderzeichen werden nur auf Zeichenterminals unterstützt (also bei Zugang zur POSIX-Shell über *rlogin*)

a-z oder **[a-z]**
steht für die Folge der Zeichen von *a* bis *z* einschließlich. Die Zeichen sind gemäß der aktuell gültigen Sortierreihenfolge geordnet. Im Unterschied zu internationalisierten regulären Ausdrücken dürfen a und z nur einfache Zeichen, d.h. keine Ausdrücke für Äquivalenzklassen [=c=] oder Zeicheneinheits-Symbole [.cc.] sein.

Das für *a* eingesetzte Zeichen muss in der aktuellen Sortierreihenfolge (siehe *LC_COLLATE*) dem für *z* eingesetzten Zeichen vorangehen.

"b-a" ist ein ungültiger Zeichenbereich und wird zurückgewiesen.

In der Schreibweise ohne Klammern steht "a-a" nur für "a" usw., "---" wird dagegen als drei "-"-Zeichen aufgefasst. In der Klammer-Schreibweise führen "[a-a]" und "[---]" zu undefinierten Ergebnissen.

"a-c" bedeutet je nach Lokalität "abc" oder "aäbc", "n-p" bedeutet je nach Lokalität "nop" oder "noöp" oder "nöop", "t-v" bedeutet je nach Lokalität "tuv" oder "tuüv" oder "tüuv", "r-t" kann je nach Lokalität "rst", "rsßt" oder "rßst" bedeuten.

[:klasse:] *klasse* gibt eine Zeichenklasse an, ähnlich wie bei internationalen regulären Ausdrücken. Folgende Werte für *klasse* sind möglich:

```
alnum, alpha, blank, cntrl, digit, graph, lower, print, punct,
space, upper, xdigit
```

Zeichenklassen dürfen nicht in einer Ersetzungszeichenkette angegeben werden. Ausnahme: Die Klassen *lower* und *upper* sind zulässig, wenn in *zeichenkette1* an derselben Stelle die korrespondierende Zeichenklasse angegeben ist.

[=äquivalenz=]
äquvalenz gibt eine Äquivalenzklasse an, ähnlich wie bei internationalen regulären Ausdrücken.

Äquivalenzklassen dürfen nicht in einer Ersetzungszeichenkette angegeben werden.

[a*n**]** steht für *n*-mal das Zeichen *a*. Z.B. [a*3] steht für aaa.

Ist die erste Ziffer von *n* 0, so wird *n* als Oktalzahl interpretiert, andernfalls als Dezimalzahl.
Wenn *n* fehlt oder 0 ist, dann steht der Ausdruck für so viele Wiederholungen des Zeichens *a*, wie nötig sind, um *zeichenkette2* auf die Länge von *zeichenkette1* aufzufüllen (siehe *Beispiel 1*).

zeichenkette2 nicht angegeben:
Für *zeichenkette2* wird die (evtl. komplementierte, siehe *-c*) *zeichenkette1* genommen.

zeichenkette1 und *zeichenkette2* nicht angegeben:
zeichenkette1 ist die leere Zeichenkette. Für *zeichenkette2* wird entweder die leere Zeichenkette (ohne Option *-c*) oder der gesamte, aktuell gültige Zeichensatz (mit Option *-c*) genommen.

Zeichen löschen

Format 3 **tr**␣**-d**[␣**-c**]␣zeichenkette1

Format 4 **tr**␣**-ds**[␣**-c**]␣zeichenkette1␣zeichenkette2

Ein Aufruf in diesem Format ist nur dann sinnvoll, wenn *zeichenkette1* angegeben ist. *zeichenkette2* wird ignoriert, wenn Option *-s* nicht angegeben ist.

-d (d - delete) Alle Eingabezeichen, die in *zeichenkette1* vorkommen, werden gelöscht. Wenn Option *-s* nicht angegeben ist, wird *zeichenkette2* ignoriert.

-c (c - complement) *zeichenkette1* wird bezüglich des aktuell gültigen Zeichensatzes komplementiert. Die komplementierte *zeichenkette1* enthält dann alle Zeichen des aktuell gültigen Zeichensatzes außer den, in der ursprünglichen *zeichenkette1* angegebenen Zeichen.
Anschließend löscht *tr* alle Eingabezeichen, die in der komplementierten *zeichenkette1* vorkommen.

-s (s - squeeze) *tr* verkürzt in der Ausgabe jede Folge von gleichen Zeichen, die in *zeichenkette2* vorkommen, zu einem Zeichen. Ist *zeichenkette2* nicht angegeben, ist die Option *-s* wirkungslos.

zeichenkette1␣[zeichenkette2]
In *zeichenkette1* geben Sie die Zeichen an, die gelöscht werden sollen. *zeichenkette2* enthält die Zeichen, die nur einmal ausgegeben werden sollen, falls sie in der Ausgabe mehrfach hintereinander auftreten (siehe Option *-s*).

Im Abschnitt *Format 1* ist beschrieben, wie Sie die beiden Zeichenketten eingeben können.

zeichenkette1 und *zeichenkette2* nicht angegeben:
Ist Option *-c* nicht angegeben, werden die Eingabezeichen unverändert auf die Standard-Ausgabe kopiert. Ist *-c* angegeben, wird nichts auf die Standard-Ausgabe ausgegeben, da *tr* sämtliche Eingabezeichen löscht.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *tr*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE In geklammerten Ausdrücken bestimmt diese Variable die Bedeutung von Zeichenbereichen (z.B. *[a-z]*).

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten). Zudem bestimmt *LC_CTYPE*, welche Zeichen bei Verwendung der Option *-c* in der aktuell gültigen Zeichenmenge enthalten sind.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Hinweise Zeichenbereiche von Groß- und Kleinbuchstaben lassen sich in Lokalitäten, die das Zeichen ’ß’ enthalten, nicht immer eindeutig aufeinander abbilden (siehe dazu oben unter "Zeichen ersetzen"). Eine Ausnahme bilden die Zeichenbereiche “a-z” und “A-Z”, die ab dieser POSIX-Version gesondert behandelt werden.

Versuchen Sie trotzdem nicht, mit dem folgenden Kommando Kleinbuchstaben in Großbuchstaben umzuwandeln. Das Ergebnis ist in einer anderen als der POSIX- oder C-Lokalität nicht definiert und kann in älteren POSIX-Versionen oder auf anderen Plattformen verschieden ausfallen:

```
$ tr 'a-z' 'A-Z' <datei
```

Verwenden Sie zum Umwandeln von Kleinbuchstaben in Großbuchstaben (oder umgekehrt) stattdessen die Zeichenklassen:

```
$ tr '[:lower:]' '[:upper:]' <datei
```

```
$ tr '[:upper:]' '[:lower:]' <datei
```

Voraussetzung für eine korrekte Umsetzung ist hier allerdings, dass *LC_CTYPE* und *LC_COLLATE* auf die gleiche Lokalität verweisen. Dies können Sie durch die Zuweisung von *LANG* oder *LC_ALL* sicherstellen.

Beispiel **Zeichen ersetzen (Format 1)**

Beispiel 1 *tr* ohne Option - einfache Beispiele, die die Wirkungsweise von *tr* zeigen

```
$ cat tage
Montag Dienstag Mittwoch Donnerstag Freitag Samstag Sonntag
$ tr MD md <tage
montag dienstag mittwoch donnerstag Freitag Samstag Sonntag
```

tr ersetzt alle *M* durch *m* und alle *D* durch *d*.

```
$ tr MDF md <tage
montag dienstag mittwoch donnerstag Freitag Samstag Sonntag
```

Hier ist die zweite Zeichenkette kürzer als die erste. *tr* ersetzt alle *M* durch *m* und alle *D* durch *d*, das *F* bleibt jedoch unverändert.

Nun soll jeder Kleinbuchstabe durch *x* ersetzt werden. Der folgende Aufruf leistet das Gewünschte *nicht*:

```
$ tr '[a-z]' x <tage
Montxg Dienstxg Mittwoch Donnerstxg Freitxg Sxmstxg Sonntxg
```

Hier ersetzt *tr* nur das *a* durch *x*. Sollen alle Kleinbuchstaben ersetzt werden, dann muss man *tr* folgendermaßen aufrufen:

```
$ tr '[a-z]' '[x*]' <tage
Mxxxxx Dxxxxxxx Mxxxxxxx Dxxxxxxxxx Fxxxxxx Sxxxxxx Sxxxxxx
```

Hier gehört zu jedem Zeichen in Zeichenkette1 ein *x* in Zeichenkette2, da durch den Stern * die Zeichenkette2 mit *x* aufgefüllt wird. Die Hochkommas sind notwendig, da die Zeichenketten Shell-Sonderzeichen enthalten.

Beispiel 2 Es soll eine Liste aller Wörter, die in der Datei *text* vorkommen, erstellt werden, wobei jedes Wort auf einer eigenen Zeile steht. Ein Wort sei eine Zeichenkette, die nur aus Buchstaben besteht.

```
$ cat text

"An der Liebe Niederlagen
laesst der Dichter Lieder nagen."

$ tr -cs [A-Z][a-z] '[\025*]' <text

An
der
Liebe
Niederlagen
laesst
der
Dichter
Lieder
nagen
```

Zeichen löschen (Format 2)

Beispiel 3 Nichtdruckbares Zeichen aus einer Datei löschen (*tr -d*):

```
$ tr -d '\016' <datei
```

tr löscht aus der Datei das Zeichen, dessen Code oktal den Wert 016 hat, und gibt das Ergebnis auf die Standard-Ausgabe aus.

Siehe auch *ed*, *sh*, *sed*

trap Signalbehandlung ändern (trap signals)

Mit dem in die POSIX-Shell *sh* eingebauten Kommando *trap* können Sie vereinbaren, wie die aktuelle Shell auf zukünftig eintreffende Signale reagieren soll. In Shell-Prozeduren können Sie auf diese Weise festlegen, welche „Aufräumarbeiten“ vor dem Abbruch erledigt werden sollen oder an welcher Stelle keine Unterbrechung stattfinden darf. *trap* hat zwei Funktionen:

- *trap* legt fest, wie die Shell auf ein Signal reagieren soll:
  - Die Shell führt die beim Aufruf von *trap* angegebenen Kommandos aus. Wenn diese Kommandos ausgeführt sind, wird aber das eventuell abgebrochene Kommando nicht nochmals gestartet. Shell-Prozeduren werden mit dem Kommando fortgesetzt, das dem unterbrochenen folgt.
  - Die Shell ignoriert das angegebene Signal.
  - Die Shell reagiert auf das angegebene Signal wieder standardmäßig. Sie können mit *trap* die Signalbehandlung wieder auf den Standard zurücksetzen.
- *trap* gibt die Signale aus, für die sich in der aktuellen Shell die Behandlung geändert hat.

Syntax

trap[␣kommandoliste␣signalnummer|␣signalname␣...]

kommandoliste

legt fest, wie die Shell auf die nachfolgend angegebenen Signale reagieren soll, ob sie also

- Kommandos ausführen soll, wenn das angegebene Signal eintrifft,
- das angegebene Signal ignorieren soll, oder
- auf das angegebene Signal wieder standardmäßig reagieren soll.

Kommandos ausführen

Für *kommandoliste* geben Sie ein oder mehrere Kommandos an. Diese Kommandos sollen ausgeführt werden, wenn das angegebene Signal eintrifft. Mehrere Kommandos trennen Sie durch Strichpunkte voneinander. Den Strichpunkt müssen Sie für die Shell entwerten.

Die Kommandoliste muss ein einziges Argument sein. Sobald in dieser Liste Argument-Trennzeichen oder Strichpunkte enthalten sind, müssen Sie *kommandoliste* in Hochkommas '...' bzw. Anführungszeichen "..." einschließen.

Wenn die angegebene *kommandoliste* nicht die leere Zeichenkette ist, gilt die so vereinbarte Signalbehandlung nur in der aktuellen Shell. In jeder Subshell muss diese mit *trap* neu vereinbart werden; andernfalls gilt die Standard-Behandlung (siehe Abschnitt „Signalbehandlung in der Shell“ auf Seite 814).

Beachten Sie dabei, dass die Shell die angegebenen Kommandos zweimal interpretiert:
– Das erste Mal, wenn die Shell *trap* ausführt.
– Das zweite Mal, wenn das entsprechende Signal eintrifft und die Shell die vereinbarten Kommandos ausführt.

Deshalb haben Hochkommas und Anführungszeichen unterschiedliche Bedeutung:

'*kommandoliste*'
Die Shell interpretiert die Sonderzeichen erst bei der Ausführung der Kommandos. Shell-Variablen werden also erst bei der Ausführung durch ihren Wert ersetzt.

"*kommandoliste*"
Die Shell interpretiert die Zeichen $, \ und ` ...` bereits beim Ausführen von *trap*. Oft sind aber zu diesem Zeitpunkt Shell-Variablen noch nicht definiert.

Wenn Sie verhindern wollen, dass die Shell-Prozedur nach dem Eintreffen eines Signals weiter ausgeführt wird, geben Sie *exit* als letztes Kommando in der Kommandoliste an.

Signal ignorieren

Die Angabe "" oder '', also die leere Zeichenkette, für *kommandoliste* bedeutet, dass die angegebenen Signale ignoriert werden.

Die entsprechenden Signale werden auch in jeder Subshell ignoriert.

Signalbehandlung auf den Standard zurücksetzen

Wenn Sie *kommandoliste* nicht angeben, reagiert die Shell auf die angegebenen Signale wieder standardmäßig (siehe Abschnitt „Signalbehandlung in der Shell“ auf Seite 814).

signalnummer | signalname
Nummer oder Name des Signals, auf das die Shell wie angegeben reagieren soll (siehe *signal()* [4]). Sie können mehrere Signalnummern oder -namen angeben, jeweils getrennt durch Leerzeichen. Sobald eines dieser Signale eintrifft, wird *kommandoliste* ausgeführt.

Sinnvolle Angaben (Signalnummer/Signalname) für die Shell sind:
0 / EOF (Beendigung der Shell)
1 / SIGHUP
2 / SIGINT
3 / SIGQUIT
15 / SIGTERM

Die Angabe 0 für *signalnummer* bewirkt, dass die angegebene *kommandoliste* ausgeführt wird, bevor sich die aktuelle Shell beendet; 0 ist kein Signal. Das bedeutet:
– Wenn Sie *trap* im Dialog aufgerufen haben, wird *kommandoliste* ausgeführt, sobald Sie die Taste END drücken.

– Wenn *trap* in einer Shell-Prozedur steht, wird *kommandoliste* nach der Ausführung dieser Prozedur ausgeführt.

Das Signal 9 (SIGKILL) führt immer zum Abbruch, daher ist *trap "9* wirkungslos.

kein Argument angegeben
Wenn Sie *trap* ohne Argumente aufrufen, schreibt es die Signale auf die Standard-Ausgabe, für die sich in der aktuellen Shell die Behandlung geändert hat. Die Ausgabe hat folgendes Format:

signalnummer: kommandoliste
.
.
.

Es werden aber nur die Signalnummern ausgegeben, für die Sie mit dem Kommando *trap* vorher eine Behandlung abweichend vom Standard vereinbart haben (siehe Abschnitt „Signalbehandlung in der Shell“ unten).

Signalbehandlung in der Shell

Ein Prozess kann jederzeit ein Signal erhalten, das entweder er selbst, ein anderer Prozess, oder der Benutzer an der Datensichtstation z.B. durch DEL erzeugt haben. Er kann darauf wie folgt reagieren:
– er ignoriert das eintreffende Signal.
– er bricht ab.
– er ruft eine Funktion auf, in der dieses Signal behandelt wird.

Für die Benutzer, die über *rlogin* Zugang zur POSIX-Shell haben, sind folgende Signale (Nummer/Name) von Bedeutung:

1 / SIGHUP
Verbindung zur Datensichtstation ist unterbrochen

2 / SIGINT
Taste DEL

3 / SIGQUIT
Tasten CTRL \

9 / SIGKILL
Kommando *kill -9 PID*; *PID* ist die Prozessnummer der entsprechenden Shell

15 / SIGTERM
Kommando *kill -15 PID*; *PID* ist die Prozessnummer der entsprechenden Shell

Abhängig davon, ob Sie mit *trap* eine Signalbehandlung vereinbart haben oder nicht, reagiert die Dialog-Shell (DS) oder die Prozedur-Shell im Hintergrund (PSH) auf diese Signale wie folgt:

Signal-Nummer 1
DS: ignorieren
PSH: abbrechen

Signal-Nummer 2
DS: vereinbarte Signalbehandlung nach dem nächsten Kommando oder nach Eingabe von ↵ ausführen, sonst: ignorieren
PSH: ignorieren

Signal-Nummer 3
DS: vereinbarte Signalbehandlung nach dem nächsten Kommando oder nach Eingabe von ↵ ausführen, sonst: ignorieren
PSH: ignorieren

Signal-Nummer 9
DS: abbrechen
PSH: abbrechen

Signal-Nummer 15
DS: ignorieren
PSH: ignorieren

Abhängig davon, ob Sie mit *trap* eine Signalbehandlung vereinbart haben oder nicht, reagiert eine Shell-Prozedur (ShP) bzw. ein aktueller Vordergrundprozess in der Shell-Prozedur (VgP) wie folgt (*PID* steht jeweils für die Prozessnummer der Shell-Prozedur):

– Keine Vereinbarung für Shell-Prozedur und für aktuellen Vordergrundprozess der Shell-Prozedur

  kill -1 PID
  ShP: bricht sofort ab
  VgP: läuft weiter

  kill -2 PID
  ShP: bricht nach normaler Beendigung des Vordergrundprozesses ab
  VgP: läuft weiter

  DEL
  ShP: bricht sofort ab
  VgP: bricht sofort ab

  kill -3 PID
  ShP: bricht nach normaler Beendigung des Vordergrundprozesses ab
  VgP: läuft weiter

CTRL \
ShP: bricht sofort ab
VgP: bricht sofort ab, core-dump wird auf Platte geschrieben

kill -9 PID
ShP: bricht sofort ab
VgP: läuft weiter

kill -15 PID
ShP: bricht nach normaler Beendigung des Vordergrundprozesses ab
VgP: läuft weiter

– Keine Vereinbarung für Shell-Prozedur; Vereinbarung für aktuellen Vordergrundprozess der Shell-Prozedur

kill -1 PID
ShP: bricht sofort ab
VgP: läuft weiter

kill -2 PID
ShP: bricht nach normaler Beendigung des Vordergrundprozesses ab
VgP: führt Vereinbarung aus

DEL
ShP: bricht nach normaler Beendigung des Vordergrundprozesses ab
VgP: führt Vereinbarung aus

kill -3 PID
ShP: bricht nach normaler Beendigung des Vordergrundprozesses ab
VgP: führt Vereinbarung aus

CTRL \
ShP: bricht nach normaler Beendigung des Vordergrundprozesses ab
VgP: führt Vereinbarung aus

kill -9 PID
ShP: bricht sofort ab
VgP: läuft weiter

kill -15 PID
ShP: bricht nach normaler Beendigung des Vordergrundprozesses ab
VgP: führt Vereinbarung aus

– Vereinbarung für Shell-Prozedur; keine Vereinbarung für aktuellen Vordergrundprozess der Shell-Prozedur

kill -1 PID
ShP: nach normaler Beendigung des Vordergrundprozesses Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: läuft weiter
(Signal wird nicht zugestellt)

kill -2 PID
ShP: nach normaler Beendigung des Vordergrundprozesses Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: läuft weiter
(Signal wird nicht zugestellt)

DEL
ShP: Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: bricht sofort ab

kill -3 PID
ShP: nach normaler Beendigung des Vordergrundprozesses Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: läuft weiter
(Signal wird nicht zugestellt)

CTRL \
ShP: Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: bricht sofort ab, core-dump wird auf Platte geschrieben

kill -9 PID
ShP: bricht sofort ab
VgP: läuft weiter

kill -15 PID
ShP: nach normaler Beendigung des Vordergrundprozesses Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: läuft weiter
(Signal wird nicht zugestellt)

– Vereinbarung für Shell-Prozedur und aktuellen Vordergrundprozess der Shell-Prozedur

kill -1 PID
ShP: nach normaler Beendigung des Vordergrundprozesses Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: läuft weiter
(Signal wird nicht zugestellt)

kill -2 PID
ShP: nach normaler Beendigung des Vordergrundprozesses Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: läuft weiter
(Signal wird nicht zugestellt)

DEL
ShP: Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: Ausführung der Vereinbarung; danach Fort setzung des Ablaufs

kill -3 PID
ShP: nach normaler Beendigung des Vordergrundprozesses Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: läuft weiter
(Signal wird nicht zugestellt)

CTRL \
ShP: Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs

kill -9 PID
ShP: bricht sofort ab
VgP: läuft weiter

kill -15 PID
ShP: nach normaler Beendigung des Vordergrundprozesses Ausführung der Vereinbarung; danach Fortsetzung des Ablaufs
VgP: läuft weiter
(Signal wird nicht zugestellt)

In C-Programmen vereinbaren Sie mit der Funktion *signal()*, wie das Programm auf eintreffende Signale reagieren soll (siehe *signal()* [4]).

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *trap*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 In einer Shell-Prozedur soll das Signal 2 ignoriert werden. Deshalb enthält diese Prozedur die folgende Zeile:

```
trap '' 2
```

Die zwei Hochkommas, also die leere Zeichenkette, bewirken, dass das Signal 2 ignoriert wird. Die Shell-Prozedur kann also nicht von außen durch DEL oder mit *kill -2 prozessnummer* abgebrochen werden.

Beispiel 2 Das Kommando *trap* in einer Dialog-Shell:

```
$ trap 'echo Zuletzt abgemeldet: `date` >>$HOME/logdatei' 0
$ trap
0, echo Zuletzt abgemeldet: `date` >>$HOME/logdatei
$ END
.
.
.
login: rosa
Password:
$ cat logdatei
Zuletzt abgemeldet: Mon Mar 9 18:17:23 MEZ 2009
```

Hier wird vereinbart, dass bei Beendigung der aktuellen Shell eine Meldung in die Datei *$HOME/logdatei* geschrieben werden soll. Die Kommandoliste muss in Hochkommas eingeschlossen sein, damit das Kommando *date* erst bei Beendigung der Shell ausgeführt wird.

Beispiel 3 Die Shell-Prozedur *traptest* zeigt, wie temporäre Dateien gelöscht werden sollten, falls Signale während des Ablaufes eintreffen. Sie enthält die folgenden Zeilen, allerdings ohne Zeilennummern:

```
1  TMP=/usr/rtmp/$$
2  trap "rm -f $TMP; trap 0; exit 1" 1 2 3 15
3  trap "rm -f $TMP; exit 0" 0
4  ls > $TMP
   .
   .
   .
```

Zeile 1:
Der Variablen TMP wird der Dateiname */usr/rtmp/$$* zugewiesen. Die Shell ersetzt *$$* durch die Prozessnummer der aktuellen Shell. Deshalb ist der Dateiname eindeutig.

Zeile 2:
Die Kommandoliste ist in Anführungszeichen eingeschlossen, weil der Variablen *TMP* bereits ein Wert zugewiesen ist. Für die Signale 1, 2, 3 und 15 ist folgende Reaktion vereinbart: Die Datei */usr/rtmp/$$* wird gelöscht, die Vereinbarung für das Ende der Prozedur (0) wird rückgängig gemacht (siehe Zeile 3) und die Prozedur mit Endestatus 1 abgebrochen.

Zeile 3:
Als einzige Signalnummer ist 0 angegeben, d.h. für das Ende der Prozedur ist vereinbart: Die Datei */usr/rtmp/$$* wird gelöscht und die Prozedur mit Endestatus 0 beendet. Diese Vereinbarung muss in Zeile 2 mit *trap 0* zurückgesetzt werden, denn das Kommando *exit* beendet die Prozedur-Shell (0). Laut Vereinbarung in Zeile 3 wäre aber dann der Endestatus 0.

Zeile 4:
Hier wird die Datei /usr/rtmp/$$ angelegt. Diese Zeile darf nicht vor den *trap*-Kommandos stehen. Wenn nämlich die Prozedur bereits unterbrochen wird, bevor die Shell das erste *trap*-Kommando ausgeführt hat, würde die Datei nicht gelöscht.

Die Kommandoliste sollte in diesem Fall mit dem Kommando *exit* enden, da sonst möglicherweise die restlichen Kommandos der Prozedur in einem undefinierten Zustand ausgeführt werden.

Siehe auch *exit*, *kill*
signal() [4]

true Endestatus 0 zurückgeben (return true value)

true gibt den Endestatus 0 zurück und führt sonst nichts aus.

true verwenden Sie in Shell-Prozeduren, um die Bedingung *wahr* zu erzeugen.
Die Bedingung *falsch* (Endestatus ungleich 0) erzeugen Sie mit dem Kommando *false*.

Syntax **true**

Endestatus 0

Beispiel Die folgende Shell-Prozedur erzeugt eine Endlosschleife, die Sie z.B. mit der Taste DEL abbrechen können:

```
while true
do
.
.
.
done
```

Siehe auch *false*, *:* (Doppelpunkt)

tsort Topologisch sortieren (topological sort)

Das Kommando *tsort* (topological sort) schreibt auf die Standardausgabe eine vollständig geordnete Liste von Wörtern, die aus einer teilweise geordneten Liste von Wörtern der Datei hervorgeht.

Die Eingabe besteht aus Paaren von Wörtern (nichtleere Zeichenketten), die durch Leerzeichen voneinander getrennt sind. Paare mit verschiedenen Wörtern erfordern Einordnung gemäß der Sortierordnung. Paare mit zwei gleichen Wörtern bedeuten, dass das Wort ohne Einordnung aufgenommen werden soll.

Die Sortierreihenfolge von *tsort* wird von der Umgebungsvariablen *LC_COLLATE* nicht verändert.

Syntax **tsort**␣[datei]

datei
Name der Datei, die sortiert werden soll.

datei nicht angegeben:
Es wird die Standardeingabe genommen

Fehler `Odd data`: Es liegt eine ungerade Anzahl von Feldern in der Eingabedatei vor.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *tsort*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Kommando *tsort* mit folgender Eingabe

```
$ tsort <<@@d
> a b c c d e
> g g
> f g e f
> g g
> h h
@@d
```

ergibt die Ausgabe

```
a
b
c
d
e
f
g
h
```

tty Pfadnamen der aktuellen Datensichtstation ausgeben (return user's terminal name)

tty gibt den Pfadnamen der Datensichtstation aus, mit der der Prozess verbunden ist. Der Endestatus sagt aus, ob die Standard-Eingabe eine Datensichtstation ist.

Ist der Prozess mit einer virtuellen Datenstation verbunden, gibt *tty* deren Namen aus, nicht den der realen Datenstation.

Syntax **tty**[␣**-s**]

-s *tty* gibt nichts aus, sondern liefert nur den Endestatus.

-s nicht angegeben:
Ist die Standard-Eingabe keine Datensichtstation, meldet *tty* dies.

Endestatus

0	Standard-Eingabe ist eine Datensichtstation.
1	Standard-Eingabe ist keine Datensichtstation.
>1	Eine ungültige Option wurde angegeben.

Fehler `not a tty`
Die Standard-Eingabe ist keine Datensichtstation und die Option *-s* wurde nicht angegeben.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *tty*:

LANG	Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.
LC_ALL	Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.
LC_CTYPE	Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
: Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH
: Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Den Namen der aktuellen Datensichtstation ausgeben:

```
$ tty
/dev/term/003
```

Beispiel 2 In einer Prozedur soll eine Ausgabe auf den Bildschirm gelenkt werden, auch wenn die Standard-Ausgabe in eine Datei umgelenkt wird:

```
  .
  .
echo 'Ausgabe auf die Datensichtstation' > `tty`
  .
  .
```

Beispiel 3 Falls die Standard-Eingabe nicht die Datensichtstation ist, soll in der folgenden Prozedur eine Fehlermeldung erzeugt werden:

```
  .
  .
if tty -s
then
read eingabe
  .
  .
else
echo 'Standard-Eingabe ist keine Datensichtstation' >&2
fi
  .
  .
```

type Typ eines Kommandos abfragen (write a description of command type)

Das in die POSIX-Shell *sh* eingebaute Kommando *type* schreibt zu jedem angegebenen Namen auf die Standard-Ausgabe, welches Kommando die Shell ausführen würde, wenn sie diesen Namen liest. So können Sie prüfen, wie die Shell den angegebenen Namen interpretiert.

In der POSIX-Shell *sh* ist *type* eine Alias-Variable für *whence -v*, wobei *whence* ein eingebautes POSIX-Shell-Kommando ist.

Syntax

type␣name␣...

name

Name des Kommandos, dessen Typ Sie abfragen wollen. Sie können mehrere Namen angeben, jeweils getrennt durch ein Leerzeichen.

Ist *name* keine Shell-Funktion und kein eingebautes *sh*-Kommando, sucht die Shell eine ausführbare Datei dieses Namens in den Dateiverzeichnissen, deren Pfadnamen der Variablen *PATH* zugewiesen sind.

Das Kommando *type* gibt aus, ob *name* eine Shell-Funktion oder ein eingebautes *sh*-Kommando ist. Falls *name* eine ausführbare Datei ist, gibt *type* den absoluten Pfadnamen bzw. einen Pfadnamen der Form *./name* aus.
Ist *name* eine Shell-Funktion, gibt *type* zusätzlich die Funktionsdefinition aus. In der POSIX-Shell wird die Definition nicht mit ausgegeben.

Falls *name* eine nicht ausführbare Datei ist, gibt *type* eine Fehlermeldung aus. Die gleiche Fehlermeldung wird auch ausgegeben, wenn *name* in den Dateiverzeichnissen nicht existiert, deren Pfadnamen der Variablen *PATH* zugewiesen sind.

Fehler

`name not found`

Diese Fehlermeldung bedeutet:

– Eine Datei dieses Namens existiert nicht in den Dateiverzeichnissen, die über die Variable *PATH* erreichbar sind, oder
– die Datei dieses Namens ist nicht ausführbar.

Variable

PATH

Suchpfad der Shell

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *type*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Der folgende Bildschirm-Dialog zeigt, welche Informationen das eingebaute *sh*-Kommando *type* liefert:

```
$ type mount
mount is /etc/mount
$ type echo
echo is a shell builtin
$ type ls
ls is a tracked alias for /usr/bin/ls
$ type typetest
typetest is ./typetest
```

Siehe auch *whence, file*

typeset Attribute für Shell-Variable setzen (set attributes for variables)

Mit *typeset* können Sie zwei Dinge tun:
– Variablen mit Zusatzinformation auf die Standard-Ausgabe schreiben
– Attribute und Werte von Variablen setzen.

Syntax **typeset**[␣option][␣name[**=**wert]]...

Sind weder *option* noch *name*[=*wert*] angegeben, dann schreibt die POSIX-Shell die Namen und Attribute aller Variablen auf die Standard-Ausgabe.
Mit Option, aber ohne *name* werden alle Variablen (und deren Attribute) aufgelistet, auf welche die Option zutrifft.
Benutzen Sie das Pluszeichen + anstelle des Bindestrichs -, werden die Werte der Variablen nicht gedruckt.

Durch *option* können Sie der Variablen *name*[=*wert*] ein oder mehrere Attribute geben.
Rufen Sie *typeset* innerhalb einer Funktion auf, wird eine neue Instanz der Variablen *name* angelegt. Attribut und Wert werden nach Verlassen der Funktion wiederhergestellt.
Durch ein Minuszeichen - vor dem Buchstaben werden die Attribute eingeschaltet, durch ein Pluszeichen + ausgeschaltet.

Die folgende Liste von Attributen können Sie als Optionen angeben:

-L[zahl]
Der Wert wird linksbündig abgelegt und führende Blanks werden gelöscht.
zahl ungleich 0 definiert die Länge des Feldes, sonst wird die Länge des Feldes durch die Länge der ersten Wertzuweisung bestimmt.
Diese Länge ist im Folgenden bestimmend. Wird der Variablen ein neuer Wert zugewiesen, dann wird entweder bei zu langer Zuweisung rechts abgeschnitten oder mit Leerzeichen aufgefüllt.
Führende Nullen werden ebenfalls gelöscht, wenn die Option *-Z* gesetzt ist.

Die Option *-R* wird von *-L* ausgeschaltet.

-R[zahl]
Rechtsbündige Ablage des Werts, führende Blanks werden am Anfang von *wert* eingefügt.
zahl ungleich 0 definiert die Länge des Feldes, sonst wird die Länge des Feldes durch die Länge der ersten Wertzuweisung bestimmt. Diese Länge ist im Folgenden bestimmend.
Wird der Variable ein neuer Wert zugewiesen, dann wird entweder bei zu langer Zuweisung rechts abgeschnitten oder links mit Leerzeichen aufgefüllt.

Die Option *-L* wird von *-R* ausgeschaltet.

-Z[zahl]
Rechtsbündige Ablage des Werts, führende Nullen werden am Anfang von *wert* eingefügt, wenn das erste Zeichen ungleich Blank eine Ziffer ist und die Option *-L* nicht eingeschaltet ist. *zahl* ungleich 0 definiert die Länge des Feldes, sonst wird die Länge des Feldes durch die Länge der ersten Wertzuweisung bestimmt.

-f Die Namen gehören zu Funktionen und nicht zu Variablen. Die POSIX-Shell legt die Funktionen in der Datei *.sh_history* ab. Sie können deshalb die Definition einer Funktion am Bildschirm nicht ausgeben lassen, wenn die Datei *.sh_history* nicht vorhanden ist, oder wenn die Option *nolog* beim Lesen der Funktion gesetzt war. Sie können keine Wertzuweisungen vornehmen und nur die folgenden Optionen sind in Verbindung mit *-f* zulässig:

-t Die Ausführungsüberwachung wird für die Funktion eingeschaltet.

-u Die Funktion wird als nichtdefiniert markiert. Die Variable *FPATH* wird zum Suchen der Funktion verwendet, wenn die Funktion das nächste Mal aufgerufen wird.

-x Die Funktionsdefinition bleibt über namentliche Aufrufe von POSIX-Shell-Prozeduren erhalten, d.h. die Funktion wird exportiert.

-i[zahl]
Die Variable ist eine ganze Zahl (integer). Diese Option beschleunigt die Berechnung. *zahl* ungleich 0 definiert die Basis der Ausgabe, sonst bestimmt die erste Wertzuweisung die Basis.

-l Alle Großbuchstaben werden in Kleinbuchstaben umgewandelt. Die Option *-u* wird ausgeschaltet.

-r Die angegebenen Variablen werden als nur lesbar (readonly) markiert. Die Werte dieser Variablen können nicht mehr durch Zuweisungen verändert werden.

-t Markiert die Variable: Marken sind benutzerspezifisch und haben für die POSIX-Shell keine spezielle Bedeutung.

-u Alle Kleinbuchstaben werden in Großbuchstaben umgewandelt. Die Option *-l* wird ausgeschaltet.

-x Die angegebenen Namen werden so markiert, dass sie automatisch in neue Umgebungen exportiert werden.

Fehler `sh: variable: bad number`
wenn bei Option *-i* der Variablen keine Integerzahl zugewiesen wurde.

Beispiel

```
$ typeset -L var1="␣␣31"
$ typeset -L
..
var1='31␣␣'
...
$ typeset -l var2=WERT2
$ typeset -l
var2=wert2
```

Siehe auch *readonly, export, set, env*

ulimit Datei-Größe für das Schreiben begrenzen oder aktuellen Grenzwert abfragen (set or report file size limit)

Mit dem in die POSIX-Shell *sh* eingebauten Kommando *ulimit* können Sie
- abfragen, welche Grenzwerte für die aktuelle Shell oder ihre Sohnprozesse festgelegt sind
- die Grenzwerte einzeln für die aktuelle Shell und alle ihre Sohnprozesse ändern. Als Benutzer ohne POSIX-Verwalter-Privilegien können Sie diese Werte nur herabsetzen. Die neuen Werte gelten für die aktuelle Shell und ihre Sohnprozesse.
  Ein herabgesetzter Wert kann erst wieder erhöht werden, wenn die Shell, in der Sie den Grenzwert herabgesetzt haben, beendet wurde.

Syntax

Format 1: ulimit[␣**-H**][␣**-S**][␣option]

Format 2: ulimit[␣**-H**][␣**-S**][␣option]␣grenzwert

Format 1 **Grenzwerte abfragen**

ulimit[␣**-H**][␣**-S**][␣option]...

ulimit schreibt die durch *option* abgefragten Grenzwerte auf die Standard-Ausgabe.

-H Abfragen eines *harten* Grenzwerts (hard limit).

-S Abfragen eines *weichen* Grenzwerts (soft limit).

Weder **-H** noch **-S** angegeben:
ulimit schreibt die *weichen* Grenzwerte auf die Standard-Ausgabe.

option

Durch Optionen können Sie die abzufragenden Grenzwerte angeben. Sie können die Optionen beliebig kombinieren.

Keine Option angegeben:
ulimit verwendet die Option *-f* (siehe *Format 2*).

-a Abfrage aller Grenzwerte.

Die weiteren Optionen sind unter *Format 2* beschrieben.

Format 2 **Grenzwerte setzen**

ulimit[␣**-H**][␣**-S**][␣option]␣grenzwert

ulimit setzt den durch *option* bezeichneten Grenzwert auf *grenzwert*. Sie können mit jedem Aufruf immer nur einen Grenzwert neu setzen.

-H Setzen eines *harten* Grenzwerts (hard limit). Als Benutzer ohne POSIX-Verwalter-Privilegien können Sie jeden *harten* Grenzwert herabsetzen. Aber nur der POSIX-Verwalter darf einen *harten* Grenzwert erhöhen.

-S Setzen eines *weichen* Grenzwerts (soft limit). Jeder Benutzer kann einen *weichen* Grenzwert auf einen Wert kleiner dem *harten* Grenzwert setzen.

Weder **-H** noch **-S** angegeben:
ulimit setzt *harte* und *weiche* Grenzwerte auf den angegebenen Wert.

option
Durch Optionen können Sie die zu setzenden Grenzwerte angeben. Die folgenden Grenzwerte, die in *getrlimit()* [4] genauer beschrieben sind, stehen Ihnen für die aktuelle Shell und alle ihre Sohnprozesse zur Verfügung:

-a (all) Abfrage aller Grenzwerte.

-f (file size) Maximale Dateigröße (in 512-byte-Blöcken), die Sie anlegen (schreiben) dürfen; das Lesen ist nicht beschränkt. Ist *file size* gleich 0, können keine Dateien angelegt werden. Wenn Sie den festgelegten Standardwert überschreiten, erhalten Sie (abhängig vom Kommando mit dem Sie die Datei erzeugen) entweder eine Fehlermeldung vom entsprechenden Kommando oder die neue Datei enthält nur die Daten bis zum Erreichen des Grenzwertes.

Beispiel für die Dateigröße

Nach *ls -lR >datei* enthält *datei* nur so viele Bytes, wie der aktuelle Grenzwert erlaubt.
Bei *cp* erhalten Sie eine Fehlermeldung, wenn die Datei, die kopiert werden soll, größer ist als der aktuelle Grenzwert.

-m (memory) Maximale Größe des *heap* (in Kbyte) eines Prozesses.

-n (number of filedescriptors) Maximale Anzahl (geöffneter) Dateikennzahlen eines Prozesses plus 1.

-t (t - time) Maximal verbrauchbare CPU-Zeit (in Sekunden) für einen Prozess.

Keine Option angegeben.
ulimit verwendet die Option *-f*.

grenzwert
legt den Grenzwert für die aktuelle Shell und jeden ihrer Sohnprozesse entsprechend der angegebenen Option fest. Für *grenzwert* können Sie als Benutzer ohne POSIX-Verwalter-Privilegien nur Werte angeben, die kleiner sind als der aktuelle Grenzwert. Als POSIX-Verwalter können Sie mit *grenzwert* den aktuellen Grenzwert auch erhöhen. Geben Sie für *grenzwert* die Zeichenkette *unlimited* an, dann wird der Grenzwert auf den maximal möglichen Wert gesetzt.

Fehler `sh: ulimit; exceeds allowable limit`
Sie haben versucht, den aktuellen Grenzwert zu erhöhen. Dies darf nur der POSIX-Verwalter.

Endestatus

0 bei erfolgreicher Ausführung des Kommandos

>0 Rückweisen eines höheren Limits oder Fehler

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *ulimit*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Der aktuelle weiche Grenzwert für die maximale Anzahl der Dateikennzahlen wird erhöht. Dies ist jedem Benutzer möglich, solange der neue Grenzwert unterhalb des harten Grenzwerts liegt. Der neue Grenzwert ist auch in einer Subshell gültig.

```
$ ulimit -Sn 80
$ ulimit -Sn
80
```

Beispiel 2 Der aktuelle Grenzwert für die maximale Dateigröße wird abgefragt und anschließend herabgesetzt. Diesen Grenzwert kann nur der POSIX-Verwalter wieder erhöhen.

```
$ ulimit
unlimited
$ ulimit 20000
$ sh
$ ulimit
20000
```

Der neue Grenzwert ist auch in einer Subshell gültig. Ab jetzt können nur noch Dateien angelegt werden, die kleiner sind als 20.000 * 512 byte.

Siehe auch *getrlimit(), signal(), ulimit()* [4]

umask Standard-Vergabe der Zugriffsrechte ausgeben oder ändern (get or set the file mode creation mask)

Das in die POSIX-Shell *sh* eingebaute Kommando *umask* gibt die aktuell gültige Schutzbit-Maske aus oder ändert sie. Diese Schutzbit-Maske legt fest, welche Zugriffsrechte die Dateien und Dateiverzeichnisse erhalten, die Sie ab jetzt in der aktuellen Shell oder in einer ihrer Subshells neu anlegen.

Wenn Sie mit *umask* die Schutzbit-Maske ändern, gilt diese Änderung so lange, bis Sie mit *umask* einen neuen Wert vereinbaren oder die Shell beenden, in der Sie *umask* aufgerufen haben.

Der POSIX-Verwalter kann mit *umask* den Wert der Schutzbit-Maske in der Datei */etc/profile* festlegen. Da */etc/profile* von jeder Login-Shell ausgeführt wird, gelten die so bestimmten Zugriffsrechte für jeden am System angemeldeten Benutzer (siehe *Beispiel 1*).

Syntax **umask**[␣**-S**][␣maske]

-S Die Maske wird symbolisch in folgender Form ausgegeben:
`u=<zugriffsrechte>,g=<zugriffsrechte>,o=<zugriffsrechte>`
wobei u = user, g = group, o = others und zugriffsrechte = r, w, x

maske
eine dreistellige Oktalzahl für die Schutzbit-Maske. Diese Schutzbit-Maske legt fest, welche Zugriffsrechte die Dateien und Dateiverzeichnisse erhalten, die Sie ab jetzt in der aktuellen Shell oder in einer ihrer Subshells neu anlegen (siehe *Aus der Schutzbit-Maske die Zugriffrechte bestimmen*).

Da Sie mit *umask* von der Grundeinstellung nur Rechte wegnehmen können, ist es unabhängig von der Angabe für *maske* nicht möglich, an Dateien standardmäßig das Ausführrecht zu vergeben. Verwenden Sie dazu *chmod*.

maske nicht angegeben:
umask gibt die derzeit gültige Schutzbit-Maske aus. Die Ausgabe hat das Format `0nnn` (`0` steht für Oktaldarstellung und `nnn` steht für die aktuelle Schutzbit-Maske).

Aus der Schutzbit-Maske die Zugriffsrechte bestimmen

Beim Anlegen von Dateien und Dateiverzeichnissen werden in der Regel folgende Zugriffsrechte als Grundeinstellung vergeben (siehe *open()* [4]):
– `rw-rw-rw-` an Dateien; das ist `110110110` in Binärdarstellung
– `rwxrwxrwx` an Dateiverzeichnisse; das ist `111111111` in Binärdarstellung

Mit *umask* können Sie von dieser Grundeinstellung nur Rechte wegnehmen. Das bedeutet, es ist mit *umask* nicht möglich, an Dateien standarmäßig das Ausführrecht zu vergeben. Das entsprechende x-Bit können Sie nur mit dem Kommando *chmod* setzen.

Die Schutzbit-Maske ist eine dreistellige Oktalzahl, die Sie beim Aufruf von *umask* angeben. Die anschließend gültigen Zugriffsrechte ergeben sich wie folgt:

1. Rechnen Sie die Schutzbit-Maske um in eine Binärzahl.
2. Bilden Sie zu dieser Binärzahl das Komplement; d.h. ersetzen Sie jede Null durch eine Eins und jede Eins durch eine Null.
3. Verknüpfen Sie dieses Komplement und den Binärwert der Grundeinstellung mit UND; d.h das Ergebnis hat nur an den Stellen eine Eins, an der beide Summanden eine Eins haben, an allen anderen Stellen steht eine Null.

Beispiel

Die Schutzbit-Maske 022 ändert die Zugriffsrechte wie folgt:

– Dateien:
Für Dateien gilt die Grund-Einstellung `110110110`.

```
1. Die Binärzahl zu 022 ist   000010010
2. Das Komplement dazu ist    111101101
3. UND-Verknüpfung:           111101101
                              110110110
                              _________
                              110100100
```

Also haben alle neu angelegten Dateien die Zugriffsrechte `rw-r--r--`

– Dateiverzeichnisse:
Für Dateiverzeichnisse gilt die Grund-Einstellung `111111111`.

```
UND-Verknüpfung:          111101101
                          111111111
                          _________
                          111101101
```

Also haben alle neu angelegten Dateiverzeichnisse die Zugriffsrechte `rwxr-xr-x`

Datei *`/etc/profile`*
Datei, die von jeder Login-Shell ausgeführt wird. Sie dient zur Einstellung einer Shell-Umgebung. Der POSIX-Verwalter legt hier in der Regel einen Schutzmasken-Wert für Benutzer ohne Sonderrechte und einen für *root* fest.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *umask*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die Datei */etc/profile* enthält häufig die folgenden Zeilen:

```
if [ "$USER" != "admin" -a "$USER" != "root" ]
then
        umask 066
fi
```

Da jede Login-Shell die Datei */etc/profile* ausführt, hat die Schutzbit-Maske für alle Benutzer außer *root* und *admin* den Wert 066.

Deshalb werden folgende Zugriffsrechte vergeben:
– für neu angelegte Dateien: `rw-------`
– für neu angelegte Dateiverzeichnisse: `rwx--x--x`

Beispiel 2 Schutzbit-Maske ändern und ausgeben:

```
$ umask 033
$ > neu
$ mkdir neudvz
$ ls -ld neu neudvz
-rw-r--r--   1 ANNA     other          0   Mar 22 09:49 neu
drwxr--r--   2 ANNA     other        520   Mar 22 16:40 neudvz
$ umask
0033
```

Die Ausgabe des Kommandos *ls -ld ...* zeigt, welche Zugriffrechte an neue Dateien und Dateiverzeichnisse vergeben werden, wenn die Schutzbit-Maske auf den Wert 033 gesetzt wird.

Siehe auch *chmod*
chmod(), *creat()*, *open()*, *umask()* [4]

umount Dateisysteme und ferne Ressourcen aushängen (unmount a filesystem)

Das Kommando *umount* hängt ein über *mount* eingehängtes Dateisystem wieder aus. Der Eintrag des Dateisystems wird aus der Tabelle */etc/mnttab* gelöscht.

Syntax **umount**[␣**-V**]␣{ressource | einhängepunkt}

option

-V Gibt die gesamte Kommandozeile auf dem Bildschirm aus, führt das Kommando jedoch nicht aus. Die Kommandozeile wird mit den vom Benutzer eingegebenen Optionen und Argumenten sowie aus */etc/vfstab* abgeleiteten Werte erstellt. Diese Option sollte verwendet werden, um die Kommandozeile einer allgemeinen Prüfung und einer Gültigkeitsprüfung zu unterziehen.

ressource

Gibt die Ressource an, die ausgehängt werden soll. Format wie bei *mount*.

Für bs2fs-Dateisysteme muss die Option entsprechend dem Eintrag in den internen Tabellen in GROSSBUCHSTABEN angegeben werden. Sonderzeichen der POSIX-Shell wie '$' oder '*' müssen explizit entwertet werden.

Wird bei NFS-Ressourcen durch den Namen des bereitstellenden Servers ersetzt, auf den ein Doppelpunkt und der Pfadname der Ressource folgen müssen.

einhängepunkt

Gibt an, wo die *ressource* lokal ausgehängt werden soll. Es muss ein absoluter Pfadname angegeben werden.

Hinweis Für das Aushängen von bs2fs-Dateisystemen wird grundsätzlich die Angabe der Option *einhängepunkt* empfohlen. Ist ein bs2fs-Dateisystem mehrfach an verschiedenen Stellen im POSIX-Dateisystem gemounted (d.h. identische *ressource*-Angabe bei *mount*) und bei *umount* wird nur *ressource* angegeben, dann wird nur das zuletzt eingehängte Dateisystem ausgehängt. Bei Angabe von *einhängepunkt* wird dagegen immer das entsprechende Dateisystem ausgehängt.

Das Kommando *umount* wird abgewiesen, wenn es sich auf ein mit der Option *bs2fscontainer* gemountetes Dateisystem bezieht und noch mindestens ein bs2fs-Dateisystem eingehängt ist. Bei einem erfolgreichen *umount* des bs2fs-Containers werden die bs2fsd-Kopierdämonen automatisch beendet.

Datei *etc/mnttab*
Tabelle der eingehängten Dateisysteme

etc/vfstab
Tabelle der automatisch eingehängten Dateisysteme

Siehe auch *mount, umountall*
mount, umount [4]

umountall Aushängen mehrerer Dateisysteme (unmount filesystems)

umountall bewirkt das Aushängen aller eingehängten Dateisysteme außer *root*, */proc*, */var* und */usr*. Ist **nur** der *dateisystemtyp* angegeben, beschränkt sich die Wirkung von *umountall* auf Dateisysteme des angegebenen Typs. Die Dateisysteme werden in der Reihenfolge *nfs* - *bs2fs* - *ufs* ausgehängt. Somit ist gewährleistet, dass das für bs2fs-Dateisysteme erforderliche bs2fscontainer-Dateisystem im ufs erst ausgehängt wird, wenn es nicht mehr benötigt wird, d.h. wenn keine bs2fs-Dateisysteme mehr eingehängt sind.

Syntax **umountall**[␣**-F**␣dateisystemtyp][␣**-k**][␣**-l** |␣**-r**|␣**-b**]

-**F** Angabe des Dateisystem-Typs, der ausgehängt werden soll.

-k Sendet das Signal SIGKILL an Prozesse, die Dateien im Dateisystem geöffnet haben.

-**l** Begrenzt den Vorgang auf lokale Dateisysteme (*ufs* und *bs2fs*).

-r Begrenzt den Vorgang auf ferne Dateisystem-Typen (*nfs*).

-b Begrenzt den Vorgang auf bs2fs-Dateisysteme.

Fehler Wenn die Dateisysteme aushängbar sind, wird keine Meldung ausgegeben. Fehler- und Warnmeldungen stammen von *fsck* und *mount*.

Hinweis Wenn die Option *-F* in Kombination mit einer oder mehrerer der Optionen *-l*, *-r* und *-b* angegeben wird und die Optionen miteinander verträglich sind, so haben die Optionen *-l*, *-r* und *-b* Vorrang. Beispielsweise haben *umountall -F bs2fs -l* und *umountall -F ufs -l* dieselbe Wirkung wie *umountall -l* : Es werden alle lokalen Dateisysteme (also alle ufs- und bs2fs-Dateisysteme) ausgehängt. Ebenso führen die Angaben *umountall -F bs2fs* und *umountall -b* zum gleichen Ergebnis: alle bs2fs-Dateisysteme werden ausgehängt.

Datei */etc/mnttab*
Tabelle der eingehängten Dateisysteme

/etc/vfstab
Tabelle der automatisch eingehängten Dateisysteme

Siehe auch *fsck, mount*, *mountall, umount*

unalias Variablen aus der alias-Tabelle löschen (remove alias definitions)

Das in die POSIX-Shell *sh* eingebaute Kommando *unalias* löscht die mit *alias* vereinbarten Alias-Variablen.

Syntax

Format 1: unalias␣name␣...

Format 2: unalias␣**-a**

Format 1 **unalias**␣name␣..

name
Die durch *name* gegebenen Aliasdefinition wird gelöscht.

Format 2 **unalias**␣**-a**

-a Alle Aliasdefinitionen werden gelöscht.

Endestatus

0 Das Kommando wurde erfolgreich ausgeführt.

>0 Einer der Operanden *name* ist kein gültiger Aliasname oder Fehler.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *unalias*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die bestehende Aliasdefinitionen werden ausgegeben und die mit dem Namen lx definierte Aliasvariable wird gelöscht.

```
$ alias
autoload='typeset -fu'
cat=/usr/bin/cat
command='command '
false='let 0'
functions='typeset -f'
hash='alias -t -'
history='fc -l'
integer='typeset -i'
ll='ls -al'
local=typeset
ls=/usr/bin/ls
lx='ls -l'
nohup='nohup '
...
$ unalias lx
$ alias
autoload='typeset -fu'
cat=/usr/bin/cat
command='command '
false='let 0'
functions='typeset -f'
hash='alias -t -'
history='fc -l'
integer='typeset -i'
ll='ls -al'
local=typeset
ls=/usr/bin/ls
nohup='nohup '
...
```

Siehe auch *alias*

uname Basisdaten über das aktuelle Betriebssystem ausgeben (return system name)

uname schreibt Informationen über das aktuelle Betriebssystem auf die Standard-Ausgabe.

Syntax **uname**[␣option]...

Keine Option angegeben
Der Name des Betriebssystems wird ausgegeben, z.B. *POSIX-BC*.

option

-a (a - all) Es werden Informationen ausgegeben über den Namen des Betriebssystems, den Knotennamen der Anlage, die Versionsnummer des Betriebssystems, den Nachtragsstand der Betriebssystemsversion, den Maschinentyp und über die Prozessor-Bezeichnung, z.B. *POSIX-BC D016ZE07 09.0A A43 BS2000 /390*

-m (m - machine type) Der Name des Maschinentyps wird ausgegeben, z.B. *BS2000*.

-n (n - node) Der Knotenname des Betriebssystems wird ausgegeben. Unter diesem Namen kann das System innerhalb eines Netzes angesprochen werden, z.B. *D016ZE07*.

-p (p - processor type) Die Prozessor-Familie des Rechners, mit dem der Benutzer aktuell arbeitet, wird ausgegeben, z.B. */390*.

-r (r - release) Die Versionsnummer des Betriebssystems wird ausgegeben, z.B. *09.0A*.

-s (s - system) Der Name des Betriebssystems wird ausgegeben. Unter diesem Namen ist das Betriebssystem auf der lokalen Installation bekannt, z.B. *POSIX-BC*.

-v (v - version) Die installierte POSIX-Version wird ausgegeben, z.B. *A43*.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *uname*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Variablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Sie möchten sich über den Knotennamen des Betriebssystems und über die Versionsnummer des Betriebssystems informieren lassen:

```
$ uname -nr
D016ZE07 09.0A
```

Siehe auch *uname()* [4]

uncompress Komprimierte Dateien dekomprimieren (expand compressed data)

uncompress dekomprimiert Dateien, die mit *compress* komprimiert wurden. Die Dateien werden wieder in ihren Originalzustand zurückgebracht. Wenn der aufrufende Prozess die geeigneten Privilegien hat, bleiben Eigentümer, Zugriffsrechte sowie Zugriffs- und Änderungsdatum unverändert.

Falls möglich, wird jede angegebene Datei ersetzt durch eine Datei gleichen Namens ohne *.Z*-Suffix.

Eine Dekomprimierung wird nicht durchgeführt, wenn
- die anzulegende Datei bereits existiert und *uncompress* im Hintergrund abläuft
- die angegebene Datei mit einem zu hohen Wert für *maxbits* (siehe *compress -b*) komprimiert wurde, den Ihr aktueller Rechner nicht verarbeiten kann
- auf die komprimierte Datei Verweise bestehen

Syntax **uncompress**[␣**-cfv**][␣datei...]

Keine Option angegeben
Die angegebenen Dateien werden dekomprimiert.

Optionen

-c *uncompress* schreibt die dekomprimierten Daten auf die Standard-Ausgabe. Es werden keine Dateien verändert oder angelegt. *uncompress -c* ist in seiner Funktion identisch mit *zcat*.

-f (f - force) *datei.Z* wird dekomprimiert, auch wenn bereits eine Datei mit Namen *datei* existiert. *datei* wird überschrieben.

-f nicht angegeben:
uncompress fragt nach, ob die existierende Datei überschrieben werden soll oder nicht. Diese Nachfrage erfolgt jedoch nicht, wenn *uncompress* im Hintergrund abläuft. Die angegebene Datei wird nicht dekomprimiert.

-v (v - verbose) *uncompress* gibt eine Meldung über die erfolgreiche Dekomprimierung aus:
`datei.Z: -- replaced with datei`

datei
Name einer mit *compress* komprimierten Datei. Sie können mehrere Dateien angeben. Jeder Dateiname muss ein *.Z*-Suffix enthalten, das Sie jedoch beim *uncompress*-Aufruf nicht angeben müssen. Die dekomprimierte Datei erhält den Namen *datei*. *datei.Z* wird nach erfolgreicher Dekomprimierung gelöscht (außer bei *-c*). Zugriffsrechte, Zugriffs- und Änderungsdatum bleiben unverändert.

Fehler

datei.Z: not in compressed format
Die angegebene Datei liegt nicht in komprimiertem Datenformat vor oder ist ein Dateiverzeichnis.

datei.Z: compressed with xxbits, can only handle yybits
Die angegebene Datei wurde mit einem zu hohen Wert für *maxbits* (siehe *compress -b*) komprimiert. Der Rechner, an dem Sie *uncompress* aufgerufen haben, kann diesen Wert nicht verarbeiten. Komprimieren Sie die Datei nochmals mit dem Wert, der in der Fehlermeldung mit *yy* angegeben wird oder mit einem niedrigeren Wert.

datei.Z: -- has xx other links: unchanged
Auf die angegebene Datei bestehen *xx* Verweise. Diese Datei wird nicht dekomprimiert.

uncompress: corrupt input
Das Signal SIGSEGV (Segmentation Violation; Adressfehler wegen unerlaubten Segmentzugriffs) wurde empfangen, was normalerweise bedeutet, dass die Eingabedatei beschädigt ist.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *uncompress*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Datei *topsecret.Z* soll dekomprimiert werden. Im gleichen Dateiverzeichnis existiert bereits eine Datei namens *topsecret*, die überschrieben werden kann.

```
$ ls -l
total 62
-rw-------  1  felix    gruppe1    4862 Mar 12 10:16 topsecret
-rw-------  1  felix    gruppe1   26326 Feb 08 10:27 topsecret.Z

$ uncompress -v topsecret.Z
topsecret already exists; do you wish to overwrite topsecret (y or n)? y
topsecret.Z: -- replaced with topsecret

$ ls -l
total 138
-rw-------  1  felix    gruppe1   69845 Feb 08 10:27 topsecret
```

Siehe auch *compress*, *zcat*

unexpand Leerzeichen in Tabulatorzeichen umwandeln (convert spaces to tabs)

Das Kommando *unexpand* kopiert Dateien oder die Standard-Eingabe auf die Standard-Ausgabe. Dabei werden Leerzeichen am Beginn jeder Zeile in die maximale Anzahl von Tabulatorzeichen umgewandelt, gefolgt von der Mindestzahl von Leerzeichen, die zum Auffüllen der Zeile bis zu den Spaltenpositionen erforderlich sind, die ursprünglich von den umgewandelten Zeichen ausgefüllt waren.

Standardmäßig wird alle acht Spaltenpositionen ein Tabulatorstopp gesetzt. Alle Backspace-Zeichen werden auf die Ausgabe kopiert und verringern dabei den Spaltenzähler zur Berechnung der Tabulatorstopposition jeweils um 1. Der Spaltenzähler kann dabei nicht kleiner als eins werden.

Syntax **unexpand**[␣**-a**|␣**-t**␣tablist][␣datei...]

-a Nicht nur die Leerzeichen am Anfang jeder Zeile werden umgewandelt, sondern auch alle Folgen von zwei oder mehr Leerzeichen unmittelbar vor einem Tabulatorstopp werden in die maximale Anzahl von Tabulatorzeichen gefolgt von der Mindestzahl von Leerzeichen umgewandelt, die zum Auffüllen der Zeile bis zu den Spaltenpositionen erforderlich sind, die ursprünglich von den umgewandelten Leerzeichen ausgefüllt waren.

-t␣tablist
Gibt die Tabulatorstopps an. Das Argument *tablist* muss aus einer oder mehreren Zahlen in aufsteigender Reihenfolge bestehen, die durch Leerzeichen oder Kommas getrennt werden. Eine durch Leerzeichen getrennte Liste muss dabei in Anführungszeichen gesetzt werden. Wenn nur eine Zahl angegeben ist, werden die Standard-Tabulatorstopps nicht alle 8 Spaltenpositionen gesetzt, sondern alle *tablist* Spaltenpositionen. Sind mehrere Zahlen angegeben, werden die Tabulatorstopps an den angegebenen Spaltenpositionen gesetzt.

Jede Tabulatorstopposition N muss ein ganzzahliger Wert größer null sein, und die Angaben müssen unbedingt in aufsteigender Reihenfolge erfolgen. Dies bedeutet, dass beim Springen mit der Tabulatortaste vom Anfang der Ausgabezeile zu Position N die nächste Zeichenausgabe in der (N+1)ten Spaltenposition in der Zeile erfolgt.

Für Zeichen, die sich an einer Position hinter der letzten Position, die in einer Liste mit mehreren Tabulatorstopps definiert wurde, befinden, erfolgt keine Umwandlungen von Leer- in Tabulatorzeichen.

Bei Angabe von *-t* ist die Umwandlung von Leerzeichen nicht auf die führenden Leerzeichen beschränkt. *-a* wird in diesem Fall ignoriert.

datei
Die Datei, deren Leerzeichen durch Tabulatorzeichen ersetzt werden sollen.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *unexpand*:

LANG Gibt einen Standardwert für die Internationalisierungsvariablen an, die nicht gesetzt oder null sind. Ist *LANG* nicht gesetzt oder null, wird der entsprechende Standard-Wert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als wäre keine der Variablen definiert.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d.h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation von Byte-Folgen als Zeichen fest (z.B. Einzelbytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien). Außerdem wird die internationale Umgebung für die Verarbeitung von Tabulator- und Leerzeichen sowie für die Angabe der Spalten festgelegt, die jedes Zeichen auf einem Ausgabegerät mit konstant breiter Schriftart einnimmt.

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest.

NLSPATH Legt die Position der Meldungskataloge für die Verarbeitung von *LC_MESSAGES* fest.

Standard-Ausgabe (stdout)

Die Standard-Ausgabe entspricht den Eingabedateien, wobei jedoch Leerzeichen in Tabulatorzeichen umgewandelt wurden.

Hinweis Das standardmäßige Verhalten von *unexpand*, das nur führende Leerzeichen berücksichtigt, kann in manchen Anwendungsfällen unerwünscht sein. Benutzer, die immer alle Leerzeichen in einer Datei umwandeln möchten, sollten mittels Alias-Definition eine Version von *unexpand* erstellen, die immer mit den Optionen *-a* oder *-t␣8* aufgerufen wird.

Siehe auch *expand*, *tabs*

uniq Mehrfache Zeilen suchen (report or filter out repeated lines in a file)

uniq sucht in einer Datei nach aufeinanderfolgenden gleichen Zeilen, schreibt die Datei auf die Standard-Ausgabe und lässt dabei die Wiederholungen weg. Nur bei aufeinanderfolgenden Zeilen können Übereinstimmungen festgestellt werden, d.h die Eingabedatei muss sortiert sein.

Syntax

Format 1: uniq[␣**-c**|␣**-d**|␣**-u**][␣**-n**][␣**+m**][␣eingabe_datei[␣ausgabe_datei]]

Format 2: uniq[␣**-c**|␣**-d**|␣**-u**][␣**-f**␣feld][␣**-s**␣zeichen][␣eingabe_datei[␣ausgabe_datei]]

Die beiden Formate werden gemeinsam beschrieben, da die Option *-n* in Format 1 äquivalent zu der Option *-f␣feld* in Format 2 und die Option *+m* in Format 1 äquivalent zu der Option *-s␣zeichen* in Format 2 ist.

Keine Option angegeben
eingabe_datei wird ausgegeben und Wiederholungen werden weggelassen.

option

-c Alle Zeilen werden ohne Wiederholungen mit einer Dezimalzahl am Zeilenanfang ausgegeben. Die Zahl gibt an, wie oft die entsprechende Zeile in *eingabe_datei* nacheinander vorkommt. *uniq* ignoriert zusätzlich gesetzte Optionen *-u* oder *-d*.

-d Nur die in *eingabe_datei* mehrfach vorkommenden Zeilen werden jeweils einmal ausgegeben.

-u Nur die Zeilen werden ausgegeben, die in *eingabe_datei* nicht wiederholt vorkommen.

-n Die ersten *n* Zeichen ab Zeilenanfang werden beim Vergleichen der Zeilen nicht berücksichtigt.
Wird die Option *-n* zusammen mit der Option *-m* verwendet, so werden die ersten *n* Zeichen nach dem *m*-ten Feld beim Vergleich nicht berücksichtigt. Leerzeichen nach dem *m*-ten Feld werden nicht ignoriert, sondern müssen im Wert *n* berücksichtigt werden.

-n nicht angegeben:
Die Zeilen werden ab Zeilenanfang bzw. ab Anfang von Feld *m+1* verglichen.

Die Option *-n* ist äquivalent zu der Option *-f␣feld* in Format 2.

+m Die ersten *m* Felder ab Zeilenanfang, zusammen mit vor einem Feld stehenden Tabulator- oder Leerzeichen, werden beim Vergleichen der Zeilen nicht berücksichtigt. Ein Feld ist eine nichtleere Zeichenfolge, die durch ein Tabulator- oder Leerzeichen vom Nachbarfeld getrennt ist.

+m nicht angegeben:
Die Zeilen werden ab Zeilenanfang bzw. ab Zeichen *n+1* verglichen.

Die Option *+m* ist äquivalent zu der Option *-s␣zeichen* in Format 2.

eingabe_datei
Name der Datei, die untersucht werden soll.

eingabe_datei nicht angegeben:
uniq liest von der Standard-Eingabe.

ausgabe_datei
Name der Datei, in welche die Ausgabe geschrieben werden soll.

ausgabe_datei nicht angegeben:
uniq schreibt auf die Standard-Ausgabe.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *uniq*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) sowie die Zeichenklassen und die Zeichenkonvertierung.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Sie möchten eine Datei nach gleichen Zeilen durchsuchen, unabhängig davon, wo diese in der Datei stehen. Für jede dieser Zeilen ist auszugeben, wie oft sie vorkommt.

```
$ sort datei | uniq -c
```

Beispiel 2 Sie möchten die 10 häufigsten Wörter der Datei *text* ausgeben lassen.

```
$ cat text \
>    | sed 's/  */ /g' \
>    | tr ' ' '\n' \
>    | sed '/^$/d' \
>    | sort \
>    | uniq -c \
>    | while read N W; do printf "%06d %s\n" $N "$W"; done \
>    | sort -r \
>    | head -n 10
```

Erläuterung:

- *sed* erzeugt aus *text* eine Liste, in der ein oder mehrere Leerzeichen durch ein Leerzeichen ersetzt werden.
- *tr* ersetzt in dieser Liste Leerzeichen durch Neue-Zeile-Zeichen.
- *sed* entfernt leere Zeilen aus dieser Liste.
- *sort* sortiert diese Liste nach EBCDIC.
- *uniq -c* gibt alle Zeilen ohne Wiederholungen aus und schreibt vor jede die Häufigkeit ihres Auftretens.
- Die *while*-Schleife ersetzt die Häufigkeit durch eine 6-stellige Zahl mit führenden Nullen.
- *sort -r* sortiert diese Häufigkeitsliste rückwärts, d.h. die häufigste Zeile steht in der ersten Zeile.
- *head* gibt die ersten 10 Zeilen dieser Liste aus.

Siehe auch *comm*, *sort*

unset Shell-Variablen oder Shell-Funktionen aus der Umgebung löschen (unset values and attributes of variables and functions)

Das in die POSIX-Shell *sh* eingebaute Kommando *unset* löscht die angegebene Shell-Funktion oder Shell-Variable aus der aktuellen Umgebung.

Die Variablen *IFS, MAILCHECK, PATH, PS1* und *PS2* können mit *unset* nicht aus der Umgebung gelöscht werden.

Syntax **unset**[␣**-f**][␣**-v**]␣name␣...

-f Ist die Option *-f* angegeben, bezieht sich *name* auf eine Shell-Funktion.

-v Ist die Option *-v* angegeben, bezieht sich *name* auf eine Shell-Variable. Readonly-Shellvariablen können nicht gelöscht werden.

Ist weder *-f* noch *-v* angegeben, bezieht sich *name* auf eine Shell-Variable. Existiert keine Variable mit dem angegebenen Namen, dann ist nicht sichergestellt, ob eine eventuell vorhandene Funktion gleichen Namens gelöscht wird.

name
Name der Shell-Variablen oder der Shell-Funktion, die aus der aktuellen Umgebung gelöscht werden sollen. Sie können mehrere Namen angeben, jeweils getrennt durch ein Leerzeichen.

Endestatus

0 alle Shell-Variablen oder Shell-Funktionen wurden erfolgreich gelöscht.
>0 mindestens eine Shell-Variable oder Shell-Funktion konnte nicht gelöscht werden.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *unset*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Shell-Funktion *ll* löschen:

```
$ type ll
ll is a function
ll(){
ls -al $* | pg
}
$ unset ll
$ ll
ll not found
```

Siehe auch *set*

usp Dynamisches Setzen von POSIX-Steuerparametern

Dieses Kommando kann nur vom Superuser aufgerufen werden. Das Kommando *usp* hat folgende Funktionen:

- Anzeige der aktuellen Werte aller POSIX-Steuerparameter (Option *-s*).
- Modifizieren der aktuellen Werte folgender POSIX-Steuerparameter: *DBLPOOL*, *FORCEDTERM*, *HDSTNI*, *HDPTNI*, *HEAPSZ*, *MAXTIMERC*, *MAXUP*, *NOSTTY*, *NOTTY* und *NPROC*.
  Abhängig von der angegebenen Option (*-p* oder *-P)* ist die Modifikation nur bis zur Beendigung des POSIX-Subsystems oder auch beim nächsten POSIX-Neustart gültig.
- Prüfen, ob eine Wertangabe für einen ausgewählten Steuerparameter erlaubt ist (Option *-c*).

Syntax

usp␣-h

usp␣-s[␣parameter]

usp␣-c␣parameter **-v␣**wert

usp␣-p␣parameter **-v␣**wert

usp␣-P␣parameter **-v␣**wert

Optionen

-h (h - help) Die Syntax des Kommados *usp* und eine Liste aller POSIX-Steuerparameter werden ausgegeben. Neben dem Namen und der Bedeutung der POSIX-Steuerparameter enthält diese Liste auch die Angabe, ob die Modifikation eines Parameters durch das Kommando *usp* unterstützt wird (`SUPPORTED`) oder nicht (`unsupported`).

-s (s - show) Der aktuelle Wert des mit *parameter* angegebenen Steuerparameters wird ausgegeben. Ist *parameter* nicht angegeben werden die aktuellen Werte **aller** (der unterstützten und der nicht unterstützten) Steuerparameter ausgegeben.

parameter
Name des Steuerparameters. Bei den Optionen *-c*, *-p* und *-P* können nur die Steuerparameter *DBLPOOL*, *FORCEDTERM*, *HDSTNI*, *HDPTNI*, *HEAPSZ*, *MAXTIMERC*, *MAXUP*, *NOSTTY*, *NOTTY* oder *NPROC* angegeben werden, deren Modifikation durch das Kommando *usp* unterstützt wird. Bei der Option *-s* können **alle** Steuerparameter angegeben werden. Die Angabe kann in Groß- oder Kleinschreibung erfolgen. Die Information, ob die Modifikation eines Steuerparameters durch das Kommando *usp* unterstützt wird oder nicht, erhalten Sie auch mit `usp -h`.

-c (c - check) Prüft, ob der mit der Option *-v* angegebene Wert innerhalb der für *parameter* erlaubten Grenzen liegt (Minimalwert ≤ *wert* ≤ Maximalwert).

-p Der aktuelle Wert des mit *parameter* angegebenen Steuerparameters wird durch den mit der Option *-v* angegebene Wert ersetzt, sofern dieser Wert gültig ist (siehe Option *-c*). Für folgende Steuerparameter muss zusätzlich die Bedingung (*wert* > aktueller Wert) erfüllt sein: *HDSTNI*, *HDPTNI*, *HEAPSZ*, *MAXUP*, *NOSTTY*, *NOTTY* und *NPROC*. Die Änderung gilt nur für den aktuellen POSIX-Subsystemlauf. Nach einem Neustart des POSIX-Subsystems gilt wieder der Wert aus der POSIX-Informationsdatei SYSSSI.POSIX-BC.<version>.

Eine Modifikation des Steuerparameters *DBLPOOL* wird erst beim nächsten Aufruf des Kommandos *posdbl* wirksam. Die Inhalte des globalen Programm-Caches werden nicht gesichert. Zur Veränderung der Größe des globalen Programm-Caches ist daher eine Kombination der Kommandos *usp* und *posdbl* notwendig, Näheres dazu siehe POSIX-Handbuch „Grundlagen für Anwender und Systemverwalter“ [1].

-P (wie Option *-p*). Zusätzlich wird der neue Wert in die POSIX-Informationsdatei SYSSSI.POSIX-BC.<version> eingetragen und gilt somit auch nach einem Neustart des POSIX-Subsystems.

Bevor die Änderung in der POSIX-Informationsdatei durchgeführt wird, wird eine Sicherungskopie mit dem Namen SYSSSI.POSIX-BC.<version>.SAVE.<datum>.<uhrzeit> angelegt.

-v wert
(v -value) Angabe des Wertes, der gesetzt oder geprüft werden soll.

Hinweis

Es wird empfohlen, die geänderten Werte in die SYSSSI-Datei einzutragen (z.B. mit der Option *–P*) und beim nächsten Hochfahren zu nutzen. Damit entfällt die erneute dynamische Anpassung während des Systemlaufs.

Bei Vergrößerung und Nutzung des Maximalwerts *HDPTNI* (Anzahl lokale Dateisysteme), muss der neue Wert beim nächsten POSIX-Neustart in der POSIX-Informationsdatei SYSSSI.POSIX-BC.<version> stehen: Falls mehr lokale Dateisysteme in den Verwaltungstabellen (z.B. `/etc/vfstab`) eingetragen sind, als es der Wert in der POSIX-Informationsdatei zulässt, wird der Start des POSIX-Subsystems mit Fehler abgebrochen.

Alle dynamischen Änderungen von Steuerparametern werden mittels Logging in POSIX dokumentiert. Jede durchgeführte Änderung eines Wertes wird an der Konsole dokumentiert. Zusätzlich werden alle Änderungen in einer POSIX-Datei `/var/adm/messages` fortgeschrieben.

uudecode Datei nach der Übertragung per mailx decodieren (decode a binary file)

uuencode und *uudecode* werden zusammen benutzt, um eine Text- oder Binärdatei über *mailx* zu versenden. Sie können die Datei nicht nur direkt, sondern auch über eine *mailx*-Kette von jeweils unmittelbar benachbarten Systemen senden.

uudecode liest eine codierte Datei und stellt die ursprüngliche Datei wieder her. Die Datei erhält den Namen und die Zugriffsrechte, die in der Codierungsinformation angegeben sind (siehe *uuencode*).

Für die codierte Datei müssen Sie Schreibrecht besitzen.

uudecode arbeitet mit der Benutzernummer, die *uucp* zugewiesen wird. In einem Dateiverzeichnis, in dem für „Andere“ kein Schreibrecht gesetzt ist, kann deshalb für *uudecode* die Ausführung verweigert werden.

Weitere Informationen siehe *uuencode*.

Syntax **uudecode**[␣datei]

datei
Datei, die decodiert werden soll.

datei nicht angegeben:
uudecode liest von der Standard-Eingabe.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *uudecode*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Der Benutzer *theo* hat eine Datei über *mailx* in codierter Fassung erhalten:

```
$ mailx
From kmdo Wed Mar 11 14:42 MET 2009
Content_Length

begin 744 proz
M9F]R(&YA;64@*:6X@*@ID;R @:68@6R M9" B)&YA;64B(%T*(" @('!H)&X@
M)&-H;R B*&1V>BD@)&YA;64B" @("!;'-E(&%C:&\@(B!@;',@+7,@)&YA
1;65@(@H@(" @)FD*9&]N9OHB

end
? q
```

In der ersten Zeile des codierten Textes stehen hinter dem Wort *begin* die Zugriffsrechte (744) und der Name (proz) der Datei. Diese soll nun wieder in ihren ursprünglichen Zustand versetzt werden. Sie trägt dann den angegebenen Namen *proz*:

```
$ mailx | uudecode
$ cat proz
for name in *
do  if [ -d "$name" ]
    then echo "(dvz) $name"
    else echo " `ls -s $name`"
    fi
done
```

Siehe auch *mailx*, *uuencode*

uuencode Datei für die Übertragung per mailx codieren (encode a binary file)

uuencode und *uudecode* werden zusammen benutzt, um eine Text- oder Binärdatei über *mailx* zu versenden. Sie können die Datei nicht nur direkt, sondern auch über eine *mailx*-Kette von jeweils unmittelbar benachbarten Systemen senden.

uuencode erhält als Eingabe eine Binärdatei oder Daten von der Standard-Eingabe, erzeugt daraus eine codierte Version und schreibt diese auf die Standard-Ausgabe. Zur Codierung werden nur druckbare ASCII-Zeichen verwendet. Damit können z.B. 8-bit-Daten auch über Systeme gesendet werden, die nicht 8-bit-transparent sind. In die Codierung eingeschlossen sind die Zugriffsrechte der Datei und der ferne Dateiname, damit die Datei nach dem Senden wiederhergestellt werden kann.

Syntax

uuencode[␣ausgangsdatei]␣zieldatei

ausgangsdatei
: Datei, die codiert werden soll.

  ausgangsdatei nicht angegeben:
  uuencode liest von der Standard-Eingabe.

zieldatei
: Name der Zieldatei. Hier geben Sie einen Pfadnamen an, der sich auf den fernen Rechner bezieht.

Hinweis

uuencode und *uudecode* sollten folgendermaßen eingesetzt werden:

Sie rufen *uuencode* auf:

```
uuencode [ausgangsdatei] zieldatei | mailx system1!system2!..!benutzer
```

Die codierte Datei wird dann an den Benutzer *benutzer* auf dem fernen System gesendet. Dieser ferne Benutzer muss Schreibrecht für die Datei besitzen. Er kann dann die Datei mit *uudecode* decodieren.

Die codierte Datei, die *uuencode* erzeugt, ist eine normale Textdatei. Sie können sie mit jedem beliebigen Editor editieren, um die Zugriffsrechte oder den Namen der Zieldatei zu ändern.

Durch die Codierung wird die Ausgangsdatei um 35% größer (aus 3 Bytes werden 4 plus Kontroll-Information); dadurch dauert die Übertragung länger.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung *uuencode*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten und Eingabedateien).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Datei *brief* soll in codierter Fassung an den Benutzer *theo* geschickt werden, der auf dem Rechner *roland* arbeitet. Die Datei soll dort nach dem decodieren den Namen *proz* haben:

```
$ uuencode brief proz | mailx roland!theo
```

Siehe auch *mailx*, *uudecode*

uuname Namen des Systems auflisten (list names of known systems)

uuname gibt den lokalen Systemnamen auf Standardausgabe aus.

Syntax **uuname**[␣**-l**]

-l gibt den lokalen Systemnamen des lokalen Systems im Format

„%s\n“, <system name>

aus. Dieser Name kann sich von dem Systemnamen, den das Kommando *uname -n* ausgibt, unterschieden.

-l nicht angegeben:
uuname gibt den Endestatus 0 zurück.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung von *uuname*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Meldungen fest, die in die Standard(fehler)ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Endestatus abfragen und lokalen Systemnamen ausgeben

```
$ uuname
$ echo $?
0
$ uuname -l
D016ZE07
```

Siehe auch *uname*

vi Bildschirmorientierter Editor (screen oriented (visual) display editor)

Der *vi* (visual) ist ein bildschirmorientierter Texteditor.

Den vi (außer im Line-Mode) können nur die Benutzer verwenden, die sich über *rlogin* Zugang zur POSIX-Shell verschafft haben.

Bei Aufruf von *vi* auf einem Blockterminal wird automatisch der Editor *ex* gestartet.

Aufbau dieser Beschreibung

Die Beschreibung des *vi* gliedert sich in acht Teile:

Kommandoübersicht (Auszug)

In der folgenden Übersicht sind die gebräuchlichsten *vi*-Kommandos in Funktionsgruppen zusammengefasst. Im Abschnitt *Kommandos* finden Sie eine ausführliche Beschreibung in alphabetischer Reihenfolge.

- **vi laden**

vi ↵	*vi* mit leerem Editorpuffer aufrufen
vi *datei* ↵	*vi* mit der zu editierenden *datei* laden

- **vi beenden**

ZZ	*vi* verlassen, Editorpuffer in die editierte Datei schreiben; wie *:wq* ↵
:w ↵	Editorpuffer in editierte Datei schreiben
:w *datei* ↵	Editorpuffer in *datei* schreiben
:q ↵	*vi* verlassen falls Änderungen gesichert
:q! ↵	*vi* verlassen, Änderungen gehen verloren
:wq ↵	*vi* verlassen, Editorpuffer in die editierte Datei schreiben; wie *ZZ*

- **Schreibmarke positionieren**

h	ein Zeichen nach links
j	in der gleichen Spalte eine Zeile vor
+	eine Zeile vor, an Zeilenanfang
↵	eine Zeile vor, an Zeilenanfang
k	in der gleichen Spalte eine Zeile zurück
-	eine Zeile zurück, an Zeilenanfang
l	ein Zeichen nach rechts
H	Bildschirmanfang
M	Bildschirmmitte
L	Bildschirmende
0	Bildschirmrand links
l	Bildschirmrand links
$	letztes Zeichen der Zeile
^	erstes Zeichen der Zeile, das nicht Leerzeichen oder Tabulator ist
w	Wortanfang des nächsten Wortes

e	Wortende des nächsten Wortes
b	Wortbeginn des vorhergehenden Wortes
W	Wortanfang des nächsten Wortes (Langwort)
E	Wortende des nächsten Wortes (Langwort)
B	Wortbeginn des vorhergehenden Wortes (Langwort)

- **Fenster positionieren**

CTRL **d**	halben Bildschirm vorwärts
CTRL **u**	halben Bildschirm zurück
CTRL **f**	einen Bildschirm vorwärts
CTRL **b**	einen Bildschirm zurück
CTRL **e**	eine Zeile vorwärts
CTRL **y**	eine Zeile zurück
z ↵	aktuelle Zeile an Bildschirmanfang
z.	aktuelle Zeile an Bildschirmmitte
z-	aktuelle Zeile an Bildschirmende
1G	Dateianfang
n**G**	Zeile n auf Bildschirmmitte
G	Dateiende

- **Textteile löschen und zurückholen**

x	löscht Zeichen an der Position der Schreibmarke
X	löscht Zeichen links von der Schreibmarke
dd	löscht aktuelle Zeile
dG	löscht ab aktuelle Zeile bis Dateiende
dH	löscht ab Bildschirmanfang bis einschließlich der aktuellen Zeile
D	löscht Zeilenrest ab Schreibmarke
u	gelöschten Text aus Standardpuffer zurückholen
U	aktuelle Zeile wieder herstellen
p	Inhalt des Puffers hinter aktuelle Position der Schreibmarke schreiben
P	Inhalt des Puffers vor aktuelle Position der Schreibmarke schreiben

- **Textteile ändern**

r	ersetzt aktuelles Zeichen durch nächste Eingabe
R	ab aktuellem Zeichen Text überschreiben
o	nach aktueller Zeile Leerzeile einfügen
O	vor aktueller Zeile Leerzeile einfügen
a	Text nach aktuellem Zeichen einfügen
A	Text am Zeilenende anfügen
i	Text an der aktuellen Schreibmarkenposition einfügen
I	Text am Zeilenanfang einfügen

- **Kopieren**

Y	aktuelle Zeile merken
yy	aktuelle Zeile merken
Yp	aktuelle Zeile verdoppeln
yyp	aktuelle Zeile verdoppeln
:10,13**t** 20	Zeilen *10* bis *13* ab neuer Zeile *21* einfügen
:.t.	aktuelle Zeile verdoppeln
:t.	aktuelle Zeile verdoppeln
:r *datei*	Inhalt von *datei* hinter aktuelle Zeile kopieren

- **Übertragen**

3**dd**	*3* Zeilen löschen, einschließlich der aktuellen Zeile
p	gespeicherte Zeilen einfügen
"a3**dd**	*3* Zeilen löschen und in Puffer a ablegen
"ap	gespeicherte Zeilen nach aktueller Zeile einfügen

- **Suchen und ersetzen**

/xyz	im ganzen Editorpuffer nach *xyz* vorwärts suchen
/xyz␣␣**/**	im ganzen Editorpuffer nach *xyz* und 2 Leerzeichen suchen
:s/xyz**/**abc**/**	in aktueller Zeile *xyz* durch *abc* ersetzen
:/alt**/s/**alt**/neu**	erst *alt* suchen, dann *alt* durch *neu* ersetzen
:1,$s/alt**/**neu**/g**	im ganzen Editorpuffer *alt* durch *neu* ersetzen

Einführung

Der *vi* stellt auf dem Bildschirm in der Regel 23 Zeilen der zu editierenden Datei dar. Sie lassen sich Ausschnitte der Datei anzeigen, indem Sie dieses Fenster verschieben. *vi* bietet Ihnen eine breite Palette von Möglichkeiten zur Textedition. Sie können:

– Text erstellen, ändern, kopieren und löschen

– die Schreibmarke mittels einfacher Kommandos positionieren (z.B. auf den Anfang oder das Ende eines Wortes, einer Zeile, eines Satzes, eines Absatzes oder der Datei)

– Textmuster mit regulären Ausdrücken suchen und ersetzen

– eine Subshell aufrufen

– mit einem POSIX-Kommando einen Bereich der aktuellen Datei bearbeiten

– mehrere Dateien während einer Sitzung bearbeiten und Text von einer Datei in eine andere kopieren

– eine Datei wiederherstellen, die durch eine Unterbrechung während einer Sitzung verlorenging

Der *vi* ist eine erweiterte Form des Zeileneditors *ex* (siehe *ex*). Sie können zwischen diesen beiden Editoren hin- und herwechseln und vom *vi* aus *ex*-Kommandos ausführen.

Modi des vi

Der *vi* stellt Ihnen verschiedene Modi zur Verfügung, in denen Sie arbeiten können:

– *vi*-Kommandomodus

– *vi*-Eingabemodus

– *vi*-Zeilen-Kommandomodus

– *ex*-Kommandomodus

– *ex*-Eingabemodus

Nach Aufruf des *vi* befinden Sie sich immer im *vi*-Kommandomodus. Von dort aus können Sie in die anderen Modi wechseln. Je nach Modus interpretiert der *vi* Ihre Tastatureingabe unterschiedlich.

vi-Kommandomodus

Nachdem Sie den *vi* aufgerufen haben, befindet sich der *vi* im *vi*-Kommandomodus. In diesem Modus ist keine Texteingabe möglich. Jede Eingabe über die Tastatur wird sofort als *vi*-Kommando interpretiert, ohne dass sie am Bildschirm ausgegeben wird. Handelt es sich dabei um ein zulässiges Kommando, wird es ausgeführt, und das Ergebnis ist sofort am Bildschirm sichtbar.

vi-Eingabemodus

Im Eingabemodus ergänzen oder ändern Sie Text im Editorpuffer. Den Eingabemodus schalten Sie ein, wenn Sie eines der *vi*-Kommandos *A, a, C, c, I, i, O, o, S, s* oder *R* eingeben (siehe im Abschnitt *Kommandos, Kommandos des vi-Kommandomodus*). Alle folgenden Eingabezeichen, auch verschiedene nicht druckbare Zeichen, werden auf den Bildschirm und in den Editorpuffer geschrieben.
Im Eingabemodus können Sie die meisten *vi*-Kommandos nicht verwenden. Es gibt jedoch einige *vi*-Kommandos, die im Eingabemodus eine spezielle Bedeutung haben. Diese sind:

ESC
: Eingabemodus verlassen, zurück in den *vi*-Kommandomodus.

DEL
: Eingabemodus abbrechen, zurück in den *vi*-Kommandomodus.

↵ Neue Zeile.

CTRL @
: Zuletzt eingegebenen Text nochmals eingeben. Dieses Kommando geben Sie ein, unmittelbar nachdem Sie in den Einfügemodus gewechselt haben. *vi* fügt dann den beim letzten Einfügen eingegebenen Text ein, falls dieser aus nicht mehr als 128 Zeichen besteht. Danach wird automatisch zurück in den Kommandomodus geschaltet.

CTRL **d**
Wenn Sie eine automatische Einrückung definiert haben, können Sie mit CTRL d den linken Rand um eine Tabulatorposition nach links positionieren. Die Schrittbreite können Sie mit der Option *shiftwidth* festlegen. Dieses Kommando ist nur am Anfang einer neuen Eingabezeile und mit gesetzter Option *autoindent* sinnvoll.

CTRL **h**
Ein Zeichen zurück.

CTRL **t**
Einen Tabulatorschritt nach rechts, entsprechend *shiftwidth*. Dies ist nur am Anfang einer neuen Eingabezeile möglich.

CTRL **v**
Nicht druckbare Zeichen einfügen.

CTRL **w**
Ein Wort zurück.

**** Escape für Korrekturzeichen.

vi-Zeilen-Kommandomodus

Im *vi*-Zeilen-Kommandomodus können Sie *vi*-, *ex*- und POSIX-Kommandos in der Statuszeile eingeben. Den Zeilen-Kommandomodus schalten Sie durch Eingabe eines Doppelpunktes : oder eines Schrägstriches / oder eines Fragezeichens ? ein. Eine weitere Möglichkeit ist die Eingabe eines Ausrufezeichens !, gefolgt von einer Positionieranweisung auf eine Zeile. Kommandoeingaben im Zeilen-Kommandomodus schließen Sie mit ↵ oder ESC ab. Mit DEL brechen Sie die Eingabe ab.

Achtung!
Im *vi*-Zeilen-Kommandomodus löscht ESC ein gerade eingegebenes Kommando nicht, sondern schickt es ab. Das Kommando wird also ausgeführt! Im *vi*-Zeilen-Kommandomodus müssen Sie DEL verwenden, um ein eingegebenes Kommando zu löschen.

: Die Eingabe eines Doppelpunktes : im *vi*-Kommandomodus bewirkt, dass die nachfolgende Eingabe bis zum nächsten ESC oder ↵ als *ex*-Kommando interpretiert wird. (Alle *ex*-Kommandos sind unter *ex*, *ex-Kommandos* beschrieben.) Nach Eingabe des Doppelpunktes springt die Schreibmarke in die unterste Zeile, die Statuszeile. Der Doppelpunkt und das eingegebene Kommando werden nun angezeigt. Sie können fast alle *ex*-Kommandos verwenden; ausgenommen sind davon jedoch solche *ex*-Kommandos, die den *ex* in den *ex*-Eingabemodus schalten würden.
Für den Namen der im Editorpuffer befindlichen Datei kann das Prozentzeichen % als Kurzbezeichnung verwendet werden. Das Nummernzeichen # steht für den Namen der in der gleichen Sitzung vorher bearbeiteten Datei (siehe *ex*, *Aktuelle und sekundäre Datei*).

Beispiel

```
:! diff # % ↵
```

vi gibt auf die Standard-Ausgabe die Unterschiede zwischen der vorher editierten Datei und der aktuellen Datei aus. Keine der beiden Dateien wird verändert.

/ oder **?**

Nach Eingabe eines Schrägstriches / bzw. Fragezeichens ? springt die Schreibmarke in die Statuszeile. Geben Sie nun einen regulären Ausdruck an, nach dem der *vi* vorwärts (bei Einagbe von /) oder rückwärts (bei Eingabe von ?) sucht. Mit dem *vi*-Kommando *n* wiederholen Sie die Suche in Suchrichtung, mit *N* entgegen der Suchrichtung.

! Nach Eingabe eines Ausrufezeichens ! im *vi*-Kommandomodus springt die Schreibmarke nicht sofort in die Statuszeile. Sie tut dies erst, wenn Sie eine *vi*-Positionieranweisung auf eine Zeile gegeben haben. Am Anfang der Statuszeile wird ein Ausrufezeichen ausgegeben. Geben Sie als Positionieranweisung ein Ausrufezeichen ein, gilt als Position die aktuelle Zeile. Ihre Positionieranweisung wird nicht angezeigt. Sie können nun ein POSIX-Kommando eingeben, das Sie mit ↵ oder ESC freigeben. Der Bereich von der aktuellen Zeile bis zur angegebenen Position wird, falls das POSIX-Kommando von der Standard-Eingabe liest, zur Eingabe des POSIX-Kommandos und nach Ende des Kommandos durch die Ausgabe (Standard-Ausgabe und Standard-Fehlerausgabe) des POSIX-Kommandos ersetzt. Falls das Kommando nichts ausgibt (z.B. *true*), wird der angegebene Bereich gelöscht, also durch die leere Zeichenkette ersetzt.

ex-Kommandomodus

Den *ex*-Kommandomodus schalten Sie ein, indem Sie im *vi*-Kommandomodus das Kommando *Q* eingeben. Damit verlassen Sie den bildschirmorientierten *vi* und gelangen in den zeilenorientierten *ex*. Sie wechseln in den *vi* zurück, indem Sie im *ex*-Kommandomodus das Kommando *vi*↵ eingeben.

ex-Eingabemodus

Den *ex*-Eingabemodus schalten Sie ein, indem Sie im *ex*-Kommandomodus *a, i* oder *c* eingeben. Alles, was Sie in diesem Modus eingeben, wird in den Editorpuffer geschrieben. Sie beenden diesen Modus, indem Sie eine einzelne Zeile mit einem Punkt in der ersten Spalte eingeben.

Die meisten *ex*-Kommandos können Sie auf zwei Arten eingeben:

– vom *vi*-Kommandomodus aus, indem Sie das *vi*-Kommando Doppelpunkt : eingeben, gefolgt von dem gewünschten *ex*-Kommando

– vom *ex*-Kommandomodus aus

Programmaufruf

Nach dem Aufruf befindet sich der *vi* im *vi*-Kommandomodus. Falls Sie in den *vi*-Eingabemodus gewechselt haben, kehren Sie durch Eingabe von ESC (bzw. DEL) wieder in den *vi*-Kommandomodus zurück.

Syntax **vi**[␣option]...[␣datei]...

option

-t␣markierung
(t - tag) Die Datei mit *markierung* wird zum Editieren aufgerufen. Der Editor positioniert die Zeile mit der Definition der Markierung in die Mitte des Bildschirms. Im gleichen Dateiverzeichnis muss eine Datei *tags* vorhanden sein, die Suchzeichenketten für die Definitionen enthält. Diese Option hilft u.a. C-Programmierern, die beim Editoraufruf sofort auf die Definition einer Funktion oder eines Makros positionieren wollen. Die dafür benötigte Markierungsdatei *tags* muss vorher mit dem Kommando *ctags* erzeugt worden sein.

-l (l - lisp) Der Lisp-Modus wird eingeschaltet. Der Text wird so eingerückt, dass er ein für Lisp-Programme typisches Aussehen bekommt.
Die folgenden *vi*-Kommandos haben für Lisp eine eigene Bedeutung: *(*, *)*, *{*, *}*, *[[* und *]]*.

-r[␣datei]
(r - recover) Mit dieser Option *-r* können Sie Ihre Sitzung wiederherstellen, falls das System oder der *vi* während der vorhergehenden Sitzung abgestürzt sind.
Bei einem Absturz des Systems oder des *vi*-Editors werden die Änderungen, die nur im Editorpuffer stehen, nicht in die Datei geschrieben. Das Betriebssystem POSIX versucht jedoch, den Inhalt des Puffers zu retten, indem eine Kopie des Pufferinhalts angelegt wird, sofern dies möglich ist. *datei* wird mit den Änderungen, die Sie vor dem Absturz gemacht haben, in den *vi*-Puffer geholt. Sie können nun die Datei weiter editieren oder die Änderungen in eine beliebige Datei schreiben.

-L (L - List) Nach einem Absturz des Systems oder des Editors wird eine Liste aller geretteten Dateien ausgegeben.

-R (R - Read-only) *datei* wird nur zum Lesen geöffnet. Damit können Sie ein versehentliches Überschreiben von *datei* verhindern.

Achtung!
Die Datei kann mit dem *ex*-Kommando *w!* dennoch überschrieben werden.

-w␣n
(w - window) Der Bildschirm wird aus *n* Textzeilen aufgebaut.

-w␣n nicht angegeben:
Die Bildschirmlänge wird durch die Umgebungsvariable *LINES* festgelegt. Ist diese nicht definiert, wird der Wert für die Anzahl der in der Datei *terminfo* zur Definition der Bildschirmausgabe herangezogen

-c␣kommando

(c - command) Mit dieser Option können Sie beim Aufruf des *vi* ein *ex*-Kommando ausführen. Wenn Sie beim Aufruf des *vi* ein Positionier-Kommando angeben, steht die gewünschte Zeile danach in der Mitte des Bildschirms. Wenn Sie ein *ex*-Kommando ausführen, wird nach Ausführung des *ex*-Kommandos auf die letzte Zeile der Datei positioniert, sofern das *ex*-Kommando (z.B. ein Suchvorgang) nichts anderes bewirkt.

kommando nicht angegeben:
Der *vi* gibt eine *usage*-Meldung aus und bricht ab.

kommando angegeben:
kommando kann entweder eine Zeilenangabe (*n*) sein, ein Suchkommando (*/muster*, siehe Abschnitt *Kommandos des vi-Kommandomodus*) oder ein sonstiges Kommando des *ex* ("*ex-kommando*" oder '*ex-kommando*'). Es wird beim Aufruf des *vi* ausgeführt:

n — *n* ist eine ganze Zahl. *vi* positioniert auf die *n*-te Zeile der Datei.

/muster — *vi* positioniert auf die Zeile, die *muster* enthält. *muster* ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*). Falls *muster* Sonderzeichen enthält, müssen Sie die Sonderzeichen jeweils mit einem Gegenschrägstrich \ entwerten oder *muster* in Hochkommas '*muster*' einschließen, damit die Shell die Sonderzeichen nicht interpretiert.

'ex-kommando' oder "ex-kommando"
ex-kommando kann ein beliebiges *ex*-Kommando sein. Das *ex*-Kommando muss in Hochkommas oder Anführungszeichen eingeschlossen werden, damit es von der Shell nicht interpretiert wird. Falls nicht schon durch das *ex*-Kommando positioniert wird, positioniert der *vi* auf die letzte Zeile der Datei.

Beispiel

Nach dem Aufruf des *vi* mit

```
vi -c "set number showmode" termine
```

wird die Datei *termine* geöffnet, die Schreibmarke auf die letzte Zeile der Datei positioniert, und auf dem Bildschirm werden Zeilennummern sowie der *vi*-Modus angezeigt (vgl. das Beispiel bei *ex*).

datei

Name der zu editierenden Datei. Sie können mehrere Dateien angeben. *vi* beginnt dann mit der ersten angegebenen Datei. Mit dem *ex*-Kommando *n* können Sie in die nächste Datei wechseln (siehe auch *ex*-Kommandos *rew* und *ar*).

datei nicht angegeben:
Wenn Sie keine Datei angeben, arbeiten Sie zunächst mit einem leeren Editorpuffer. Den Inhalt dieses Puffers können Sie dann mit dem *ex*-Kommando *w datei* in die Datei *datei* schreiben.

Akustisches Signal (nicht für Blockterminals)

Sie erhalten ein akustisches Signal wenn

– Sie im Kommandomodus die Taste ESC oder DEL drücken

– Sie ein unerlaubtes Kommando verwenden

– *vi* ein Signal SIGINT erhält

Erhält der *vi* ein Signal SIGINT (DEL) während der Texteingabe oder während der Eingabe eines Kommandos in der untersten Bildschirmzeile, so wird die Eingabe beendet (bzw. das Kommando abgebrochen) und der Editor wieder in den Kommandomodus geschaltet. Der Empfang eines Signals SIGINT im Kommandomodus bewirkt ein akustisches Signal.

Bildschirmaufbau

Ein zeichenorientierter Bildschirm einer Datensichtstation stellt in der Regel 24 Zeilen und 80 Spalten dar. Davon werden standardmäßig 23 Bildschirm-Zeilen für die Darstellung des Textes der zu editierenden Datei verwendet. In xterm-Sitzungen können mehr als 24 Zeilen dargstellt werden. Die Anzahl der vom *vi* verwendeten Bildschirmzeilen wird in diesem Fall automatisch festgestellt.

Die unterste Zeile auf dem Bildschirm ist die Status- und Kommandozeile. Sie dient zur Eingabe von Kommandos und zur Ausgabe von Informationen und Meldungen des *vi*.

Falls eine Textzeile breiter ist als der Bildschirm, wird sie in der nächsten Bildschirmzeile fortgesetzt.

vi verwendet zur Kennzeichnung des Zeilenstatus zwei Sonderzeichen, die in der ersten Spalte der jeweiligen Zeile sichtbar sind:

~ Die Tilde ~ kennzeichnet Zeilen, die auf dem Bildschirm, aber nicht im Editorpuffer vorhanden sind. Das sind Zeilen hinter dem Dateiende. Wenn sich z.B. die letzte Textzeile der Datei in der Mitte des Bildschirms befindet, dann beginnen alle dahinterliegenden Bildschirmzeilen mit ~. Sobald Sie in die erste so gekennzeichnete Zeile Text eingeben, verschwindet die Tilde. Wenn Sie eine neue oder leere Datei mit *vi* aufrufen, enthalten alle Bildschirmzeilen bis auf die erste und die letzte (Statuszeile) dieses Zeichen.

@ Der Klammeraffe @ kennzeichnet Zeilen, die der *vi* auf dem Bildschirm nicht ordnungsgemäß darstellen kann. Dies kann zwei Ursachen haben:
- Eine Textzeile kann sich über mehrere Zeilen auf dem Bildschirm erstrecken. Deshalb können Textzeilen im Editorpuffer vorhanden sein, aber der Platz auf dem Bildschirm nach der letzten, noch vollständig dargestellten Textzeile reicht nicht mehr für eine weitere ganze Textzeile.
- Zeilenweises Löschen aus einer Datei kann dazu führen, dass die Bildschirmdarstellung nicht mehr mit dem tatsächlichen Inhalt des Editorpuffers übereinstimmt. In diesem Fall können Sie den Bildschirm mit dem *vi*-Kommando CTRL L neu aufbauen.

Die letzte Zeile auf dem Bildschirm ist die Statuszeile. Sie dient zur:
- Eingabe von *vi*-Kommandos, die mit **/**, **?**, **!** oder **:** beginnen
- Ausgabe von Meldungen
- Ausgabe von *ex*-Kommandos
- Anzeige des Modus (Kommandomodus wird nicht angezeigt)

Arbeitsweise

Der *vi* arbeitet immer mit einem Editorpuffer. Bei Beginn einer *vi*-Sitzung wird eine Kopie der Datei, die Sie bearbeiten, im Editorpuffer angelegt. Falls Sie keinen Dateinamen angeben oder die Datei noch nicht existiert, beginnen Sie mit einem leeren Editorpuffer. Während einer Sitzung erfolgen alle Änderungen am Editorpuffer. Die Orginaldatei wird erst verändert, wenn Sie den Inhalt des Editorpuffers in die Datei schreiben. Dies geschieht durch explizites Sichern während der Sitzung (mit dem *ex*-Kommando *w*) oder beim Verlassen des *vi* (mit den *ex*-Kommandos *wq*, *x* oder dem *vi*-Kommando *ZZ*). Sie können den *vi* auch verlassen, ohne die Orginaldatei zu verändern (durch das *ex*-Kommando *q!*). Falls Sie eine neue Datei erstellen wollen, wird die Datei erst angelegt, wenn der Editorpuffer in die Datei geschrieben wird.

Editorpuffer sichern und vi verlassen

Um den Editorpuffer zu sichern oder den *vi* zu verlassen, muss sich der *vi* im *vi*-Kommandomodus befinden.

Mit folgendem *ex*-Kommando (siehe *Arbeitsweise des vi, ex-Kommandos eingeben*) sichern Sie den Inhalt Ihres Editorpuffers in die editierte oder eine angegebene Datei:

:w[␣datei]
(w - write, *ex*-Kommando) Der Inhalt des Editorpuffers wird in die angegebene Datei gesichert.

datei nicht angegeben:
Der Inhalt des Editorpuffers wird in die editierte Datei geschrieben.

Um den *vi* zu verlassen, haben Sie folgende Möglichkeiten:

:q (q - quit, *ex*-Kommando) *vi* verlassen. Funktioniert nur, falls noch keine Änderungen am Editorpuffer vorgenommen wurden oder der geänderte Editorpuffer in eine Datei gesichert wurde.

:q! (q - quit, *ex*-Kommando) *vi* verlassen. Am Editorpuffer vorgenommene Änderungen gehen verloren.

:x (x - exit, *ex*-Kommando) *vi* verlassen und den geänderten Editorpuffer in die editierte Datei schreiben.

ZZ (*vi*-Kommando)
vi verlassen und den geänderten Editorpuffer in die editierte Datei schreiben.

:wq[␣datei] (wq - write and quit, *ex*-Kommando) *vi* verlassen und den Editorpuffer in die angegebene Datei *datei* schreiben.

datei nicht angegeben:
Der Inhalt des Editorpuffers wird in die editierte Datei geschrieben.

Editorpuffer des vi

Der *vi* verfügt über

– einen temporären Puffer

– 9 nummerierte Puffer

– 26 alphabetische Puffer

Auf den Inhalt dieser Puffer können Sie im *vi*-Kommandomodus durch *vi*-Kommandos zugreifen.

Temporärer Puffer

Der *vi* sichert in den temporären Puffer die letzte Änderung, die Sie am Editorpuffer vornehmen. Wenn Sie z.B. eine Zeile gelöscht haben, dann enthält der Puffer die gelöschte Zeile. Die *vi*-Kommandos *U* und *u* (*undo*) benutzen diesen Puffer. Mit *u* können Sie Ihr letztes Kommando, das den Editorpuffer geändert hat, rückgängig machen. *u* macht auch *u* wieder rückgängig.

Mit den *vi*-Kommandos *P* und *p* (*put*) können Sie ebenfalls auf den temporären Puffer zugreifen, falls das letzte *vi*-Kommando ein Kopier- (*y*), ein Lösch- (*d*) oder sonstiges Änderungskommando war (außer *R* und *r*).

P und *p* kopieren den Inhalt des temporären Puffers vor (*P*) bzw. hinter (*p*) die aktuelle Zeile. Im Gegensatz zu *U* (*u*) ändert *P* (*p*) den temporären Puffer nicht. Dadurch können Sie, indem Sie *P* (*p*) mehrfach verwenden, den gleichen Text auch mehrfach an verschiedene Stellen der Datei einfügen.

Achtung!
Bei jedem Kommando, das den Editorpuffer ändert, wird der Inhalt des temporären Puffers überschrieben. Deshalb können Sie den temporären Puffer nicht verwenden, um Text von einer Datei in eine andere zu kopieren.

Zwischen einem Kopier-, Lösch- oder Änderungskommando und dem *vi*-Kommando *P* (*p*) dürfen Sie nur die *vi*-Kommandos zur Positionierung der Schreibmarke (siehe *Positionieranweisungen*) verwenden, da andere Kommandos den Inhalt des temporären Puffers verändern.

Wenn *vi* im Eingabemodus ist und Sie eine der Pfeil-Tasten benutzen, funktioniert nach Benutzung der Pfeil-Tasten das *vi*-Kommando *U* (*u*) nicht.

Nummerierte Puffer

Wenn Sie eine oder mehrere Zeilen löschen (*vi*-Kommando *d*), wird der gelöschte Text in den ersten von neun nummerierten Puffer kopiert. Wenn Sie nun nochmals Zeilen löschen, wird der Inhalt von Puffer 1 in Puffer 2 kopiert, und die zuletzt gelöschten Zeilen werden wiederum in Puffer 1 gesichert. In Puffer 1 steht also immer der zuletzt gelöschte Text, in Puffer 2 der vorletzte ... und in Puffer 9 der neunt-letzte gelöschte Text.

Den Inhalt der nummerierten Puffer können Sie mit den *vi*-Kommandos *P* und *p* zurückholen. Die nummerierten werden mit Anführungszeichen " und der Nummer des Puffers angesprochen, zum Beispiel: `"4P`

Dieses *vi*-Kommando fügt den Inhalt des Puffers 4 vor die aktuelle Zeile (bzw. aktuelle Position der Schreibmarke) ein.
Mit dem *vi*-Kommando Punkt . holen Sie den Inhalt des Puffers mit der nächsthöheren Nummer zurück, vorausgesetzt, Sie haben mit Ihrem letzten *vi*-Kommando Text aus einem nummerierten Puffer zurückgeholt. Nach obigem Beispiel holen Sie also den Inhalt des Puffers 5 zurück. Der Inhalt der nummerierten Puffer geht bei Dateiwechsel (mit den *ex*-Kommandos *n* oder *e*) nicht verloren.

Alphabetische Puffer

Mit einem Kopier- (*y*) oder Löschkommando (*d*) können Sie Text in einen der 26 bezeichneten Puffer schreiben. Die Puffer bezeichnen Sie mit den Kleinbuchstaben *a-z* oder mit den Großbuchstaben *A-Z*. Klein- und Großbuchstaben sprechen den gleichen Puffer an, mit folgendem Unterschied:
Verwenden Sie Kleinbuchstaben, wird der alte Pufferinhalt durch den neuen überschrieben, der alte wird also gelöscht. Verwenden Sie Großbuchstaben, wird der alte Pufferinhalt nicht überschrieben, sondern der neue Text wird an den alten angehängt. Der *vi* ändert den Inhalt eines bezeichneten Puffers während einer Editor-Sitzung solange nicht, bis Sie den Pufferinhalt ausdrücklich überschreiben. Der *vi* schreibt Text in einen bezeichneten Puffer, wenn Sie ein Anführungszeichen " eingeben, gefolgt von dem Namen des Puffers und einem der *vi*-Kommandos *d* oder *y*.

Beispiele

`"A10dd`
Aktuelle und neun weitere Zeilen (10) löschen (dd) und an den Inhalt in Puffer A anhängen.

`"by4w`
Das aktuelle Wort und drei folgende (*4w*) in den Puffer *b* kopieren (*y*).

`"cCZeichenkette` ESC
Die aktuelle Zeile von der Position der Schreibmarke aus bis zum Zeilenende mit *Zeichenkette* überschreiben (*C*) und den überschriebenen Text in Puffer *c* sichern.

`"Ap`
Den Text aus Puffer A zurückholen und hinter die aktuelle Zeile bzw. Position einfügen (*p*).

Positionieranweisungen

Der *vi* bietet eine große Menge von Kommandos und Suchkommandos zur Positionierung der Schreibmarke. Eine Positionieranweisung wird im *vi*-Kommandomodus eingegeben. In der folgenden Übersicht sind die wichtigsten Positionieranweisungen zusammengestellt. Die Begriffe Absatz, Abschnitt, Satz, Wort und Langwort sind im Abschnitt *Kommandos*, *Definitionen* erklärt.

Zeichenweise positionieren

[zahl] CTRL **h** um *zahl* Zeichen nach links

[zahl]**h** um *zahl* Zeichen nach links

[zahl]**l** um *zahl* Zeichen nach rechts

0 auf den Zeilenanfang

^ auf das erste Zeichen in der Zeile außer Leerzeichen und Tabulator

[zahl]**|** auf die erste (*zahl*-te) Spalte der Zeile

$ auf das letzte Zeichen in der Zeile

[zahl]**f**zeichen auf *zahl*-tes Zeichen *zeichen* vorwärts

[zahl]**F**zeichen auf *zahl*-tes Zeichen *zeichen* rückwärts

[zahl]**t**zeichen vor *zahl*-tes Zeichen *zeichen* vorwärts

[zahl]**T**zeichen hinter *zahl*-tes Zeichen *zeichen* rückwärts

Wortweise positionieren

[zahl]**b** auf den Anfang des *zahl*-ten vorhergehenden Wortes

[zahl]**B** auf den Anfang des *zahl*-ten vorhergehenden Langwortes

[zahl]**e** auf das Ende des *zahl*-ten Wortes

[zahl]**E** auf das Ende des *zahl*-ten Langwortes

[zahl]**w** auf den Anfang des *zahl*-ten Wortes

[zahl]**W** auf den Anfang des *zahl*-ten Langwortes

Zeilenweise positionieren

[zahl]**k** um *zahl* Zeilen in der gleiche Spalte nach oben

[zahl]**j** um *zahl* Zeilen in der gleichen Spalte nach unten

[zahl] CTRL **J** um *zahl* Zeilen in der gleichen Spalte nach unten

[zahl] ↵	um *zahl* Zeilen nach unten, auf erstes Zeichen außer Leerzeichen und Tabulatorzeichen
[zahl]**+**	um *zahl* Zeilen nach unten, auf erstes Zeichen außer Leerzeichen und Tabulatorzeichen
[zahl]**-**	um *zahl* Zeilen nach oben, auf erstes Zeichen außer Leerzeichen und Tabulatorzeichen
[zeile]**G**	auf Zeile Nummer *zeile* bzw. Dateiende (*zeile* nicht angegeben)
[zahl]**$**	auf das letzte Zeichen der *zahl*-ten Zeile

Satz-, absatz- und abschnittsweise positionieren

[zahl]**(**	auf den Anfang des *zahl*-ten vorherigen Satzes
[zahl]**)**	auf den Anfang des *zahl*-ten Satzes
[zahl]**{**	zurück zum Anfang des *zahl*-ten Absatzes
[zahl]**}**	auf den Anfang des *zahl*-ten Absatzes
[zahl]**[[**	zurück auf die *zahl*-te vorangehende Abschnittsgrenze
[zahl]**]]**	auf die *zahl*-te Abschnittsgrenze
%	zur korrespondierenden Klammer in C-Quellprogrammen

Bildschirmbezogen positionieren

[zahl]**H**	auf das erste Zeichen in der *zahl*-ten Textzeile auf dem Bildschirm
M	auf die Bildschirmmitte
[zahl]**L**	auf das erste Zeichen in der *zahlt*-letzten Textzeile auf dem Bildschirm
[zeile]**z+**	aktuelle Zeile oder Zeile mit Nummer *zeile* an den oberen Bildschirmrand
[zeile]**z.**	aktuelle Zeile oder Zeile mit Nummer *zeile* auf die Bildschirmmitte
[zeile]**z-**	aktuelle Zeile oder Zeile mit Nummer *zeile* an den unteren Bildschirmrand

Muster und Marken positionieren

/Muster	auf *Muster* vorwärts
?Muster	auf *Muster* rückwärts
'marke	auf markierte Zeile (Zeilenanfang)
` marke	auf *marke* positionieren

Kommandos

Nach dem Aufruf befindet sich der *vi* im *vi*-Kommandomodus. Falls Sie in den *vi*-Eingabemodus gewechselt haben, kehren Sie durch Eingabe von ESC (bzw. DEL) wieder in den *vi*-Kommandomodus zurück.

ex-Kommandos können Sie, wie bereits oben beschrieben, entweder vom *vi*-Kommandomodus aus oder vom *ex*-Kommandomodus aus eingeben. Im Folgenden sind nur die *vi*-Kommandos beschrieben. Die *ex*-Kommandos finden Sie beim Kommando *ex*.

Im *vi*-Kommando-Modus wird mit ESC und DEL ein teilweise eingegebenes Kommando gelöscht. Wenn weder der Editor sich im Eingabemodus befindet, noch ein Kommando teilweise eingegeben wurde, so löst ESC bei Zeichenterminals ein akustisches Signal aus.

Achtung!
Im *vi*-Zeilen-Kommandomodus löscht ESC ein gerade eingegebenes Kommando nicht, sondern schickt es ab. Das Kommando wird also ausgeführt! Im *vi*-Zeilen-Kommandomodus müssen Sie DEL verwenden, um ein eingegebenes Kommando zu löschen.

Sofern nicht anders angegeben, gelten die Kommandos nur im *vi*-Kommandomodus und haben im Eingabemodus keine spezielle Bedeutung.

Definitionen

Absatz
: Ein Absatz ist festgelegt durch den Wert der *ex*-Option *paragraphs*. Leere Zeilen und solche, an denen ein Abschnitt beginnt, sind ebenfalls Absatzgrenzen.

Abschnitt
: Ein Abschnitt ist festgelegt durch den Wert der *ex*-Option *sections*. Zeilen, die mit Formularvorschub CTRL L oder der öffnenden geschweiften Klammer { beginnen, zählen ebenfalls als Abschnittsgrenze.

  Ist die *ex*-Option *lisp* gesetzt, dann zählt jede öffnende runde Klammer (am Anfang einer Zeile ebenfalls als Abschnittsgrenze.

Langwort
: Ein Langwort ist eine Folge von Zeichen, die kein Leer-, Tabulator- oder Neue-Zeile-Zeichen enthalten. Vergleiche *Wort*.

Satz

Ein Satz wird beendet durch eines der folgenden Satzende-Zeichen:

Punkt .
Ausrufezeichen !
Fragezeichen ?

jeweils gefolgt von *zwei* Leerzeichen (um Verwechslungen mit einer Abkürzung zu vermeiden) oder dem Neue-Zeile-Zeichen. Zwischen dem Satzende-Zeichen und den Leerzeichen bzw. dem Neue-Zeile-Zeichen kann eine beliebige Anzahl von schließenden runden oder eckigen Klammern),], Anführungszeichen " und Gegenhochkommas ` stehen.

Wort

Ein Wort ist entweder eine Folge von alphanumerischen Zeichen oder eine Folge von Sonderzeichen. Ein Wort wird abgeschlossen von einem Leerzeichen, Tabulatorzeichen, Neue-Zeile-Zeichen oder dem nächsten folgenden Wort.
Vergleiche *Langwort*.

Beispiele

((!!++;-.	ist ein Wort
Me&You2	sind drei Worte (Me,&,You2), aber ein Langwort

Zeichenkette

Eine beliebige Folge von Zeichen.

Control-Kommandos

Um ein Control-Kommando einzugeben, drücken Sie gleichzeitig die Taste CTRL und die Taste mit dem gewünschten Kommando. Einigen der Control-Kommandos kann eine positive ganze Zahl *zahl* vorangestellt werden. Die eckigen Klammern in der Beschreibung bedeuten, dass die Angabe von *zahl* optional ist. Während der Eingabe von *zahl* dürfen Sie die Taste CTRL nicht drücken. Die eingegebene *zahl* wird nicht angezeigt, sondern beeinflusst nur die Wirkung des nachfolgenden Kommandos. Haben Sie eine falsche Zahl angegeben, dann können Sie die Eingabe durch ESC rückgängig machen. Durch *zahl* geben Sie an:

– eine Größe bei einem Positionierkommando, z.B. rollt

  5 CTRL U
  den Bildschrirm fünf Zeilen zurück.

– einen Wiederholfaktor, z.B. blättert

  4 CTRL B
  den Schirm um vier Seiten zurück.

Der Standardwert für *zahl* ist 1, es sei denn es ist ausdrücklich anders angegeben.

CTRL **@**
Im *vi*-Eingabemodus:
Wenn Sie dieses als erstes und einziges Zeichen eingeben, wird der zuletzt eingegebene Text an der Stelle eingefügt, an der Sie CTRL @ eingeben. Danach wird automatisch in den Kommandomodus zurückgewechselt.

CTRL @ funktioniert nur bis maximal 127 Zeichen.

[zahl] CTRL **b**
(b - backward) Die Bildschirmausgabe wird um 21 Bildschirm-Zeilen zurückgeblättert. Mit *zahl* können Sie angeben, um wieviele Bildschirmseiten zurückgeblättert werden soll. Falls möglich, werden auf die neue Bildschirmseite zwei Zeilen der vorhergehenden Bildschirmseite übernommen.

[zahl] CTRL **d**
Im *vi*-Kommandomodus:
zahl nicht angegeben:
zahl = 11, bzw. der zuletzt bei CTRL d oder CTRL u verwendete Wert von *zahl*

(d - down) Die Bildschirmausgabe wird um ein halbes Fenster vorwärts gerollt. Mit *zahl* kann angegeben werden, um wieviele Textzeilen die Bildschirmausgabe gerollt werden soll; sie wird während Ihrer *vi*-Sitzung für spätere Aufrufe der Kommandos CTRL D und CTRL U gespeichert.

Im *vi*-Eingabemodus:
zahl nicht angegeben:
Im Eingabemodus wird die Schreibmarke entsprechend der *ex*-Option *autoindent* oder CTRL t um *shiftwidth* Spalten nach links bewegt (siehe *ex*, *ex*-Optionen). CTRL d funktioniert nur in einer Zeile, in der noch keine Zeichen stehen.

zahl kann sein: ^ oder *0*

^ CTRL d unterbricht das automatische Einrücken für diese Zeile.

0 CTRL d beendet das automatische Einrücken.

[zahl] CTRL **e**
Die Bildschirmausgabe wird um *zahl* Zeilen vorwärts gerollt. Falls möglich, bleibt die Position der Schreibmarke dabei unverändert.

[zahl] CTRL **f**
(f - forward) Die Bildschirmausgabe wird um 21 Bildschirmzeilen in Vorwärtsrichtung geblättert. Mit *zahl* kann angegeben werden, um wieviele Bildschirmseiten vorwärts geblättert werden soll.

CTRL **g**
In der Statuszeile werden folgende Informationen ausgegeben:
- der Name der aktuellen Datei
- evtl. eine Zustandsangabe *[Modified]*, falls die Datei nach der letzten Sicherung mit dem *ex*-Kommando *w* verändert wurde
- die aktuelle Zeilennummer
- die Gesamtzahl aller Textzeilen im Editorpuffer
- der Prozent-Anteil der aktuellen Zeile an der Gesamtlänge der Datei in Zeilen.

Entspricht dem *ex*-Kommando *f*.

[*zahl*] CTRL **h**
(gleichbedeutend mit dem Korrekturzeichen)
Im *vi*-Kommandomodus:

Die Schreibmarke wird um *zahl* Zeichen nach links bewegt, höchstens bis zum linken Rand des Bildschirms (siehe auch *vi*-Kommando *h*).

Im *vi*-Eingabemodus:
Keine Angabe für *zahl* möglich.

Die Schreibmarke wird um ein Zeichen nach links bewegt, höchstens bis zum Anfang des gerade eingegebenen Textes.

[zahl] CTRL **j**
(gleichbedeutend mit MENU und ↓) Die Schreibmarke wird innerhalb der aktuellen Spalte um *zahl* Textzeilen nach unten bewegt.

Entspricht CTRL n und *j*.

CTRL **l**

Die aktuelle Bildschirmausgabe wird gelöscht und wieder neu aufgebaut. Dieses Kommando können Sie verwenden, wenn Meldungen oder Nachrichten auf dem Bildschirm ausgegeben wurden (z.B. mit *write*).

[zahl] CTRL **m**

(gleichbedeutend mit ↵) Die Schreibmarke wird zum ersten Zeichen in der *zahl*-ten Zeile bewegt, bei dem es sich nicht um ein Leer- oder Tabulatorzeichen handelt.

[zahl] CTRL **n**

Entspricht CTRL j und *j*.

[zahl] CTRL **p**

Die Schreibmarke wird innerhalb der aktuellen Spalte in die *zahl*-te darüberliegende Zeile bewegt.

Entspricht dem *vi*-Kommando *k*.

CTRL **r**

(r - Refresh) Die aktuelle Bildschirmausgabe wird gelöscht und wieder neu aufgebaut. Zuvor gelöschte Zeilen, durch das Platzhalterzeichen @ gekennzeichnet, werden entfernt. Siehe CTRL l.

CTRL **t**

Im *vi*-Eingabemodus:
(t - Tabulator) Tabulatorzeichen einfügen. Die Schrittweite eines Tabulators entspricht dem Wert der *ex*-Option *shiftwidth* (siehe *ex*, *ex-Optionen*). Soll die Schreibmarke diese Leer- oder Tabulatorzeichen nach vorne überspringen, so müssen Sie das Kommando CTRL d verwenden.

[zahl] CTRL **u**

zahl nicht angegeben:
zahl = 11, bzw. der zuletzt bei CTRL d oder CTRL u verwendete Wert der *zahl*

(u - up) Die Bildschirmausgabe wird um ein halbes Fenster rückwärts in Richtung Dateianfang gerollt. Mit *zahl* können Sie angeben, um wieviele Textzeilen die Bildschirmausgabe gerollt werden soll; sie wird während Ihrer *vi*-Sitzung für spätere Aufrufe der Kommandos CTRL d und CTRL u gespeichert.

CTRL **v**

Im *vi*-Eingabemodus:
Ermöglicht, ein nicht druckbares Zeichen in den Text einzufügen (z.B. Escape- oder Control-Zeichen). Die Tastenfolge CTRL v ESC fügt das Zeichen ESC in die Datei ein.

CTRL **w**

Im *vi*-Eingabemodus:
(w - word) Die Schreibmarke wird zum Anfang des vorherigen Wortes bewegt. Sie können dabei den Bereich des gerade eingegebenen Textes nicht verlassen. Die nach links übersprungenen Worte sind nur noch auf dem Bildschirm vorhanden, sollen sie im Editorpuffer stehen, müssen sie nochmals eingegeben werden.

[zahl] CTRL **y**

Die Bildschirmausgabe wird um *zahl* Zeilen rückwärts in Richtung Dateianfang gerollt. Falls möglich, bleibt die Position der Schreibmarke dabei unverändert.

CTRL **[**

(gleichbedeutend mit ESC) Im *vi*-Kommandomodus:
Ein teilweise eingegebenes Kommando wird gelöscht; wurde kein Kommando teilweise eingegeben, so wird ein akustisches Signal ausgegeben.

Im *vi*-Zeilen-Kommandomodus:
Ein gerade eingegebenes Kommando wird abgeschlossen und ausgeführt.

Im *vi*-Eingabemodus:
Der *vi*-Eingabemodus wird beendet.

[ESC]

Siehe CTRL [.

Kommandos des vi-Kommandomodus

Den meisten *vi*-Kommandos können Sie eine ganze Zahl voranstellen. Der Standardwert für diese Zahl ist 1, es sei denn, es ist ausdrücklich anders angegeben. Die eckigen Klammern in der Beschreibung bedeuten, dass die Angabe optional ist. Die eingegebene Zahl wird nicht angezeigt, sondern beeinflusst nur die Wirkung des nachfolgenden Kommandos. Haben Sie eine falsche Zahl angegeben, dann können Sie die Eingabe mit ESC löschen.

Folgende Fälle sind zu unterscheiden:

[zahl]**kommando**
Durch die Angabe von *zahl* wird das Kommando *zahl*-mal ausgeführt. *zahl* ist eine positive ganze Zahl. *zahl* kann nicht 0 sein, da die 0 ein *vi*-Kommando darstellt.

zahl nicht angegeben:
vi nimmt für *zahl* den Wert 1 an.

[zeile]**z**
Der *vi* positioniert die Zeile mit der Zeilennummer *zeile* an den oberen oder unteren Rand bzw. in die Mitte des Bildschirms (siehe ausführliche Beschreibung des *vi*-Kommandos *z*).

zeile nicht angegeben:
vi positioniert die aktuelle Zeile an den oberen oder unteren Rand bzw. in die Mitte des Bildschirms.

[spalte]**|**
Angabe der Spalte, auf die Sie die Schreibmarke positionieren.

Auf die folgenden *vi*-Kommandos kann ein Positionierkommando *position* folgen, um einen Bereich anzugeben, der von dem Kommando bearbeitet wird:

c, d, y, Kleinerzeichen <, Größerzeichen >, Ausrufezeichen ! und Gleichheitszeichen =.

Bei den Kommandos <, >, ! und =, die nur ganze Zeilen bearbeiten, werden alle Zeilen bearbeitet, die sich ganz oder teilweise im angegebenen Bereich befinden.

Bei den Kommandos *c, d* und *y* erstreckt sich der zu bearbeitende Bereich von der aktuellen Position der Schreibmarke (einschließlich) bis zur Position, die die Schreibmarke nach dem Positionierkommando hat. Dabei ist entscheidend, ob z.B. wortweise oder zeilenweise positioniert wurde. Falls zeilenweise positioniert wurde, werden alle Zeilen (einschließlich der aktuellen) als ganze Zeilen bearbeitet. Falls wortweise positioniert wurde, wird der Bereich bis vor das Wort (auf Wortanfang positioniert) bzw. bis hinter das Wort (auf Wortende positioniert) bearbeitet.

[zahl]**a**

(a - append) *vi*-Eingabemodus einschalten. Der eingegebene Text wird hinter der aktuellen Position der Schreibmarke eingefügt. Mit *zahl* können Sie angeben, wieviele Kopien des Textes eingefügt werden sollen; der eingefügte Text darf dann aber nicht länger als eine Textzeile sein.

[zahl]**A**

(A - append) *vi*-Eingabemodus einschalten. Der eingegebene Text wird hinter dem Ende der aktuellen Textzeile eingefügt. Durch *zahl* kann angegeben werden, wieviele Kopien des Textes eingefügt werden sollen; der eingefügte Text darf dann aber nicht länger als eine Textzeile sein. Entspricht *$a*.

[zahl]**b**

(b - back) Die Schreibmarke wird auf den Anfang des *zahl*-ten vorhergehenden Wortes in der aktuellen Zeile bewegt.

[zahl]**B**

(B - back) Die Schreibmarke wird zum Anfang des *zahl*-ten vorhergehenden Langwortes bewegt.

[zahl]**c**position

(c - change) Textbereich überschreiben.

Auf dieses Kommando muss eine Positionieranweisung folgen.

– Wenn sich die angegebene Position noch in der gleichen Textzeile befindet, wird sie durch das Dollar-Zeichen $ markiert, und für den Bereich zwischen der Schreibmarke und $ wird der *vi*-Eingabemodus eingeschaltet. Der Text in diesem Bereich wird durch den Text ersetzt, der nun eingegeben wird.

– Wenn sich die angegebene Position in einer neuen Textzeile befindet, hängt das Verhalten von *c* davon ab, ob Sie zeilenweise oder nicht zeilenweise positioniert haben. Haben Sie zeilenweise positioniert, dann werden alle ganz oder teilweise betroffenen Zeilen gelöscht. Haben Sie nicht zeilenweise, z.B. wortweise, positioniert, dann wird nur der betreffende Bereich gelöscht. In jedem Fall wird der *vi* in den Eingabemodus geschaltet. Der gelöschte Text wird durch den Text ersetzt, der nun eingegeben wird. Der gelöschte Text wird vom *vi* in den temporären Puffer gesichert, d.h. er kann z.B. mit P oder *"1p* wiederhergestellt werden.

zahl

Zahl wirkt als Multiplikator auf den angegebenen Bereich.

[zahl]**cc**

Das Kommando *zahlcc* bewirkt, dass die aktuelle Zeile bzw. *zahl* Zeilen vollständig überschrieben werden, unabhänging von der Position der Schreibmarke in der aktuellen Zeile.

[zahl]**C**
(C - change) Der Rest der aktuellen Zeile wird überschrieben. Entspricht *zahlc$*.

[zahl]**d**position
[zahl]**dd**
(d - delete) Textbereich löschen.

Auf dieses Kommando muss eine Positionierungsanweisung folgen. *d* löscht den Bereich von der aktuellen Position der Schreibmarke bis zum Ende des angegebenen Bereiches. Wenn mehr als ein Teil einer einzigen Zeile gelöscht wird, wird der gelöschte Text in den nummerierten Puffern 1 - 9 gespeichert, d.h. er kann mit *p* wiederhergestellt werden.

zahl Zahl wirkt als Multiplikator auf den angegebenen Bereich.

d Das Kommando *zahldd* löscht die aktuelle bzw. *zahl* Zeilen.

"Puffer[zahl]**d**position
"Puffer[zahl]**dd**
Text löschen und in alphabetischen Puffer sichern.
Der Text wird von der aktuellen Position der Schreibmarke bis zum Ende des angegebenen Bereichs gelöscht, zusätzlich aber in den mit *Puffer* bezeichneten alphabetischen Puffer gesichert. *Puffer* ist ein Groß- oder ein Kleinbuchstabe.

D (D - delete) Der Rest der aktuellen Zeile wird gelöscht. Entspricht *d$*.

[zahl]**e**
(e - end) Die Schreibmarke wird auf das Ende des *zahl*-ten Wortes positioniert.

[zahl]**E**
(E - end) Die Schreibmarke wird auf das Ende des *zahl*ten Langwortes positioniert.

[zahl]**f**zeichen
(f - find) Auf *f* muss ein einzelnes Zeichen folgen. *vi* durchsucht die aktuelle Zeile in Vorwärtsrichtung (nach rechts) nach diesem Zeichen und positioniert die Schreibmarke gegebenenfalls auf das gefundene Zeichen. Wenn ein Wiederholungsfaktor angegeben wird, positioniert *vi* auf das *zahl*-te Zeichen.

Sie wiederholen dieses Suchkommando mit dem Strichpunkt ; in Suchrichtung und mit dem Komma , entgegen der Suchrichtung.

[zahl]**F**zeichen
(F - find) Auf *F* muss ein einzelnes Zeichen folgen. *vi* durchsucht die aktuelle Zeile in Rückwärtsrichtung (nach links) nach diesem Zeichen und positioniert die Schreibmarke gegebenenfalls auf das gefundene Zeichen. Wenn ein Wiederholungsfaktor angegeben wird, positioniert *vi* auf das *zahl*-te Zeichen.

Sie wiederholen dieses Suchkommando mit dem Strichpunkt ; in Suchrichtung und mit dem Komma , entgegen der Suchrichtung.

[zeile]**G**

(G - go to) *vi* positioniert die Schreibmarke auf den Anfang der Zeile mit der Nummer *zeile*.

zeile nicht angegeben:
vi positioniert die Schreibmarke auf die letzte Zeile der Datei.

[zahl]**h**

(h - home) Die Schreibmarke wird um *zahl* Zeichen nach links bewegt, jedoch nur in der aktuellen Zeile

[zahl]**H**

(h - home) Die Schreibmarke wird auf den Anfang der ersten Bildschirmzeile positioniert. Wird *zahl* angegeben, so wird die Schreibmarke zu der Zeile bewegt, die um *zahl* Zeilen vom oberen Rand des Bildschirms entfernt ist. Die Schreibmarke wird in beiden Fällen zum ersten Zeichen in der Zeile bewegt, bei dem es sich nicht um ein Leer- oder Tabulatorzeichen handelt.

[zahl]**i**

(i - insert) *vi*-Eingabemodus einschalten. Der eingegebene Text wird vor der aktuellen Position der Schreibmarke eingefügt. Mit *zahl* können Sie angeben, wieviele Kopien des Textes eingefügt werden sollen. Der eingefügte Text darf dann aber nicht länger als eine Textzeile sein.

[zahl]**I**

(I - insert) *vi*-Eingabemodus einschalten. Der eingegebene Text wird vor dem ersten Zeichen in der aktuellen Zeile eingefügt. Mit *zahl* können Sie angeben, wieviele Kopien des Textes eingefügt werden sollen. Der eingefügte Text darf dann aber nicht länger als eine Textzeile sein.

[zahl]**j**

Die Schreibmarke wird um *zahl* Textzeilen in der gleichen Spalte nach unten bewegt. Entspricht CTRL j und CTRL n.

[zahl]**J**

(J - join) Die nachfolgende Zeile wird an die aktuelle Zeile angehängt, wobei an der Nahtstelle die geeignete Anzahl von Leer- oder Tabulatorzeichen eingefügt wird: ein Leerzeichen zwischen Wörtern, zwei Leerzeichen nach einem Punkt, kein Leerzeichen, wenn das erste Zeichen in der nächsten Zeile eine schließende runde Klammer) ist. Über *zahl* kann die Anzahl der zusammenzufügenden Zeilen angegeben werden.

[zahl]**k**

Die Schreibmarke wird um *zahl* Textzeilen in der gleichen Spalte nach oben bewegt. Entspricht CTRL P.

[zahl]**l**

Die Schreibmarke wird um *zahl* Zeichen nach rechts bewegt, jedoch nur in der aktuellen Zeile.
Entspricht *Leerzeichen*.

[zahl]**L**

(L - last) Die Schreibmarke wird zum ersten Zeichen in der letzten Textzeile auf dem Bildschirm bewegt, bei dem es sich nicht um ein Leer- oder Tabulatorzeichen handelt. Wird *zahl* angegeben, so wird die Schreibmarke zu der Textzeile bewegt, die sich *zahl* Textzeilen oberhalb des unteren Bildschirmrandes befindet.

Beispiel

`d3L` löscht alle Zeilen von der aktuellen Zeile (einschließlich) bis zur dritten Textzeile (einschließlich) oberhalb des unteren Bildschirmrandes.

mmarke

(m - mark) An der aktuellen Position der Schreibmarke wird eine Marke gesetzt. Auf *m* muss ein Kleinbuchstabe *marke* folgen; mit diesem Buchstaben wird die aktuelle Position der Schreibmarke markiert.

Adressieren

` marke Mit Gegenhochkomma ` wird die Schreibmarke auf *marke* bewegt.

’marke Mit Hochkomma ’ wird die Schreibmarke auf das erste Zeichen in der Zeile mit *marke* bewegt.

M (M - middle) Die Schreibmarke wird zum ersten Zeichen in der mittleren Bildschirmzeile bewegt, bei dem es sich nicht um ein Leer- oder Tabulatorzeichen handelt.

n Wiederholt das letzte / bzw. ? Suchkommando.

N Wiederholt das letzte / bzw. ? Suchkommando in entgegengesetzter Suchrichtung.

o (o - open) *vi*-Eingabemodus einschalten. Unter der aktuellen Zeile wird eine Zeile eingefügt.

O (O - open) *vi*-Eingabemodus einschalten. Oberhalb der aktuellen Zeile wird eine neue Zeile eingefügt.

p Text aus temporärem Puffer holen

(p - put) Schreibt den Text aus dem temporären Puffer hinter die aktuelle Position der Schreibmarke. Wenn der Text im Puffer mit den *vi*-Kommandos, die Zeilen bearbeiten (z.B. *dd*, *Y*), gesichert wurde, dann wird der Text als Zeile zwischen der aktuellen Textzeile und der nächsten Textzeile eingefügt.

"puffer**p**
(p - put) Text aus nummeriertem oder alphabetischem Puffer holen

puffer kann sein:
– eine Ziffer von 1-9: der Text wird aus dem entsprechenden nummerierten Puffer geholt
– ein Buchstabe: der Text wird aus dem entsprechenden alphabetischen Puffer geholt.

P
"puffer**P**
(P - put) Ähnlich *p*; der Text wird jedoch vor die aktuelle Position der Schreibmarke bzw. die aktuelle Zeile eingefügt.

Q (Q - quit) *vi* wird beendet, und der *ex*-Kommandomodus wird eingeschaltet. Um in den *vi*-Kommandomodus zurückzukehren müssen Sie *vi*↵ eingeben.

[zahl]**r**zeichen
(r - replace) Das Zeichen, auf dem die Schreibmarke steht, wird durch das nächste eingegebene Zeichen *zeichen* ersetzt. Wird eine Zahl angegeben, so werden *zahl* Zeichen jeweils durch das nächste Zeichen ersetzt.

R (R - replace) Mehrere Zeichen ersetzen. *vi* schaltet auf der aktuellen Position der Schreibmarke in den *vi*-Eingabemodus Die eingegebenen Zeichen werden jedoch nicht eingefügt, sondern überschreiben jeweils das Zeichen unter der Schreibmarke. Mit ESC kann dieser Modus beendet werden.

[zahl]**s**
(s - substitute) Das Zeichen unter der Schreibmarke wird gelöscht, und *vi* wird in den *vi*-Eingabemodus umgeschaltet. Das gelöschte Zeichen wird dann durch den eingegebenen Text ersetzt. Über eine Zahl kann angegeben werden, wieviele Zeichen in der aktuellen Zeile gelöscht werden sollen. Das letzte zu löschende Zeichen wird wie bei *c* durch ein Dollar-Zeichen $ markiert.

Beispiel

```
5sbla bla bla ESC
```

ersetzt das Zeichen unter der Schreibmarke und die vier rechts folgenden Zeichen durch die Zeichenkette *bla bla bla*. Das Gleiche erreichen Sie mit:

```
c5lbla bla bla ESC
```

[zahl]**S**
(S - substitute) *vi*-Eingabemodus einschalten. Ganze Zeilen werden ersetzt (entspricht *cc*). Mit *zahl* kann angegeben werden, wieviele Zeilen ersetzt werden sollen.

[zahl]**t***zeichen*

(t - to) Auf *t* muss ein einzelnes Zeichen folgen. *vi* durchsucht die aktuelle Zeile in Vorwärtsrichtung (nach rechts) nach diesem Zeichen. Wenn das Zeichen gefunden wurde, positioniert der *vi* die Schreibmarke unmittelbar vor dieses Zeichen. Wenn Sie einen Wiederholungsfaktor angeben, positioniert *vi* vor das *zahl*-te Zeichen. Eine Wiederholung des Suchkommandos mit *,* bzw. *;* ist nicht sinnvoll.

[zahl]**T***zeichen*

(T - to) Auf *T* muss ein einzelnes Zeichen folgen. *vi* durchsucht die aktuelle Zeile in Rückwärtsrichtung (nach links) nach diesem Zeichen. Wenn das Zeichen gefunden wurde, positioniert der *vi* die Schreibmarke unmittelbar hinter dieses Zeichen. Wenn Sie einen Wiederholungsfaktor angeben, positioniert *vi* hinter das *zahl*-te Zeichen. Eine Wiederholung des Suchkommandos mit *,* bzw. *;* ist nicht sinnvoll.

u (u - undo) Das letzte Kommando, durch das der temporäre Puffer geändert wurde, wird rückgängig gemacht. Durch ein erneutes *u*-Kommando kann auch das letzte *u*-Kommando rückgängig gemacht werden, usw.

Beispiel

`d3w`	löscht drei Worte ab aktueller Position der Schreibmarke.
`u`	holt die drei Worte wieder zurück
`u`	löscht die drei Worte wieder

Mit *p* und *P* können Sie auf den Inhalt des temporären Puffers zugreifen.

Folgt ein *u*-Kommando auf ein *insert*-Kommando, mit dem mindestens eine neue Textzeile eingefügt wurde, so wird der mit *u* gelöschte Text im nummerierten Puffer 1 gesichert.

U (U - undo) Die letzten Änderungen an der aktuellen Zeile werden rückgängig gemacht. *U* funktioniert nur, solange Sie die Zeile nicht verlassen haben.

[zahl]**w**

(w - word) Die Schreibmarke wird auf den Anfang des *zahl*-ten Wortes bewegt.

[zahl]**W**

(W - word) Die Schreibmarke wird auf den Anfang des *zahl*-ten Langwortes bewegt.

[zahl]**x**

Das Zeichen an der aktuellen Schreibmarkenposition wird gelöscht. Wird *zahl* angegeben, so werden, beginnend bei der aktuellen Position der Schreibmarke, *zahl* Zeichen in der aktuellen Zeile gelöscht.

[zahl]**X**

Das Zeichen vor der Schreibmarke wird gelöscht. Wird *zahl* angegeben, so werden in der aktuellen Zeile *zahl* Zeichen vor der Schreibmarke gelöscht.

[zahl]**y**position
[zahl]**yy**

Text in temporären Puffer sichern

(y - yank) Auf dieses Kommando muss eine Positionieranweisung folgen. Der Text von der aktuellen Position der Schreibmarke bis zum Ende der angegebenen Position wird in den temporären Puffer kopiert.

zahl *zahl* wirkt als Multiplikator auf den angegebenen Bereich.

y Das Kommando *yy* bewirkt, dass die aktuelle Zeile vollständig in den Puffer gesichert wird.

Beispiele

`yw` kopiert von der Position der Schreibmarke bis zum Wortende in den temporären Puffer.

`4y3w` 12 Worte in den temporären Puffer sichern.

`4yy` die aktuelle und drei folgende Textzeilen in den temporären Puffer sichern.

"puffer[zahl]**y**position
"puffer[zahl]**yy**

Text in alphabetischen Puffer sichern

(y - yank) Ist dem Kommando der Name eines alphabetischen Puffers (*"puffer*) vorangestellt, so wird der Text zusätzlich in diesen Puffer gesichert. *puffer* ist ein Großbuchstabe oder ein Kleinbuchstabe.

Beispiele

`"a4yy` die aktuelle und drei folgende Textzeilen in den alphabetischen Puffer *a* sichern.

`"ap` Inhalt von *a* im Anschluss an aktuelle Position ausgeben.

[zahl]**Y**

Zeile(n) in temporären Puffer sichern

(Y - yank) Eine Kopie der aktuellen Zeile bzw. von *zahl* Zeilen wird in den temporären Puffer gesichert (entspricht *yy*).

"puffer[zahl]**Y**

Zeile(n) in alphabetischen Puffer sichern

(Y - yank) Eine Kopie der aktuellen Zeile bzw. von *zahl* Zeilen wird in den mit *puffer* bezeichneten alphabetischen Puffer gesichert. *puffer* ist ein Großbuchstabe oder ein Kleinbuchstabe. Entspricht *"pufferyy*.

[zeile]**z**[zahl]zeichen
/Muster**/z**[zahl]zeichen

Die durch *zeile* bzw. *Muster* bestimmte Textzeile wird auf die durch *zeichen* bestimmte Stelle des Bildschirms positioniert.

zeichen muss auf *z* folgen. *zeichen* kann sein:

↵ oder Plus-Zeichen +
Positionierung auf den oberen Rand des Bildschirms

Punkt . Positionierung auf die Mitte des Bildschirms

Minuszeichen -
Positionierung auf den unteren Rand des Bildschirms

zeile ist die Zeilennummer der Textzeile, die positioniert wird.

zeile nicht angegeben:
Die aktuelle Zeile wird positioniert.

Muster bezeichnet die Zeile, in der *Muster* zum ersten Mal vorkommt.

zahl ist eine ganze Zahl zwischen 1 und 23, die auf *z* folgt. *zahl* gibt die Anzahl der Bildschirmzeilen an, die dargestellt werden sollen.

zahl nicht angegeben:
zahl = 23.

ZZ *vi* beenden. Wenn am Editorpuffer seit der letzten Sicherung (mit dem *ex*-Kommando *w*) Änderungen vorgenommen wurden, dann wird der Inhalt des Editorpuffers in die Datei mit dem aktuellen Dateinamen (siehe *ex*-Kommando *f*) gesichert. Entspricht *ex*-Kommando *x*.

[zahl]**Leerzeichen**

Die Schreibmarke wird um *zahl* Zeichen nach rechts bewegt (höchstens bis zum Zeilenende).
Entspricht *l*.

[zahl]**!**position

Mit Ausrufezeichen ! und einer Positionieranweisung können Sie in den *vi*-Zeilen-Kommandomodus wechseln, um ein POSIX-Kommando einzugeben. Der Bereich von der aktuellen Zeile bis zur angegebenen Zeile wird zur Standard-Eingabe des POSIX-Kommandos und durch die Ausgabe (Standard-Ausgabe und Standard-Fehlerausgabe) des POSIX-Kommandos ersetzt. Die Zeilen werden immer ganz übergeben, nie als Teilzeilen.
Die Schreibmarke springt erst nach der Eingabe der Positionieranweisung in die Statuszeile. Die Eingabe des POSIX-Kommandos muss mit ↵ abgeschlossen werden.

position
Positionieranweisung auf eine Zeile. Positionieranweisungen auf Zeilen sind z.B.:

3j 3 Zeilen nach unten

zeileG auf Zeile mit Zeilennummer *zeile* positionieren

L auf letzte Bildschirmzeile positionieren

position kann auch ein zweites Ausrufezeichen ! sein. Dann wird die aktuelle Zeile als Standard-Eingabe dem POSIX-Kommando übergeben. Dem zweiten Ausrufezeichen kann eine ganze Zahl vorangestellt werden; damit wird angegeben, wieviele Zeilen einschließlich der aktuellen Zeile an das POSIX-Kommando übergeben werden.

zahl
ist ein Wiederholungsfaktor.

Beispiele

`!!wc` ↵
übergibt die aktuelle Zeile als Eingabe an *wc* und ersetzt sie durch die Ausgabe von *wc*.

`!1Gsort` ↵
sortiert den Bereich von der aktuellen Zeile bis Dateianfang.

`3!$tr a b` ↵
ersetzt alle *a* durch *b* in der ganzen aktuellen Zeile, sowie in den beiden folgenden Zeilen.

[zahl]**$**
Mit Dollar-Zeichen $ wird die Schreibmarke zum Ende der aktuellen Zeile bewegt. *zahl* ist ein Wiederholungsfaktor (mit *2$* z.B. wird die Schreibmarke zum Ende der nächsten Zeile bewegt).

" Das Anführungszeichen " wird dem Namen eines alphabetischen oder nummerierten Puffers vorangestellt. Siehe *vi*-Kommandos *d*, *p* und *y*.

% Mit Prozent-Zeichen % wird die Schreibmarke zur korrespondierenden Klammer bewegt. Korrespondierende Klammern sind öffnende und schließende runde Klammern (und), eckige Klammern [und] sowie geschweifte Klammern { und }.

& Mit dem kommerziellen Und & wird das letzte *ex*-Kommando *s* wiederholt. Entspricht dem *ex*-Kommando *&*.

'marke
(Hochkomma)

Schreibmarke auf markierte Zeile bewegen

Mit Hochkomma Marke '*marke* wird die Schreibmarke zum ersten Zeichen der markierten Zeile bewegt, das kein Leer- oder Tabulatorzeichen ist. *marke* ist ein Kleinbuchstabe, der eine Marke bezeichnet, die mit dem *vi*-Kommando *m* gesetzt wurde.

Wird das Hochkomma-Kommando bei einem anderen Kommando als Positionierkommando verwendet, werden alle Zeilen im angegebenen Bereich als ganze Zeilen behandelt - einschließlich der aktuellen und der markierten Zeile.

Beispiel

```
d'a
```

Die Zeilen von der aktuellen Zeile bis zur markierten Zeile einschließlich werden gelöscht.

'' (doppeltes Hochkomma)

Schreibmarke auf vorherige aktuelle Zeile bewegen

Durch das Kommando zweifaches Hochkomma '' wird die Schreibmarke auf den Anfang der Zeile bewegt, in der sie sich vor der letzten Ausführung eines der folgenden *vi*-Kommandos befunden hat.

H, L, M, n, N, %, ', (,), [[,]], ` , {, },

/Muster ↵,
?*Muster* ↵.

Wird das Doppelte-Hochkomma-Kommando bei einem anderen Kommando als Positionierkommando verwendet, so werden alle Zeilen im angegebenen Bereich als ganze Zeilen behandelt - einschließlich der aktuellen und der markierten Zeile.

Beispiel

```
d''
```

Der Bereich von der aktuellen bis zur vorherigen aktuellen Zeile wird gelöscht. Vergewissern Sie sich vorher mit '', wo die vorherige aktuelle Zeile war. Wenn keine vorherige aktuelle Zeile existiert, löscht *vi* von der aktuellen Zeile bis Datei-Anfang. Mit *u* können Sie den gelöschten Text zurück holen.

[zahl]**(**
Mit der öffnenden runden Klammer (wird die Schreibmarke zurück zum Anfang des *zahl*-ten vorhergehenden Satzes bewegt.

Ist die *ex*-Option *lisp* gesetzt, so wird die Schreibmarke an den Anfang eines *lisp s*-Ausdruckes bewegt (siehe *ex*, *ex-Optionen*). Auch der Beginn eines Abschnittes oder Absatzes wird als Beginn eines Satzes interpretiert (siehe { und [[).

[zahl]**)**
Mit der schließenden runden Klammer) wird die Schreibmarke zum Anfang des *zahl*-ten Satzes bewegt.

[zahl]**+**
Mit dem Pluszeichen + wird die Schreibmarke zum ersten Zeichen in der nächsten Zeile bewegt, bei dem es sich nicht um ein Leer- oder Tabulatorzeichen handelt. Mit *zahl* kann angegeben werden, um wieviele Zeilen die Schreibmarke in Vorwärtsrichtung bewegt werden soll (wie bei CTRL m).

[zahl]**,**
Komma , bewirkt die Umkehrung des letzten *vi*-Kommandos (Suche nach einem einzelnen Zeichen): *f*, *F*, *t* oder *T*.
Die Zeile wird in der entgegengesetzten Richtung durchsucht. Durch *zahl* kann der Wiederholungsfaktor angegeben werden.

[zahl]**-**
Mit Minuszeichen - wird die Schreibmarke zum ersten Zeichen in der *zahl*-ten vorhergehenden Zeile bewegt, bei dem es sich nicht um ein Leer- oder Tabulatorzeichen handelt.

[zahl]**.**
Mit Punkt wird das letzte *vi*-Kommando, das den Editorpuffer verändert hat, *zahl*-mal wiederholt.

Wenn Ihr letztes Kommando den Inhalt eines nummerierten Puffers ausgegeben hat, dann gibt ein darauf folgendes Punkt-Kommando den Inhalt des nummerierten Puffers mit der um eins erhöhten Nummer aus usw.

/Muster↵
Mit Schrägstrich / wechseln Sie sofort in den *vi*-Zeilen-Kommandomodus, in dem Sie einen regulären Ausdruck *Muster* angeben können. Die Eingabe von *Muster* beenden Sie mit ↵. *vi* positioniert dann die Schreibmarke vorwärts auf die erste gefundene Zeichenkette, die zu *Muster* passt. Wenn keine passende Zeichenkette gefunden wird, bleibt die Schreibmarke auf der aktuellen Position. Die Suche kann mit SIGINT beendet werden.

Standardmäßig ist die *ex*-Option *wrapscan* (abgekürzt *ws*) gesetzt, deshalb erfolgt die Suche über Dateiende hinaus und wird am Dateianfang fortgesetzt bis die aktuelle Position der Schreibmarke wieder erreicht ist. Mit dem *ex*-Kommando *set nows* endet die Suche am Dateiende (siehe *ex*, *ex-Optionen* und *ex*-Kommando *set*).

Mit dem *vi*-Kommando *n* wiederholen Sie die Suche in Suchrichtung, mit *N* in Gegenrichtung.

Wird das Schrägstrich- Kommando bei einem anderen Kommando als Positionierkommando verwendet, so wird der Bereich von der Position der Schreibmarke (einschließlich) bis zur angegebenen Position (ausschließlich) bearbeitet.

Beispiel

In einer Zeile steht folgender Text:
```
Ein Taucher, der nicht taucht, taugt nix.
```

Die Schreibmarke steht auf dem Komma hinter Taucher. Geben sie nun ein:
```
d/ taugt↵
```

Es bleibt:
```
Ein Taucher taugt nix.
```

? Das Fragezeichen bewirkt eine Suche in Rückwärtsrichtung; ist also das Gegenstück zu / (siehe /).

0 Mit Null 0 wird die Schreibmarke zum ersten Zeichen in der aktuellen Zeile bewegt. Wenn der Null eine andere Ziffer vorangestellt wird, wird die Null nicht als *vi*-Kommando interpretiert.

: Das *vi*-Kommando Doppelpunkt : wechselt in den *vi*-Zeilen-Kommandomodus. Hier können Sie nun ein *ex*-Kommando angeben. Das *ex*-Kommando wird mit↵ abgeschickt.

Falls Sie ein *ex*-Kommando eingeben wollen, das länger als eine Eingabezeile ist, müssen Sie mit *Q* in den *ex*-Kommandomodus wechseln.

[zahl]**;**
Strichpunkt ; bewirkt die Wiederholung des letzten *vi*-Kommandos (Suche nach einem einzelnen Zeichen): *f*, *F*, *t*, *T*.
Durch *zahl* kann der Wiederholungsfaktor angegeben werden.

[zahl]**<**position
Auf Kleinerzeichen < muss ein Positionierkommando auf eine andere Zeile folgen. Die Zeilen von der aktuellen Zeile bis einschließlich der angegebenen Zeile werden um *shiftwidth* Positionen nach links verschoben (siehe *ex*, *ex-Optionen*). *zahl* wirkt als Multiplikator.

position kann auch ein zweites Kleinerzeichen sein. << bewirkt, dass die aktuelle Zeile verschoben wird (oder *zahl* Zeilen, einschließlich der aktuellen Zeile).

[zahl]**=**position

Auf das Gleichheitszeichen = muss ein Positionierkommando auf eine andere Zeile folgen. Ist die *ex*-Option *lisp* gesetzt, so werden die Zeilen im angegebenen Bereich so eingerückt, als wäre bei ihrer Eingabe auch die Option *autoindent* gesetzt. Mit *zahl* kann angegeben werden, wieviele Zeilen bearbeitet werden sollen. *position* kann auch ein zweites Gleichheitszeichen sein. == bewirkt, dass die aktuelle Zeile eingerückt wird (oder zahl Zeilen , einschließlich der aktuellen).

[zahl]**>**position

Das *vi*-Kommando Größerzeichen > verschiebt Zeilen um *shiftwidth* Positionen nach rechts (siehe <).

? Das Fragezeichen bewirkt eine Suche in Rückwärtsrichtung; ist also das Gegenstück zu / (siehe /).

` marke

(Gegenhochkomma)

Schreibmarke auf gesetzte Marke bewegen

marke ist ein Kleinbuchstabe, der eine Marke bezeichnet, die mit dem *vi*-Kommando *m* gesetzt wurde. Mit Gegenhochkomma Marke ` *marke* wird die Schreibmarke auf das markierte Zeichen positioniert.

Wird das ` -Kommando bei einem anderen Kommando als Positionierkommando verwendet, wird der Bereich zwischen der aktuellen Postion der Schreibmarke (einschließlich) und der vorherigen aktuellen Position (ausschließlich) bearbeitet.

Beispiel

```
d` a
```

Der Bereich von der aktuellen Position der Schreibmarke bis zur Marke *a* wird gelöscht.

`` (doppelte Gegenhochkommas)

Schreibmarke auf vorherige aktuelle Position bewegen

Durch das Kommando doppeltes Gegenhochkomma `` wird die Schreibmarke auf die Position bewegt, auf der sie sich vor der letzten Ausführung eines der folgenden *vi*-Kommandos befunden hat:

H, L, M, n, N, %, ”, (,), [[,]], `` , {, },
/*Muster* ↵,
?*Muster* ↵.

Wird das `` -Kommando bei einem anderen Kommando als Positionierkommando verwendet, wird der Bereich zwischen der aktuellen Postion der Schreibmarke (einschließlich) und der vorherigen aktuellen Position (ausschließlich) bearbeitet.

[zahl]**[[**
Mit 2 öffnenden eckigen Klammern wird zurück auf die *zahl*-te Abschnittsgrenze positioniert.

[zahl]**]]**
Mit 2 schließenden eckigen Klammern wird vorwärts auf die *zahl*-te Abschnittsgrenze positioniert.

^ Mit Dach ^ wird die Schreibmarke auf das erste Zeichen in der aktuellen Zeile bewegt, bei dem es sich nicht um ein Leer- oder Tabulatorzeichen handelt.

[zahl]**{**
Mit der öffnenden geschweiften Klammer { wird die Schreibmarke zum Anfang des aktuellen Absatzes bewegt. Durch die Angabe von *zahl* kann die Schreibmarke auf den Anfang des *zahl*-ten Absatzes gesetzt werden.

[zahl]**}**
Mit der schließenden geschweiften Klammer } wird die Schreibmarke zum Anfang des nächsten Absatzes bewegt. Durch *zahl* kann ein Wiederholungsfaktor angegeben werden.

[spalte]**|**
Mit dem senkrechten Strich | wird die Schreibmarke auf die mit *spalte* angegebene Spalte positioniert.

spalte nicht angegeben:
spalte = 1

~ Die Tilde ~ bewirkt eine Zeichenkonvertierung. Steht an der aktuellen Schreibmarke ein Kleinbuchstabe, so wird er in einen Großbuchstaben umgewandelt. Steht an der aktuellen Schreibmarkenposition ein Großbuchstabe, so wird er in einen Kleinbuchstaben umgewandelt. Die Schreibmarke bewegt sich anschließend um ein Zeichen nach rechts weiter.

Anpassung an die Datensichtstation

Die Bildschirmausgabe des *vi* hängt vom Typ der verwendeten Datensichtstation ab. In POSIX brauchen Sie den *vi* nicht anzupassen.

vi benutzt den Inhalt der Umgebungsvariablen *TERM*, um in einer Datenbank (*/usr/lib/terminfo/*/**) die notwendigen Informationen über die verwendete Datensichtstation zu finden. Falls *TERM* nicht definiert ist oder Sie an einer Datensichtstation eines anderen Typs arbeiten, müssen Sie *TERM* neu setzen. Dazu gibt es verschiedene Möglichkeiten:

– Shell-Variable *TERM* setzen und exportieren

– *TERM* in der Datei *.profile* definieren und exportieren

– *ex*-Kommando *set term* ausführen (nur im *ex*-Kommandomodus möglich)

– *ex*-Kommando *set term* in die Datei *.exrc* eintragen

– Umgebungsvariable *EXINIT* in der Datei *.profile* definieren und exportieren

– Den Systemverwalter konsultieren

Achtung!
Geben Sie nur den tatsächlichen Namen Ihrer Datensichtstation für *TERM* an. Falls andere Namen verwendet werden, kann dies zur Störung Ihrer Datensichtstation führen.

Voreinstellung des vi

Voreinstellungen werden mit *ex*-Optionen gesetzt; beim Kommando *ex*, Abschnitt *ex-Optionen*, ist beschrieben, welche Voreinstellungen möglich sind.

Das *ex*-Kommando *set all* listet alle aktuell geltenden Voreinstellungen auf, zum Beispiel:

directory=/tmp	Dateiverzeichnis für Editorpuffer
nonumber	Zeilennummern werden nicht ausgegeben
report=5	Meldung bei mehr als 5 veränderten Zeilen
scroll=11	Bildschirm wird um 11 Zeilen bei CTRL D verschoben
noshowmode	Keine Anzeige des *vi*-Modus
term=97801	Terminaltyp
window=23	Bildschirmzeilen für das Textfenster
wrapmargin=0	Kein automatischer Zeilenumbruch

Es gibt zwei Typen von Optionen: Optionen mit Boole'schen Werten und Optionen mit nicht-Boole'schen Werten. Ein Beispiel für den Boole'schen Typ ist die Option *showmode*. Ist sie gesetzt, gibt *set all* die Zeichenkette *showmode* aus, ist sie nicht gesetzt, gibt *set all* die Zeichenkette *noshowmode* aus. Bei nicht-Boole'schen Typen gibt *set all* den Wert aus, auf den die jeweilige Option gesetzt ist. Ein Beispiel für den nicht-Boole'schen Typ ist *scroll*.

Wenn im *vi* Probleme mit den Pfeiltasten auftreten (z.B. wenn die Betätigung der Pfeiltasten zu einem Wechsel vom Eingabe- in den Kommandomodus führt), sollte die Option *timeout* abgeschaltet werden (*set notimeout*).

Sie können den *vi* an Ihre Bedürfnisse und Gewohnheiten anpassen. Der *vi* kann zum Beispiel Zeilennummern anzeigen oder Sie in der Statuszeile informieren, in welchem Modus er sich gerade befindet.

Wenn Sie die Voreinstellungen des *vi* ändern möchten, haben Sie dazu drei Möglichkeiten:

- während einer *vi*-Sitzung
- mit der Umgebungsvariablen *EXINIT*
- mit der Datei *.exrc*

Möchten Sie sich zum Beispiel die Zeilennummern und den *vi*-Modus anzeigen lassen, haben Sie dazu die nachfolgend aufgeführten Möglichkeiten:

- Während einer *vi*-Sitzung:

  Um sich die Zeilennummern anzeigen zu lassen, benutzen Sie während einer *vi*-Sitzung das *ex*-Kommando *set number*; mit dem *ex*-Kommando *set showmode* bewirken Sie, dass der *vi* Ihnen anzeigt, wenn er sich im Eingabemodus befindet. Die Anpassung gilt dann nur für die Dauer der Sitzung.

– Mit der Umgebungsvariablen *EXINIT*:

  In die Datei *.profile* in Ihrem *HOME*-Dateiverzeichnis müssen Sie folgende Zeilen schreiben:

```
EXINIT='set number showmode'
export EXINIT
```

  Mit dem Kommando . *.profile* machen Sie die neue Variable Ihrer Shell bekannt (siehe Kommando Punkt .). Rufen Sie nun den *vi* auf, so werden die Zeilennummern ausgegeben, und der *vi* zeigt Ihnen an, wenn Sie im Eingabe-Modus sind. Diese Voreinstellung ist nun immer wirksam.

– Mit der Datei *.exrc*:

  Diese Datei müssen Sie selbst anlegen. In *.exrc* definieren Sie ständige Voreinstellungen des *vi*. Damit der *vi* Zeilennummern ausgibt und den Modus anzeigt, müssen Sie in diese Datei folgende Zeile eintragen:

```
set number showmode
```

  Die Datei *.exrc* muss im *HOME*-Dateiverzeichnis stehen. *.exrc*-Dateien in Unterdateiverzeichnissen werden aus Sicherheitsgründen vom *vi* nicht ausgewertet. *.exrc* im *HOME*-Dateiverzeichnis wird immer gelesen!

Die für *vi* definierten Voreinstellungen gelten auch für *ex*.

Die Kombination der verschiedenen Möglichkeiten ist erlaubt, unterliegt jedoch gewissen Bedingungen:
Während einer *vi*-Sitzung definierte Voreinstellungen haben höchste Priorität. Sie können daher während einer Sitzung jederzeit die aktuellen Voreinstellungen mit dem *ex*-Kommando *set* ändern.
Wenn *EXINIT* definiert ist, wird die Datei *$HOME/.exrc*. nicht ausgewertet

Datei *$HOME/.exrc*
Datei mit Voreinstellungen für *ex* und *vi*. Diese Voreinstellungen sind nicht wirksam, wenn *EXINIT* definiert ist oder während einer *vi*-Sitzung etwas Widersprüchliches definiert wird.

Variable *LINES*
Anzahl der Zeilen pro Bildschirm

TERM
Typ der benutzten Datensichtstation

EXINIT
Umgebungsvariable mit Voreinstellungen für *ex* und *vi*.

COLUMNS
Wenn diese Variable gesetzt ist, dann wird ihr Wert für die Definition der Breite des Editierfensters beim Editiermodus der POSIX-Shell und für die Ausgabe der *select*-Liste verwendet. Diese Variable ist sinnvoll, wenn der Zugang zur POSIX-Shell über *rlogin* erfolgt.

PATH
Der Suchpfad für Shell-Kommandos, die in den Editorkommandos *shell*, *read* und *write* angegeben werden können.

SHELL
Der Kommandozeilen-Interpreter, der bei *!*, *shell*, *read* oder anderen Kommandos mit einem Operanden der Form *!string* verwendet wird. Für das *shell* Kommando wird das Programm nur dem *-i* Argument aufgerufen, bei allen anderen mit zwei Argumenten *-c* und *string*. Ist *SHELL* nicht gesetzt oder hat den Wert 0, wird *sh* verwendet.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *vi*:

LANG	Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.
LC_ALL	Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.
LC_CTYPE	Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).
LC_MESSAGES	Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.
NLSPATH	Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *edit*, *ex*

el Lozy, M.: *Editing in a UNIX Environment. The vi/ex Editor* [18]

wait Auf die Beendigung von Hintergrundprozessen warten (await process completion)

Das in die POSIX-Shell *sh* eingebaute Kommando *wait* wartet,
- bis der vorher gestartete Hintergrundprozess mit der angegebenen Prozessnummer (PID) beendet ist, oder
- bis alle vorher gestarteten Hintergrundprozesse beendet sind, wenn Sie beim Aufruf keine Prozessnummer angeben.

Im Dialog wartet die Shell standardmäßig nicht auf die Beendigung eines vorher gestarteten Hintergrundkommandos, sondern gibt sofort das Bereitzeichen aus. Bei Kommandos, die nicht im Hintergrund gestartet werden, wartet die Shell immer auf das Ende der Ausführung.

Für Shell-Prozeduren gilt das gleiche. Wenn eine Shell-Prozedur ein Kommando enthält, das die Ausgabe eines vorher im Hintergrund gestarteten Kommandos bearbeiten soll, können Sie mit *wait* sicherstellen, dass dieses Hintergrundkommando rechtzeitig beendet ist.

Erhalten Sie beim Starten eines Hintergrundprozesses die Fehlermeldung

```
fork failed - too many processes
```

dann können Sie Ihr System durch den Gebrauch von *wait* entlasten. Wenn Sie hierbei keinen Erfolg erzielen, ist die Systemprozesstabelle voll oder es sind zu viele Prozesse im Vordergrund aktiv.

Hinweis

Besteht eine Pipeline aus 3 oder mehr Teilen, so gruppiert die Shell jeweils zwei Prozesse und ordnet diesen einen Steuerprozess zu. Diese beiden Prozesse sind damit keine Sohnprozesse der Shell. Beachten Sie deshalb, dass *wait* auf Prozesse, die keine Sohnprozesse der Shell sind, nicht wartet.

Syntax

wait[␣prozessnummer...]

prozessnummer

Nummer des Hintergrundprozesses, auf dessen Ende *wait* warten soll.

Sie können nur eine Prozessnummer angeben. Wenn Sie mehrere Prozessnummern angeben, gibt *wait* keine Fehlermeldung aus, sondern wartet auf die Beendigung des Hintergrundprozesses, den Sie beim Aufruf als ersten angegeben haben.
Das Kommando *wait* gibt als Endestatus den Endestatus des Hintergrundprozesses zurück. Haben Sie die Prozessnummer eines bereits beendeten Hintergrundprozesses angegeben, dann verhält sich *wait* wie bei nicht angegebener Prozessnummer (siehe *Endestatus*).

Der Variablen *!* (Ausrufezeichen) weist die Shell immer die Prozessnummer des zuletzt im Hintergrund gestarteten Kommandos zu. Mit *$!* können Sie auf den Inhalt dieser Variablen zugreifen.

prozessnummer nicht angegeben:
wait wartet auf die Beendigung aller Hintergrundprozesse, die vor dem Aufruf von *wait* gestartet wurden. In diesem Fall gibt *wait* den Endestatus 0 zurück.

Endestatus Wenn Sie *wait* ohne Argument aufgerufen haben und alle bekannten Prozessnummern sich beendet haben, ist der Endestatus immer gleich 0.

1-126 Fehler

127 Das Kommando der letzten Prozessnummer ist unbekannt.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *wait*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Die Shell soll auf den gestarteten Hintergrundprozess warten:

```
$ find / -name tst -print 2>/dev/null &
3456
$ wait 3456
home/markus/PASCAL/tst
.
.
.
```

Beispiel 2 Die Shell-Prozedur *warte* zeigt, wie man auf das Ende eines bestimmten Hintergrundkommandos warten kann. Diese Shell-Prozedur hat den folgenden Inhalt:

```
: Aufruf mit sh warte
sort liste > sortiert &
pid1=$!
tar cvf /dev/dsk/f0t /home/anna &
pid2=$!
.
.
.
wait $pid1
pg sortiert
```

Die jeweiligen Prozessnummern der Hintergrundkommandos werden in den Variablen *pid1* und *pid2* gespeichert. So sind diese Prozessnummern für den späteren Gebrauch gesichert, da die Variable *!* immer nur die Prozessnummer des zuletzt im Hintergrund gestarteten Kommandos enthält.

Das Kommando *wait* wartet auf das Ende des Kommandos *sort*. Erst dann gibt *pg* den Inhalt der Datei *sortiert* aus.

Siehe auch *waitpid*() [4]

wc Wörter, Zeichen und Zeilen zählen (word, line and byte or character count)

wc gibt die Anzahl der Zeilen, Wörter und Zeichen von Dateien auf die Standard-Ausgabe aus.

Syntax **wc**[␣**-c**|␣**-m**][␣**-lw**][␣datei...]

Keine Option angegeben
wc gibt drei Zahlenwerte aus für die Anzahl der Zeilen, Wörter, Zeichen.

option

-c *wc* gibt die Anzahl der Bytes aus. Leer-, Tabulator- und Neue-Zeile-Zeichen werden mitgezählt. Die Option *-c* verhält sich genauso wie die Option *-m*, da ein Byte einem Zeichen entspricht.

-m *wc* gibt die Anzahl der Zeichen aus. Leer-, Tabulator- und Neue-Zeile-Zeichen werden mitgezählt. Die Option *-m* verhält sich genauso wie die Option *-c*, da ein Zeichen einem Byte entspricht.

-l *wc* gibt die Anzahl der Zeilen aus. Die Anzahl ermittelt *wc* aus der Anzahl der Neue-Zeile-Zeichen.

-w *wc* gibt die Anzahl der Wörter aus. Wörter sind nicht-leere Zeichenketten, die durch Zwischenraumzeichen voneinander getrennt sind. Zwischenraumzeichen sind Leerzeichen, Tabulatoren und Neue-Zeile-Zeichen.

datei
Name der Datei, deren Zeilen, Wörter und Zeichen gezählt werden sollen. Der Name der Datei wird zusammen mit den ermittelten Werten ausgegeben.
Sie können mehrere Dateien angeben. Bei mehreren Dateien gibt *wc* zusätzlich eine Zeile aus, in der die Summe der einzelnen Angaben steht.
datei nicht angegeben:
wc liest von der Standard-Eingabe.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *wc*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) und welche Zeichen als Zwischenraumzeichen eingestuft werden.

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Sie möchten die Anzahl der Zeilen, Wörter und Zeichen für die Dateien *logik*, *plan* und *rest* ausgeben lassen.

```
$ wc logik plan rest
     27    139    1077  logik
      5     15     140  plan
      3      6      51  rest
     35    160    1268  total
```

Beispiel 2 Sie möchten feststellen, wieviele Dateien im aktuellen Dateiverzeichnis eingetragen sind.

```
$ ls | wc -l
       31
```

Beispiel 3 Sie möchten feststellen, wieviele Benutzer gerade am Rechner arbeiten.

```
$ who | wc -l
        6
```

Beispiel 4 Sie möchten zählen lassen, wieviele verschiedene Wörter in einer Datei stehen.

```
$ cat datei | sed 's/ ␣␣*/\
> /g' | sort -u | wc -l
```

Erläuterung:

sed erstellt eine Liste aller Wörter von *datei*, indem ein oder mehrere Leerzeichen durch Neue-Zeile-Zeichen ersetzt werden. *sort -u* sortiert diese Liste und entfernt dabei alle Wiederholungen. *wc -l* zählt dann die Zeilen dieser Liste und gibt die ermittelte Anzahl aus.

whence Kommando-Lokalisation (query command type)

Das in die POSIX-Shell *sh* eingebaute Kommando *whence* gibt für jeden Namen an, wie er als Kommandoname interpretiert werden würde.

Syntax **whence**[␣**-pv**]␣name␣**...**

-v gibt eine ausführlichere Liste aus.

-p durchsucht den Suchpfad auch dann, wenn *name* eine Alias-Variable, eine Funktion oder ein reserviertes Wort ist.

name
Name des zu interpretierenden Kommandos.

Endestatus

0 das Kommando wurde erfolgreich ausgeführt

>0 kein Kommando *name* gefunden

Beispiel Verschiedene Ausgabemöglichkeiten von *whence*

```
$ whence echo
echo
$ whence -pv echo
echo is /usr/bin/echo
$ whence -v echo
echo is a shell builtin
$ whence -v cat
cat is a tracked alias for /usr/bin/cat
```

Siehe auch *type*

who Aktive Benutzerkennungen anzeigen (display who is on the system)

who informiert Sie darüber,
- unter welcher Benutzerkennung und an welcher Datensichtstation Sie gerade arbeiten
- welche Benutzer seit wann am System angemeldet sind und an welcher Datensichtstation sie arbeiten
- welche Prozessnummer der verwendete Kommandointerpreter (die verwendete Shell) hat
- wann zuletzt an einer Datensichtstation gearbeitet wurde
- wann welche An- und Abmeldungen am System erfolgt sind und wann Systemabstürze stattgefunden haben, wobei diese Informationen bis zu jenem Zeitpunkt zurückreichen, an dem der Systemverwalter die Datei */var/adm/wtmp* zum letzten Mal auf die Größe 0 reduziert hat
- wann die Systemzeit zuletzt verändert wurde
- welche Prozesse vom *init*-Prozess gestartet wurden.

Hinweis *who* bezieht seine Informationen standardmäßig aus der Datei */var/adm/utmp*. Bei jedem Login werden die entsprechenden Informationen in dieser Datei auf den neuesten Stand gebracht. Im Single-User-Modus erfolgt kein Login. Nach einem Shutdown in den Single-User-Modus kann *who* deshalb keine korrekten Informationen über den augenblicklichen Login-Status liefern. Verwenden Sie statt dessen *who am i*.

Syntax

Format 1: who[␣**-mu**]␣**-s**[␣**-bHlprt**][␣datei]

Format 2: who[␣**-mTu**][␣**-abdHlprt**][␣datei]

Format 3: who␣**-q**[␣**-n**␣zahl][␣datei]

Format 4: who␣**am**␣**i**

Format 5: who␣**am**␣**I**

Ausführliche Informationen ausgeben

Format 1 **who**[␣**-mu**]␣**-s**[␣**-bHlprt**][␣datei]

Format 2 **who**[␣**-mTu**][␣**-abdHlprt**][␣datei]

Keine Option angegeben

who gibt für jeden aktuell am System angemeldeten Benutzer folgendes aus:
- Benutzerkennung, unter der sich der Benutzer angemeldet hat
- Name der Datensichtstation, an der sich der Benutzer angemeldet hat
- Anmeldezeitpunkt

Die Bedeutung der Ausgabespalten ist im Abschnitt *Ausgabe* genauer erläutert.

Optionen

-a (a - all) Es werden die Optionen *-b*, *-d*, *-l*, *-p*, *-r*, *-t*, *-T* und *-u* aktiviert.

-b (b - boot) *who* gibt Zeit und Datum des letzten Systemstarts aus.

-d (d - dead) *who* gibt alle Prozesse aus, die sich beendet haben und von *init* nicht neu gestartet wurden. Für beendete Prozesse werden der *Endestatus* und die Nummer des Signals, das den Prozess beendet hat, angegeben. Damit können Sie eventuell feststellen, weshalb ein Prozess beendet wurde.

-H (H - headings) Die einzelnen Spalten erhalten Überschriften.

-l (l - login) Es werden die Prozesse aufgelistet, bei denen das System auf eine Anmeldung wartet. Das Ausgabefeld *Name* enthält in diesem Fall den Namen des Programms (bzw. *LOGIN)*, das Feld *Zustand* existiert nicht. Die anderen Felder haben die übliche Bedeutung.

-m *who* gibt nur Informationen zu der aktiven Datensichtstation aus.

-p (p - process) *who* listet alle Prozesse auf, die vom *init*-Prozess gestartet wurden. Das Ausgabefeld *Name* enthält den Namen des von *init* gestarteten Programms.
Das Feld *Leitung (Line)* hat im Zusammenhang mit *-p* keine Bedeutung, daher wird in diesem Feld nur ein Punkt ausgegeben.

-r (r - run level) *who* gibt den aktuellen Run Level des Prozesses *init* aus. Im Feld *Leitung* wird der momentane Run Level ausgegeben, im Feld *Zeit* das Enstehungsdatum. *inaktiv* enthält den aktuellen Run Level als Zahl, *PID* zeigt, wie oft man sich zuvor bereits in diesem Status befunden hat, *Kommentar* gibt den Run Level an, in dem man sich zuvor befunden hat. Das Ausgabefeld *Name* hat bei der Option *-r* keine Bedeutung.

-s (s - standard) *who* listet die Benutzer auf, die momentan angemeldet sind. Das Feld *Name* ist der jeweilige Benutzername, *Leitung* ist der Name der Datensichtstation (ohne */dev/*), an der sich der Benutzer angemeldet hat. Das Ausgabefeld *Zeit* gibt an, wann sich der Benutzer angemeldet hat.

Diese Option ist die Standardeinstellung. *-s* kann nicht mit *-a*, *-d* oder *-T* kombiniert werden.

-T (T - Terminal) Der Zustand der Datensichtstation wird zusätzlich zu den Standardwerten ausgegeben.
Das Feld *Zustand* gibt an, ob ein anderer Benutzer auf diese Datensichtstation schreiben darf: Ein Pluszeichen + bedeutet, dass jeder Benutzer auf diese Datensichtstation schreiben darf, ein Minuszeichen - bedeutet, dies ist nicht möglich. Der POSIX-Verwalter kann auf alle Datensichtstationen schreiben. Kann diese Information nicht bestimmt werden, so wird ein Fragezeichen ? ausgegeben.

-t (t - time) *who* gibt den Zeitpunkt der letzten Änderung der Systemzeit durch den Systemverwalter (mit dem Kommando *date*) aus.

-u (u - user) *who* listet die Benutzer auf, die momentan angemeldet sind. Das Feld *Name* ist der jeweilige Benutzername. *Leitung* ist der Name der Datensichtstation (ohne */dev/*), an der sich der Benutzer angemeldet hat. *Zeit* gibt an, wann sich der Benutzer angemeldet hat.
Das Ausgabefeld *inaktiv* gibt Auskunft über die letzte Ein- oder Ausgabe auf der Datensichtstation: Ein Punkt . bedeutet, dass innerhalb der letzten Minute Ein- oder Ausgabe erfolgte. Falls die Datensichtstation seit mehr als 24 Stunden oder seit dem Hochfahren des Systems nicht benutzt wurde, wird der Eintrag mit *old* gekennzeichnet.
Das Ausgabefeld *PID* gibt die Prozessnummer des Kommandointerpreters (der Shell) an, mit dem ein Benutzer arbeitet.

datei
Name der Datei, aus der *who* seine Informationen bezieht. Möglich ist hier die Angabe von */var/adm/wtmp*. *who* informiert dann über An- und Abmeldungen am System sowie über Systemabstürze, wobei diese Informationen bis zu jenem Zeitpunkt zurückreichen, an dem der Systemverwalter die Datei */var/adm/wtmp* zum letzten Mal auf die Größe 0 reduziert hat.

datei nicht angegeben:
who holt die Informationen aus der Datei */var/adm/utmp*

Ausgabe

Im Folgenden werden die Überschriften und die Bedeutung der Spalten in der Ausgabe von *who* erläutert.
Die Besonderheiten der Ausgabe bei den Optionen *-b*, *-d*, *-p*, *-r* und *-t* werden bei der Beschreibung der jeweiligen Option näher beschrieben.

Name
Das Feld *Name* gibt den jeweiligen Benutzernamen an.

Zustand
Das Feld *Zustand* gibt an, ob ein anderer Benutzer auf diese Datensichtstation schreiben darf: Ein Pluszeichen + bedeutet, dass jeder Benutzer auf diese Datensichtstation schreiben darf, ein Minuszeichen - bedeutet, dies ist nicht möglich. Der Systemverwalter kann auf alle Datensichtstationen schreiben. Bei einer defekten Leitung wird ein Fragezeichen ? ausgegeben.

Leitung
Das Feld *Leitung* gibt den Namen der Datensichtstation (ohne */dev/*) an, an der sich der jeweilige Benutzer angemeldet hat.

Zeit
Das Feld *Zeit* gibt an, wann sich der jeweilige Benutzer angemeldet hat.

inaktiv
: Das Feld *inaktiv* gibt Auskunft über die letzte Ein- oder Ausgabe auf der Datensichtstation: Ein Punkt . bedeutet, dass innerhalb der letzten Minute Ein- oder Ausgabe erfolgte. Falls die Datensichtstation seit mehr als 24 Stunden oder seit dem Hochfahren des Systems nicht benutzt wurde, wird der Eintrag mit *old* gekennzeichnet.

PID
: Das Feld *PID* gibt die Prozessnummer des Kommandointerpreters (der Shell) an, mit dem ein Benutzer arbeitet.

Kommentar
: Das Feld *Kommentar* gibt den Run Level an, in dem man sich zuvor befunden hat.

Endestatus
: Das Feld *Endestatus* gibt für beendete Prozesse den Endestatus und die Nummer des Signals an, das den Prozess beendet hat. Damit können Sie eventuell feststellen, weshalb ein Prozess beendet wurde.

Die Informationen zu *Name* (Benutzerkennung), *Leitung* (Datensichtstation) und *Zeit* werden bei den Optionen *-a*, *-l*, *-s*, *-T* und *-u* ausgegeben.
Zustand wird nur bei *-T* ausgegeben, *inaktiv*, *PID* (Prozessnummer) *-a*, *-l* und *-u*. Die Option *-a* informiert zusätzlich über den *Endestatus*.

Format 3 **Kurzinformation ausgeben**

who␣-q[␣**-n**␣zahl][␣datei]

-q (q - quick) *who* gibt nur die Namen und die Zahl aller aktuell angemeldeten Benutzer aus. Außer *-n* werden weitere Optionen ignoriert.

-n␣zahl
zahl gibt an, wieviele Benutzernamen pro Zeile ausgegeben werden. *zahl* muss größer 0 sein.

-n␣zahl nicht angegeben:
es werden standardmäßig 8 Benutzernamen pro Zeile ausgegeben.

datei
wie bei Format 1

Eigeninformation ausgeben

Format 4 **who␣am␣i**

Format 5 **who␣am␣I**

Die Formate 4 und 5 sind identisch mit *who -m*.

who gibt aus:
- Benutzerkennung, unter der Sie sich angemeldet haben
- Name der Datensichtstation (ohne */dev/*), an der Sie sich angemeldet haben
- Anmeldezeitpunkt

Datei */var/adm/utmp*
Datei, aus der *who* standardmäßig seine Informationen holt. Diese Datei enthält den aktuellen Zustand.

/var/adm/wtmp
Diese Datei kann als Alternative zu */var/adm/utmp* angegeben werden. Der Systemverwalter reduziert die Größe dieser Datei regelmäßig auf 0.

Internationale Umgebung
Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *who*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

LC_TIME Legt das Format der Datums- und Zeitangaben fest.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel 1 Ausgabe einer Übersicht über alle angemeldeten Benutzer durch Aufruf von *who* ohne Optionen.

```
$ who
CMPQD004   term/002     Mon Aug 18 16:48:21 MSZ 2008
QM212JNA   term/003     Mon Aug 18 17:24:54 MSZ 2008
```

Beispiel 2 Ausgabe einer ausführlichen, mit Spaltenüberschriften versehenen Übersicht über alle angemeldeten Benutzer durch Aufruf von *who* mit den Optionen *-H* und *-u*.

```
$ who -Hu
NAME       LINE         TIME          IDLE    PID
markus     tty001       Mar  7 10:53   .     6252
ulrich     tty002       Mar  7 11:10  0:22   6289
```

Beispiel 3 Ausgabe einer Kurzübersicht über alle angemeldeten Benutzer durch Aufruf von *who* mit der Option *-q*.

```
$ who -q
CMPQD004 QM212JNA
# users=2
```

Beispiel 4 Ausgabe der eigenen Daten durch Aufruf von *who am i*.

```
$ who am i
 QM212JNA   term/003     Mon Aug 18 17:24:54 MSZ 2008
```

Siehe auch *date*, *last*, *mesg*
wait() [4]

write Nachricht an einen Benutzer senden (write to another user)

Es können nur die Benutzer Nachrichten empfangen, die mit Zeichenterminals arbeiten, d.h. über *rlogin* Zugang zur POSIX-Shell erhalten haben.
Dagegen können alle Benutzer Nachrichten senden, also auch die Benutzer, die mit Blockterminals arbeiten.

write sendet Nachrichten an einen anderen Benutzer. *write* liest zeilenweise von der Standard-Eingabe und schickt die eingelesenen Zeilen als Nachrichten an den angegebenen Benutzer. Beim Aufruf von *write* erscheint auf dem Bildschirm des Empfängers zuerst ein Nachrichtenkopf, der die Kennung des Senders, seine Datensichtstation und die Sendezeit enthält. Danach erscheinen die Nachrichten.

Benutzer können miteinander kommunizieren, wenn sie sich gegenseitig mit *write* Nachrichten senden (siehe *Benutzer-Dialog*).

Vor dem Aufruf beachten

An Benutzer, die ihre Datensichtstation mit *mesg -n* für Nachrichten gesperrt haben oder an Benutzer, die auf einem Blockterminal arbeiten, können Sie keine Nachrichten mit *write* schicken. Ein Benutzer mit POSIX-Verwalterberechtigung kann Nachrichten an alle Datensichtstationen schicken, auch wenn diese für die Nachrichtenübermittlung mittels *mesg -n* gesperrt sind.

Syntax

write␣empfänger[␣tty-name]
text
...
...
... CTRL D

empfänger
: Kennung eines Benutzers, der an einer Datensichtstation angemeldet ist. Sie können auch Nachrichten an sich selbst schicken. Wenn ein Benutzer an mehreren Datensichtstationen gleichzeitig angemeldet ist, können Sie zusätzlich die Datensichtstation angeben.

  Mit *who* erfahren Sie alle aktuell angemeldeten Benutzer und ihre Datensichtstationsnummern.

tty-name
Nummer der Datensichtstation, an der der Empfänger angemeldet ist.

tty-name nicht angegeben:
write sucht die Datensichtstation aus der Datei */var/adm/utmp*. Wenn ein Benutzer mehrfach angemeldet ist und es daher mehrere Einträge gibt, verwendet *write* den ersten Eintrag, der dort verzeichnet ist. Es wird folgende Meldung ausgegeben:

```
user is logged on more than one place.
you are connected to "tty-name".
Other locations are
tty-name
```

text
Text, der als Nachricht geschickt werden soll. *write* liest zeilenweise von der Standard-Eingabe bis zum Dateiende-Zeichen:

- eine Zeile, die mit einem Ausrufezeichen ! beginnt, interpretiert *write* als Kommando und übergibt den Rest der Zeile der Shell. Das Kommando wird ausgeführt, *write* bleibt aktiv. Ausgaben, die das Kommando auf die Standard-Ausgabe schreibt, werden nicht in die Nachrichten aufgenommen.
- jede andere Zeile wird als Nachricht an den Empfänger geschickt
- wenn *write* das Dateiende-Zeichen liest, gibt es zum Abschluss beim Empfänger Dateiende („EOT\n“) aus und beendet sich.
- Nichtdruckbare Zeichen (mit Ausnahme von *\d*, *\v*, *\n*, *\r* und *\a*) werden vor dem Senden umgewandelt.
  Steuerzeichen werden als Folge Dach ^<Großbuchstabe> dargestellt.
  <Großbuchstabe> ist der Buchstabe, der sich im ASCII-Zeichensatz durch Setzen des 7. Bits ergibt. So wird zum Beispiel „\003“ als „^C“ dargestellt.

Benutzer-Dialog

Benutzer können miteinander kommunizieren, wenn sie sich gegenseitig mit *write* Nachrichten senden. Der Ablauf bei zwei Benutzern ist wie folgt:

1. Der erste Benutzer ruft *write* mit der Kennung des zweiten Benutzers auf. Der zweite Benutzer erhält den Nachrichtenkopf und erfährt, dass der erste Benutzer mit ihm kommunizieren möchte.

   ```
   Message from sender (terminal) [zeit]
   ```

   Der erste Benutzer erkennt an einem zweifachen Klingelzeichen, dass die Verbindung zustande gekommen ist und der zweite Benutzer Nachrichten entgegennehmen kann.

2. Der zweite Benutzer ruft nunmehr *write* mit der Kennung des ersten Benutzers auf.

   `write` *sender [terminal]*

   Der erste Benutzer erhält den Nachrichtenkopf als Antwort.

3. Jetzt können beide Benutzer sich gegenseitig Nachrichten senden. Jeder Benutzer sollte das Ende einer Nachricht eindeutig kennzeichnen, damit der andere weiß, wann er antworten kann. Sinnvoll ist auch ein Kennzeichen für das Dialogende.

4. Sie beenden den Dialog, indem Sie CTRL D oder die DEL-Taste drücken. Wenn Sie zusätzlich verhindern möchten, dass der andere Benutzer weiterhin Nachrichten sendet, dann rufen Sie *mesg -n* auf.

Fehler

`user is not logged on.` oder `user is not at " tty "`
Der Empfänger ist nicht angemeldet.

`Permission denied.`
Die Datensichtstation des Empfängers ist schreibgeschützt (siehe *mesg*) oder der Empfänger arbeitet mit einem Blockterminal.

`Warning: You have your terminal set to "mesg -n". No reply possible.`
Die eigene Datensichtstation ist für Nachrichten anderer Benutzer gesperrt.

`Warning: You are on a Block terminal. No reply possible.`
Die eigene Datensichtstation ist für Nachrichten anderer Benutzer gesperrt.

`Can no longer write to` *tty-name*
Nach Beginn der Übertragung wurde Schreibschutz für die Datensichtstation des Empfängers gesetzt (siehe *mesg*).

Datei

/var/adm/utmp
Datei, in der alle angemeldeten Benutzer registriert sind.

/usr/bin/sh
Kommandointerpreter für das Kommando Ausrufezeichen *!*.

Endestatus

0 Erfolgreiche Ausführung

>0 Der Empfänger ist nicht angemeldet oder die Datensichtstation des Empfängers ist schreibgeschützt.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *write*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt das Format und den Inhalt von Fehlermeldungen fest. Die hier angegebene internationale Umgebung gilt auch für informative Meldungen, die auf die Standard-Ausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel An die Benutzerin *karin* eine Meldung schicken:

```
$ write karin
!date
Mon Oct 13 19:00:13 MSZ 2008
heute am 15.10.
bin ich gespannt,
was passiert CTRL D
```

Siehe auch *mailx*, *mesg*, *pr*, *sh*, *talk*, *who*

xargs Argumentliste(n) aufbauen und Kommando ausführen (construct argument lists and involve utility)

xargs verbindet beim Aufruf angegebene Argumente und solche, die es von der Standard-Eingabe liest, miteinander. Danach führt *xargs* das beim Aufruf angegebene Kommando ein- oder mehrmals aus. Wie viele Argumente für jeden Kommando-Aufruf verwendet werden und auf welche Weise diese Argumente kombiniert werden, können Sie durch Optionen steuern.

Die von der Standard-Eingabe gelesenen Argumente müssen zusammenhängende Zeichenketten sein, die von einem oder mehreren Leer-, Tabulator- oder einem Neue-Zeile-Zeichen abgeschlossen werden. Leere Zeilen werden gelöscht. Wenn Leer- oder Tabulatorzeichen Bestandteil eines Arguments sein sollen, müssen sie entweder durch einen Gegenschrägstrich \ entwertet oder in Anführungszeichen "..." oder Hochkommas '...' eingeschlossen werden. Ansonsten würden sie als Trennzeichen zwischen den Argumenten interpretiert. Auch sonst gelten die üblichen Entwertungsmechanismen, d.h., Sonderzeichen werden dadurch entwertet, dass sie in Anführungszeichen "..." oder Hochkommas '...' eingeschlossen werden oder ihnen ein Gegenschrägstrich \ vorangestellt wird.

Syntax **xargs**[␣option]...[␣kommando[␣anfangs_argument]...]

option

Mit den Optionen *-I* (bzw. *-i)* , *-L* (bzw. *-l*) und *-n* legen Sie fest, wie die beim Aufruf von *xargs* angegebenen Anfangsargumente und die von der Standard-Eingabe eingelesenen Argumente für einen Aufruf des Kommandos *kommando* verwendet werden.

Keine der Optionen *-I* (bzw. *-i)* , *-L* (bzw. *-l*) oder *-n* angegeben:
Zunächst werden die beim Aufruf von *xargs* angegebenen Anfangsargumente, dann die Argumente von der Standard-Eingabe eingelesen, bis ein interner Puffer voll ist. Dann wird *kommando* mit all diesen Argumenten ausgeführt. Dieser Vorgang wird so lange wiederholt, bis alle Argumente abgearbeitet sind.

Kombinationen der Optionen *-I* (bzw. *-i)* , *-L* (bzw. *-l*) oder *-n*:
Wenn sich Optionen gegenseitig aufheben, etwa *-l* und *-n*, so gilt die zuletzt angegebene Option.

-I␣ersetzungszeichenkette (bzw. **-i**[ersetzungszeichenkette])
(I - insert) *kommando* wird für jede von der Standard-Eingabe eingelesene Zeile ausgeführt. Dabei wird jede Zeile als ein Argument interpretiert und für jedes Vorkommen von *ersetzungszeichenkette* in die Liste der beim Aufruf von *xargs* angegebenen Anfangsargumente eingefügt.
In der Liste der Anfangsargumente können maximal fünf Argumente jeweils ein- oder mehrmals *ersetzungszeichenkette* enthalten. Leer- und Tabulatorzeichen zu Beginn einer Zeile werden ignoriert. Die erstellten Argumente dürfen aus maximal 255 Zeichen bestehen.

Bei *-I* wird automatisch *-x* gesetzt.

Dies entspricht der alten Option *-i*, die weiterhin unterstützt wird. Für *ersetzungszeichenkette* wird bei *-i* ein Paar geschweifter Klammern {} angenommen, falls nicht explizit angegeben.

-L␣zeilenanzahl (bzw. **-l[**zeilenzahl])
(L - line) *kommando* wird für jede *zeilenanzahl* nicht-leerer Argumentzeilen, die *xargs* von der Standard-Eingabe liest, ausgeführt. Bleiben für den letzten Aufruf von *kommando* weniger als *zeilenanzahl* Zeilen übrig, so wird *kommando* mit diesen verbleibenden Zeilen ausgeführt.
Eine Zeile gilt beim ersten Auftreten eines Neue-Zeile-Zeichens als abgeschlossen, es sei denn, das letzte Zeichen in der Zeile ist ein Leer- oder Tabulatorzeichen. In diesem Fall wird die Zeile in der nächsten nicht leeren Zeile fortgesetzt.

Bei *-L* wird automatisch *-x* gesetzt.

Dies entspricht der alten Option *-l*, die weiterhin unterstützt wird. Ist *zeilenanzahl* bei *-l* nicht angegeben, wird 1 angenommen.

-n␣arganzahl
kommando wird unter Verwendung möglichst vieler von der Standard-Eingabe eingelesener Argumente ausgeführt, maximal jedoch mit *arganzahl* Argumenten. Wenn die Gesamtgröße der Argumentliste die durch *maxgröße* (siehe Option *-s*) festgelegte Obergrenze übersteigt, so werden weniger Argumente verwendet. Wenn für den letzten Aufruf von *kommando* weniger als *arganzahl* Argumente übrigbleiben, so werden diese verbleibenden Argumente verwendet. Wenn auch die Option *-x* gesetzt ist, so dürfen jeweils *arganzahl* Argumente die durch *maxgröße* festgelegte Obergrenze (siehe Option *-s*) nicht überschreiten, andernfalls wird die Ausführung von *xargs* beendet.

-E␣dateiende (bzw. **-e**[dateiende])
xargs liest die Standard-Eingabe entweder bis zum Erreichen des tatsächlichen Dateiendes oder bis es das angegebene logische Dateiende *dateiende* erkennt.

Für *dateiende* geben Sie eine Zeichenkette an. Diese Zeichenkette wird bei der Ausführung von *xargs* als logisches Dateiende interpretiert.

Dies entspricht der alten Option *-e*, die weiterhin unterstützt wird. Wird *-e* ohne *dateiende* angegeben, ist kein logisches Dateiende mehr definiert. Der Unterstrich _ hat keine Sonderbedeutung und wird als normales Zeichen verarbeitet.

Weder *-E* noch *-e* angegeben:
Der Unterstrich _ wird als logisches Dateiende interpretiert.

-p (p - prompt) Bei jedem Aufruf von *kommando* werden Sie gefragt, ob dieser Aufruf ausgeführt werden soll oder nicht. Dazu wird der trace-Modus (Option *-t*) eingeschaltet, und der Aufruf von *kommando* wird, gefolgt von der Eingabeaufforderung *?...*, angezeigt.

-s␣maxgröße
Die maximale Anzahl von Zeichen in einer Argumentliste wird auf *maxgröße* gesetzt (Eine Argumentliste ist eine Kombination von Argumenten, die nach den durch die Optionen *-I* (bzw. *-i)* , *-L* (bzw. *-l*) oder *-n* festgelegten Regeln erzeugt wurde). Zu beachten ist, dass in *maxgröße* ein zusätzliches Zeichen für jedes Argument und die Anzahl der Zeichen im Kommandonamen bereits enthalten sind.

-t (t - trace) Das *kommando* und jede erstellte Argumentliste werden unmittelbar bevor sie abgearbeitet werden auf die Standard-Fehlerausgabe ausgegeben.

-x Die Ausführung von *xargs* wird beendet, wenn die Länge einer Argumentliste die angegebene Obergrenze *maxgröße* (siehe Option *-s*) übersteigen würde.

Die Option *-x* ist standardmäßig gesetzt, wenn die Optionen *-I* (bzw. *-i*) oder *-l* gesetzt sind.
Wenn keine der Optionen *-I* (bzw. *-i)* , *-L* (bzw. *-l*) oder *-n* gesetzt ist, dann wird die Ausführung von *xargs* beendet, wenn die Gesamtlänge aller Argumente *maxgröße* (siehe Option *-s*) überschreitet.

kommando
Für *kommando* können Sie ein beliebiges Kommando angeben. Wenn die Ausführung von *kommando* den Ende-Status 255 liefert oder *kommando* nicht ausgeführt werden kann, wird die Ausführung von *xargs* beendet. Wenn *kommando* ein Shell-Skript ist, sollte es explizit mit *exit* einen passenden Ende-Status liefern, um den zufälligen Wert 255 zu vermeiden (z.B. *exit -1*).

kommando nicht angegeben:
Für *kommando* wird *echo* angenommen.

anfangs_argument
Die beim Aufruf von *xargs* angegebenen Anfangsargumente und die von der Standard-Eingabe eingelesenen Argumente werden wie oben beschrieben zu Argumentlisten zusammengestellt (siehe Optionen *-I*, *-L* und *-n*) und *kommando* wird mit diesen Argumentlisten ausgeführt.
Am Anfang einer Argumentliste stehen immer die beim Aufruf angegebenen Anfangsargumente, es sei denn die Option *-I* (bzw. *-i*) ist gesetzt.

anfangs_argument nicht angegeben:
Die Argumentlisten werden nur aus den von der Standard-Eingabe eingelesenen Argumenten aufgebaut.

Endestatus

0 Jeder Aufruf von *kommando* hat den Ende-Status 0 geliefert.

1-125 Die Argumente konnten nicht wie gefordert zusammengestellt werden oder mind. ein Aufruf von *kommando* hat einen Ende-Status ungleich 0 geliefert oder ein anderer Fehler ist aufgetreten.

126 Das angegebene *kommando* existiert, kann aber nicht aufgerufen werden.

127 Das angegebene *kommando* ist nicht auffindbar.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *xargs*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_COLLATE Legt die internationale Umgebung für das Verhalten von Bereichen, Äquivalenzklassen und Zeicheneinheiten in erweiterten regulären Ausdrücken für ja/nein-Abfragen fest.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten) sowie das Verhalten von Zeichenklassen in erweiterten regulären Ausdrücken bei ja/nein-Abfragen.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

PATH Legt den bei der Kommandosuche verwendeten Suchpfad fest.

Beispiel 1 Mit der folgenden Shell-Prozedur *schiebe* werden alle Dateien in einem Dateiverzeichnis, deren Namen nicht mit einem Punkt . beginnen, in ein anderes Dateiverzeichnis übertragen:

```
$ cat schiebe
ls $1 | xargs -I {} -t mv $1/{} $2/{}
$ schiebe dir1 dir2
mv dir1/datei1 dir2/datei1
mv dir1/datei2 dir2/datei2
mv dir1/datei3 dir2/datei3
```

Für die beiden Stellungsparameter *$1* und *$2* werden die beim Aufruf von *schiebe* angegebenen Argumente *dir1* und *dir2* eingesetzt. Das Kommando *ls $1* gibt den Inhalt des Dateiverzeichnisses *dir1* aus, wobei in einer Zeile jeweils ein Dateiname steht. Diese Dateinamen werden nacheinander für *{}* eingesetzt (Option *-I {}*). Vor jedem Aufruf des Kommandos *mv* wird das Kommando und die Argumentliste ausgegeben (Option *-t*).

Beispiel 2 Die folgende Shell-Prozedur *wer_und_wann* hängt die Ausgabe der in runden Klammern (...) zusammengefassten Kommandos in einer Zeile an das Ende der Datei *log* an:

```
$ cat wer_und_wann
(logname; date; echo $0 $*) | xargs >>log
$ cat log
michael Mon Mar 9 14:21:06 MEZ 2009 wer_und_wann
```

Beispiel 3 Die folgende Shell-Prozedur *archiviere* archiviert die Dateien im aktuellen Dateiverzeichnis, deren Namen nicht mit einem Punkt . beginnen, in einem Archiv *archiv.a* (siehe *ar*):

```
$ cat archiviere
ls | xargs -p -L 1 ar r archiv.a
$ archiviere
ar r archiv.a datei1 ?... y
ar: creating archiv.a
ar r archiv.a datei2 ?... n
ar r archiv.a datei3 ?... y
ar r archiv.a datei4 ?... n
$ ar t archiv.a
datei1
datei3
```

ls gibt den Inhalt des aktuellen Dateiverzeichnisses auf die Standard-Ausgabe aus, wobei in jeder Zeile jeweils ein Dateiname steht. *xargs* ruft dann *ar* mit den Argumenten *r*, *archiv.a* und jeweils einem Dateinamen, den *ls* liefert auf. Weil die Option *-p* gesetzt ist, werden Sie jedesmal gefragt, ob der entsprechende *ar*-Aufruf ausgeführt werden soll oder nicht. Wenn Sie diese Frage das erste Mal bejahen, legt *ar* das Archiv *archiv.a* an und gibt eine entsprechende Meldung aus und archiviert die aktuelle Datei in *archiv.a*. Danach werden weitere Dateien in *archiv.a* archiviert, wenn Sie die Fragen bejahen. Zum Schluss können Sie sich mit *ar t* das Inhaltsverzeichnis von *archiv.a* ausgeben lassen.

Siehe auch *echo*

yacc Parser erstellen (yet another compiler-compiler)

Das Kommando *yacc* (yet another compiler-compiler) wandelt eine kontextfreie Grammatik in eine Menge von Tabellen für einen einfachen Automaten um, der einen *LALR(1)* Syntaxanalyse-Algorithmus ausführt. Die Grammatik kann mehrdeutig sein; für die Auflösung dieser Mehrdeutigkeiten werden bestimmte Regeln angewendet.

Die Ausgabedatei *y.tab.c* (bzw. *datei_präfix.tab.c*, falls die Option *-b* angegeben ist) muss zur Erstellung eines Programms *yyparse* vom C-Übersetzer übersetzt werden. Dieses Programm muss mit dem lexikalischen Analyse-Programm *yylex* sowie mit *main* und *yyerror*, einer Fehlerbehandlungsroutine, gebunden werden. Diese Routinen sind vom Benutzer bereitzustellen. Das Kommando *lex* ist zum Erstellen lexikalischer Analysatoren, die von *yacc* verwendet werden können, nützlich.

Syntax **yacc**␣[␣**-dltv**][␣**-b**␣datei_präfix][-p␣sym_präfix]|␣**-y**␣driver_file][␣**-V**][␣**-Q[y**|**n]**][␣datei ...]

Folgende Optionen können beim Aufruf von *yacc* verwendet werden:

-d erzeugt die Datei *y.tab.h* mit den *#define*-Anweisungen, die die von `yacc` zugewiesenen Token-Codes mit den vom Benutzer angegebenen Token-Namen in Verbindung setzen. Damit können auch andere Quelldateien als *y.tab.c* auf die Token-Codes zugreifen.

-l gibt an, dass der in *y.tab.c* erzeugte Code (bzw. *datei_präfix.tab.c*, falls die Option *-b* angegeben ist) keine *#line*-Anweisungen enthält. Verwenden Sie diese Option nur, wenn Grammatik und zugehörige Aktionen völlig ausgetestet, d.h. fehlerfrei sind.

-t übersetzt standardmäßig Code zur Unterstützung der Fehlersuche zur Laufzeit. Der Fehlersuchcode wird in *y.tab.c* (bzw. *datei_präfix.tab.c*, falls die Option *-b* angegeben ist) immer generiert, steht jedoch unter bedingter Übersetzungssteuerung. Standardmäßig wird dieser Code bei der Übersetzung von *y.tab.c* nicht berücksichtigt. Unabhängig von der Verwendung der Option *-t* wird die Bereitstellung des Codes für die Laufzeit-Fehlersuche durch das Präprozessor-Symbol `YYDEBUG` gesteuert. Wenn `YYDEBUG` einen Wert ungleich Null hat, wird der Fehlersuchcode eingebunden. Ist der Wert gleich Null, wird der Code nicht eingebunden. Größe und Ablaufzeit eines ohne Fehlersuchcode erstellten Programms ist kleiner bzw. kürzer.

-v stellt die Datei *y.output* (bzw. *datei_präfix.output*, falls die Option *-b* angegeben ist) bereit. Sie enthält eine Beschreibung der Syntaxanalyse-Tabellen und eine Meldung über Konflikte, die aus Mehrdeutigkeiten in der Grammatik entstanden sind.

-b␣datei_präfix
Die Ausgabedateien beginnen mit dem Präfix *datei_präfix* anstelle von y. Auch sonstige von *yacc* benötigte Dateien wie y.tab.c, y.tab.h und y.output erhalten dann die Dateinamen *datei_präfix*.tab.c, *datei_präfix*.tab.h und *datei_präfix*.output.

-p␣sym_präfix
Alle von *yacc* erzeugten externen Namen beginnen mit dem Präfix *sym_präfix* anstelle von yy. Davon betroffen sind auch die Funktionen *yyparse()*, *yylex()* und *yyerror()* und die Variablen *yylval*, *yychar* und *yydebug*. Interne Namen können auch von der Option *-p* betroffen sein. Die Option *-p* wirkt jedoch nicht auf #define-Symbole, die von *yacc* erzeugt werden.

-Q[y/n]
legt fest, ob die Versionsinformation über die erstellte *yacc*-Version in *y.tab.c* (bzw. *datei_präfix.tab.c*, falls die Option *-b* angegeben ist) geschrieben werden soll (y) oder nicht (n).

y/n steht für eine ja/nein-Angabe in der jeweils eingestellten Sprachumgebung. In einer englischsprachigen Umgebung geben Sie *-Qy* an, um die Versionsinformation in die Datei *y.tab.c* zu schreiben und *-Qn*, um keine Information zu schreiben. In einer deutschsprachigen Umgebung müssen Sie beispielsweise *-Qj* oder *-Qn* angeben. Standardmäßig wird keine Versionsinformation geschrieben.

-V druckt die Versionsinformation für *yacc* auf die Standard-Fehlerausgabe.

-y␣driver_file
definieren einer persönlichen *yaccpar*-Datei.

Datei
y.output, y.tab.c
y.tab.h Definitionen von Token-Namen
(Anstelle von *y.* wird *datei_präfix.* verwendet, wenn die Option *-b* angegeben ist)
yacc.tmp, yacc.debug, yacc.acts temporäre Dateien
/usr/lib/yaccpar Analysealgorithmus-Prototyp für C-Programme
/usr/ccs/lib/liby.a für das Binden benötigte Module

Endestatus Die Anzahl der reduziere/reduziere- und lies/reduziere-Konflikte wird auf der Standard-Fehlerausgabe gemeldet; ein detaillierterer Bericht ist in der Datei *y.output* (bzw. *datei_präfix.output*, falls die Option *-b* angegeben ist) zu finden. Eine Meldung erfolgt auch dann, wenn einige Regeln nicht vom Startsymbol aus erreichbar sind.

Hinweis Da standardmäßig die Dateinamen festgelegt sind (Option *-b* nicht angegeben), kann jeweils nur höchstens ein *yacc*-Prozess zu einem gegebenen Zeitpunkt in einem gegebenen Dateiverzeichnis aktiv sein.

Wird mit *c89* [5] ein *yacc*-Programm gebunden, muss als Bibliotheksparameter *-ly* angegeben werden.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *yacc*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten). Außerdem legt *LC_CTYPE* fest, welche Zeichen in der aktuell gültigen Umgebung als Buchstaben und Ziffern definiert sind. Dies ist bei Symbolnamen von Bedeutung.
Hat die Variable *LC_CTYPE* den Wert `De_DE.646`, werden z.B. der senkrechte Strich | und die öffnende geschweifte Klammer { als Buchstaben interpretiert (ihr Code entspricht in der deutschen Variante des Zeichensatzes ISO 646 den Umlauten `ö` und `ä`). Dadurch kann *yacc* die Grammatiken nicht mehr richtig auswerten. In einer solchen Umgebung können Sie deshalb nicht mit *yacc* arbeiten. Setzen Sie in diesem Fall entweder die Variable *LC_CTYPE* auf den Wert `En_US.ASCII` oder weisen Sie der Variablen *LANG* und ggf. auch der Variablen *LC_CTYPE* die leere Zeichenkette zu.

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Siehe auch *lex*

zcat Komprimierte Dateien ausgeben (expand and concatenate compressed data)

zcat gibt den Original-Zustand einer mit *compress* komprimierten Datei auf die Standard-Ausgabe aus. Die komprimierte Datei bleibt unverändert. Mit dem Kommando *uncompress* bringen Sie eine Datei, die Sie mit *compress* komprimiert haben, in ihren Original-Zustand.

zcat gehört, zusammen mit dem Programmen *compress* und *uncompress* zu einer Gruppe von Kommandos, mit der Sie Dateien komprimieren, dekomprimieren und komprimierte Dateien ausgeben können. *zcat* ist identisch mit *uncompress -c*.

Syntax **zcat**[␣datei...]

datei
: Name der komprimierten Datei, deren Originalzustand ausgegeben werden soll. Sie können mehrere Dateien angeben.
Die Namen der komprimierten Dateien können Sie mit oder ohne das Suffix *.Z* angeben. Wenn Sie den Namen ohne *.Z* angeben, sucht *zcat* nach der entsprechenden *.Z*-Datei.

Fehler Bei folgenden Fehlern wird das Kommando *zcat* nicht ausgeführt.

dateiname: `no such file or directory`
Die angegebene Datei ist nicht vorhanden.

dateiname: `not in compressed format`
Die angegebene Datei liegt nicht in komprimiertem Datenformat vor.

dateiname: `compressed with xxbits, can only handle yybits`
Die Datei wurde von einem Programm komprimiert, dessen Code mehr Bits verarbeiten kann, als der Komprimierungscode dieser Maschine. Sie können versuchen, die Datei nochmals mit einer kleineren Bitanzahl zu komprimieren.

`uncompress: corrupt input`
Das Signal SIGSEGV (Adressfehler wegen unerlaubten Segmentzugriffs) wurde empfangen. Dies bedeutet normalerweise, dass die Eingabedatei beschädigt ist.

Internationale Umgebung

Die folgenden Umgebungsvariablen beeinflussen die Ausführung des Kommandos *zcat*:

LANG Gibt einen Standardwert für die Variablen für die internationale Umgebung an, die nicht gesetzt oder Null sind. Ist *LANG* nicht gesetzt oder Null, wird der entsprechende Standardwert der internationalen Umgebung verwendet. Enthält eine der Internationalisierungsvariablen eine ungültige Einstellung, verhält sich das Kommando so, als sei keine der Variablen definiert worden.

LC_ALL Ist diese Variable auf einen Wert gesetzt, d. h. ist sie nicht leer, überschreibt dieser Wert die Werte aller übrigen Internationalisierungsvariablen.

LC_CTYPE Legt die internationale Umgebung für die Interpretation der Byte-Folgen eines Datentexts als Zeichen fest (z.B. Singlebytezeichen im Unterschied zu Mehrbytezeichen in Argumenten).

LC_MESSAGES
Legt die internationale Umgebung für Format und Inhalt der Diagnosemeldungen fest, die in die Standardfehlerausgabe geschrieben werden.

NLSPATH Legt den Pfad der Meldungsdateien für *LC_MESSAGES* fest.

Beispiel Die Datei *zcat_bsp* wird mit *cat* ausgegeben, dann mit *compress* komprimiert und anschließend wird die komprimierte Datei mit *zcat* ausgegeben.

```
$ cat zcat_bsp
Noch bin ich nicht komprimiert!
$ compress -fv zcat_bsp
zcat_bsp: Compression: -16.12% -- replaced with zcat_bsp.Z

$ zcat zcat_bsp.Z
Noch bin ich nicht komprimiert!
```

Siehe auch *cat*, *compress*, *uncompress*

: Endestatus 0 zurückgeben (return true value)

Das in die POSIX-Shell *sh* eingebaute Kommando *:* (Doppelpunkt) gibt den Endestatus 0 zurück und tut sonst nichts. In Shell-Prozeduren wird es wie folgt verwendet:

– Ohne Aufruf-Argumente verhält es sich wie das Kommando *true*. Sie können also auch mit dem eingebauten *sh*-Kommando *:* die Bedingung *wahr* erzeugen.

– Geben Sie beim Aufruf zusätzlich Argumente an, interpretiert die Shell alle vorkommenden Sonderzeichen.
Auf diese Weise können Sie mit dem eingebauten *sh*-Kommando *:* nicht belegten Shell-Variablen einen Standard-Wert zuweisen, ohne dass eine Aktion ausgelöst wird.

– Wenn Sie in den Aufruf-Argumenten keine Sonderzeichen der Shell verwenden bzw. diese entwerten, hat das eingebaute *sh*-Kommando *:* dieselbe Funktion wie das Nummernzeichen #. Es leitet Kommentare ein. Das erste, nicht entwertete Kommando-Trennzeichen nach *:* beendet aber, im Gegensatz zu #, diese Art von Kommentar.

Syntax

```
:[␣argument]...
```

argument
beliebige Zeichenkette, die jeweils durch Leer- oder Tabulatorzeichen begrenzt wird. Das letzte Argument wird durch ein Kommando-Trennzeichen abgeschlossen.

Sie können beliebig viele Argumente angeben, jeweils getrennt durch mindestens ein Tabulator- bzw. Leerzeichen.

Wie bei jedem anderen Kommando auch wird die Zeichenkette zunächst von der Shell interpretiert. Das eingebaute *sh*-Kommando *:* gibt nur den Endestatus 0 zurück.

argument nicht angegeben:
Das eingebaute *sh*-Kommando *:* gibt nur den Endestatus 0 zurück und tut sonst nichts.

Endestatus 0 in jedem Fall

Beispiel 1 Sie können mit dem eingebauten *sh*-Kommando *:* den Zweig einer if- oder case-Anweisung füllen, falls in diesem Zweig nichts passieren soll. Die Shell-Prozedur *kein_x* hat folgenden Inhalt:

```
if test -x $1
then :
else echo $1 ist nicht ausfuehrbar!
fi
```

Die Shell-Prozedur testet die Datei, die Sie beim Aufruf als erstes Argument angeben. Ist diese Datei ausführbar, so tut die Shell-Prozedur nichts. Ist die Datei nicht ausführbar, so erhalten Sie die Meldung "*datei* ist nicht ausfuehrbar!".

Beispiel 2 In der folgenden Shell-Prozedur wird der Shell-Variablen *name* die Prozessnummer des aktuellen Shell-Prozesses zugewiesen, falls diese Variable noch nicht definiert ist oder die leere Zeichenkette enthält:

```
: ${name:=$$}
echo $name
```

Siehe auch *true*

. Shell-Prozeduren in der aktuellen Shell ausführen (execute commands in current environment)

Das in die POSIX-Shell *sh* eingebaute Kommando . (Punkt) führt die angegebene Shell-Prozedur in der aktuellen Shell aus.

Wenn Sie in der Shell-Prozedur Shell-Variablen neu definieren oder die Werte vorhandener Shell-Variablen ändern, sind diese Variablen in der Ablaufumgebung der aktuellen Shell eingetragen. Wenn Sie in der Shell-Prozedur mit dem eingebauten Kommando *set* Shell-Optionen setzen oder zurücksetzen, sind diese Optionen in der aktuellen Shell gesetzt oder zurückgesetzt.

An die Shell-Prozedur können Sie beim Aufruf nur Schlüsselwortparameter übergeben, aber nicht die Stellungsparameter neu setzen. Innerhalb der Shell-Prozedur können Sie jedoch auf die Stellungsparameter der aktuellen Shell zugreifen.
Wenn Sie in der Shell-Prozedur mit *set* die Stellungsparameter neu setzen, sind diese Stellungsparameter in der aktuellen Shell gesetzt.

Syntax

```
.␣datei
```

datei
: Name der Shell-Prozedur, die von der aktuellen Shell ausgeführt werden soll. Die Shell sucht *datei* in den Dateiverzeichnissen, deren Pfadnamen der Variablen *PATH* zugewiesen sind. Für die angegebene Datei brauchen Sie das Leserecht.

Variable *PATH*
Suchpfad der Shell

Internationale Umgebung
Die Umgebungsvariable *LC_MESSAGES* bestimmt die Sprache der Meldungstexte. Wenn *LC_MESSAGES* nicht oder als leere Zeichenkette definiert ist, wird der Wert von *LANG* als Standardwert herangezogen. Ist auch *LANG* nicht oder als leere Zeichenkette definiert,verhält sich das System so, als wäre es nicht internationalisiert.
Die Umgebungsvariable *LC_ALL* bestimmt die gesamte internationale Umgebung. *LC_ALL* hat Vorrang vor allen anderen Umgebungsvariablen im Bereich der Internationalisierung.

Beispiel Wenn die Datei *$HOME/.profile* neu angelegt oder verändert wird, werden die darin enthaltenen Kommandos und Zuweisungen erst von der nächsten Login-Shell ausgeführt.
Mit der folgenden Eingabe werden diese Änderungen bereits in der aktuellen Shell wirksam; dabei ist vorausgesetzt, dass der Variablen *PATH* das aktuelle Dateiverzeichnis zugewiesen ist:

```
$ . .profile
```

Siehe auch *sh*

[␣␣] Bedingungen prüfen (evaluate expression)

Das in die POSIX-Shell *sh* eingebaute Kommando *[␣␣]* prüft, ob Bedingungen erfüllt sind. Bedingungen können sein: Eigenschaften von Dateien, Eigenschaften und Vergleiche von Zeichenketten und algebraische Vergleiche ganzer Zahlen.

Sie können Bedingungen auch verneinen; mehrere Bedingungen können Sie miteinander verknüpfen. Als Ergebnis liefert *[␣␣]* zurück:
- Endestatus 0 (wahr), falls die Bedingung erfüllt ist.
- Endestatus 1 (falsch), falls die Bedingung nicht erfüllt ist oder falls Sie die Bedingung nicht vollständig angegeben haben. Den gleichen Endestatus erhalten Sie, wenn Sie keine Bedingung angeben.

Abhängig vom Endestatus können Sie unterschiedliche Kommandos ausführen, Schleifen abbrechen usw.

Für das eingebaute *sh*-Kommando *[␣␣]* gibt es zwei Schreibweisen (siehe Syntax). Die Wirkung ist dieselbe.

Syntax

test[␣ausdruck]
[[␣ausdruck␣**]]**

Die Beschreibung der Operanden finden Sie beim eingebauten *sh*-Kommando *test*.

Beispiel

Die Prozedur *dr* gibt eine Liste des aktuellen Dateiverzeichnisses aus. Dabei wird für jeden in dem Verzeichnis enthaltenen Namen mit dem Kommando *[-d "$name"]* geprüft, ob es sich um ein weiteres Dateiverzeichnis handelt. Entsprechend wird vor dem Namen die Kennzeichnung *"(dvz)"* oder die Dateilänge ausgegeben:

```
$ cat dr
for name in *
do if [ -d "$name" ]
   then echo "(dvz) $name"
   else echo " `ls -s $name`"
   fi
done
$ dr
    2 angebot
(dvz) briefe
    2 dr
    2 notiz
(dvz) schreiben
(dvz) texte
```

Weitere Beispiele und Erläuterungen finden Sie bei der Beschreibung des gleichwertigen Kommandos *test*.

Siehe auch *test*

5 Tabellen und Verzeichnisse

5.1 Übersicht über XPG4-Konformität der Kommandos

Kommandos, die genau dem XPG4-Standard entsprechen

asa, at, awk, basename, batch, bc, cal, cancel, cmp, comm, compress, cp, csplit, cut, dd, dirname, du, env, expand, expr, fold, getconf, head, join, locale, localedef, logger, logname, mesg, mkdir, mkfifo, mv, nice, nm, nohup, paste, patch, pathchk, pax, pr, printf, renice, rm, rmdir, sed, sleep, split, sum, tabs, talk, tee, time, tput, tr, tsort, tty, touch, uncompress, unexpand, uniq, uudecode, uuencode, wc, who, write, xargs, zcat

Kommandos, die Abweichungen gegenüber dem XPG4-Standard zeigen

Die Abweichungen treten meist in Form zusätzlicher Optionen auf.

POSIX-Kommando	zusätzliche Option(en)	andere Abweichungen
ar	-S, -V	
cat	-s	
chgrp	-h	
chmod		bei Angabe von <perm> wird zusätzlich zum XPG4-Standard auch die Angabe von l (lock) und t (t-bit) sowie u, g und o unterstützt
chown	-h	
cksum	-C	
crontab	-ll-rl-e [user]	
date	-a	die Systemuhr kann nur um Sekunden und Sekundenbruchteile nach-/vorgestellt werden
df	-F, -n, -b, -c, -e, -g, -l, -v, -V, -o	
diff	-a, -h, -i, -l. -n, -s, -t, -w, -D, -S	

POSIX-Kommando	zusätzliche Option(en)	andere Abweichungen
ed	-W	
egrep	-b, -h	
ex	-r, -L	
fc	-s	
fgrep	-b, -h	
file	-c, -f, -h, -m	
find	-follow, -fstype -local, -mount	für die Optionen -links, -size, -atime, -ctime, -mtime und -size ist +/-<num> zulässig (statt nur <num>) für -type <char> ist auch ein symbolischer Verweis zulässig
gencat	-l, -m	
grep	-b, -h, -r	
iconv		zusätzliche Konvertierung zwischen ISO646 und EDF03 (von ASCII-7-bit nach EBCDIC)
id	-a[<user>]	
lex	-V, -Q(y/n)	
ln	-n	zusätzliche Syntax gegenüber XPG4-Standard: ln -s <name><verweis> ln -s <name>...<dateiverzeichnis>
lp		Die Optionen -m und -w des XPG4-Standards werden nicht unterstützt.
lpstat		Die Option -v des XPG4-Standards wird nicht unterstützt. Für die Option -u kann nur die eigene Benutzerkennung angegeben werden.
ls	-L, -b	
mailx	-F, -i, -n -V	im Sendemodus im Lesemodus
make	-b, -d	
man	-k, -x	
more	-d, -f, +<zeichen>, +<zeilennr>, +/-<muster>	Die Option -t des XPG4-Standards wird nicht unterstützt.
nice		kein Einfluss auf die BS2000-Taskprioritäten
nl	-f␣typ	
od	-f, -D, -F, -O, -S, -X	
ps	-c, -j, -s, -T	
renice		kein Einfluss auf die BS2000-Taskprioritäten

POSIX-Kommando	zusätzliche Option(en)	andere Abweichungen
sh		in die Shell eingebaute XPG4-Funktionen: ., :, [<expr>], alias, bg, break, case, cd, command, continue, do, echo, else, eval, exec, exit, export, false, fc, fg, for, getopts, hash, if, jobs, kill, newgrp, pwd, read, readonly, return, set, shift, test, times, trap, true, type, ulimit, umask, unalias, unset, until, wait, while Sonstige in die Shell eingebaute Funktionen: edt, let, print, typeset, whence
sort	-M, -T␣<tempdir>	
strings	-o	
tail	-r	
tput	-S	
uname	-p	
uuname		Es wird nur der lokale Systemname ausgegeben.
vi	-r, -L	
yacc	-V, -Q(y/n) y driver-file	

Kommandos, die nicht oder nicht mehr im XPG4-Standard definiert sind

POSIX-Kommando	Funktion
bs2cmd	BS2000-Kommando ausführen
bs2cp	BS2000-Dateien kopieren
bs2do	BS2000-Prozeduren aus der POSIX Shell aufrufen
bs2file	Dateiattribute für BS2000-Dateien festlegen
bs2lp	Dateien ausdrucken
bs2pkey	P-Tasten belegen
cpio	Dateien und Dateiverzeichnisse ein- und auslagern
debug	POSIX-Programme testen
dumpfs	Interne Dateisystem-Information ausgeben
edt, edtu	BS2000-Dateibearbeiter EDT aufrufen
fsck	Konsistenz eines Dateisystems prüfen und korrigieren
fsexpand	Existierende Dateisysteme vergrößern
ftyp	Bearbeitungsart für Dateien festlegen
fuser	Dateinutzer anzeigen

POSIX-Kommando	Funktion
hd	Dateiinhalt hexadezimal ausgeben
info	Online-Diagnosetool
ipcrm	Interprozess-Kommunikationseinrichtungen löschen
ipcs	Zustand von Interprozess-Kommunikationseinrichtungen anzeigen
last	Zuletzt angemeldete Benutzer anzeigen
logrotate	Wechsel der Protokolldateien des syslog-Dämonen
mkfs	Dateisystem erzeugen
mknod	Gerätedatei erzeugen
mkpart	Dienstprogramm zur Festplattenverwaltung
mount	Dateisysteme einhängen
mountall	Mehrere Dateisysteme einhängen
pdbl	Benutzerspezifische Programm-Caches einrichten und verwalten
ping	Senden von Echo-Request-Paketen an Netzwerkkomponenten
pkginfo	Informationen über Software-Pakete im POSIX ausgeben
posdbl	POSIX-Lader verwalten
rcp	Dateien von oder zu einem fernen Rechner kopieren
rsh	Shell-Kommando am fernen Rechner ausführen
rmpart	Partition aus Partionstabelle entfernen
show_pubset_export	vom EXPORT-PUBSET betroffene Dateisysteme anzeigen
start_bs2fsd	Kopierdämonen starten
su	Benutzerkennung wechseln
sync	Systempuffer zurückschreiben
tar	Dateien archivieren
umount	Dateisysteme aushängen
umountall	Mehrere Dateisysteme aushängen

5.2 Reguläre Ausdrücke der POSIX-Shell

Reguläre Ausdrücke werden verwendet, um in einem Text nach Stellen zu suchen, die zu einem vorgegebenen Muster passen. Ein regulärer Ausdruck steht für eine Menge von Zeichenketten. Von jeder Zeichenkette in dieser Menge sagt man, dass sie zu dem regulären Ausdruck passt. Ein oder mehrere reguläre Ausdrücke bilden ein Muster.

Ein regulärer Ausdruck besteht aus einer Folge von Zeichen. Bei diesen Zeichen unterscheidet man

– einfache Zeichen und

– Sonderzeichen.

Einfache Zeichen sind alle Zeichen im Zeichensatz außer dem Neue-Zeile-Zeichen und den Sonderzeichen. Einfache Zeichen in einem Muster stehen für sich selbst, z.B. passen zu dem Muster *abc* nur diejenigen Zeichenketten, in denen die Folge *abc* an irgendeiner Stelle enthalten ist.

Sonderzeichen stehen nicht für sich selbst, sondern haben eine besondere Bedeutung. Diese ist unten erläutert.

Man unterscheidet zwei Arten regulärer Ausdrücke:

– einfache reguläre Ausdrücke

– erweiterte reguläre Ausdrücke

Wie die verschiedenen Arten regulärer Ausdrücke gebildet werden, wird in den folgenden Abschnitten beschrieben.

Die folgende Tabelle gibt einen Überblick darüber, welche Kommandos reguläre Ausdrücke verarbeiten:

Kommando	**Art der regulären Ausdrücke**
awk	erweiterte
ed	einfache
egrep	erweiterte
ex	*)
expr	einfache
grep	einfache
lex	erweiterte
nl	einfache
sed	einfache
vi	*)

*) Die Kommandos *ex* und *vi* verarbeiten reguläre Ausdrücke, die von den einfachen regulären Ausdrücken in einigen Punkten abweichen. Dies ist bei *ex* und *vi* beschrieben.

Einfache reguläre Ausdrücke

Einfache reguläre Ausdrücke werden wie folgt gebildet:

Nr.	regulärer Ausdruck	Bedeutung	Beispiel	passende Zeichenketten
1	c	Das Zeichen *c*, wobei *c* kein Sonderzeichen sein darf.	a	a
2	**** c	Das Zeichen *c*, wobei *c* jedes Zeichen sein darf außer *( ) { } 1 2 3 4 5 6 7 8 9* Sinnvoll ist ein regulärer Ausdruck der Form \ *c*, wenn *c* ein Sonderzeichen ist. \ *c* steht dann für das Zeichen *c*, Sonderzeichen verlieren durch einen vorangestellten Gegenschrägstrich ihre Sonderbedeutung (Entwertung von Sonderzeichen).	\ a \ *	a *
3		Ein beliebiges Zeichen.	.	a, x, *, ...
4	**[**s**]**	Eines der Zeichen, die in der Zeichenkette *s* enthalten sind. Wenn eines der Zeichen die schließende eckige Klammer] sein soll, muss diese an erster Stelle stehen. Wenn eines der Zeichen der Bindestrich - sein soll, muss dieser an erster oder an letzter Stelle stehen. Wenn eines der Zeichen das Dach ^ sein soll, darf dieses an beliebiger Stelle stehen außer an erster.	[mz] []a] [-a] [a-] [a^]	m, z], a -, a -, a a, ^
	[c1**-**c2**]**	Ein beliebiges Zeichen aus dem Bereich von *c1* bis *c2*, gemäß der EBCDIC-Sortierreihenfolge (Grenzen *c1* und *c2* eingeschlossen). *c1* muss in der EBCDIC- Sortierreihenfolge vor *c2* stehen. Ist dies nicht der Fall, dann steht *c1-c2* nicht für einen Bereich, sondern für die beiden Zeichen *c1* und *c2*. Eine Kombination der beiden Formen ist möglich: **[**s1c1**-**c2s2**]**	[a-m] [m-a] [ado-qxz]	a, m sowie jedes Zeichen, das in der EBCDIC-Sortierreihenfolge dazwischen steht m, a a, d, o, q, x, z sowie jedes Zeichen zwischen o und q (EBCDIC)

Nr.	regulärer Ausdruck	Bedeutung	Beispiel	passende Zeichenketten
5	**[^s]**	Eines der Zeichen, die nicht in der Zeichenkette *s* enthalten sind.	[^xyz] [^0-9]	jedes Zeichen außer x, y, z
	[^c1-c2]	Ein beliebiges Zeichen, das nicht im Bereich von *c1* bis *c2* liegt. Das für [*c1-c2*] Gesagte gilt analog.		jedes Zeichen außer 0, 9 und allen Zeichen, die zwischen 0 und 9 stehen (EBCDIC)
		Eine Kombination der beiden Formen ist möglich: **[^s1c1-c2s2]**	[^a0-9b]	jedes Zeichen außer a, b, 0, 9 und allen Zeichen, die zwischen 0 und 9 stehen (EBCDIC)
6	r*	Null-, ein- oder mehrmals der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 12, 15 oder 16 sein.	a*	nichts, a, aa, aaa, ...
7	r\ {m,n\ }	Mindestens *m*- und höchstens *n*-mal der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 12, 15 oder 16 sein.	a\ {1,2\ }	a oder aa
	r\ {m\ }	Genau *m*-mal der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 12, 15 oder 16 sein.	a\ {3\ }	aaa
	r\ {m,\ }	Mindestens *m*-mal der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 12, 15 oder 16 sein.	a\ {3,\ }	aaa, aaaa, aaaaa, ...
8	rx	(Verkettung) Aufeinanderfolge einer zum regulären Ausdruck *r* passenden Zeichenkette und einer zum regulären Ausdruck *x* passenden Zeichenkette. *r* und *x* dürfen beliebige reguläre Ausdrücke sein.	[ab].	ax, a3, a*, bz, ...
9	^r	Eine zum regulären Ausdruck *r* passende Zeichenkette am Zeilenanfang, d.h. direkt nach einem Neue-Zeile-Zeichen oder am Dateianfang. *r* darf ein beliebiger regulärer Ausdruck sein, außer von der Form 9.	^[aA]pfel	apfel oder Apfel am Zeilenanfang

Nr.	regulärer Ausdruck	Bedeutung	Beispiel	passende Zeichenketten
10	**r$**	Eine zum regulären Ausdruck *r* passende Zeichenkette am Zeilenende, d.h. direkt vor einem Neue-Zeile-Zeichen. *r* darf ein beliebiger regulärer Ausdruck sein, außer von der Form 10.	[bB]irne$	birne oder Birne am Zeilenende
11	**\ (r\ )**	Zeichenketten, die zum regulären Ausdruck *r* passen. *r* darf ein beliebiger regulärer Ausdruck sein. Nur sinnvoll zusammen mit Nr. 12	\ ([aA]pfel\ )	apfel, Apfel
12	**** n	*n* ist eine ganze Zahl von 1 bis 9. Kommt \ *n* in einem zusammengesetzten regulären Ausdruck vor, steht es für den regulären Ausdruck *x*, wobei *x* der *n*-te in \ (und \ ) eingeschlossene reguläre Ausdruck ist, der in dem zusammengesetzten regulären Ausdruck vorkommt.	\ (a\ (b\ )\ )\ 2 \ (hal\ )li\ 1lo \ (ab\ )x\ 1*	abb hallihallo abx, abxab, abxabab, ...

Priorität

Die folgende Tabelle zeigt die Priorität der Operatoren in regulären Ausdrücken.

Operator	Priorität
[. .] [= =] [: :]	höchste Priorität
\<zeichen>	.
[ ]	.
( )	.
* ? + \ {m,n\ }	.
Verkettung	.
^ $	.
|	niedrigste Priorität

Sonderzeichen

Sonderzeichen	Das linksstehende Zeichen ist ein Sonderzeichen, wenn ...
\	- ihm kein Gegenschrägstrich \ vorangestellt ist.
. [	- ihm kein Gegenschrägstrich \ vorangestellt ist und - es nicht in eckigen Klammern [...] steht.
*	- ihm kein Gegenschrägstrich \ vorangestellt ist, - es nicht in eckigen Klammern [...] steht, - es nicht das erste Zeichen eines Musters ist und - es nicht nach \ ) steht.
$	- es das letzte Zeichen eines Musters ist.
^	- es das erste Zeichen eines Musters ist oder - es das erste Zeichen in eckigen Klammern [...] ist.
-	- es in eckigen Klammern steht, aber nicht an erster oder letzter Stelle.
Begrenzungszeichen für reguläre Ausdrücke, wie z.B. /.../	- ihm kein Gegenschrägstrich \ vorangestellt ist.
[. [= [:	Die linksstehenden Zeichenpaare sind Sonderzeichen, wenn sie innerhalb einer Bracket Expression (also eines Ausdrucks in eckigen Klammern [ ]) vorkommen. Sie müssen mit dem äquivalenten Zeichenpaar .], =] oder :] abgeschlossen werden. Beispiel: [[:upper:]] kennzeicnet alle Großbuchstaben.

Erweiterte reguläre Ausdrücke

Erweiterte reguläre Ausdrücke umfassen die einfachen regulären Ausdrücke mit folgender Ausnahme:

Die bei einfachen regulären Ausdrücken verwendete Konstruktion \(...\) hat bei erweiterten regulären Ausdrücken *keine* Sonderbedeutung, z.B. steht der erweiterte reguläre Ausdruck \(*ab*\) für die Zeichenkette (*ab*).

Darüberhinaus bieten erweiterte reguläre Ausdrücke die folgenden zusätzlichen Syntaxelemente zur Bildung von Mustern:

Nr.	regulärer Ausdruck	Bedeutung	Beispiel	passende Zeichenketten
7	r **{**m**,**n**}**	Mindestens *m*- und höchstens *n*-mal der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 12, 15 oder 16 sein.	a {1,2 }	a oder aa
	r **{**m**}**	Genau *m*-mal der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 12, 15 oder 16 sein.	a {3}	aaa
	r **{**m**, }**	Mindestens *m*-mal der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 12, 15 oder 16 sein.	a {3, }	aaa, aaaa, aaaaa, ...
13	r**+**	Ein- oder mehrmals der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 15 oder 16 sein.	u+	u, uu, uuu, ...
14	r**?**	Null- oder einmal der reguläre Ausdruck *r*. *r* muss von der Form 1 - 5, 15 oder 16 sein.	u?	nichts oder u
15	**(**r**)**	Zeichenketten, die zu dem regulären Ausdruck *r* passen. *r* kann ein beliebiger regulärer Ausdruck sein.	(ok(abc))	okabc
			(au)*	nichts oder au, auau, ...
16	**(**r1/r2**)**	Zeichenketten, die zu dem regulären Ausdruck *r1* oder zu dem regulären Ausdruck *r2* passen.	(ok?ko)	ok oder ko

Priorität

Die folgende Tabelle zeigt die Priorität der Operatoren in erweiterten regulären Ausdrücken.

Operator	Priorität
[. .] [= =] [: :]	höchste Priorität
\<zeichen>	.
[ ]	.
( )	.
* ? + {m,n}	.
Verkettung	.
^ $	.
|	niedrigste Priorität

Beispiele

1. Einfache reguläre Ausdrücke

Muster	Bedeutung	passende Zeichenketten
ab.d	a - b - ein beliebiges Zeichen - d	abcd, abXd, ab*d, ...
ab.*d	a - b - beliebige Zeichenkette (kann auch leer sein) - d	abd, abxd, abX*Yd, ...
ab[xyz]d	a - b - entweder x oder y oder z - d	abxd, abyd, abzd
ab[^c]d	a - b - beliebiges Zeichen ungleich c - d	abbd, abXd, ab*d, ...
^abcd$	eine Zeile, die nur die Zeichenkette abcd enthält	

2. Erweiterte reguläre Ausdsrücke

Muster	Bedeutung	passende Zeichenketten
ab.+d	a - b - beliebige Zeichenkette aus einem oder mehreren Zeichen - d	abjd, abX*Yd, ...
abc?d	a - b - c oder nichts - d	abd, abcd
(abc|xyz)	abc oder xyz	abc, xyz

5.3 Sonderzeichen der POSIX-Shell

Argument- und Kommando-Trennzeichen

Sonderzeichen	Bedeutung
Leerzeichen Neue-Zeile-Zeichen Tabulatorzeichen	 Argument-Trenner, abhängig vom Inhalt der Variablen IFS
Neue-Zeile-Zeichen	Kommando-Abschluss
|	Pipe-Zeichen
;	Kommando-Abschluss
&	Kommando-Abschluss; das so abgeschlossene Kommando wird im Hintergrund gestartet
| |	ORIF; das nachfolgende Kommando wird nur ausgeführt, wenn das vorausgehende Kommando einen Ende-Status ungleich 0 zurückgibt
&&	ANDIF; das nachfolgende Kommando wird nur ausgeführt, wenn das vorausgehende Kommando als Ende-Status 0 zurückgibt

Kommandos klammern

Sonderzeichen	Bedeutung
(kommando_folge**)**	*kommando_folge* in einer Subshell ausführen
{ kommando_folge**;}**	Ausgaben aller Kommandos aus *kommando_folge* zusammenfassen

Kommando ausführen und durch Ausgabe ersetzen

Sonderzeichen	Bedeutung
` kommando`	durch die Ausgabe von *kommando* ersetzen
$(kommando)	durch die Ausgabe von *kommando* ersetzen

Argumente durch passende Dateinamen ersetzen

Sonderzeichen	Bedeutung
*	– als eigenständiges Muster: wird ersetzt durch die Liste aller Dateinamen im aktuellen Dateiverzeichnis, die nicht mit einem Punkt . beginnen. – als Bestandteil eines Musters: wird ersetzt durch kein, ein oder mehrere Zeichen entsprechend den Dateinamen, zu denen das Muster passt.
?	als eigenständiges Muster: wird ersetzt durch die Liste aller Dateinamen im aktuellen Dateiverzeichnis, die aus genau einem Zeichen bestehen, allerdings nicht durch einen Punkt. als Bestandteil eines Musters: wird ersetzt durch genau ein Zeichen entsprechend den Dateinamen, zu denen das Muster passt.
[s]	wird ersetzt durch genau eines der Zeichen, die in der Zeichenkette *s* enthalten sind, entsprechend den Dateinamen, zu denen das Muster passt.
[c1-c2]	wird ersetzt durch genau ein Zeichen aus dem Bereich von *c1* bis *c2* (Grenzen *c1* und *c2* eingeschlossen), entsprechend den Dateinamen, zu denen das Muster passt. c1 und c2 müssen einfache Zeichen sein. Welche Zeichen im Bereich *c1-c2* enthalten sind, hängt von der EBCDIC-Sortierreihenfolge ab. Eine Kombination der Ausdrücke [*s*] und [*c1-c2*] ist möglich: [s1c1-c2s2]
[!s]	wird ersetzt durch genau eines der Zeichen, die nicht in der Zeichenkette *s* enthalten sind, entsprechend den Dateinamen, zu denen das Muster passt.
[!c1-c2]	wird ersetzt durch genau ein Zeichen, das nicht im Bereich von *c1* bis *c2* liegt, entsprechend den Dateinamen, zu denen das Muster passt (siehe [*c1-c2*]). Eine Kombination der Ausdrücke [!*s*] und [!*c1-c2*] ist möglich: [!s1c1-c2s2]

Standard-Ausgabe umlenken

Sonderzeichen	Bedeutung
[n]> datei	Standard-Ausgabe [oder Filedeskriptor *n*] auf *datei* umlenken; alter Inhalt wird gelöscht
[n]> > datei	Standard-Ausgabe [oder Filedeskriptor *n*] auf *datei* umlenken; alter Inhalt bleibt erhalten
[n]> &zahl	Standard-Ausgabe [oder Filedeskriptor *n*] auf die Datei umlenken, der die Dateikennzahl *zahl* zugeordnet ist
[n]> &-	Standard-Ausgabe [oder Filedeskriptor *n*] schließen

Standard-Eingabe umlenken

Sonderzeichen	Bedeutung
[n]<datei	Standard-Eingabe [oder Filedeskriptor *n*] auf *datei* umlenken
[n]<<argument	Here-Dokument einleiten oder auf Filedeskriptor *n* umlenken
[n]<<-argument	Here-Dokument einleiten oder auf Filedeskriptor *n* umlenken; führende Tabulator-Zeichen werden entfernt
[n]<&zahl	Standard-Eingabe auf die Datei oder auf Filedeskriptor *n* umlenken, der die Dateikennzahl *zahl* zugeordnet ist
[n]<&-	Standard-Eingabe oder Filedeskriptor *n* schließen
[n]<>datei	*datei* [oder Filedeskriptor *n*] wird zum Lesen und Schreiben als Standard-Eingabe geöffnet

Shell-Variablen und -Parameter

Sonderzeichen	Bedeutung
name=wert	der Variablen *name* einen Wert zuweisen
$name	Wert der Variablen *name*; Schlüsselwort-Parameter
${name}	wie *$name*; die geschweiften Klammern grenzen den Variablennamen von nachfolgenden Ziffern bzw. Buchstaben ab
${name-standard_wert}	Ersetzung durch *standard_wert*, falls Variable *name* nicht definiert
${name=standard_wert}	Zuweisung von *standard_wert*, falls Variable *name* nicht definiert
${name?standard_wert}	Shell bricht Ausführung ab mit der Fehlermeldung *parameter : standard_wert*, falls *name* nicht definiert
${name+standard_wert}	Ersetzung durch die leere Zeichenkette, falls *name* nicht definiert Ersetzung durch *standard_wert*, falls *name* definiert
${name:-standard_wert}	Ersetzung durch *standard_wert*, falls Variable *name* nicht definiert oder ihr Wert die leere Zeichenkette ist
${name:=standard_wert}	Zuweisung von *standard_wert*, falls Variable *name* nicht definiert oder ihr Wert die leere Zeichenkette ist
${name:?standard_wert}	Shell bricht Ausführung ab mit der Fehlermeldung *parameter : standard_wert*, falls *name* nicht definiert oder ihr Wert die leere Zeichenkette ist
${name:+standard_wert}	Ersetzung durch die leere Zeichenkette, falls *name* nicht definiert oder ihr Wert die leere Zeichenkette ist Ersetzung durch *standard_wert*, falls *name* definiert und ihr Wert nicht die leere Zeichenkette ist
${#name}	Ersetzung durch die Länge der Zeichenkette, die *name* bildet. Ist *name* eines der Zeichen Stern * oder Klammeraffe @, dann ist die Ersetzung undefiniert.

Sonderzeichen	Bedeutung
${name**%**muster**}**	Passt das POSIX-Shell-Muster *muster* auf das Ende des Wertes von *name*, dann wird der Ersetzungstext aus dem Wert von *name* ohne den gelöschten auf *muster* passenden Teil gebildet. Es wird das kürzeste passende Muster gelöscht.
${name**%%**muster**}**	Passt das POSIX-Shell-Muster *muster* auf das Ende des Wertes von *name*, dann wird der Ersetzungstext aus dem Wert von *name* ohne den gelöschten auf *muster* passenden Teil gebildet. Es wird das längste passende Muster gelöscht.
${name**#**muster**}**	Sollte das POSIX-Shell-Muster *muster* auf den Anfang des Wertes von *name* passen, dann besteht der Ersetzungstext aus dem Wert von *name*, aus dem der, auf *muster* passende, Teil gelöscht wurde. Sonst wird der Wert von *name* eingesetzt. Es wird das kürzeste passende Muster gelöscht.
${name**##**muster**}**	Sollte das POSIX-Shell-Muster *muster* auf den Anfang des Wertes von *name* passen, dann besteht der Ersetzungstext aus dem Wert von *name*, aus dem der, auf *muster* passende, Teil gelöscht wurde. Sonst wird der Wert von *name* eingesetzt. Es wird das längste Auftreten des Musters gelöscht.
$0	1. Argument des Aufrufs, also Name des Kommandos, der Shell-Prozedur bzw. der aktuellen Shell
$1, $2, ... , $9	Stellungsparameter
$* "$*"	alle Aufruf-Argumente alle Aufruf-Argumente als ein einziges Argument
$@** **"$@"	alle Aufruf-Argumente alle Aufruf-Argumente als eigenständige Argumente
$#	Anzahl Aufruf-Argumente, also ohne $0
$$	Prozessnummer (PID) der aktuellen Shell
$!	Prozessnummer (PID) des zuletzt im Hintergrund gestarteten Kommandos
$?	Ende-Status des zuletzt ausgeführten Kommandos, das nicht im Hintergrund gestartet wurde
$-	alle in der aktuellen Shell gesetzten Optionen

Shell-Funktionen

Sonderzeichen	Bedeutung
name**() {** kommando_folge**;}**	Shell-Funktion; bei Aufruf von *name* werden die Kommandos aus *kommando_folge* ausgeführt

Sonderzeichen entwerten

Sonderzeichen	Bedeutung
****	entwertet das nachfolgende Sonderzeichen
' ...'	entwertet alle Sonderzeichen; ein einziges Argument
"..."	entwertet nicht die Sonderzeichen $ ` ...` \ ein einziges Argument

Sonstiges

Sonderzeichen	Bedeutung
#	Kommentar-Zeichen in Shell-Prozeduren
;;	Abschluss für Kommandofolgen innerhalb einer *case*-Anweisung
~ **~-** **~+**	Tildeersetzung
(( )) **$((expression))**	Arithmetische Berechnungen
[[ ]]	Bedingte Ausdrücke

5.4 Zeichensatz ASCII (ISO 646)

Die folgende Tabelle beinhaltet die internationale Variante des Zeichensatzes ISO 646, die US-amerikanische Variante (entspricht dem 7-bit-ASCII-Zeichensatz), die britische und die deutsche Variante.

dezimal	oktal	hexadez.		Bedeutung/Bemerkung	Control
0	00	00	NUL	Null, keine Operation	@
1	01	01	SOH	Start of Heading Vorspannanfang	A
2	02	02	STX	Start of Text Textanfang	B
3	03	03	ETX	End of Text Textende	C
4	04	04	EOT	End of Transmission Taste END Übertragungsende	D
5	05	05	ENQ	Enquiry Stationsanruf	E
6	06	06	ACK	Acknowledge Bestätigung	F
7	07	07	BEL	Bell Klingel	G
8	10	08	BS	Backspace Korrekturtaste	H
9	11	09	HT	Horizontal Tabulation Tabulatorzeichen	I
10	12	0A	LF	Line Feed Zeilenvorschub, neue Zeile	J
11	13	0B	VT	Vertical Tabulation	K
12	14	0C	FF	Form Feed Formularvorschub	L
13	15	0D	CR	Carriage Return Wagenrücklauf	M
14	16	0E	SO	Shift Out Umschalten Zeichensatz	N
15	17	0F	SI	Shift In Zurückschalten Zeichensatz	O
16	20	10	DLE	Data Link Escape Austritt aus der Datenverbindung	P

dezimal	oktal	hexadez.		Bedeutung/Bemerkung	Control
17	21	11	DC1	Device Control 1 Gerätesteuerung 1, Ausgabe fortsetzen	Q
18	22	12	DC2	Device Control 2	R
19	23	13	DC3	Device Control 3 Ausgabe anhalten	S
20	24	14	DC4	Device Control 4	T
21	25	15	NAK	Negative Acknowledge Fehlermeldung	U
22	26	16	SYN	Synchronous Idle Synchronisierung	V
23	27	17	ETB	End of Transmission Block Datenblockende	W
24	30	18	CAN	Cancel ungültig, Zeilen löschen	X
25	31	19	EM	End of Medium Datenträgerende, quit (Signal3)	\ oder |
26	32	1A	SUB	Substitute Character Zeichen ersetzen	Z
27	33	1B	ESC	Escape Rücksprung	
28	34	1C	FS	File Separator Dateitrennung	\
29	35	1D	GS	Group Separator Gruppentrennung	]
30	36	1E	RS	Record Separator Satztrennung	^
32	40	20	SP	SPACE Leerzeichen	
33	41	21	!		
34	42	22	"		
35	43	23	#	siehe 1)	
36	44	24	$	siehe 2)	
37	45	25	%		
38	46	26	&		
39	47	27	'		
40	50	28	(		
41	51	29	)		
42	52	2A	*		

dezimal	oktal	hexadez.		Bedeutung/Bemerkung	Control
43	53	2B	+		
44	54	2C	,		
45	55	2D	-		
46	56	2E	.		
47	57	2F	/		
48	60	30	0		
49	61	31	1		
50	62	32	2		
51	63	33	3		
52	64	34	4		
53	65	35	5		
54	66	36	6		
55	67	37	7		
56	70	38	8		
57	71	39	9		
58	72	3A	:		
59	73	3B	;		
60	74	3C	<		
61	75	3D	=		
62	76	3E	>		
63	77	3F	?		
64	100	40	@		
65	101	41	A		
66	102	42	B		
67	103	43	C		
68	104	44	D		
69	105	45	E		
70	106	46	F		
71	107	47	G		
72	110	48	H		
73	111	49	I		
74	112	4A	J		
75	113	4B	K		

dezimal	oktal	hexadez.		Bedeutung/Bemerkung	Control
76	114	4C	L		
77	115	4D	M		
78	116	4E	N		
79	117	4F	O		
80	120	50	P		
81	121	51	Q		
82	122	52	R		
83	123	53	S		
84	124	54	T		
85	125	55	U		
86	126	56	V		
87	127	57	W		
88	130	58	X		
89	131	59	Y		
90	132	5A	Z		
91	133	5B	[	siehe 1); Deutsch: Ä	
92	134	5C	\	siehe 1); Deutsch: Ö	
93	135	5D	]	siehe 1); Deutsch: Ü	
94	136	5E	^		
95	137	5F	_		
96	140	60	`		
97	141	61	a		
98	142	62	b		
99	143	63	c		
100	144	64	d		
101	145	65	e		
102	146	66	f		
103	147	67	g		
104	150	68	h		
105	151	69	i		
106	152	6A	j		
107	153	6B	k		
108	154	6C	l		

dezimal	oktal	hexadez.		Bedeutung/Bemerkung	Control
109	155	6D	m		
110	156	6E	n		
111	157	6F	o		
112	160	70	p		
113	161	71	q		
114	162	72	r		
115	163	73	s		
116	164	74	t		
117	165	75	u		
118	166	76	v		
119	167	77	w		
120	170	78	x		
121	171	79	y		
122	172	7A	z		
123	173	7B	{	siehe 1); Deutsch: ä	
124	174	7C	|	siehe 1); Deutsch: ö	
125	175	7D	}	siehe 1); Deutsch: ü	
126	176	7E		siehe 2); Deutsch: ß	
127	177	7F	DEL	Delete Löschzeichen, Interrupt (Signal2)	

1) ASCII: wie internationale Variante (siehe Tabelle)
Britisch: wie internationale Variante (siehe Tabelle)
Deutsch: Umlaute (siehe Tabelle)

2) ASCII: ~
Britisch: ~
Deutsch: ß

5.5 Zeichensatz EBCDIC (EDF04)

Code-Tabelle EBCDIC.DF.04

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0					SP	&	-	ø	Ø	°	µ	¢	ù	¦	Ù	0
1					NBSP	é	/	É	a	j	‾	£	A	J	÷	1
2					â	ê	Â	Ê	b	k	s	¥	B	K	S	2
3					ä	ë	Ä	Ë	c	l	t	•	C	L	T	3
4					à	è	À	È	d	m	u	©	D	M	U	4
5					á	í	Á	Í	e	n	v	§	E	N	V	5
6					ã	î	Ã	Î	f	o	w	¶	F	O	W	6
7					å	ï	Å	Ï	g	p	x	¼	G	P	X	7
8					ç	ì	Ç	Ì	h	q	y	½	H	Q	Y	8
9					ñ	ß	Ñ	¨	i	r	z	¾	I	R	Z	9
A					`	!	^	:	<<	ª	;	¬	SHY	¹	²	³
B					.	$	,	#	>>	º	¿	[	ô	û	Ô	{
C					<	*	%	@	ð	æ	Ð	\	ö	ü	Ö	Ü
D					(	)	_	‘	ý	,	Ý	]	ò	Û	Ò	}
E					+	;	>	=	Þ	Æ	þ	´	ó	ú	Ó	Ú
F					|		?	“	±	¤	®	×	õ	ÿ	Õ	~

Literatur

Die Handbücher finden Sie im Internet unter *http://manuals.ts.fujitsu.com*. Handbücher, die auch in gedruckter Form vorliegen, können Sie unter *http://manualshop.ts.fujitsu.com* bestellen.

[1] **POSIX** (BS2000/OSD)
Grundlagen für Anwender und Systemverwalter
Benutzerhandbuch

[2] **POSIX** (BS2000/OSD)
BS2000-Dateisystem bs2fs
Benutzerhandbuch

[3] **POSIX** (BS2000/OSD)
SOCKETS/XTI für POSIX
Benutzerhandbuch

[4] **C-Bibliotheksfunktionen** (BS2000/OSD)
für POSIX-Anwendungen
Referenzhandbuch

[5] **C/C++** (BS2000/OSD)
POSIX-Kommandos des C/C++-Compilers
Benutzerhandbuch

[6] **CRTE (BS2000/OSD)**
Common Runtime Environment
Benutzerhandbuch

[7] **EDT** (BS2000/OSD)
Anweisungen
Benutzerhandbuch

[8] **EDT** (BS2000/OSD)
Unicode-Modus Anweisungen
Benutzerhandbuch

[9] **NFS** (BS2000/OSD)
Network File System
Benutzerhandbuch

[10] **JV** (BS2000/OSD)
Jobvariablen
Benutzerhandbuch

[11] **BS2000/OSD-BC**
Kommandos
Benutzerhandbuch

[12] **Reliant UNIX**
Referenzhandbuch für Benutzer
Referenzhandbuch (nur Online)

[13] **Reliant UNIX**
Referenzhandbuch für Systemverwalter
Referenzhandbuch (nur Online)

[14] **SINIX**
Leitfaden für Programmierer: Internationalisierung - Lokalisierung
Programmierhandbuch

Sonstige Literatur

[15] XBD, Issue 4
X/Open CAE Specification, August 1994
System Interfaces Definitions, Issue 4, Version 2
(ISBN: 1-85912-036-9, C434)
X/Open Company Limited

[16] XSH, Issue 4
X/Open CAE Specification, August 1994
System Interfaces and Headers, Issue 4, Version 2
(ISBN: 1-85912-037-7, C435)
X/Open Company Limited

[17] XSH, Issue 4
X/Open CAE Specification, August 1994
Commands and Utilities, Issue 4, Version 2
(ISBN: 1-85912-034-2, C436)
X/Open Company Limited

[18] el Lozy, M.:
Editing in a UNIX Environment
The vi/ex Editor
Prentice-Hall, 1985

Stichwörter

B

C

F

G

L