SIEMENS
NIXDORF

SINIX

SINIX V5.40

Kommandos
Band 2: L–Z

Beschreibung

Sie haben

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

Siemens Nixdorf Informationssysteme AG
Manualredaktion STM QM 2
Otto-Hahn-Ring 6
W-8000 München 83

Fax: (089) 636-40443

email im EUnet:
man @ sieqm2.uucp

SINIX V5.40

Kommandos
Band 2: L–Z

Beschreibung

Ausgabe Dezember 1990 (SINIX V5.40)

... und Schulung?

Zu dem nachstehend beschriebenen Produkt, wie zu fast allen DV-Themen, bieten wir unsere regionalen Training Center in Berlin, Essen, Frankfurt, Hannover, Hamburg, München, Mainz, Stuttgart, Wien und Zürich Kurse an.

Auskunft und Info-Material:

Systemfamilien 7·500 und 8890	**Telefon (089) 636-48987**
Ein- und Mehrplatzsysteme	**Telefon (089) 636-42480**

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

X/Open XPG3-konform
Warenzeichen beantragt

Herausgegeben von
Siemens Nixdorf Informationssysteme AG

Inhalt

last
Letzten Benutzer oder letztes Terminal-Login ausgeben

Mit *last* können Sie alle An- und Abmeldezeiten des letzten Benutzers einer Datensichtstation erfahren.

```
last[␣-␣nummer][␣-f␣dateiname][␣name|tty]
```

-n␣nummer
-nummer
begrenzt die Anzahl der auszugebenden An- und Abmeldezeiten auf *nummer*.

-f␣dateiname
Die Datei *dateiname* wird hinsichtlich der Fehlermeldungen ausgewertet. Voreinstellung ist die Datei */var/adm/wtmp*.

Arbeitsweise

In der Datei */var/adm/wtmp* sind An- und Abmeldezeiten vermerkt. *last* holt sich aus dieser Datei alle Informationen über einen Benutzer, eine Datensichtstation oder über Benutzergruppen und deren Datensichtstationen.

Die Argumente zu *last* bezeichnen die Benutzernamen oder Datensichtstationen, die abgerufen werden sollen. Die Namen der Datensichtstationen können vollständig oder nur teilqualifiziert angegeben werden.

Folgende Informationen werden von *last* angezeigt:

- Login-Name
- Datensichtstation
- ggf. Rechnername des Benutzers, der sich eingeloggt hat.
- An- und Abmeldezeiten

Standardmäßig gibt *last* ein Protokoll aller An- und Abmeldezeiten in absteigender Reihenfolge aus. Wenn Sie bestimmte Benutzer oder Datensichtstationen als Argumente übergeben, werden nur davon die Daten ausgegeben. Übergeben Sie dagegen mehrere Argumente, wird zu jedem angegebenen Argument Information ausgegeben. *last* vermerkt aber auch, ob die abgefragte Sitzung noch nicht beendet ist oder durch ein *reboot* abgebrochen wurde. Der Pseudo-Benutzer *reboot* loggt sich ein, wenn am System ein Reboot gestartet wurde. Daher gibt die Meldung *letztes reboot* die Zeit zwischen mehreren reboots an. Wenn *last* abgebrochen wird, schreibt es auf die Standard-Ausgabe, wie weit die Suche bis zum Abbruch vorangeschritten ist. Wird es aber mit Q (quit) beendet, setzt *last* nach der Meldung auf die Standard-Ausgabe die Suche fort.

DATEIEN

/var/adm/wtmp
Abrechnungsdatei

BEISPIEL

Benutzer Udo möchte seine letzten drei Anmeldezeiten auf dem Rechner *tty06* erfahren:

```
$ last -3 udo tty06
udo    tty06              Fri  Aug   3   13:37 - 15:53  (02:15)
udo    tty06              Thu  Aug   2   10:25 - 12:56  (02:31)
udo    tty06              Wed  Aug   1   09:54 - 11:47  (01:53)
```

SIEHE AUCH

utmp() [5] bzw. [14]

layers
Multiplexer für Bildschirme mit Fensterdarstellung

layers verwaltet asynchrone Shell-Fenster auf einem Bildschirm mit Fensterdarstellung. Beim Aufruf sucht *layers* eine freie *xt*-Kanalgruppe und verbindet sie mit der Bildschirmleitung auf der Standard-Ausgabe. Dann wartet *layers* auf Kommandos von der Datensichtstation.

Jedes Shell-Fenster verhält sich wie eine eigene Datensichtstation. Zeichen von der Tastatur werden an die Standard-Eingabe des Prozesses gesandt, der mit dem aktuellen Shell-Fenster verbunden ist (der sogenannte *host*-Prozeß). Zeichen von der Standard-Ausgabe des *host*-Prozesses erscheinen in diesem Shell-Fenster. Wenn ein Shell-Fenster erzeugt wird, wird ein eigener Kommandointerpreter gestartet und mit diesem Shell-Fenster verbunden. Ist die Umgebungsvariable SHELL belegt, wird der angegebene Kommandointerpreter aufgerufen, anderenfalls */usr/bin/sh*. Um den Austausch von Nachrichten mit anderen Benutzern durch *write* zu ermöglichen, ruft *layers* das Kommando *relogin* auf, wenn das erste Shell-Fenster erzeugt wird. *relogin* definiert dieses Shell-Fenster als aktuelle Datensichtstation des Benutzers. Um ein anderes Shell-Fenster als aktuelle Datensichtstation zu definieren, rufen Sie *relogin* direkt auf. Bei Beendigung von *layers* wird wieder die ursprüngliche Datensichtstation als login-Terminal definiert.

Die Art und Weise, wie Shell-Fenster erzeugt, gelöscht, vergrößert, verkleinert oder sonst verändert werden, hängt ebenso von der verwendeten Datensichtstation ab wie die Beendigung einer *layers*-Sitzung.

Wenn ein Programmpaket für eine spezielle Datensichtstation verwendet werden soll, sollte die Umgebungsvariable DMD den Pfadnamen des Dateiverzeichnisses enthalten, in dem das Softwarepaket installiert wurde. Anderenfalls sollte die Variable DMD nicht belegt werden.

Der *xt*-Treiber unterstützt ein besonderes Verfahren zur Datenübertragung, den sogenannten *ENCODING MODE*. Mit diesem Verfahren ist der Einsatz von layers auch bei solchen Verbindungen möglich, die keine Übertragung von Steuerzeichen oder 8-bit-Zeichen zulassen. *ENCODING MODE*-Übertragung wird entweder durch eine Setup-Option an der Datensichtstation aktiviert oder durch Belegung der Umgebungsvariablen *DMDLOAD* mit dem Wert *hex* vor dem Aufruf von *layers*.

Um die korrekte Version von *layers* aufzurufen, muß */usr/bin* im Pfad vor allen anderen Dateiverzeichnissen erscheinen, die ein *layers*-Programm enthalten, etwa *$DMD/bin*. Falls eine bildschirmabhängige Version von *layers* vorhanden ist, könnte sonst diese statt der korrekten aufgerufen werden.

Fehlermeldungen und Informationen zur Fehlersuche werden von *layers* auf die Standard-Fehlerausgabe geschrieben. Deshalb sollte die Standard-Fehlerausgabe beim Aufruf mit den Option *-D*, *-d* und *-p* auf eine Datei umgelenkt werden. Wurde die Stan-

dard-Fehlerausgabe nicht umgelenkt und es tritt ein Fehler auf, so wird bei Beendigung von *layers* die letzte Fehlermeldung ausgegeben.

Die minimale Übertragungsgeschwindigkeit beim Einsatz von *layers* beträgt 1200 Baud.

```
layers [option] ...
```

option

-s
Nach dem Ende der *layers*-Sitzung wird auf der Standard-Fehlerausgabe eine Protokoll-Statistik ausgegeben. Während der Sitzung kann diese Statistik mit dem Kommando *xts* erzeugt werden.

-t
(t - trace) Die Ablauf-Verfolgung von Paketen des *xt*-Treibers wird eingeschaltet. Das Ergebnis dieser Ablauf-Verfolgung wird nach dem Ende der *layers*-Sitzung auf der Standard-Fehlerausgabe ausgegeben. Während der Sitzung kann es mit dem Kommando *xtt* erzeugt werden.

-D
(D - debugging) Auf der Standard-Fehlerausgabe werden Informationen zur Fehlersuche ausgegeben.

-m␣max-pkt
Setzt die maximale Größe für den Datenteil gewöhnlicher *xt*-Pakete, die vom Rechner an die Datensichtstation geschickt werden. Es sind Werte zwischen 32 und 252 erlaubt. Diese Option bewirkt außerdem, daß ein gewöhnliches und kein Netzwerk-Protokoll verwendet wird (siehe *xtproto*).

-d
Wenn über die Leitung ein Firmware-Programm zur Datensichtstation geladen wurde, wird auf der Standard-Fehlerausgabe die Größe des Text-, Daten- und BSS-Teils dieses Programms ausgegeben.

-p
Wenn über die Leitung ein Firmware-Programm zur Datensichtstation geladen wurde, wird auf der Standard-Fehlerausgabe eine Statistik und eine Ablaufverfolgung des Protokolls ausgegeben.

-h␣modul-liste
modul-liste ist eine Liste von STREAMS-Modulen, die für ein Shell-Fenster aktiviert werden. Die Module in der Liste werden durch Komma getrennt.

-f␣datei

datei enthält die Startkonfiguration für *layers*. Jede Zeile in der Datei beschreibt ein Shell-Fenster, das erzeugt werden soll. Die Zeilen haben das folgende Format:

x0 y0 x1 y1 kommandoliste

Für *x0, y0, x1, y1* sind sinnvolle Werte einzusetzen. *x0*, *y0* (linke obere Ecke) bzw. *x1*, *y1* (rechte untere Ecke) beschreiben die Größe und Lage des Shell-Fensters im Koordinatensystem der Datensichtstation. Wenn alle vier Koordinaten den Wert 0 haben, muß der Benutzer das Shell-Fenster interaktiv positionieren. *kommandoliste* ist eine Liste von einem oder mehreren Kommandos und muß angegeben werden. Die Kommandos werden in dem neuen Shell-Fenster ausgeführt, wobei mit dem Kommando *$SHELL -i -c "kommandoliste"* der Kommandointerpreter des Benutzers aufgerufen wird. Deshalb sollte das letzte Kommando einen Kommandointerpreter aufrufen, etwa */usr/bin/sh*. Wenn das letzte Kommando keinen Kommandointerpreter aufruft, ist das Shell-Fenster nach Beendigung des letzten Kommandos nutzlos. Reagiert die Datensichtstation nach dem Aufruf *layers -f datei* in einem oder mehreren Shell-Fenstern nicht auf Eingaben, wurde wahrscheinlich vom letzten Kommando in *kommandoliste* kein Kommandointerpreter aufgerufen.

programm

programm ist ein Firmware-Programm, das von *layers* zur Datensichtstation geladen wird, bevor Shell-Fenster erzeugt werden und die *kommandoliste* ausgeführt wird.

DATEIEN

/dev/xt??[0-7]
/usr/lib/layersys/lsys.8;7;3
$DMD/lib/layersys/lsys.8;?;?

UMGEBUNGSVARIABLEN

SHELL

enthält den aufgerufenen Kommandointerpreter.

DMD

enthält den Pfadnamen des Verzeichnisses, in dem spezielle Anwendungsprogramme für eine bestimmte Datensichtstation installiert sind.

DMDLOAD

wird auf den Wert *hex* gesetzt, wenn die die Datenübertragung im sogenannten *ENCODING MODE* erfolgen soll.

BEISPIELE

1. Ein typisches Startkommando lautet:

   ```
   $ layers -f startup
   ```

   Die Datei *startup* enthält dabei folgende Anweisungen:

   ```
   8 8 700 200 date ; pwd ; exec $SHELL
   8 300 780 850 exec $SHELL
   ```

2. Mit dem Kommando

   ```
   $ layers -h FILTER,LDTERM
   ```

   werden die STREAMS-Module FILTER und LDTERM für jedes Shell-Fenster aktiviert, das geöffnet wird.

SIEHE AUCH

ismpx, *jterm*, *jwin*, *relogin*, *sh*
jagent(), *layers()*, *libwindows()*, *write()*, *wtinit()*,
xt(), *xtproto()*, *xts()*, *xtt()* [5] bzw. [14]

line
Eine Zeile lesen

line liest eine Zeile von der Standard-Eingabe einschließlich Neue-Zeile-Zeichen und schreibt sie auf die Standard-Ausgabe.
Es wird immer mindestens ein Neue-Zeile-Zeichen ausgegeben. Nützliche Anwendung findet *line* in Shell-Prozeduren, wenn Eingaben des Benutzers interaktiv verarbeitet werden sollen.

line

ENDE-STATUS

0 Kommando wurde normal beendet.
1 Dateiende-Zeichen (EOF) auf der Eingabe.

BEISPIEL

Mit der Shell-Prozedur *kunden* soll überprüft werden, ob ein bestimmter Kunde in der Datei *liste* steht.

Inhalt von *kunden*:

```
echo "Bitte geben Sie den Namen des Kunden ein und schliessen Sie mit\
der Return-Taste ab!"
NAME=`line`
if grep $NAME liste
then echo "Der Kunde $NAME steht in der Datei liste."
else echo "Der Kunde $NAME steht nicht in der Datei liste."
fi
```

Nach dem Aufruf von *kunden* (Ausführrecht ist erforderlich) erscheint am Bildschirm die Aufforderung, den Namen einzugeben. Die Eingabe wird dann von *line* gelesen und der Variablen *NAME* als Wert zugewiesen. Die *if*-Abfrage überprüft den Ende-Status des Kommandos *grep*. Wenn dieser den Wert 0 hat, war das Kommando *grep* erfolgreich beim Durchsuchen der Datei *liste* nach dem Wert der Variablen *NAME* (*$NAME*). Dann wird ausgegeben: "Der Kunde $NAME steht in der Datei liste."
Wenn der Ende-Status einen Wert ungleich 0 hat, war *grep* nicht erfolgreich und es wird ausgegeben: "Der Kunde $NAME steht nicht in der Datei liste."

SIEHE AUCH

sh
read() [14]

listusers
Informationen über Benutzerkennungen ausgeben

listusers gibt Informationen über Benutzerkennungen aus.
Eine Benutzerkennung ist definiert als eine Kennung, deren Benutzernummer größer oder gleich 100 ist.

```
listusers[␣option]...
```

Keine Option angegeben
listusers gibt eine Liste aller Benutzerkennungen aus, sowie den entsprechenden Abrechnungswert für jede Benutzerkennung, die in */etc/passwd* eingetragen ist.

Die Ausgabe erfolgt sortiert, wobei das Sortierkriterium die Benutzerkennung ist.

option

-g␣gruppe
(g - group) Alle Benutzer der angegebenen Benutzergruppe *gruppe* werden aufgelistet. Mehrere Benutzergruppen können, durch Kommata getrennt, angegeben werden.

-l␣kennung
(l - login) Informationen über die in *kennung* angegebenen Benutzer werden ausgegeben. Mehrere Benutzerkennungen können, durch Kommata getrennt, angegeben werden.

Die Optionen *-g* und *-l* können kombiniert werden. Benutzerkennungen werden nur jeweils einmal aufgelistet, auch wenn sie zu mehr als einer der selektierten Gruppen gehören.

DATEIEN

/etc/passwd
Datei mit Benutzerkennungen

BEISPIELE

```
$ listusers -l wie,der,on
```

kann folgende Ausgabe produzieren:

```
wie         Arnulf Wiedmann
der         Dale Reed  Tel. 12345
on          Onyx Software
```

ln Verweis auf eine Datei eintragen (link)

ln erzeugt Verweise auf vorhandene Dateien. Sie können dann auf diese Dateien 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:

- einfacher Verweis (hard link):
  Wenn Sie einen einfachen Verweis auf eine Datei erzeugen, dann ist diese Datei als Eintrag in mehreren Dateiverzeichnissen vorhanden; physisch gibt es 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.
- symbolischer 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, sondern auf die Datei zu, zu der der Pfadname führt.
  Symbolische Verweise können Sie auf beliebige Dateien oder Dateiverzeichnisse über Dateisystem-Grenzen hinaus einrichten.

```
ln[␣option]␣datei␣verweis                                    Format 1
ln[␣option]␣datei␣...␣dateiverzeichnis                       Format 2
ln␣-s␣name␣verweis                                           Format 3
ln␣-s␣name␣...␣dateiverzeichnis                              Format 4
```

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.

option

-f

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.

-f nicht 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 Zugriffs-

rechte aus und fragt, ob es den Verweis erzeugen soll. Nur wenn Sie bejahen, erzeugt *ln* den Verweis, sonst nicht.
Ist die Standard-Eingabe keine Datensichtstation, dann unterbleibt die Frage (siehe *verweis* und *Beispiel 2*).

-n
Existiert bereits eine Datei mit dem Namen *verweis*, dann wird der Inhalt der Datei nicht überschrieben.
Die Option *-f* überschreibt 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.

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/name*:
ln trägt den einfachen Dateinamen *name* 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*).

ln in dieser Form legt keine Verweise über Dateisystemgrenzen hinweg an. Zum Beispiel ist folgender Aufruf nicht möglich:

```
ln /usr/gast/... /usr1/...
```

Verweise über Dateisystemgrenzen hinweg können Sie mit der Option *-s* einrichten (siehe *Format 3* und *Format 4*).

Format 2: Einfache Verweis(e) unter dem gleichen Namen in 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.

option

-f

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.

-f nicht 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. Nur wenn Sie bejahen, erzeugt *ln* den Verweis, sonst nicht.
Ist die Standard-Eingabe keine Datensichtstation, dann unterbleibt die Frage (siehe *verweis* und *Beispiel 2*).

-n

Existiert in *dateiverzeichnis* bereits eine Datei mit dem Namen *datei*, dann wird der Inhalt der Datei nicht überschrieben.
Die Option *-f* überschreibt die *-n* Option.

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, und muß nicht unbedingt existieren. *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/name*:
ln trägt den einfachen Dateinamen *name* in das Dateiverzeichnis *präfix* ein.

Existiert bereits eine Datei mit dem Namen *verweis*, erfolgt eine Fehlermeldung.

Format 4: Symbolische Verweis(e) unter dem gleichen 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.

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 einfachen Dateinamen oder einen absoluten oder relativen Pfadnamen angeben.

einfacher Dateiname:
ln -s trägt in *dateiverzeichnis* den symbolischen Verweis *name* 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 *name* hat, meldet *ln* einen Fehler:

```
ln: cannot create name ln: File exists
```

dateiverzeichnis
Name des Dateiverzeichnisses, in das die symbolischen Verweise eingetragen werden sollen. Das Dateiverzeichnis muß existieren.

ln legt Verweise über Dateisystemgrenzen hinweg an.

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 (und damit die gleichen Zugriffsrechte, der gleiche Eigentümer, die gleichen Daten usw.) gültig. Die Datei, auf die sich die Dateinamen beziehen, ist physisch 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!

FEHLERMELDUNGEN

```
ln: no permission for verweis
```

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
```

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> 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: different file system
```

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.

BEISPIELE

1. Der Benutzer Max besitzt die Dateien *bolte* und *laempel*; sie stehen im Dateiverzeichnis */usr1/max*. Nun soll auch Moritz mit diesen Dateien arbeiten. Damit Moritz weiterhin in seinem eigenen Dateiverzeichnis */usr1/moritz* arbeiten kann und dort nicht immer die langen Pfadnamen eingeben muß, legt Moritz im Dateiverzeichnis */usr1/moritz* Verweise auf diese Dateien an. Voraussetzung dafür ist, daß Moritz

   - für sein Dateiverzeichnis */usr1/moritz* Schreibrecht hat
   - für das Dateiverzeichnis */usr1/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 /usr1/max/bolte /usr1/max/laempel /usr1/moritz
```

   Dasselbe leisten die folgenden zwei *ln*-Aufrufe in Format 1:

```
$ ln /usr1/max/bolte /usr1/moritz/bolte
$ ln /usr1/max/laempel /usr1/moritz/laempel
```

   Wie die folgenden *ls*-Aufrufe zeigen, hat *ln* die Dateinamen *bolte* und *laempel* in das Dateiverzeichnis */usr1/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 /usr1/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 /usr1/max/bolte /usr1/moritz/bolte
16435 /usr1/max/bolte
16435 /usr1/moritz/bolte

$ ls -i /usr1/max/laempel /usr1/moritz/laempel
24766 /usr1/max/laempel
24766 /usr1/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 /usr1/max/bolte /usr1/moritz/spitz
$ ln /usr1/max/laempel /usr1/moritz/pfeife
```

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.

SIEHE AUCH

chmod, cp, ls, mv, rm
link(), readlink(), stat(), symlink() [14]

login
Sich neu am System anmelden

In der Korn-Shell ist *login* ein eingebautes Shell-Kommando, in der Bourne-Shell wird */bin/login* aufgerufen.

login kann sowohl vom Anwender als auch vom System aufgerufen werden:

- Wenn Sie sich neu am System anmelden, wird das Kommando *login* vom System aufgerufen.
- Wollen Sie *login* in der Bourne-Shell als Kommando selbst aufrufen, dann muß die aktuelle Shell damit überlagert werden. Sie müssen deshalb das Kommando mit *exec login* aufrufen. Die aktuelle Shell ist beendet, wenn *login* gestartet wird. Es entsteht kein neuer Prozeß; das erkennen Sie daran, daß sich die Prozeß-Nummer nicht ändert.

login führt folgendes aus:

- Es initialisiert die Standard-Shell-Variablen mit dem entsprechenden Standard-Wert (siehe *Arbeitsweise des Kommandos login*).
- Es überlagert sich mit dem Programm, das für diese Benutzerkennung in der Datei */etc/passwd* festgelegt ist.

Ist für die angegebene Benutzerkennung in der Datei */etc/passwd* als Startprogramm */usr/bin/sh* eingetragen, so überlagert sich *login* mit einer neuen Login-Shell. Diese Shell erledigt folgendes:

- Sie macht das HOME-Dateiverzeichnis zum aktuellen Dateiverzeichnis. Der Variablen HOME hat *login* den Namen des Login-Dateiverzeichnisses zugewiesen, das für die angegebene Benutzerkennung in */etc/passwd* eingetragen ist.
- Sie führt die Datei */etc/profile* aus.
- Sie führt die Datei *$HOME/.profile* aus, falls Sie diese Datei angelegt haben.

Deshalb gilt ab jetzt die Shell-Umgebung, die für diese Benutzerkennung definiert ist. Wenn Sie Ihre aktuelle Shell-Umgebung und Ihr aktuelles Dateiverzeichnis beibehalten, aber trotzdem Ihre Benutzerkennung wechseln wollen, verwenden Sie statt *login* das Kommando *su*.

Vor dem Aufruf beachten

Damit Sie sich neu am System anmelden können, muß die angegebene Benutzerkennung in der Datei */etc/passwd* eingetragen sein.
Wenn nicht, wenden Sie sich an den Systemverwalter.

```
login[␣-d␣gerät][␣benutzerkennung[␣umgebung . . ]]
```

Bestehen die Angaben für *login* ausschließlich aus Großbuchstaben, dann geht *login* davon aus, daß die angesprochene Datensichtstation nur Großbuchstaben verarbeitet.

gerät
Pfadname der Datensichtstation, an der Sie sich anmelden wollen. Mit dieser Angabe arbeitet *login* schneller, denn es muß nicht selbst die Datensichtstation feststellen.

gerät nicht angegeben:
login ruft *ttyname* auf um die Datensichtstation zu erfahren.

benutzerkennung
Benutzerkennung, unter der Sie sich neu am System anmelden wollen. Diese Benutzerkennung muß in der Datei */etc/passwd* eingetragen sein.

benutzerkennung nicht angegeben:
Das Kommando *login* fordert die Benutzerkennung in der nächsten Zeile an.

Ist in der Datei */etc/passwd* ein Kennwort für diese Benutzerkennung vereinbart, fordert *login* Sie anschließend in beiden Fällen auf, das Kennwort einzugeben.

Ihre Eingabe wird am Bildschirm nicht angezeigt und wird auch in keiner Datei mitprotokolliert.

umgebung
Wertzuweisung an eine Umgebungsvariable, die Sie entweder in der Form *yyy* oder *xxx=yyy* angeben. Die Wertzuweisung können Sie zur Ausführungszeit oder bei der Eingabe Ihrer Benutzerkennung eingeben. *xxx* ist der Name einer Umgebungsvariablen. *yyy* ist ein Wert, der einer Umgebungsvariablen zugewiesen werden soll. Mit dem Gegenschrägstrich entwerten Sie einzelne Zeichen innerhalb von *yyy*, wie z.B. Leer- oder Tabulatorzeichen.

xxx=yyy
Die Umgebungsvariable *xxx* erhält den Wert *yyy*. Ältere Werte von bereits gesetzten Umgebungsvariablen, werden überschrieben, mit zwei Ausnahmen: PATH und SHELL können auf diese Weise nicht verändert werden.

yyy
Die Umgebungsvariable *Lnummer* erhält den Wert *yyy*. *nummer* beginnt bei 0 und wird weitergezählt, solange noch Werte *yyy* vorhanden sind.

Arbeitsweise des Kommandos login

Das Kommando *login* vergleicht die angegebene Benutzerkennung und das angegebene Kennwort mit dem entsprechenden Eintrag in der Datei */etc/passwd*. Wenn alle Angaben korrekt sind, werden Sie unter dieser Benutzerkennung am System angemeldet.

Sind die Angaben nicht korrekt, wird nach fünf mißlungenen Login-Versuchen die Leitung zur Datensichtstation unterbrochen. Ebenso wird verfahren, wenn Sie nicht innerhalb einer bestimmten Zeit unvollständige Eingaben fortsetzen.

Beim Anmelden am System erledigt *login* der Reihe nach die folgenden Aufgaben:

- *login* sucht zur angegebenen Benutzerkennung die Benutzernummer (UID) und die Gruppennummer (GID) in der Datei */etc/passwd*.
  Das Betriebssystem verwaltet die Benutzerkennungen intern nur unter ihrer Benutzer- und Gruppennummer.
- *login* trägt die Benutzerkennung in die Datei */etc/utmp* ein. Diese Datei enthält die Login-Kennungen aller Benutzer, die aktuell am System angemeldet sind.
  Auf diese Datei greift das Kommando *who* zu.
- *login* trägt die Benutzerkennung in die Datei */var/adm/wtmp* ein. Wenn diese Datei existiert, werden dort alle An- und Abmeldungen fortlaufend protokolliert.
- *login* gibt aus, wann Sie sich unter dieser Benutzerkennung das letzte Mal angemeldet haben. Diese Information enthält die Datei */var/adm/lastlog*.
- */etc/motd* wird ausgegeben.
- *login* versorgt die folgenden Umgebungsvariablen mit Standard-Werten:

Variable	Zuweisung:
HOME	Name des Login-Dateiverzeichnisses (aus */etc/passwd*)
LOGNAME	Login-Benutzerkennung (aus */etc/passwd*)
MAIL	/var/mail/b*enutzerkennung*
SHELL	/usr/bin/sh (aus */etc/passwd*)
TZ	MET-1
PATH	/usr/bin

Weitere Informationen zu diesen Umgebungsvariablen finden Sie in der Beschreibung des Kommandos *sh*. Hier erfahren Sie auch, wie Sie diese Standard-Werte ändern und zusätzliche Umgebungsvariablen definieren können.

- *login* macht das Login-Dateiverzeichnis, das in der Datei */etc/passwd* für diese Benutzerkennung eingetragen ist, zum aktuellen Dateiverzeichnis.
- *login* startet das Programm, das in der Datei */etc/passwd* für diese Benutzerkennung eingetragen ist. Normalerweise wird */usr/bin/sh* als Login-Shell gestartet (siehe *sh*). Diese Shell überlagert *login.*

  Enthält */etc/passwd* den Eintrag *, dann wird das angegebene Login-Dateiverzeichnis zum Root-Verzeichnis. D.h. bei Dateinamen, die mit Schrägstrich beginnen, wird von diesem Dateiverzeichnis aus gesucht. Weiter wird *login* auf der neuen Root-Struktur erneut ausgeführt.

Die Arbeit an der Datensichtstation beginnen

Nach dem Einschalten der Datensichtstation erscheint die Schreibmarke am Bildschirm. Wenn Sie die Taste END drücken, gibt der Systemprozeß *getty* den Begrüßungsbildschirm aus und ruft das Kommando *login* ohne Angabe einer Benutzerkennung auf. Anschließend können Sie Ihre Benutzerkennung und gegebenenfalls das dazugehörige Kennwort so eingeben, als hätten Sie *login* selbst aufgerufen.
Das gleiche gilt, wenn Sie Ihre letzte Sitzung mit der Taste END beendet haben.

Bevor Sie also Ihre Arbeit an der Datensichtstation beginnen können, müssen folgende Systemprozesse tätig sein:

- */etc/init* startet für alle angeschlossenen Datensichtstationen den System-Prozeß *getty*. Für jede Datensichtstation wird also ein neuer Prozeß erzeugt.
- */etc/getty* gibt den Begrüßungsbildschirm aus und überlagert sich mit *login.*
- *login* wartet auf die Eingabe einer Benutzerkennung.

Wenn Sie sich erfolgreich am System angemeldet haben, überlagert sich *login* mit dem in der Datei */etc/passwd* eingetragenen Startprogramm. Das ist in der Regel die Login-Shell */bin/sh*. Wenn Sie diese Login-Shell mit END beenden, beenden Sie den aktuellen Prozeß, d.h. */etc/init* startet erneut den Systemprozeß */etc/getty*, und dieser überlagert sich wieder mit *login.*

FEHLERMELDUNGEN

`Login incorrect`

Diese Fehlermeldung kann folgende Ursachen haben:

- Sie haben sich bei der Eingabe der Benutzerkennung oder des Kennwortes vertippt.
  Das Kommando *login* fordert erneut Ihre Angaben an.
- Diese Benutzerkennung ist noch nicht eingerichtet.
  Wenden Sie sich an den Systemverwalter.
- Das Kennwort hat sich geändert.
  Wenden Sie sich an den Systemverwalter.

`No shell`

Der Eintrag für diese Benutzerkennung in der Datei */etc/passwd* ist fehlerhaft oder Sie haben unter dieser Benutzerkennung kein Ausführrecht für die entsprechende Shell. Wenden Sie sich an den Systemverwalter.

`unable to change directory to ""`

Das Login-Dateiverzeichnis für diese Benutzerkennung ist nicht eingerichtet.
Wenden Sie sich an den Systemverwalter.

`Cannot open password file`

Die Datei */etc/passwd* ist nicht vorhanden oder kann von *login* nicht gelesen werden.
Wenden Sie sich an den Systemverwalter.

`No utmp entry. You must exec "login" from the lowest level "sh".`

Eintrag in der Datei */var/adm/utmp* fehlt oder ist aus Platzmangel nicht möglich. Der Systemverwalter kann sich auch dann anmelden, wenn der Eintrag nicht vorhanden ist. *login* kann nur von der Login-Shell oder mit *exec login* aufgerufen werden.

DATEIEN

/etc/motd
enthält die Login-Nachricht für alle Benutzer.

/etc/passwd
enthält alle eingerichteten Benutzerkennungen.

/etc/profile
wird von jeder Login-Shell ausgeführt. Diese Datei erstellt der Systemverwalter.

/etc/shadow
enthält verschlüsselte Paßwörter

/var/adm/utmp
enthält die Login-Kennungen aller Benutzer, die aktuell am System angemeldet sind.

/var/adm/loginlog
enthält Informationen über mißlungene Login-Versuche

/var/adm/lastlog
protokolliert für jede Benutzerkennung den Zeitpunkt der letzten Anmeldung am System.

/var/adm/wtmp
Wenn diese Datei existiert, werden dort alle An- und Abmeldungen fortlaufend protokolliert.

/var/mail/benutzerkennung
enthält Post für diese Benutzerkennung.

$HOME/.profile
Shell-Prozedur, die jeder Benutzer in seinem HOME-Dateiverzeichnis einrichten kann. Sie wird von der Login-Shell nach */etc/profile* ausgeführt.

BEISPIEL

Die Benutzerin *rosa* möchte sich unter der Benutzerkennung *harald* neu am System anmelden. In der Datei */usr1/harald/.profile* ist das Bereitzeichen der Shell umdefiniert: PS1=$USER:

```
$ pwd
/usr/rosa/prog
$ id
uid=104(rosa) gid=12(gruppe12)
$ exec login harald    [in der Bourne-Shell]
password:              [blinde Eingabe des Kennworts für harald]
                       [login informiert über den Zeitpunkt der letzten
                        Anmeldung, gibt die Login-Nachricht aus und informiert,
                        falls Post für harald eingetroffen ist]
harald: pwd
/usr1/harald
```

Mit *su* bleibt beim Wechsel der Benutzerkennung die aktuelle Arbeitsumgebung erhalten:

```
$ pwd
/usr/rosa/prog
$ id
uid=104(rosa) gid=12(gruppe12)
$ su harald
password:          [blinde Eingabe des Kennworts für harald]
$ pwd
/usr/rosa/prog
$ echo $USER
rosa
$ id
uid=100(harald) gid=10(other)
```

SIEHE AUCH

exec, *mail*, *newgrp*, *sh*, *su*
loginlog [5]
passwd(), *profile()*, *environ()* [14]

logname
Login-Kennung abfragen (login name)

logname schreibt Ihre Login-Benutzerkennung auf die Standard-Ausgabe.

```
logname
```

Wechselwirkung mit su

Wenn Sie sich unter der Benutzerkennung *name* am System angemeldet haben, dann mit dem Kommando *su* die Benutzerkennung wechseln und in der neuen Benutzerkennung *logname* eingeben, dann gibt *logname*, ebenso wie *who am i*, die Login-Kennung *name* aus.
Das Kommando *id* hingegen gibt Informationen über die aktuelle Benutzerkennung aus.

UMGEBUNGSVARIABLEN

LOGNAME
Login-Kennung des Benutzers

SIEHE AUCH

env, *id*, *login*, *su*, *who*
cuserid() [14]
environ [5]

lp Dateien ausdrucken (line printer)

Unter SINIX 5.4 können Sie alternativ zwei Spool-Systeme einsetzen:

- den SINIX-Spooler und
- den Original-AT&T-Spooler.

Das Kommando *lp* können Sie für beide Spooler benutzen. Bestimmte Optionen werden vom SINIX-Spooler jedoch nicht unterstützt.

lp steuert Druckaufträge für Dateien. Mit dem Kommando *lpr* können Sie ebenfalls Dateien ausdrucken lassen.

Mit *lp* können Sie:

- Dateien am Drucker ausdrucken lassen (Druckauftrag) (Format 1)
- die Druckergruppe auswählen, auf der ein Druckauftrag ausgeführt wird (Format 1)
- eine Kopfseite ausgeben lassen (Format 1)
- Druckeroptionen für Druckaufträge ändern (Format 2)

```
lp[␣option]...[␣datei]...                                    Format 1
lp␣-i[␣auftragsnr]...[␣option]...                            Format 2
```

Format 1: Dateien am Drucker ausgeben lassen

lp[␣option]...[␣datei]...

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 Druckauftag mit *cancel* zu löschen oder abzubrechen oder um mit *lpstat* Informationen über den Druckauftrag ausgeben zu lassen.

Format 2: Optionen für Druckaufträge verändern

lp␣-i[␣auftragsnr]...[␣option]...

lp ersetzt die bisherigen Optionen der durch *auftragsnr* bestimmten Druckaufträge durch *option*. Wurde der Druckauftrag bereits ausgeführt, nimmt *lp* keine Änderung vor. Läuft er gerade, unterbricht *lp* den Druck und startet ihn neu (wenn Sie nicht die Option *-P* gesetzt haben).

Keine Option angegeben
Die angegebenen Dateien werden auf einem freien Drucker der Standard-Druckergruppe (standardmäßig ALLE) ausgedruckt.

Format 2 wird vom SINIX-Spooler nicht unterstützt.

option

Optionen, die für den Ausdruck einer Datei wirksam sein sollen, müssen beim *lp*-Aufruf vor dieser Datei angegeben werden. Ansonsten ist die Reihenfolge der Optionen beliebig.

-c
(c - copy) Die angegebenen Dateien werden beim Aufruf von *lp* kopiert, und diese Kopien werden ausgedruckt.

-c nicht angegeben:
Die auszudruckenden Dateien werden nicht kopiert. Alle Veränderungen der Dateien, die Sie zwischen dem Aufruf von *lp* und dem tatsächlichen Ausdruck der Dateien vornehmen, spiegeln sich im Ausdruck wider. Wenn Sie die Option *-c* nicht angegeben haben, müssen Sie darauf achten, keine Dateien zu löschen, für die ein Druckauftrag mit *lp* gestellt wurde, die aber noch nicht ausgedruckt worden sind.

-d␣d_gruppe
-d␣drucker
Der Druckauftrag wird auf der Druckergruppe *d_gruppe* bzw. dem Drucker *drucker* ausgeführt. Wenn zu einer Druckergruppe mehrere Drucker gehören, wird der Auftrag auf dem ersten freien Drucker dieser Druckergruppe ausgeführt.

Schreiben Sie *any* für *drucker*, gibt *lp* auf irgendeinem Drucker aus, der den Auftrag übernehmen kann.

-d␣d_gruppe und *-d␣drucker* nicht angegeben:
Wenn die Umgebungsvariable LPDEST gesetzt ist, wird deren Wert eingesetzt.

-f␣formname[**␣-d␣any**]
Diese Option gilt nicht für den SINIX-Spooler.
lp gibt den Druckauftrag auf das Formular *formname* aus. Die Druckerverwaltung stellt sicher, daß das Formular in den Drucker eingelegt wurde. Wird *formname* einem Drucker oder einer Druckergruppe zugeordnet, der oder die das Formular nicht unterstützt, wird der Auftrag zurückgewiesen. Das gleiche geschieht, wenn *formname* für das System nicht definiert oder der Benutzer nicht dazu berechtigt ist, das Formular zu verwenden. Wenn Sie *-d␣any* angegeben haben, wird der Auftrag auf irgendeinem Drucker ausgegeben, in den ein entsprechendes Formular eingelegt wurde und der auch die übrigen Erfordernisse des Druckauftrages erfüllt.

-H␣spezialbehandlung
Diese Option gilt nicht für den SINIX-Spooler.
lp gibt den Druckauftrag entsprechend *spezialbehandlung* aus. Sie können für *spezialbehandlung* folgende Werte angeben:

hold
lp gibt den Druckauftrag nicht aus, bis Sie es explizit angeben. Lief der Druckauftrag bereits, wird er unterbrochen. Andere Druckaufträge werden solange vorgezogen, bis der angehaltene Auftrag wiederaufgenommen wird.

resume
lp nimmt den angehaltenen Druckauftrag wieder auf. Lief der Druckauftrag zum Zeitpunkt des Anhaltens, wird er als nächstes ausgeführt, wenn er nicht nachträglich von einem mit *immediate* versehenen Auftrag verdrängt wird.

immediate (nur für Druckverwalter)
lp gibt den angegebenen Druckauftrag als nächstes aus. Sind mehrere Aufträge mit *immediate* versehen, werden sie in umgekehrter Reihenfolge ausgeführt. Wird ein Druckauftrag gerade auf dem gewünschten Drucker ausgeführt, müssen Sie ihn anhalten, damit ihr *immediate*-Auftrag laufen kann.

-m
(m - message) Sie erhalten über das Kommando *mail* eine Nachricht, nachdem der Druckauftrag ausgeführt wurde.

-m nicht angegeben:
Sie erhalten keine Nachricht.

-n␣anzahl
Für *anzahl* geben Sie eine ganze Zahl an, die festlegt, wie oft die Dateien ausgedruckt werden. Die kleinste wirksame Angabe für *anzahl* ist 1.

-n␣anzahl nicht angegeben:
Für *anzahl* wird 1 angenommen.

-o␣druckeroption
Diese Option gilt nicht für den SINIX-Spooler.
lp gibt den Druckauftrag entsprechend der druckerabhängigen Option aus. Wollen Sie mehrere Druckeroptionen setzen, dann können Sie die Option *-o* entsprechend oft angeben oder alle Druckeroptionen zusammenfassen und zwischen Anführungstriche stellen. Z.B. schreiben Sie statt
-o droption1 -o droption2 -o droption3 zusammengefaßt
-o "droption1 droption2 droption3".
Sie können für *druckeroption* standardmäßig folgende Werte angeben:

nobanner
lp druckt keine einleitende Seite vor dem Druckauftrag. Diese Druckeroption kann vom Verwalter unterbunden werden.

nofilebreak
lp macht keinen Seitenvorschub zwischen den Dateien, die ausgedruckt werden.

length=maßzahl
lp gibt den Druckauftrag mit Seitenlänge *maßzahl* aus. Sie dürfen diese Option nicht zusammen mit Option *-f* verwenden.

maßzahl
Dezimalzahl oder Dezimalzahl mit Appendix *i* oder Dezimalzahl mit Appendix *c*. Dabei steht *i* für Inches und *c* für Zentimeter. Z.B. bedeutet *length=66*: Seitenlänge 66 Zeilen,
length=11i: Seitenlänge 11 Inches und
length=27.94c: Seitenlänge 27.94 cm.

width=maßzahl
lp gibt den Druckauftrag mit Seitenbreite *maßzahl* aus. Sie dürfen diese Option nicht zusammen mit Option *-f* verwenden.

maßzahl
Dezimalzahl oder Dezimalzahl mit Appendix *i* oder Dezimalzahl mit Appendix *c*. Wie bei *length* steht *i* für Inches und *c* für Zentimeter.

lpi=zahl
lp gibt den Druckauftrag mit Zeilenabstand (Zeilen pro Inch) *zahl* aus. Sie dürfen diese Option nicht zusammen mit Option *-f* verwenden.

cpi=zahl
lp gibt den Druckauftrag mit Zeichenbreite (Zeichen pro Inch) *zahl* aus. Sie dürfen diese Option nicht zusammen mit Option *-f* verwenden. Hat *zahl* den Wert *pica*, dann druckt *lp* 10 Zeichen pro Inch aus. Hat *zahl* den Wert *elite*, dann druckt *lp* 12 Zeichen pro Inch aus. Hat *zahl* den Wert *compressed*, dann druckt *lp* soviele Zeichen pro Inch aus, wie der Drucker maximal kann.

stty=sttyoptionsliste
sttyoptionsliste ist eine Liste von Optionen, die *lp* an das Kommando *stty* übergibt. Die Liste muß in Hochkomma eingeschlossen werden, wenn sie Leerzeichen enthält.

-P␣seitenliste
Diese Option gilt nicht für den SINIX-Spooler.
lp druckt die in *seitenliste* angegebenen Seiten. Damit *lp* den Druckauftrag ausführen kann, muß ein Filter vorhanden sein, der die Seiteneinteilung vornimmt. Ist das nicht der Fall, denn führt *lp* den Druckauftrag nicht aus.
seitenliste kann aus einzelnen Seitenzahlen, Bereichsangaben oder beidem gemischt bestehen. *lp* druckt die Seiten in aufsteigender Reihenfolge aus.

-q␣priorität
Diese Option gilt nicht für den SINIX-Spooler.
Der Druckauftrag erhält die Prioritätsstufe *priorität*. *priorität* kann alle Werte zwischen 0, der niedrigsten Prioritätsstufe, und 39, der höchsten, annehmen. Benutzern, die keine Systemverwalterrechte haben, können vom Systemverwalter individuelle Prioritätsobergrenzen zugeordnet werden.

-q␣priorität nicht angegeben:
Die Prioritätstufe erhält einen Standardwert, der vom Systemverwalter festgelegt wurde.

-s
(s - silent) Meldungen des Kommandos *lp* werden unterdrückt.

-S␣zeichensatz[␣-d␣any]
Diese Option gilt nicht für den SINIX-Spooler.
lp bearbeitet den Druckauftrag mit dem angegebenen Zeichensatz. Soll auf ein Formular gedruckt werden, das einen anderen Zeichensatz erfordert, dann verweigert *lp* den Druckauftrag. Ist *zeichensatz* nicht in der Terminfo-Datenbank des Druckers definiert oder nicht ein Aliasname für einen solchen Zeichensatz, dann führt *lp* den Druckauftrag nicht aus.
Haben Sie zusätzlich *-d␣any* angegeben, dann führt *lp* den Druckauftrag auf einem Drucker aus, der alle angebenen Optionen verarbeiten kann.

-S␣zeichensatz nicht angegeben:
Der Standardzeichensatz des Druckers wird verwendet.

-S␣typenrad[␣-d␣any]
Diese Option gilt nicht für den SINIX-Spooler.
lp bearbeitet den Druckauftrag mit dem angegebenen Typenrad. Soll auf ein Formular gedruckt werden, das ein anderes Typenrad erfordert, dann verweigert *lp* den Druckauftrag. Ist *typenrad* nicht in der Liste der Typenräder, die der Verwalter für den zum Druckauftrag gehörenden Drucker erstellt hat, dann führt *lp* den Druckauftrag nur aus, wenn das Typenrad bereits am Drucker montiert ist.
Haben Sie zusätzlich *-d␣any* angegeben, dann führt *lp* den Druckauftrag auf einem Drucker aus, der alle angebenen Optionen verarbeiten kann.

-S␣typenrad nicht angegeben:
Das am Drucker montierte Typenrad wird verwendet.

-t␣titel
(t - title) Der Ausdruck beginnt mit einer zusätzlichen Kopfseite, auf der *titel* ausgedruckt wird.

-T␣inhaltstyp[␣-r]
Diese Option gilt nicht für den SINIX-Spooler.
lp gibt den Druckauftrag auf einem Drucker aus, der den angegebenen *inhaltstyp* unterstützt. Gibt es keinen solchen Drucker, dann wird der Inhalt den vorhandenen Druckern entsprechend durch einen Filter konvertiert. Ist das nicht möglich, dann wird der Druckauftrag nicht ausgeführt. Haben Sie zusätzlich *-r* angegeben, dann wird bei nicht unterstütztem Inhaltstyp auf keinen Fall konvertiert und der Druckauftrag wird nicht ausgeführt.

-w
Diese Option gilt nicht für den SINIX-Spooler.
(w - write) Sie erhalten auf der Datensichtstation eine Nachricht, nachdem der Druckauftrag ausgeführt wurde. Sind Sie nicht am System angemeldet, dann erhalten Sie eine Nachricht, die Sie über das Kommando *mail* erfahren können.

-w nicht angegeben:
Sie erhalten keine Nachricht.

datei
Name der auszudruckenden Datei. Sie können mehrere Dateien angeben. Wenn Sie keine Datei angeben, liest *lp* von der Standard-Eingabe. Wenn Sie mehrere Dateien angeben, werden diese in der beim Aufruf angegebenen Reihenfolge ausgedruckt. Alle Optionen, die für den Ausdruck einer Datei gelten sollen, müssen vor dieser Datei angegeben werden.

Für jede auszudruckende Datei gibt *lp* eine Meldung der folgenden Form aus:

```
request id is d_gruppe-id
```

Dabei ist *d_gruppe* der Name der Druckergruppe, auf der der Auftrag ausgeführt wird, und *id* eine ganze Zahl.

Wenn *lp* von der Standard-Eingabe liest, heißt die Meldung

```
request id is d_gruppe-id (standard input)
```

datei nicht angegeben:
lp liest von der Standard-Eingabe.

UMGEBUNGSVARIABLE

LPDEST
Name einer Druckergruppe. Wenn *-d␣d_gruppe* oder *-d␣drucker* nicht angegeben wird, wird *-d$LPDEST* angenommen.

BEISPIELE

1. Es sollen drei Dateien, *t1*, *t2* und *t3* ausgedruckt werden. Nach dem Ausdruck soll der Auftraggeber mit *mail* benachrichtigt werden. Von den Dateien *t2* und *t3* sollen jeweils drei Kopien ausgedruckt werden:

```
$ lp -m t1 -n3 t2 t3
request id is ALLE-1
request id is ALLE-2
request id is ALLE-3
```

   Für jede auszudruckende Datei gibt *lp* eine Meldung aus. Alle Druckaufträge werden auf der Druckergruppe ALLE ausgeführt, die Auftragsnummern sind 1, 2 bzw. 3.

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 -dG002 t2
request id is G001-1
request id is G002-2
```

   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 -dG002 t2
```

SIEHE AUCH

cancel, *enable*, *lpr*, *lpstat*, *mail*
accept, *lpadmin*, *lpfilter*, *lpforms*, *lpsched*, *lpsystem*, *lpusers* [5]
terminfo [5] bzw. [14]

SINIX SPOOL [15]

lpr
Dateien ausdrucken und Druckaufträge steuern

lpr steuert Druckaufträge für Dateien. Mit *lp* können Sie ebenfalls Dateien ausdrucken lassen.

Mit *lpr* können Sie:

- Dateien am Drucker ausdrucken lassen (Druckauftrag)
- das Papierformat einstellen (Zeilen pro Seite, Zeichen pro Zeile)
- ein bestimmtes Formular auswählen
- die Ausgabe auf einen bestimmten Drucker lenken, wenn mehrere angeschlossen sind
- eine Kopfseite ausgeben, eine Endseite ausgeben
- Druckaufträge löschen und modifizieren
- die Priorität eigener Druckaufträge festlegen
- den Zustand der Druckaufträge und die Betriebsbereitschaft der Drucker abfragen
- Druckergruppen-Namen abfragen
- Druckerverwalter abfragen
- abfragen, welche Formulare unterstützt werden

Als Systemverwalter oder Druckerverwalter können Sie weitere Funktionen des Kommandos *lpr* nutzen. Ein Druckerverwalter muß vom Systemverwalter in der Datei *.../CONFIG* definiert sein (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*). Weitere Informationen zur Druckerverwaltung finden Sie in *SINIX SPOOL* [15].

Als *Druckerverwalter* können Sie zusätzlich:

- Drucker sperren und wieder freigeben, mit oder ohne Abbruch laufender Druckaufträge
- für einen Drucker einen Schwellenwert (Minimalpriorität für die Ausgabe) festlegen
- ein bestimmtes Formular für einen Drucker einstellen
- einen Probedruck veranlassen

Als *Systemverwalter* können Sie darüberhinaus noch:

- den Druckerbetrieb abschalten
- die Druckerkonfiguration ändern
- Treiberprogramme (Backends) laden und beenden

Abhängig davon, welche Optionen Sie beim Kommando *lpr* angeben, führt *lpr* die gewünschte Aktion entweder sofort aus oder delegiert sie an das Steuerprogramm */etc/daemon*. In diesem Fall beendet sich das Kommando *lpr* vor der endgültigen Ausführung der Aktion. Die Beendigung des Kommandos *lpr* bedeutet also nicht zwingend, daß auch die Aktion bereits ausgeführt ist.

Es kann sein, daß an Ihr System nur ein Drucker angeschlossen ist. Funktionen oder Angaben, die sich auf mehrere Drucker beziehen, sind dann bedeutungslos.

```
lpr[␣option]...[␣datei]...[␣option]...[␣datei]...
```

option

Optionen, die für den Ausdruck einer Datei gelten sollen, müssen vor dem Namen dieser Datei angegeben werden. Auch Optionen, die für einen durch *-id=n* identifizierten Druckauftrag wirksam sein sollen, müssen vor der Option *-id* angegeben werden.

Optionen für die Auftragsverwaltung	Kurzbeschreibung
-ap=n	Priorität eigener Druckaufträge ändern
-ca	Druckauftrag löschen
+co	Dateien vor Abdruck kopieren
-co	Dateien vor Abdruck nicht kopieren
-cp	Dateien vor Abdruck kopieren
+del	Dateien nach Abdruck löschen
-del	Dateien nach Abdruck nicht löschen
-dru=dg	Auftrag für Druckergruppe *dg*
-form=fm	Formular auswählen
-id=n	Identifikation eines Druckauftrags über die Auftragsnummer
-lang=sprache	Sprache, in der der Benutzer informiert wird
-mp␣auftrag**␣...**	Ändern der für einen Druckauftrag gültigen Optionen
-mp␣-id=*nn***␣...**	Ändern der für einen Druckauftrag gültigen Optionen
+msg	Nachricht nach Ende des Druckauftrags
-msg	Keine Nachricht nach Ende des Druckauftrags
-nc=n	Druckauftrag *n*-mal wiederholen
-no	Nachricht nach Ende des Druckauftrags
-pr=n	Ändern der Priorität eines Druckauftrags
-q[=fm**]**	Zustand der Drucker und Druckaufträge ausgeben
-qadmin	Druckerverwalter ausgeben
-qdru	Druckergruppen ausgeben
-qform	Formulare ausgeben
-rl=Dnnn	Drucker im Blockierbetrieb freigeben
-rm	Dateien nach Abdruck löschen
-tl=titel	Titel für einen Druckauftrag angeben
-to=benutzerkennung	Druckauftrag für *benutzerkennung*
+v	Quittung nach Aufruf von lpr
-v	Keine Quittung nach Aufruf von lpr
-ws=dg	Auftrag für Druckergruppe *dg*

Optionen für die Backends	Kurzbeschreibung
-ab=n	Druckauftrag ab Seite *n* drucken
-band=n	Typenband *n* für Drucker 9047 auswählen
-bis=n	Druckauftrag bis Seite *n* drucken
-bits=n	Anzahl der Datenbits einstellen
+cat	Datei ohne Code-Umwandlung an den Drucker übergeben
-cat	Datei mit Code-Umwandlung an den Drucker übergeben
-ds=n	doppelseitigen Druckmodus auswählen
-dt	Deutschen Zeichensatz auswählen, laden
-filter=n	Filterprogramm auswählen
-font=n	Zeichensatz auswählen
-form=fm	Formular auswählen
+hd	Kopfseite drucken
-hd	Keine Kopfseite drucken
-hdgrp	Kopfseite bei Gruppenwechsel drucken
-hop=n	Papierzuführung wählen
-int	ASCII-Zeichensatz auswählen, laden
-mar=n	Linken Druckrand einstellen
-nk=n	Jede Seite des Druckauftrags *n*-mal drucken
-pb=n	Zeilenbreite einstellen
-pb1	Zeichenbreite 10 Zeichen pro Zoll
-pb2	Zeichenbreite 12 Zeichen pro Zoll
-pb3	Kleinste Zeichenbreite des Druckers
-pl=n	Seitenlänge einstellen
-port=n	Einstellen der TCP-Port-Nummer eine TACLAN-Druckers
+ps	Proportionalschrift einschalten
-ps	Proportionalschrift ausschalten
+tab	Hardwaretabulatur einschalten
-tab	Hardwaretabulatur ausschalten
-top=n	Kopfrand einstellen
+trl	Endeseite ausdrucken
-trl	Keine Endeseite ausdrucken
-trlgrp	Endeseite bei Gruppenwechsel drucken
+vp	Eingabedatei durch virtuellen Drucker übersetzen
-vp	Eingabedatei nicht durch virtuellen Drucker übersetzen
-za=n	Zeilenabstand einstellen
-zb=n	Zeichenbreite auswählen
-zs=<zeichensatz>	Zeichensatz auswählen

Optionen für die Konfiguration der Backends	Kurzbeschreibung
-addr=Druckername	Internet-Adresse eines TACLAN-Druckers einstellen
-bits=n	Anzahl der Datenbits einstellen
-esca	Drucker 9025 erwartet \ statt ESC (0x1b)
-escb	Drucker 9025 erwartet ^**x41** statt ESC (0x1b)
+lkmod	Blockierbetrieb einstellen
-parity=pp	Art des Paritätsbits einstellen
-perm	keine permanente TACLAN-Verbindung
+perm	permanente TACLAN-Verbindung
-port=n	Einstellen der TCP-Port-Nummer eine TACLAN-Druckers
-polltmo=t	Einstellen der Periode zwischen zwei Statusabfragen (S)
-port=n	Einstellen der TCP-Port-Nummer eines TACLAN-Druckers (S)
-speed=br	Übertragungsgeschindigkeit einstellen
-stopb=bits	Einstellen der Stoppbits (S)
-waittmo=t	Einstellen der Wartezeit bis zum Abbruch eines Druckauftrags (S)

Optionen für Systemverwalter (S) und Druckerverwalter (D)	Kurzbeschreibung
-dd=Dnnn	Drucker sperren (S)(D)
-dg	Druckerbetrieb abschalten (S)
-dk=Dnnn	Drucker sperren (S)(D)
-du=Dnnn	Drucker freigeben (S)(D)
-ex=Dnnn	Drucker aus der Druckerverwaltung herausnehmen (S)
-forminit=Dnnn	Formular einstellen (S) (D)
-ld=Dnnn	Drucker wieder in die Druckerverwaltung aufnehmen (S)
-port=n	Einstellen der TCP-Port-Nummer eines TACLAN-Druckers (S)
-rr	Aktuelle Spool-Konfigurationsdatei erzeugen (S)
-speed=br	Übertragungsgeschindigkeit einstellen (S)
-su=benutzername	eigenen Benutzernamen zeitweilig ersetzen (S)
-stopb=bits	Einstellen der Stoppbits (S)
-tst=Dnnn	Probedruck (S)(D)
-vex=Dnnn	Drucker aus der Druckerverwaltung herausnehmen (S)
-vld=Dnnn	Drucker wieder in die Druckerverwaltung aufnehmen (S)

Optionen zur Auftragsverwaltung

-ap=n
Diese Option setzt die Priorität bereits gegebener eigener Druckaufträge auf *n*.

Als Systemverwalter können Sie auch die Priorität fremder Druckaufträge ändern. Dazu müssen Sie vor dieser Option die Option *-su=benutzerkennung* angeben (siehe *Optionen für den Druckerverwalter bzw. den Systemverwalter*).

Die Option *-ap=n* ist eine Kurzform für *-pr=n -mp auftrag...*, siehe Optionen *-pr* und *-mp*.

-ca
(ca - cancel) Die eigenen Druckaufträge für die genannten Dateien werden gelöscht bzw. abgebrochen, falls sie bereits laufen.

Der Systemverwalter kann auch fremde Druckaufträge angeben, dazu muß er allerdings vorher die Option *-su=benutzerkennung* setzen (siehe *Optionen für den Drucker- bzw. Systemverwalter*).

Gibt es mehrere Druckaufträge mit gleichem Namen, wird nur der nächstanliegende Auftrag gelöscht.
Namen von Druckaufträgen können Sie teilqualifiziert angeben (z.B. durch * oder ?). Allerdings müssen Sie die Shell-Sonderzeichen entwerten, damit sie nicht von der Shell interpretiert werden.

Druckaufträge können auch mit dem Kommando *cancel* gelöscht oder abgebrochen werden.

+co
(co - copy) *lpr* kopiert die beim Aufruf angegebenen Dateien in das Dateiverzeichnis *.../sp*. (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*) Sie können dadurch sofort mit den Dateien weiterarbeiten, ohne auf die Beendigung des Druckauftrags zu warten.

-co
(co - copy) Setzt die Optionen *+co* bzw. *-cp* zurück.

-cp
(cp - copy) wie die Option *+co* (siehe oben).

+del
(del - delete) Die angegebenen Dateien werden nach erfolgreichem Ausdruck gelöscht. Der Benutzer muß Schreibrecht für die entsprechenden Dateien haben.

-del
(del - delete) Setzt die Optionen *+del* und *-rm* zurück.

-dru=dg

Für *dg* können Sie den in *.../CONFIG* (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*) definierten Namen einer Druckergruppe angeben. Der Auftrag wird dann auf einem Drucker dieser Druckergruppe ausgedruckt. Die Aufträge werden dabei gleichmäßig und dynamisch auf alle Drucker einer Gruppe verteilt. Wenn ein Drucker während einer Ausgabe ausfällt, wird der Druckauftrag in der gleichen Druckergruppe auf einem anderen Drucker, sofern vorhanden, ausgegeben.
Steht kein weiterer Drucker in der Druckergruppe im Zustand READY zur Verfügung, so kann der Druckauftrag durch ein Modifikationskommando in eine andere Druckergruppe gelenkt werden (siehe Option *-mp*).

-dru=dg nicht angegeben:
Der Auftrag wird auf einem bereiten Drucker der ersten in *.../CONFIG* eingetragenen Druckergruppe gedruckt (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*). Wenn eine terminal- oder benutzerbezogene Zuordnung existiert, wird der Auftrag auf einem Drucker der dort angegebenen Druckergruppe gedruckt. Wenn allerdings die Umgebungsvariable *TTYDEV* gesetzt ist, wird auf der dort angegebenen Druckergruppe ausgedruckt.

-form=fm

Mit *fm* wählen Sie ein Formular aus, auf dem der Druckauftrag ausgeführt wird. Für *fm* können Sie die Formular-Nummer oder den Formular-Namen angeben. Wenn die Druckerverwaltung mit Formularverwaltung betrieben wird (dazu muß die Datei *.../FORMTAB* existieren; siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*), wird der Druckauftrag in die entsprechende Formular-Warteschlange eingetragen. Der Druckauftrag wird erst dann ausgeführt, wenn der Drucker auf das entsprechende Formular eingestellt ist (siehe *Optionen für den Druckerverwalter bzw. den Systemverwalter, -forminit=Dnnn*).

Wenn die Druckerverwaltung ohne Formularverwaltung betrieben wird, wird die Option *-form=fm* lediglich an das Treiberprogramm weitergereicht (siehe *Optionen für die Backends (Treiberprogramme)*).

-id=n

Mit *n* kann ein Auftrag im Auftragspuffer über die Auftragsnummer (Spalte *ID* in der Ausgabe von *lpr -q*) identifiziert werden, wenn z.B. der Auftragsname nicht eindeutig ist. Alle Optionen, die für den so identifizierten Auftrag gelten sollen, müssen vor *id=n* angegeben werden.

-lang=Sprache

Diese Option stellt die Sprache ein, in der der Benutzer informiert wird. Der Parameter *Sprache* erlaubt nur die Werte der LANG-Variable. Diese sind als temporäre Werte folgendermaßen vordefiniert:

deutsch	LANG=De_DE.646
dänisch	LANG=Da_DK.646
niederländisch	LANG=Nl_NL.646
französisch	LANG=Fr_FR.646
italienisch	LANG=It_IT.646
spanisch	LANG=Es_SP.646
schwedisch	LANG=Sv_SE.646
englisch	LANG=En_US.ASCII

Deshalb können Sie den Parameter *Sprache* in der Form

```
-lang=LANG
```

aufrufen.

-mp␣auftrag␣...
-mp␣-id=nn␣...

Mit dieser Option können Sie die für einen oder mehrere Druckaufträge gültigen Optionen ändern. Sie müssen mindestens einen Druckauftrag angeben, können aber auch mehrere Aufträge, durch Leerzeichen getrennt, angeben. Dabei geben Sie für *auftrag* entweder die Auftragsnummer oder den Auftragsnamen an. Beide Angaben liefert das Kommando *lpr -q* in der Tabelle *Job Status* in den Spalten *ID* bzw. *Job*.

Wenn ein Druckauftrag noch nicht ausgeführt wird, dann können Sie alle Optionen zur Auftragsverwaltung modifizieren.

Wenn der Druckauftrag bereits ausgeführt wird, dann können Sie nur noch die folgenden Optionen angeben:

- Option -dru= ändert die Druckergruppe bei gestörtem Drucker.
- Option -ws= ändert die Druckergruppe bei gestörtem Drucker.
- Option -pr= ändert die Druckpriorität.

Vorsicht

Wenn Sie die Druckergruppe ändern, müssen Sie berücksichtigen, daß unter Umständen nicht alle angegebenen Optionen für die Treiberprogramme vom neuen Drucker ausgewertet werden können.

+msg

Nach der Beendigung eines Druckauftrags wird der Auftraggeber über das Kommando *mail* benachrichtigt. Wenn mit der Option *-to=benutzerkennung* ein anderer Benutzer als Empfänger des Auftrags benannt wurde, wird zusätzlich auch dieser Benutzer benachrichtigt.

-msg

Setzt die Optionen *+msg* bzw. *-no* wieder zurück.

-nc=n
Die nachfolgend angegebenen Dateien werden *n*-mal ausgedruckt. Für *n* können Sie ganze Zahlen zwischen 1 und 99 angeben.

-nc=n nicht angegeben:
Der Standard-Wert für *n* ist 1.

-no
wie die Option *+msg* (siehe oben).

-pr=n
lpr druckt die angegebenen Dateien mit der Priorität *n*. Für *n* können Sie ganze Zahlen zwischen 1 (niedrigste Priorität) und 20 angeben.
Als Systemverwalter können Sie 1 bis 30 angeben.
Aufträge werden in der Reihenfolge ihrer Priorität bearbeitet. Allerdings wird ein laufender Auftrag nicht von einem anderen Auftrag mit höherer Priorität unterbrochen.
Jeder Drucker besitzt einen gewissen Schwellenwert. An ihm werden nur Aufträge ausgegeben, deren Priorität über diesem Schwellenwert liegt.
Wenn Sie die Priorität eines laufenden Auftrags unter den Schwellenwert setzen, wird die Ausgabe unterbrochen. Erst wenn Sie die Priorität des Auftrags wieder über den Schwellenwert setzen, wird der Auftrag weiter ausgeführt. Der aktuelle Schwellenwert eines Druckers wird bei Abfrage der Druckerzustände in der Spalte LIMIT angezeigt (siehe Option *-q*).

-q[=fm]
Informationen ausgeben über den Zustand von Druckaufträgen und die Betriebsbereitschaft von Druckern. Das Kommando *lpstat* gibt ebenfalls Statusinformation über Druckaufträge aus.

Für *fm* können Sie eine Formular-Nummer oder einen Formular-Namen angeben. Es werden dann nur Informationen über Druckaufträge ausgegeben, die für das so spezifizierte Formular gestellt sind.

=fm nicht angegeben:
Es werden Informationen über alle Druckaufträge ausgegeben.

lpr gibt zwei Tabellen mit den Überschriften *Status of Printers* bzw. *Job Status* aus. Die Tabelle *Job Status* wird nur ausgegeben, wenn Druckaufträge zur Bearbeitung anstehen. Andernfalls gibt *lpr* die Meldung `"There are currently no print jobs awaiting processing".` aus.

Einträge in der Tabelle Status of Printers

PRINTER
Name des Druckers wie er in der Konfigurationsdatei .../*CONFIG* angegeben ist (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*). Diesen Namen erwartet *lpr* bei den Optionen: *-dd=, -dk=, -du=, -forminit=, -of=, -tst=, -ex=, -ld=, -rl=, -vex=, -vld=*

STATUS
zeigt den Betriebszustand des Druckers an. Folgende Betriebszustände sind möglich:

READY Der Drucker steht zur Ausführung eines Druckauftrags bereit.

RUNNING An diesem Drucker wird gerade ein Druckauftrag abgearbeitet. Der Ausgabestand dieses Auftrags wird in den folgenden Zeilen beschrieben.

LOCKED *lpr* nimmt zwar Aufträge für dieses Gerät an. Die Ausführung dieser Aufträge ist aber gesperrt.

EXCL.RES Der Drucker wird z.Zt. von einer anderen Druckerverwaltung beansprucht. Aufträge werden zwar angenommen, die Ausführung der Aufträge ist aber gesperrt.

FAULT Der Drucker ist wegen einer Störung nicht druckbereit. Ist die Leitung unterbrochen, der Drucker abgeschaltet oder die Störungsursache nicht feststellbar, so erscheint die Anzeige "Printer cannot be addressed". Je nach Druckertyp wird die Art der Störung angezeigt.

UNKNOWN Unbekannter Betriebszustand, z.B. wegen eines Fehlers in der Konfigurationsdatei. Es läuft kein Backend (Treiberprogramm). Der Systemverwalter muß das Backend neu laden oder vom Bediensystem aus neu starten.

TRIAL PRINT
Der Drucker ist von einem Benutzer für einen Probedruck belegt. Er ist z.Zt. für andere Benutzer nicht zugänglich. Aufträge können jedoch gestellt werden.

Folgende Übergangszustände sind möglich:

ABORT Ein Benutzer wünscht den Abbruch eines Druckauftrags. Das zuständige Backend hat den Abbruchbefehl jedoch noch nicht quittiert.

START OUTPUT
: Der Prozeß *.../daemon* (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*) hat dem Treiberprogramm den Druckauftrag übergeben, wartet aber noch auf eine Quittung.

POLL
: Das Backend prüft auf Veranlassung von *.../daemon* den Drucker, hat das Ergebnis aber noch nicht an *.../daemon* zurückgemeldet. (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*)

WAITING
: Die laufende Ausgabe auf dem Drucker ist möglicherweise unterbrochen. Ursachen können sein:
Papierende, Papierstau, Farbbandende, Lampe ON-LINE am Gerät brennt nicht oder dergl. Wird die Störung nicht behoben, so wird der Druckauftrag stamdardmäßig nach 4 Minuten abgebrochen.

LIMIT
: Schwellenwert für die Ausgabe-Priorität eines Druckers.

FM
: Nummer des Formulars, auf das der Drucker eingestellt ist. Wenn die Anzeige leer ist, ist der Drucker auf Formular 0 (Null) eingestellt.

ID
: Auftragsnummer des gerade laufenden Druckauftrags.

JOB
: Name der gerade ausgedruckten Datei.

P-GROUP
: Name der Druckergruppe, für die der Druckauftrag gestellt wurde. Diesen Namen erwartet *lpr* bei den Optionen *-ws=* bzw. *-dru=* .

PAGES
: Anzahl der bereits gedruckten Seiten.

Einträge in der Tabelle Job Status

ID
: Auftragsnummer des Druckauftrags.

JOB
: Auftragsname des Druckauftrags. Der Auftragsname ist der Name der auszudruckenden Datei bzw. die *titel*-Angabe bei der Option *-tl*. Nur über diesen Namen bzw. über die Auftragsnummer ist ein Druckauftrag ansprechbar.

USER
Kennung des Benutzers, der den Auftrag gegeben hat.

P_GROUP
Name der Druckergruppe, auf der dieser Auftrag ausgeführt werden soll.

FM
Nummer des Formulars, mit dem der Auftrag ausgedruckt wird. Wenn die Anzeige leer ist, wird das Formular 0 (Null) bzw. kein Formular angegeben.

LENGTH
Größe der auszudruckenden Datei in Byte.

KOP
Anzahl der insgesamt auszugebenden Kopien (siehe Option *-nc=n*).

PRIO
Priorität des Druckauftrags (siehe Option *-pr=n*).

TIME
Uhrzeit, zu der der Druckauftrag gestellt wurde.

%-CPL
Bereits ausgegebener Teil der Datei in Prozent. Die Angabe wird nach jeder ausgegebenen Seite neu berechnet.

-qadmin
Die in *.../CONFIG* konfigurierten Druckerverwalter werden ausgegeben (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*).

-qdru
Abfragen der in der Konfigurationsdatei eingetragenen Druckergruppen-Namen. *lpr* gibt eine Tabelle mit der Überschrift *Permitted Printer Groups* aus.

Einträge in der Tabelle "Permitted Printer Groups"

PRINTERGROUP
Name der Druckergruppe, wie er in der Datei *.../CONFIG* eingetragen ist (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in* SINIX V5.23 und SINIX V5.40).

PRINTERNAMES
Namen der Drucker, die zur angegebenen Druckergruppe gehören. Die Zuordnung von Druckern zu Druckergruppen ist in der Datei *.../CONFIG* festgelegt (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*).

PRINTER TYPE
Typenbezeichnung der betreffenden Drucker, wie sie in der Datei *.../CONFIG* bei der betreffenden Druckergruppe als Kommentar eingetragen ist, z.B. Drucker 9022 (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*).

-qform
Die von *.../daemon* unterstützten und in der Datei *.../FORMTAB* eingetragenen Formulare werden angezeigt (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*).

-rl=Dnnn
Ein Drucker im Blockierbetrieb (siehe *Optionen für den Druckerverwalter bzw. den Systemverwalter, +lkmod*) wird für den nächsten Auftrag freigegeben. Für *Dnnn* geben Sie den Namen des freizugebenden Druckers an. Diesen Namen erfahren Sie durch *lpr -q*. Einen Drucker im Zustand LOCKED können Sie mit *-rl=Dnnn* nicht wieder freigeben.

-rm
wie die Option *+del* (siehe oben).

-tl=titel
Der Druckauftrag wird unter dem Namen *titel* in der Auftragsverwaltung geführt und *titel* erscheint gegebenenfalls in der Titelzeile der Kopfseite des Ausdrucks.

-tl=titel nicht angegeben:
titel ist der Name der auszudruckenden Datei. Wenn Sie die Optionen -cp oder +co angeben oder *lpr* über eine Pipe aufrufen, lautet der Name der auszudruckenden Datei **sp.*.***.

-to=benutzerkennung
Empfänger des Ausdrucks ist der Benutzer mit der Benutzerkennung *benutzerkennung*, das heißt, *benutzerkennung* wird im Kopf und im Anhang des Ausdrucks anstelle der Benutzerkennung des Auftraggebers eingetragen.

Wenn die Optionen *+msg* oder *-no* gesetzt sind, werden sowohl Auftraggeber als auch Empfänger mit *mail* benachrichtigt.

+v
lpr gibt nach dem Aufruf eine Quittung aus.

-v
+*v* wird zurückgesetzt.

Standardmäßig ist -*v* gesetzt.

-ws=dg
wie die Option -*dru* (siehe oben).

Optionen für die Backends

Diese Optionen werden nicht von *lpr* ausgewertet, sondern unverändert an die Backends weitergeleitet.
Dabei ist zu beachten, daß nicht alle Optionen von allen Backends verstanden werden. Maßgebend dafür, welche Optionen an das jeweilige Backend weitergereicht werden, ist der jeweilige Eintrag in der Datei *.../CONFIG* (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*). So können Sie für ein selbst geschriebenes Treiberprogramm auch eigene Optionen definieren. Eine Option wird jedoch nur akzeptiert, wenn sie für *alle* Drucker einer Druckergruppe zugelassen ist.

-ab=n
Jede angegebene Datei wird erst ab Seite *n* ausgedruckt.

-band=n
Nur für Drucker 9047!
(*Betrieb des Druckers 9047 mit verschiedenen Typenbändern*)! Für *n* geben Sie die Band-ID eines Typenbandes an. Sie können die Band-ID sowohl dezimal (z.B. -*band=66*) als auch sedezimal angeben (z.B. -*band=0x42*); Sie müssen jedoch beachten, daß die Original-ID-Nummern der Bänder sedezimale Werte haben und deshalb die zugehörigen Banddateien im Dateiverzeichnis *.../band* ebenfalls diese sedezimalen Werte als Suffixe haben, in diesem Fall also *band.42*. Vor Beginn des Druckauftrages wird überprüft, ob das richtige Typenband im Drucker eingelegt ist. Ist ein falsches Band eingelegt, so wird der Druckauftrag zurückgestellt und der Drucker in den Zustand FAULT gesetzt. Der Auftraggeber wird über *mail* verständigt. Die Plausibilitätskontrolle wird in regelmäßigem Abstand wiederholt, und der Druckauftrag wird erst ausgeführt, wenn das richtige Band eingelegt ist.
Wenn Sie für *n* eine Band-ID angeben, zu der im Dateiverzeichnis *.../band* nicht die entsprechende Datei *band.n* existiert, dann kann Ihr Druckauftrag nicht ausgeführt werden, und Sie erhalten über *mail* eine entsprechende Nachricht.
Wenn die Option -*band=n* mit einer Band-ID, zu der keine Datei *band.n* existiert, in der Datei *.../CONFIG* eingetragen ist (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*), so wird das Laden des entsprechenden Backends generell abgelehnt, und der Systemverwalter erhält über *mail* eine entsprechende Nachricht.

-band=n nicht angegeben:
Zu Beginn des Druckauftrags wird die Band-ID des eingelegten Typenbandes abgefragt und automatisch die richtige Banddatei zur Codeumsetzung verwendet.

-bis=n
Jede angegebene Datei wird nur bis Seite *n* einschließlich ausgedruckt.

+cat
Alle Daten werden völlig unbehandelt wie beim Kommando *cat* vom Backend an den Drucker weitergegeben. Diese Option ist nur dann zu verwenden, wenn die normale Datenausgabe im Backend zu Problemen führt.

-cat
+cat wird wieder ausgeschaltet.

-ds=n
Die Variable n = 0 veranlaßt einseitiges Drucken, n = 1 dagegen führt zu doppelseitigem Drucken mit Bindung an der breiten Papierseite, n = 2 bewirkt ebenfalls doppelseitiges Drucken mit Bindung an der schmalen Papierseite. (Nur HP LaserJet II D - doppelseitiger HPLJ)

-dt
Deutschen Zeichensatz auswählen und laden. Die Angabe von *-dt* entspricht *zs=DTSH* (siehe unten).

-filter=n
Auswahl eines Filterprogramms, das die SINIX-Darstellung eines Dokumententextes in den Steuercodes eines Druckers umsetzt. Mit *n* bezeichnen Sie das Filterprogramm. Derzeit ist die Angabe der Option *-filter=* nur für das PostScript-Backend zulässig, da derzeit nur Filterprogramme für PostScript-Drucker existieren. Das aufgerufene Filterprogramm muß mit vollem Pfadnamen in der Datei *.../filtertabs* eingetragen sein.

-font=n
Aufruf eines druckerspezifischen Zeichensatzes (font). Mit *n* kennzeichnen Sie symbolisch einen bestimmten Zeichensatz. Für *n* können Sie Dezimalzahlen oder Sedezimalzahlen angeben. sedezimalzahlen müssen Sie 0x voranstellen, etwa *0x1d* für den Zeichensatz mit der Nummer 29. Die Angaben haben bei den einzelnen Druckern unterschiedliche Bedeutung (siehe unten). Existiert ein angegebener Zeichensatz nicht, so wird mit dem Standard-Zeichensatz gedruckt. Für die Drucker 9022 und HP-LaserJet werden unbekannte Fontnummern an den Drucker durchgereicht. Die zur Verfügung stehenden Zeichensätze entnehmen Sie bitte dem entsprechenden Druckerhandbuch.

-font=n nicht angegeben:
Für *n* wird 1 angenommen, d.h. es wird mit dem Standard-Zeichensatz des jeweiligen Druckers gedruckt.

Bei den druckerresidenten Zeichensätzen werden die möglichen Werte für die Zeilen- und Spaltenzahl pro Seite automatisch eingestellt, bei optionalen Zeichensätzen (Kassetten) bzw. rechnerresidenten Zeichensätzen (ladbare Fonts) müssen Sie entsprechende Angaben bei Abweichung vom Standard-Zeichensatz explizit mit den Optionen *-pb=n* bzw. *-pl=n* machen.

Zeichensatz-Angaben:

Drucker 9001 und 9011:

-font=9:	Schönschrift (NLQ)
-font=10:	Schnellschrift (DATA)

Drucker 9012 und 9013:
die Nummern der Zeichenvorräte (siehe *SINIX SPOOL* [15])

Drucker 9022:
die Identifikationscodes der Zeichensätze (siehe *SINIX SPOOL* [15]).

Beim Drucker 9022 bestimmt die Option *-font=* gleichzeitig den Schnitt (das Aussehen) der Schrift, die Größe der Zeichen und ihre Orientierung.

Wegen der in SINIX üblichen Darstellungsform ist zu beachten, daß bei sedezimaler Schreibweise die beiden ersten Zeichen des Identifikationscodes mit den beiden letzten Zeichen vertauscht werden müssen.

Beispiel

Wenn der Identifikationscode für einen Zeichensatz *0A30* ist, müssen Sie angeben:

- in sedezimaler Form: `-font=0x300a`
- in dezimaler Form: `-font=2608`

Drucker 9025:

-font=1:	COURIER10:E.N.10.P	Portrait
-font=2:	COURIER10:E.N.10.L	Landscape
-font=3:	MODERN12:E.N.10.P	Portrait
-font=4:	MODERN12:E.N.10.L	Landscape
-font=5:	BASKERVILLE.N.12.P	Portrait
-font=6:	BULLETIN.N.6.P	Portrait

Mit HP-LaserJet-Series-II kompatible Drucker:
die Nummern der Zeichenvorräte (siehe *SINIX SPOOL* [15]).

Bei mit HP-LaserJet-Series-II kompatiblen Druckern bestimmt die Option *-font=* nur den Schnitt (das Aussehen) der gedruckten Zeichen. Die Größe der Zeichen und ihre Orientierung müssen gesondert eingestellt werden.

Man kann bei **-font=** auch einen Dateinamen angeben. Dieser muß unter .../font/<drucker> (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*) stehen und enthält ladbare Zeichengeneratoren. Dieser Dateiname darf nicht mit einer Ziffer beginnen.

-form=fm

Mit *fm* wird ein bestimmtes Formular für den Ausdruck ausgewählt. Die auszudruckenden Dateien werden dann entsprechend formatiert, die bedruckbare Zeilen- und Seitenlänge wird festgelegt. Zur Zeit werden vom SPOOL-System nur 7 Formulare standardmäßig unterstützt:

-form=1	oder	-form=breit	(breites 12" Endlospapier)
-form=2	oder	-form=schmal	(schmales 12" Endlospapier)
-form=3	oder	-form=Portrait	(DIN-A 4 Papier hochkant)
-form=4	oder	-form=Landscape	(DIN-A 4 Papier quer)
-form=5	oder	-form=Executive	(engl. Format 7.25" x 10.5")
-form=6	oder	-form=Letter	(engl. Format 8.5" x 11")
-form=7	oder	-form=Legal	(engl. Format 8.5" x 14")

Laserdrucker können nur auf geschnittenes Papier, andere Drucker auch auf Endlospapier drucken. Andere Angaben für *fm* werden von den Treiberprogrammen ignoriert. Bei anderen Formularen muß der Anwender selbst für die Formatierung sorgen, das heißt -pl= und -pb= angeben. (siehe *Optionen zur Auftragsverwaltung*).

+hd

(hd - head) Kopfseite ausdrucken.

-hd

(hd - head) Kopfseite unterdrücken.

-hdgrp

Kopfseite nur bei Wechsel der Benutzergruppe ausdrucken.

-hop=n

Es wird der *n*-te Schacht für die Papierzuführung ausgewählt (bei mehreren Schächten).

-hop=n nicht angegeben:
n = 1 (Schacht 1).

-int

ASCII-Zeichensatz auswählen und laden. Die Angabe von *-int* entspricht *zs=ASCI* (siehe unten). Diese Option ist standardmäßig gesetzt, wenn kein anderer Zeichensatz ausgewählt wurde.

-mar=n
Der linke Druckrand wird in jeder Zeile um *n* Millimeter eingerückt. Das geht aber nur so gut, wie es dem Drucker möglich ist. Je nach Druckerauflösung kann es zu Auf- oder Abrundung der Millimeter-Angaben kommen.

-nk=n
Es werden *n* Kopien der angegebenen Dateien ausgedruckt. Die Ausführung der Kopien wird vom Drucker selbst verwaltet. Jede einzelne Seite wird *n*-mal ausgedruckt, bevor mit dem Druck der nächsten Seite begonnen wird. Dadurch wird eine Entlastung des Systems erreicht. Wegen des Zeitverhaltens sollten nicht mehr als 10 bis 15 Kopien auf diese Weise eingestellt werden.

-pb=n
Es werden maximal *n* Spalten pro Zeile ausgedruckt. Ein evtl. vorhandener Zeilenrest wird nicht gedruckt.

-pb=n nicht angegeben:
n ist die maximal vom jeweiligen Drucker ausdruckbare Anzahl von Zeichen, ohne daß Zeichen übereinander bzw. in der nächsten Zeile gedruckt werden.

-pb1
Zeichenbreite von 10 Zeichen pro Zoll. *-pb1* ist die Kurzform für *-zb=1*.

-pb2
Zeichenbreite von 12 Zeichen pro Zoll. *-pb2* ist die Kurzform für *-zb=2*.

-pb3
Die kleinste für einen Drucker verfügbare Zeichenbreite.

-pl=n
Mit *pl* (page length) wird die Seitenlänge eingestellt, wobei *n* die Zeilen pro Seite bei 6 Zeilen pro Zoll angibt. Die Schrifthöhe wird dabei nicht beachtet.

Diese Option erlaubt die Verwendung von anderen Formularhöhen als 12 Zoll. Bei Einzelblattzuführung wird nach *n* Zeilen ein neues Blatt eingezogen.

-pl=n nicht angegeben:
Bei Endlospapier ist *n*=72.
Bei Einzelblattzuführung ist der Wert für *n* druckerabhängig.

+ps
Proportionalschrift einschalten.

-ps
Proportionalschrift ausschalten.

+tab
Hardwaretabulatur einschalten. Horizontal-Tabulatorzeichen werden vom Backend direkt an den Drucker geschickt.

-tab
Hardwaretabulatur ausschalten. Horizontal-Tabulatorzeichen werden vom Backend durch die entsprechende Anzahl von Leerzeichen ersetzt.

Die Option *-tab* ist standardmäßig gesetzt.

-top=n
Die erste zu druckende Zeile wird auf jeder neuen Seite um *n* Millimeter nach unten verschoben (Kopfrand). Je nach Druckerauflösung kann es zu Auf- oder Abrundung der Millimeter-Angaben kommen.

+trl
Endeseite ausdrucken.

-trl
Endeseite unterdrücken.

-trlgrp
Endeseite nur bei Wechsel der Benutzergruppe ausdrucken.

+vp
Die Eingabedatei wird durch den virtuellen Drucker übersetzt.

-vp
Die Eingabedatei wird nicht durch den virtuellen Drucker übersetzt.

-za=n
Der Zeilenabstand wird so eingestellt, daß *n* Zeilen pro Zoll gedruckt werden. Die Seitenlänge sowie die Schrifthöhe werden dabei nicht verändert.

-za=n nicht angegeben:
n = 6, bei einigen Druckern ist der Zeilenabstand abhängig von der Schriftart.

-zb=zeichenbreite
Zeichenbreite auswählen. Für *zeichenbreite* sind folgende Angaben möglich:

1 entspricht 10 Zeichen pro Zoll (Standard)
2 entspricht 12 Zeichen pro Zoll
3 entspricht 13 Zeichen pro Zoll
4 entspricht 15 Zeichen pro Zoll
5 entspricht 17 Zeichen oder mehr pro Zoll (beim Seitendrucker 9025 21 Zeichen pro Zoll)

Welche Zeichenbreiten für einen Drucker jeweils verfügbar sind, entnehmen Sie bitte dem entsprechenden Druckerhandbuch.

-zb=zeichenbreite nicht angegeben:
Für *zeichenbreite* wird 1 angenommen.

-zs=zeichensatz
Zeichensatz auswählen und laden. für *zeichensatz* sind folgende Angaben möglich:

DTSH	deutscher Zeichensatz
INT	internationaler Zeichensatz
ENGL	englischer Zeichensatz
ASCI	ASCII-Zeichensatz
DAEN	dänischer Zeichensatz
FINN	finnischer Zeichensatz
FRNZ	französischer Zeichensatz
SPAN	spanischer Zeichensatz
NIED	niederländischer Zeichensatz
ITAL	italienischer Zeichensatz
SWED	schwedischer Zeichensatz
NORW	norwegischer Zeichensatz
BELG	belgischer Zeichensatz
CH	schweizer Zeichensatz
8859-1	Internationaler 8-Bit-Zeichensatz ISO-8859-1 (Siehe *SINIX SPOOL* [15], *A.2 Unterstützung des ISO-8859-1-Zeichensatzes*).

Welche Zeichensätze jeweils für einen Drucker verfügbar sind, entnehmen Sie bitte dem entsprechenden Druckerhandbuch.

Backend für den Seitendrucker 9025

Bei allen länderspezifischen Zeichensätzen ist die Funktion des Druckers 'fließende Akzente' ('floating accent') eingeschaltet, d.h. daß Akzente über dem nachfolgenden Buchstaben gedruckt werden.
Nur bei -zs=INT und -zs=ASCI wird diese Funktion ausgeschaltet (siehe *9025 - Schnittstelle für Programmierer, 3.10 Universelle Akzente* [16]).

-zs=zeichensatz nicht angegeben:
Für *zeichensatz* wird *ASCI* angenommen.

Welche Optionen für einen Drucker verfügbar sind, können Sie der folgenden Tabelle entnehmen:

Optionen	9001	9004	9011	9012	9013	9022	9025	9047	HP-LaserJet	PostScript Drucker
-ab=	*	*	*	*	*	*	*	*	*	*
-band=								*		
-bis=	*	*	*	*	*	*	*	*	*	*
+cat/-cat	*	*	*	*	*	*	*	*	*	
-ds=									* *)	
-dt	*		*	*	*	*	*		*	
-filter=	*	*	*	*	*	*	*	*	*	*
-font=	*		*	*	*	*	*		*	
-form=	*	*	*	*	*	*	*	*	*	*
+hd/-hd	*	*	*	*	*	*	*	*	*	
-hop=		*	*	*	*	*	*		*	*
-int	*		*	*	*	*	*		*	
-mar=	*	*	*	*	*	*	*	*	*	
-nk=						*	*		*	
-pb=	*	*	*	*	*	*	*	*	*	
-pb1	*	*	*	*	*	*	*	*	*	
-pb2	*	*	*	*	*	*	*		*	
-pb3	*		*	*	*	*	*		*	
-pl=	*	*	*	*	*	*	*	*	*	
+ps/-ps	*	*	*	*	*	*	*		*	
+tab/-tab	*	*	*	*	*	*	*			
-top=	*	*	*	*	*	*	*	*	*	
+trl/-trl	*	*	*	*	*	*	*	*	*	*
+vp/-vp	*		*	*	*	*	*		*	*
-za=	*	*	*	*	*	*	*		*	
-zb=	*		*	*	*	*	*		*	
-zs=	*		*	*	*	*	*		*	

*) nur HP-LaserJet IID

Wenn einzelne Optionen standardmäßig gesetzt sein sollen, so können sie auch direkt in die CONFIG-Datei eingetragen werden (siehe *SINIX SPOOL, Konfiguration* [15]).

Optionen für Konfiguration der Backends

Die Optionen dieses Abschnitts dienen zur Konfiguration des SPOOL-Systems und der Backends. Mit diesen Optionen werden notwendige Einstellungen festgelegt.

Achtung

Die Optionen dieses Abschnitts darf nur der Systemverwalter in die CONFIG-Datei eintragen. Dort darf er diese Optionen nur als Teil des Backend-Aufrufs verwenden.

-addr=Druckername
Internet-Adresse für einen TACLAN-Drucker einstellen. Für *Druckername* geben Sie entweder die Internet-Adresse oder den symbolischen Namen des Drucker an, der über TACLAN angeschlossen ist.

Hinweis

Die Optionen *-addr=* und *-port=* gehören zusammen und müssen gleichzeitig angegeben werden.

-bits=n
Für die Datenübertragung zum Drucker werden *n* Bits pro Zeichen benutzt. Erlaubt sind 7 und 8 Bits pro Zeichen.

-esca
Der Drucker 9025 wird so eingestellt, daß er statt des Zeichens *ESC* (0x1b) das Zeichen \ erwartet. Anstelle des Zeichens *ESC* wird dann das Zeichen \ vom SPOOL-System gesendet.

Hinweis

Die Option *-esca* ist nur für den Drucker 9025 zulässig.

-escb
Der Drucker 9025 wird so eingestellt, daß er statt des Zeichens *ESC* (0x1b) das Zeichen ~ erwartet. Anstelle des Zeichens *ESC* wird dann das Zeichen ~ vom SPOOL-System gesendet.

Hinweis

Die Option *-escb* ist nur für den Drucker 9025 zulässig.

+lkmod (S)(D)
Diese Option können Sie nur in der Datei *.../CONFIG* hinter dem Namen eines Treiberprogramms angeben (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*)! Die Druckerverwaltung wird dann den Drucker, der dieses Backend verwendet, in den Blockierbetrieb setzen. Nach jedem Auftrag wird der Weiterdruck angehalten, um z.B. das Papier wechseln zu können (Plotterunterstützung). Mit den Optionen *-rl=Dnnn* oder *-du=Dnnn* kann der Drucker wieder freigegeben werden.

-parity=bitval

Einstellen des Paritätsbits in der Datenübertragung. Für den Parameter *bitval* sind nur folgende Werte zulässig.
Diese Option gilt nur für das PostScript-Backend.

Parameter	Bedeutung
odd	ungerade Parität
even	gerade Parität
off	keine Parität

-perm

Verbindung zum TACLAN-Drucker ist nicht permanent. Das Backend gibt die Verbindung zum TACLAN-Drucker nach jedem abgearbeitetem Auftrag wieder frei.

+perm

Verbindung zum TACLAN-Drucker ist permanent. Das Backend gibt die Verbindung zum TACLAN-Drucker nicht frei.

-polltmo=t (S)

Mit dieser Option stellen Sie die Wartezeit zwischen den Statusabfragen des Backends ein. Das Backend wartet *t* Sekunden zwischen zwei Statusabfragen. Für *t* können Sie 0 (Null) und alle Werte zwischen 10 und 65535 einsetzen. Wenn Sie für *t* den Wert 0 (Null) einsetzen, macht das Backend keine Statusabfragen.
Wenn die Option *-polltmo=* nicht angegeben ist, wartet das Backend 80 Sekunden zwischen zwei Statusabfragen.

-port=n

Einstellen der TCP-Port-Nummer für einen TACLAN-Drucker. Der Wert *n* entspricht der TCP-Port-Nummer die bei der TACLAN-Konfigurierung angegeben wurde.

Hinweis

Die Optionen *-port=* und *-addr=* gehören zusammen und müssen gleichzeitig angegeben werden.

-speed=br

br gibt die Geschwindigkeit an, mit der die Ausgabedaten zum Drucker übertragen werden sollen. Die Geschwindigkeit muß der am Drucker eingestellten entsprechen. Für 9001 kann die Geschwindigkeit am Drucker nicht verändert werden (immer 9600 Baud). Als Parameter **br** sind folgende Werte zulässig:

Parameter	Geschwindigkeit
B110	110 Baud
B150	150 Baud
B200	200 Baud
B300	300 Baud
B1200	1200 Baud
B1800	1800 Baud
B2400	2400 Baud
B4800	4800 Baud
B9600	9600 Baud
B19200	19200 Baud

-stopb=bits
Einstellen der Anzahl der Stop-Bits in der Datenübertragung. Diese Option gilt nur für das PostScript-Backend. Für den Parameter *bits* sind nur die folgenden Werte zulässig:

Parameter	Bedeutung
1	1 Stop-Bit
2	2 Stop-Bits

-waittmo=t (S)
Mit dieser Option stellen Sie die Wartezeit ein, die das SPOOL-System zweimal verstreichen läßt, bevor es einen Druckauftrag nach einem Druckerfehler abbricht. Für *t* setzen Sie die Wartezeit in Sekunden ein. Für *t* sind alle Werte zwischen 10 und 65535 erlaubt. Es ist sinnvoll für *t* einen Wert zu wählen, der größer ist als das Doppelte der Wartezeit zwischen zwei Statusabfragen des Backends.
Wenn die Option *-waittmo=* nicht angegeben wird, ist die Wartezeit auf 160 Sekunden eingestellt.

Optionen für den Druckerverwalter bzw. den Systemverwalter

- Optionen, die nur der Systemverwalter verwenden darf, sind mit (S) gekennzeichnet.
- Optionen, die Druckerverwalter und Systemverwalter verwenden dürfen, sind mit (S)(D) gekennzeichnet.

-dd=Dnnn (S)(D)
Drucker sperren. Für *Dnnn* geben Sie den Drucker an, den Sie sperren wollen. Den Namen des Druckers erhalten Sie mit dem Kommando *lpr -q*. Weitere Aufträge für den Drucker nimmt *lpr* zwar an, führt sie aber nicht aus. Ein eventuell laufender Druckauftrag wird noch zu Ende geführt. Bei der Statusabfrage mit *lpr -q* wird der Drucker als LOCKED gekennzeichnet. Ein gesperrter Drucker wird nicht mehr von der Drucker-Verwaltung zyklisch abgefragt und kann für Anwendungen außerhalb der SINIX-Druckerverwaltung benutzt werden.

-dg (S)
Druckerbetrieb abschalten, d.h. die Programme zur Druckerverwaltung werden beendet.
Alle laufenden Aufträge werden noch ausgeführt. Aufträge in der Warteschlange bleiben erhalten.

-dk=Dnnn (S)(D)
Drucker sperren (wie Option *-dd=Dnnn*), zusätzlich gilt: Ein eventuell laufender Druckauftrag wird abgebrochen. Wird der Drucker später wieder freigegeben, wird der abgebrochene Druckauftrag von Anfang an wiederholt. Für *Dnnn* geben Sie den Drucker an, den Sie sperren wollen. *nnn* bezeichnet die Nummer des Steckplatzes. Den Namen des Druckers erhalten Sie auch mit dem Kommando *lpr -q*.

-du=Dnnn (S)(D)
Drucker freigeben. Für *Dnnn* geben Sie den Drucker an, den Sie freigeben wollen. Den Namen des Druckers erhalten Sie, wenn Sie das Kommando *lpr -q* eingeben.

-ex=Dnnn (S)
Mit dieser Option wird ein Drucker aus der Druckerverwaltung herausgenommen und in den Zustand UNKNOWN versetzt. Für *Dnnn* geben Sie den Drucker an, den Sie in den Zustand UNKNOWN versetzen wollen. Den Namen des Druckers erhalten Sie mit dem Kommando *lpr -q*. Nur wenn ein Drucker sich in diesem Zustand, im Zustand LOCKED oder im Zustand EXCL.RES befindet, dürfen Sie mit eigenen Anwenderprogrammen auf diesen Drucker zugreifen.

-forminit=Dnnn (S) (D)
Mit dieser Option wird ein Drucker auf ein bestimmtes Formular eingestellt. Für *Dnnn* geben Sie den Namen des Druckers an. Den Namen des Druckers erhalten Sie mit *lpr -q*. Das Formular, auf das der Drucker *Dnnn* eingestellt werden soll, geben Sie mit der Option *-form=fm* an. Alle für das Formular *fm* gestellten Druckaufträge für den Drucker *Dnnn* können dann ausgedruckt werden. Ein auf *Dnnn* noch laufender Auftrag wird zu Ende ausgeführt. Nach dem Neustart der Druckerverwaltung bleibt die zuletzt aktuelle Formulareinstellung der Drucker gültig. Um den Formularbetrieb für den Drucker *Dnnn* wieder aufzuheben, geben Sie *lpr -form=0 -forminit=Dnnn* ein.

-ld=Dnnn (S)
Mit dieser Option kann ein Drucker, der mit der Option *-ex* aus der Druckerverwaltung herausgenommen wurde, d.h. im Zustand UNKNOWN ist, wieder in die Druckerverwaltung hineingenommen werden. Für *Dnnn* geben Sie den Drucker an, den Sie wieder in die Druckerverwaltung hineinnehmen wollen. Den Namen des Druckers erhalten Sie mit dem Kommando *lpr -q*.

-of=Dnnn (S)(D)
Für den angegebenen Drucker den Schwellenwert für die Ausgabe-Priorität festlegen. Für *Dnnn* geben Sie den Drucker an, für den Sie die Ausgabepriorität festlegen wollen. Den Namen des Druckers erhalten Sie mit dem Kommando *lpr -q*.

Den Schwellenwert für die Ausgabepriorität müssen Sie bei der Option *-of=* mit der Option *-pr=n* angeben. Dabei kann *n* eine ganze Zahl zwischen 0 und 20 (D) bzw. 0 und 30 (S) sein.

Aufträge für diesen Drucker mit geringerer oder gleicher Priorität werden anschließend zwar angenommen, aber nicht gedruckt.

Wenn bei laufendem Druckauftrag der Schwellenwert über den Prioritätswert dieses Auftrags angehoben wird, wird der Auftrag abgebrochen. Erst wenn der Schwellenwert wieder unter dem Prioritätswert liegt, wird der Druckauftrag seitenrichtig an der Stelle fortgesetzt, an der er unterbrochen wurde.

-of=Dnnn nicht angegeben:
Der Schwellenwert für die Ausgabepriorität ist 2.

-rr (S)
lpr überprüft, ob die Binärversion der Konfigurationsdatei älteren Datums ist als die Datei *.../CONFIG* (siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*). Haben Sie diese während des laufenden Betriebs geändert, übersetzt *lpr* die Konfigurationsdatei neu. Die Änderungen wirken ab dem nächsten Auftrag, der ausgedruckt wird.
Drucker im Zustand LOCKED bzw. EXCL.RES können mit diesem Kommando *nicht* wieder freigegeben werden.

-su=benutzername (S)
Sie können Druckaufträge des Benutzers *benutzername* löschen oder modifizieren (siehe Optionen *-ca* bzw. *-mp*).

-tst=Dnnn (S)(D)
Probedruck anstoßen. Für *Dnnn* geben Sie den Drucker an, an dem Sie den Probedruck anstoßen wollen. *nnn* bezeichnet die Nummer des Steckplatzes. Den Namen des Druckers erhalten Sie auch mit dem Kommando *lpr -q*.
Anschließend geben Sie die Optionen und den Namen der Datei an, die Sie probeweise drucken wollen. Nach dem Aufruf *lpr -tst=Dnnn option... datei* beginnt ein Dialog mit dem Auftraggeber. Nach dem Ausdruck der ersten Seite wird der Druckbetrieb angehalten, und der Auftraggeber wird aufgefordert, das weitere Vorgehen anzugeben:

YES	der Ausdruck der ersten Seite wird wiederholt.
NO	der Ausdruck wird vollständig ausgeführt.
END	der Ausdruck wird abgebrochen.

Einen Probedruck können Sie nur auf einem Drucker, der sich im Zustand LOCKED befindet, anstoßen (siehe Option *-dd* bzw. *-dk*). Wenn ein Drucker von einem Benutzer für die Funktion Probedruck belegt ist, ist er für andere Benutzer nicht zugänglich. Nach Beendigung des Ausdrucks steht der Drucker allgemein wieder zur Verfügung.

-vex=Dnnn (S)
Diese Option nimmt wie die Option *-ex=Dnnn* einen Drucker aus der Druckerverwaltung heraus. Im Gegensatz zu *-ex=Dnnn* wird das Kommando nur ausgeführt, wenn sich der Drucker im Zustand READY befindet und unmittelbar der Druckerverwaltung entzogen werden kann. Für *Dnnn* geben Sie den Drucker an, den Sie aus der Druckerverwaltung herausnehmen wollen. Den Namen des Druckers erhalten Sie mit dem Kommando *lpr -q*.

Die Kommandoausführung erfolgt synchron. Sie können also den Ende-Status abfragen, um festzustellen, ob der Aufruf erfolgreich war oder warum der Auftrag nicht ausgeführt werden konnte. Auf diese Weise wird gewährleistet, daß der Drucker exklusiv nur von einem Anwenderprogramm außerhalb der Druckerverwaltung angefordert werden kann. Bei Erfolg wird der Drucker in den Zustand EXCL.RES gesetzt.

-vld=Dnnn (S)
Mit dieser Option können Sie einen Drucker im Zustand EXCL.RES wieder in die Druckerverwaltung hineinnehmen. Ein Drucker im Zustand EXCL.RES kann nur auf diese Weise wieder allgemein zugänglich gemacht werden. Für *Dnn* geben Sie den Drucker an, den Sie wieder in die Druckerverwaltung hineinnehmen wollen. Den Namen des Druckers erhalten Sie mit dem Kommando *lpr -q*.

datei
Name der Datei, die ausgedruckt werden soll.
Sie können mehrere Dateien, durch Leerzeichen voneinander getrennt, angeben. Alle Optionen, die für den Ausdruck einer Datei gelten sollen, müssen vor dieser Datei angegeben werden.

datei nicht angegeben:
lpr liest von der Standard-Eingabe.

FEHLERMELDUNGEN

Wenn Sie Optionen angeben, die nur der Systemverwalter oder ein Druckerverwalter angeben darf, wird eine entsprechende Fehlermeldung ausgegeben.

Wenn ein Druckauftrag fehlerhaft beendet wurde, erhalten Sie eine Fehlermeldung mit *mail*.

DATEIEN

.../CONFIG
Systemdatei für Druckerbeschreibungen

.../POOLDAT
Systemdatei für die Druckerverwaltung

.../band
Dateiverzeichnis mit den Banddateien für den Drucker 9047

(siehe *Tabellen und Verzeichnisse, Dateien des SPOOL-Systems in SINIX V5.23 und SINIX V5.40*)

INTERNATIONALE UMGEBUNG

Wenn Sie nicht in einer englischen, sondern in einer anderssprachigen Umgebung arbeiten, dann gibt *lpr* die Meldungstexte in dieser Sprache aus. Die Sprache wird durch die NLS-Umgebungsvariable *LANG* definiert.

Weitere Informationen zur internationalen Umgebung finden Sie im *Kapitel 3, Internationale Umgebung - NLS.*

lpr ist 8-bit-transparent.

BEISPIELE

1. Ein Auftrag mit der Auftragsnummer 2 soll auf die Priorität 1 gesetzt werden:

```
$ lpr -pr=1 -mp -id=2
$ lpr -q
                S T A T U S   O F   P R I N T E R S

 PRINTER  STATUS         LIMIT FM    ID  JOB                      P-GROUP  PAGES

 D002     RUNNING          2          1  datei1                     ALLE       6

                  J O B   S T A T U S

   ID  JOB                      USER      P_GROUP  FM LENGTH COP PRIO TIME %-CPL

    1 datei1                    michael   ALLE        15357   1   15  13:54  48
    2 datei2                    michael   ALLE        31960   1    1  13:54   0
```

2. Der Schwellenwert für den Drucker D003 soll auf 10 gesetzt werden:

```
$ lpr -pr=10 -of=D003
$ lpr -q
                S T A T U S   O F   P R I N T E R S

 PRINTER  STATUS         LIMIT FM    ID  JOB                      P-GROUP  PAGES

 D003     READY            10

There are currently no print jobs awaiting processing.
```

3. Sie löschen als Systemverwalter den Druckauftrag des Benutzers *harald* mit der Auftragsnummer 13.

```
# lpr -su=harald -ca -id=13
```

SIEHE AUCH

cancel, *cat*, *lp*, *lpstat*, *mail*, *pr*

SINIX SPOOL [15]

lpstat
Informationen über Druckaufträge ausgeben (line printer status)

Unter SINIX 5.4 können Sie alternativ zwei Spool-Systeme einsetzen:

- den SINIX-Spooler und
- den Original-AT&T-Spooler.

Das Kommando *lpstat* können Sie für beide Spooler benutzen. Die aufgeführten Optionen werden vom SINIX-Spooler jedoch nicht unterstützt.

lpstat gibt Informationen über alle mit *lp* gestellten Druckaufträge aus. Informationen über Druckaufträge, die Sie mit *lpr* gestellt haben, erhalten Sie mit dem Kommando *lpr -q*.

```
lpstat[ option]   [auftragsnummer] ...
```

auftragsnummer
: Argumente, die keine der unten aufgeführten Optionen sind, interpretiert *lpstat* als Auftragsnummern von Druckaufträgen. *lpstat* zeigt dann nur Informationen zu Aufträgen mit den entsprechenden Nummern.

keine Auftragsnummer und keine Option angegeben
: *lpstat* gibt alle vom Benutzer mit *lp* gestellten und noch nicht vollständig ausgeführten Druckaufträge aus.

option
: Sie dürfen die Optionen in beliebiger Reihenfolge und mit anderen Argumenten vermischt angeben.

liste
benutzerliste
: Bei mehreren der unten beschriebenen Optionen können Sie Listen angeben. Die einzelnen Listenelemente schreiben Sie entweder durch Kommas getrennt. Oder Sie trennen sie durch Leerzeichen und schließen sie in Anführungzeichen ein, z.B.:

  user1,user2,user3 oder
  "user1 user2 user3".

  Geben Sie statt einer Liste nichts oder *all* an, zeigt *lpstat* alle für die angegebene Option relevanten Informationen. Das Kommando *lpstat -o all* z.B. gibt den Zustand aller Druckaufträge aus. Ebenso verhält sich das Kommando *lpstat -o*.

-a[␣liste]
(a - accept) *lpstat* informiert, ob die Drucker oder Druckergruppen in *liste* Aufträge annehmen. In *liste* können Sie die Namen von beliebig vielen Druckern bzw. Druckergruppen angeben.

-c[␣liste]
(c - class) *lpstat* gibt die Namen aller bzw. der in *liste* angegebenen Druckergruppen und der darin enthaltenen Drucker aus. In *liste* können Sie die Namen von beliebig vielen Druckergruppen angeben.

-d
(d - default) *lpstat* gibt den voreingestellten Zielort für Druckaufträge aus.

-f[␣liste][-l]
(f - forms) *lpstat* gibt Auskunft, ob der Druckerverwaltung die in *liste* angegebenen Formulare bekannt sind. In *liste* können Sie einen oder mehrere Formulare eintragen. Die Voreinstellung für *liste* ist *all*. Mit der Option *-l* erhalten Sie eine Liste der Formularbeschreibungen.

-o[␣liste]
(o - output) *lpstat* informiert über die gestellten Druckaufträge. Für *liste* können Sie sowohl Druckernamen, als auch Druckergruppennamen und Auftragsnummern angeben. Die Option *-o* können Sie auch weglassen, da *lpstat -o* dem Kommando *lpstat* entspricht.

-p[␣liste][-D][-l]
(p - printer) *lpstat* informiert über den aktuellen Zustand von Druckern. *liste* enthält einen oder mehrere Druckernamen. Haben Sie die Option *-D* gesetzt, druckt *lpstat* für alle in *liste* angegebenen Drucker eine kurze Beschreibung aus. Wenn die Option *-l* gesetzt ist, und der Drucker an den lokalen Rechner angeschlossen ist, liefert *lpstat* zu jedem Drucker eine ausführliche Beschreibung. Dazu gehören unter anderem Angaben zu eingestelltem Formular, zulässigen Inhaltstypen sowie verwendeter Schnittstelle. Wenn die Option *-l* gesetzt ist, und sich der Drucker im Netz befindet, zeigt *lpstat* lediglich den fernen Rechner und die Druckernamen an, gefolgt von den Shell-Kommandos, die Sie zum Datenaustausch und für die Ausführung im Netz verwenden können.

-r
(r - request scheduler) *lpstat* gibt aus, ob der Dämon ein- oder ausgeschaltet ist.

-R
lpstat gibt die Position aus, die ein angegebener Auftrag in der Warteschlange zum Drucker hat.

-s

(s - status) *lpstat* gibt umfassende Informationen über den aktuellen Zustand der Druckerverwaltung aus. Diese Informationen beinhalten

- den Zustand des Dämon,
- den vom System voreingestellten Zielort,
- die Namen aller Druckergruppen und deren Drucker,
- eine Liste aller Druckern samt angeschlossener Geräte,
- alle die Druckerverwaltung beanspruchenden Rechner,
- eine Liste aller momentan eingelegten Formulare und
- alle bekannten Zeichensätze und Typenräder.

-S[␣liste][**-l**]

lpstat gibt Auskunft, ob die in *liste* spezifizierten Zeichensätze oder Typenräder der Druckerverwaltung bekannt sind. Die Voreinstellung für *liste* ist *all*. Haben Sie die Option *-l* gesetzt, zeigt *lpstat* zu jedem angegebenen Zeichensatz oder Typenrad alle Drucker an, die damit arbeiten können. Außerdem gibt *lpstat* aus, ob der Zeichensatz aktuell eingestellt bzw. das Typenrad aktuell montiert ist. Ist das nicht der Fall, spezifiziert *lpstat* ggf. einen druckerspezifischen Zeichensatz, der sattdessen verwendet wird.

-t

(t - total) *lpstat* zeigt die gesamte Zustandsinformation, d.h. alle mit der Option *-s* erhältlichen Daten einschließlich Informationen darüber, wie die Drucker ausgelastet sind und welche Drucker Druckaufträge akzeptieren.

-u[␣benutzerliste]

(u - user) *lpstat* zeigt den Zustand aller von den angegebenen Benutzern gestellten Druckaufträge an. In *benutzerliste* können Sie folgende Elemente angeben:

benutzernr	Benutzerkennung auf irgendeinem System
sysname!*benutzernr*	Benutzerkennung auf dem System *sysname*
sysname!*all*	alle Benutzer auf dem System *sysname*
all!*benutzernr*	Benutzerkennung auf allen Systemen
all	alle Benutzer auf allen Systemen

-v[␣liste]

lpstat zeigt für die Drucker in *liste* deren Namen und Gerätebezeichnung an. Zu lokalen Druckern ist das der Pfadname der Gerätedatei, zu Netzdruckern der Name des fernen Rechners. *liste* ist eine Liste von Druckernamen.

SIEHE AUCH

enable, *lp*

SINIX SPOOL [15]

ls
Informationen über Dateiverzeichnisse und Dateien ausgeben (list contents of directory)

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.

```
ls[␣option]...[␣datei]...
```

Kein Argument angegeben
ls gibt aus: Namen aller Dateien und Dateiverzeichnisse des aktuellen Dateiverzeichnisses, alphabetisch sortiert. Nur die Dateien, deren Name mit einem Punkt beginnt, werden nicht ausgegeben, z.B. .profile.

Keine Option angegeben
datei ist ein Dateiverzeichnis
ls gibt aus: Namen aller Dateien und Dateiverzeichnisse, die darin eingetragen sind, alphabetisch sortiert.

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. Standard ist: Dateinamen, die mit einem Punkt . beginnen, werden nicht ausgegeben.

-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 ISO 646*).

-c
Statt des Zeitpunkts der letzten Änderung der Datei wird der Zeitpunkt der letzten Indexänderung verwendet.

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*:
Der Zeitpunkt der letzten Indexänderung wird als Sortierschlüssel verwendet.

-C

Mehrspaltige Ausgabe. 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 des Dateiverzeichnisses zu erhalten. Wird *datei* nicht angegeben, gibt *ls* den Punkt für das aktuelle Dateiverzeichnis aus.

-f

Jedes Argument wird wie ein Dateiverzeichnis behandelt; die Einträge in jedem Dateiverzeichnis werden in der Reihenfolge, in der sie erfaßt wurden, ausgegeben. 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. Ausführbare Dateien kennzeichnet *ls* mit dem Suffix *. Ein Klammeraffe @ bezeichnet einen symbolischen Verweis.

-g

Wirkt wie *-l*, nur wird die Benutzerkennung des Eigentümers nicht ausgegeben.

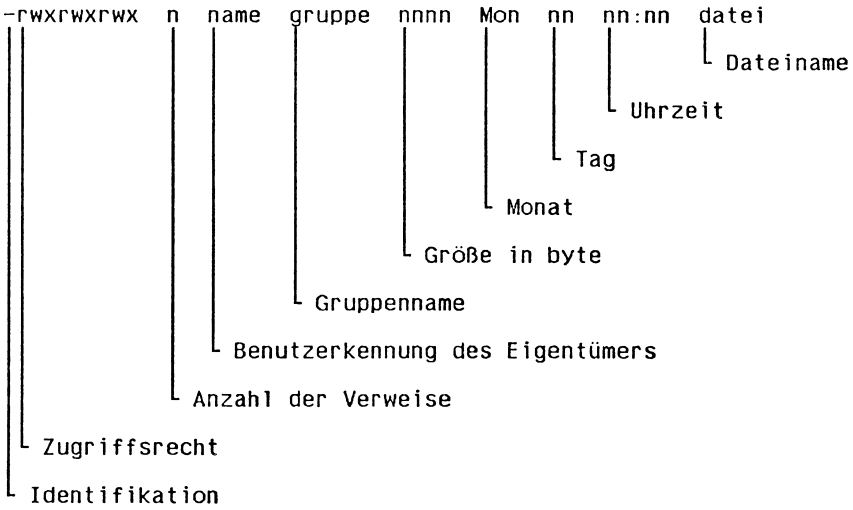
-i

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:



Identifikation

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 oder - für kein Ausführungsrecht

   Wenn für den Eigentümer oder die Gruppe das s-Bit und gleichzeitig das entsprechende x-Bit gesetzt ist, steht anstelle von *x* ein *s*. Wenn das s-Bit *ohne* das entsprechende x gesetzt ist, steht anstelle von x ein *S*.

   Ist für die Gruppe das S-Bit gesetzt (x-Bit nicht gesetzt, s-Bit gesetzt) kann statt dem S auch ein l stehen, wenn die obligatorische Dateisperre (mandatory file and record locking) eingeschaltet ist. Dieses Zugriffsrecht beschreibt die Eigenschaft einer Datei, anderen Dateien zu erlauben, seine Lese- und Schreibrechte während des Zugriffs zu sperren.

   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 mit nicht gesetztem 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) werden ausgegeben.

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 ->, gefolgt von dem Pfadnamen der referenzierten Datei, angegeben.

-L

Bei symbolischen Verweisen wird die Datei oder das Dateiverzeichnis angezeigt, auf die der symbolische Verweis referenziert, anstatt dem Verweis selbst.

-m

Ausgabe in Listenform. Die Einträge werden hintereinander, durch Kommata 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

Wirkt wie *-l*, nur wird anstelle des Namens des Eigentümers bzw. des Gruppennamens die Benutzer- bzw. die Gruppennummer ausgegeben.

-o

Wirkt wie *-l*, nur wird die Gruppe nicht ausgegeben.

-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 Dateiverzeichnisse, 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.

BEISPIELE

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
```

2. Mehrspaltige Ausgabe des aktuellen Dateiverzeichnisses auf einer Breite von 80 Zeichen pro Ausgabezeile.

```
$ ls -C
ADRESSEN    brief3      brief6      kart_c      kart_f
brief1      brief4      kart_a      kart_d      typescript
brief2      brief5      kart_b      kart_e
```

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
```

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 root       root        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/
```

5. Ausgeben aller Dateien und Dateiverzeichnisse (mit Inhalt). Die Namen der Dateiverzeichnisse werden mit einem Doppelpunkt beendet.

```
ls *

Postkasten
autoren
bvp.col

bildschirme:
aus
pause
sinix

lingua:
beispiele
car.cdr
engl.dt
fachwort
```

SIEHE AUCH

chmod, find, ln

mail
Nachrichten senden oder lesen

Mit *mail* können Sie elektronische Post senden und empfangen. Wenn Ihr Rechner an ein Netz angeschlossen ist, können Sie auch mit Benutzern an anderen Rechnern Nachrichten austauschen.

Bereits bei der Anmeldung erfährt der Benutzer, ob er Post bekommen hat. Auch während der Anwendung von *mail* wird gemeldet, wenn neue Nachrichten eintreffen.

Nachrichten sind ASCII-Texte. Sie enthalten einen Nachrichtenkopf mit Informationen, die zum Weiterleiten der Nachricht dienen und die zum Teil implementierungsabhängig sind. Auf den Nachrichtenkopf folgt eine Leerzeile und anschließend der Text der Nachricht.

Mit *mail* können Sie

- prüfen, ob Nachrichten vorliegen (Format 1, Option -e)
- Nachrichten lesen (Lesemodus, Format 1)
- Nachrichten senden (Sendemodus, Format 2)
- Post nachsenden (Nachsendemodus, Format 3)
- Sendemechanismus überprüfen (Fehlersuchemodus, Format 4 und 5).

Die Arbeitsweise ist nach den Optionen beschrieben.

```
mail[␣-ehpPqr][␣-f␣datei]                                    Format 1
mail[␣-tw][␣-m␣nachrichten_typ]␣empfänger␣...                Format 2
mail[␣-F]␣empfänger␣...                                      Format 3
mail␣-T␣testdatei␣empfänger␣...                              Format 4
mail[␣-x␣info_status][␣weitere_optionen...]␣empfänger␣...    Format 5
```

Format 1: Lesemodus

mail[␣-ehpPqr][␣-f␣datei]

Keine Option angegeben
: *mail* prüft den Standard-Briefkasten (*$MAIL*). Wenn Nachrichten vorliegen, gibt *mail* die zuletzt eingegangene Nachricht aus (last in - first out). Dann gibt *mail* ein Fragezeichen ? aus und erwartet eines der im Abschnitt *Arbeitsweise im Lesemodus* beschriebenen Kommandos. Wurde die letzte Nachricht bearbeitet, beendet sich *mail*.

-e
: *mail* prüft, ob Nachrichten vorliegen. Wenn Nachrichten vorliegen, gibt *mail* den Ende-Status 0 zurück, im anderen Fall den Ende-Status 1. Dann beendet sich *mail*.

-h
mail zeigt die Header der vorliegenden Nachrichten an, in umgekehrter Reihenfolge des Eintreffens.

-p
mail gibt alle Nachrichten ohne Anzuhalten nacheinander aus und beendet sich.

-P
mail gibt alle Nachrichten mit vollständigen Kopfzeilen aus.

.1

-q
mail beendet sich, wenn das Signal SIGINT eintrifft (Taste DEL).

-q nicht angegeben:
Das Signal SIGINT unterbricht die aktuelle Ausgabe einer Nachricht.

-r
mail gibt die Nachrichten in der Reihenfolge aus, in der sie eingetroffen sind.

-r nicht angegeben:
mail gibt die jüngste Nachricht zuerst aus.

-f␣datei
datei bezeichnet den Briefkasten, d.h. die Datei, aus der *mail* Nachrichten lesen soll.

-f␣datei nicht angegeben:
mail liest die Nachrichten aus dem Standard-Briefkasten, das ist *$MAIL*.

Format 2: Sendemodus

mail[␣-tw][␣-m␣nachrichten_typ]␣empfänger␣...

-t
mail fügt im Kopf (vor der Nachricht) eine Zeile mit dem Inhalt *From ... Datum* und eine Zeile *Content-Lentgh: nachrichtengröße* ein.

-t nicht angegeben:
mail fügt vor der Nachricht eine Zeile mit dem Inhalt *Apparently-To: empfänger, ...* ein.

-m␣nachrichten_typ
mail fügt im Nachrichtkopf eine Zeile mit dem Inhalt *Message-Type: nachrichten_typ* ein.

-w
mail schickt einen Brief an einen Empfänger im Netz, ohne auf die Beendigung des Transferprogramms zu warten.

empfänger
ist eine Benutzerkennung im lokalen System oder ein Netzpfad, falls der Rechner an ein Netz angeschlossen ist. Näheres zur Netzadressierung lesen Sie im Abschnitt *Arbeitsweise im Sendemodus.*

Format 3: Nachsendemodus

mail[␣-F]␣empfänger␣...

-F
mail leitet alle Nachrichten, die an den Eigentümer des Briefkastens geschickt werden, an den oder die *empfänger* weiter. Anfangs muß der Briefkasten leer sein. *mail* trägt dann als erste Zeile

Forward to: empfänger ...

ein und leitet alle eingehenden Nachrichten weiter. Vor dem Kopf jeder weitergeleiteten Nachricht wird die Zeile

Auto-Forwarded-From: ...

eingefügt. *mail* gibt 0 zurück, wenn die Nachsendung erfolgreich eingerichtet oder abgeschaltet wurde. Zum Abschalten des Nachsende-Mechanismus schreiben Sie

```
mail -F "".
```

empfänger
ist eine maximal 1024 Bytes große Liste, die in Anführungszeichen eingeschlossen werden sollte, um sicherzustellen, daß sie als ein einziger Operand der Option -F interpretiert wird. Die Empfänger in der Liste können durch Zwischenraumzeichen oder Kommata voneinander getrennt werden, wie z.B. in

```
mail -F "benutzer1,benutzer2@att.com,systemc!systemd!benutzer3"
```

Steht vor einem der Empfänger ein Pipe-Symbol |, wird der Rest der Zeile als Kommando interpretiert, zu dem die aktuelle Nachricht durch eine Pipe gesendet wird, z.B.

```
mail -F "|/bin/sh -c \"shell_command_line\""
```

Eine Nachricht kann gleichzeitig an verschiedene Empfänger weitergeleitet und durch eine Pipe zu einem Kommando gesandt werden, so z.B. in

```
mail -F "karl,paul,|meinurlaubprog %R".
```

Näheres hierzu finden Sie in *Arbeitsweise im Nachsendemodus.*

Format 4 und 5: Fehlersuchemodus

mail␣-T␣testdatei␣empfänger␣...
mail[␣-x␣info_status][␣weitere_optionen...]␣empfänger␣...

-T␣testdatei
In der Datei *testdatei* steht, wie *mail* die Post für die Empfänger interpretieren, behandeln und weiterleiten soll. Mit dieser Option testen Sie die Wirkungsweise von *testdatei* ohne Post wirklich zu senden oder zu empfangen. *mail* liest eine Nachricht von der Standardeingabe ein, führt *testdatei* für den angegebenen Empfänger aus und zeigt dann die Eingaben und Informationen über die Bearbeitung an. Ist *testdatei* gleich "", verwendet *mail* die Systemdatei *mailsurr*.

-x
mail erzeugt eine Datei namens */tmp/MLDBGprozeßnummer*, die Informationen darüber enthält, wie *mail* die aktuelle Nachricht bearbeitet hat. Die Informationen sind vor allem für den Systemverwalter interessant und umfassen alles, was Sie mit Option *-T* erfahren können.

info_status
Der absolute Wert von *info_status* bestimmt, wie ausführlich die Informationen sind. Ist *info_status* größer 0, werden Informationen nur dann ausgegeben, wenn *mail* bei der Abarbeitung der Nachricht Probleme hat. Ist *info_status* kleiner 0, werden alle für die Abarbeitung relevanten Daten gesichert. Ist *info_status* gleich 0, findet überhaupt keine Aufzeichnung statt. *info_status* überschreibt den Wert von *DEBUG* in */etc/mail/mailcnfg*.

ARBEITSWEISE IM LESEMODUS

mail gibt die jüngste vorliegende Nachricht aus. Wenn nicht mit der Option *-P* umdefiniert, zeigt *mail* zunächst nur die unmittelbar interessanten Kopfzeilen an. Dazu gehören die UNIX-Postmarks *From* und *>From*, die Kopfzeilen *From:*, *Date:*, *Subject:* und *Content-Length:* und Zeilen mit Angaben über den oder die Empfänger wie *To:*, *Cc:* oder *Bcc*. Danach gibt *mail* den Inhalt der Nachricht aus, falls diese keine nicht-druckbaren Zeichen enthält. Sonst erscheint lediglich eine Warnung über binären Text, und nichts weiter wird ausgegeben. Dieses Verhalten können Sie mit der Option *-p* verändern. Mit einem Fragezeichen verlangt *mail* die Eingabe eines der folgenden Kommandos. Jedes Kommando ist mit der Taste ↵ abzuschließen.

mail gibt auf die Standard-Ausgabe aus. Nachrichten, die über eine Bildschirmseite hinausgehen, können Sie nur mit CTRL S anhalten (weiter mit CTRL Q). Sie können die Ausgabe auch in eine Datei umlenken und damit weiterarbeiten oder Sie können das Kommando *mailx* verwenden, das Möglichkeiten bietet, die Ausgabe zu steuern. Nach einem Interrupt wird die nächste Nachricht nicht ausgegeben. Drücken Sie dann Taste *p*.

KOMMANDOS ZUM LESEN

↵
nächste Nachricht ausgeben.

END
mail beenden. Nicht gelöschte Nachrichten bleiben im Briefkasten.

#
Nummer der aktuellen Nachricht ausgeben

!shell-kommando
shell-kommando ausführen.

*****
Übersicht über die Kommandos von *mail* ausgeben.

+ oder **n**
nächste Nachricht ausgeben (wie ↵).

-
vorhergehende Nachricht ausgeben.

a
Nachricht ausgeben, die während der laufenden *mail*-Sitzung angekommen ist.

d oder **dp** (delete/print)
Nachricht löschen und nächste Nachricht ausgeben. Wenn *mail* mit *x* beendet wird, bleiben gelöschte Nachrichten erhalten.

d*n*
Nachricht Nummer *n* löschen und nicht die nächste Nachricht ausgeben.

dq (delete/quit)
Nachricht löschen und *mail* verlassen.

h (header)
Ein Fenster mit den Kopfzeilen der Nachrichten ausgeben, die in der Nähe der aktuellen Nachricht stehen.

h*n*
Ein Fenster mit den Kopfzeilen der Nachrichten ausgeben, die in der Nähe der Nachricht Nummer *n* stehen.

h␣a (header all)
Die Nachrichtenköpfe aller Nachrichten im Briefkasten ausgeben.

h␣d (header/delete)
Die Nachrichtenköpfe aller Nachrichten ausgeben, die zum Löschen vorgesehen sind.

m[␣*empfänger*...] (mail)
mail sendet die Nachricht an die angegebenen Empfänger und löscht sie dann.

empfänger nicht angegeben:
mail nimmt die eigene Benutzerkennung an.

nummer
Nachricht mit der Nummer *nummer* ausgeben.

p (print)
mail gibt aktuelle Nachricht nochmals aus, ohne auf (nicht druckbare) Binärzeichen Rücksicht zu nehmen.

P (print)
mail gibt die aktuelle Nachricht aus und zeigt, entgegen der Voreinstellung, alle dazugehörigen Kopfzeilen an.

q (quit)
mail beenden. Nicht gelöschte Nachrichten bleiben im Briefkasten.

r[␣benutzer...] (reply)
dem Absender und weiteren Benutzern antworten und die Nachricht löschen.

s[␣datei␣...] (save)
schreibt die Nachricht in die angegebenen Dateien und löscht die Nachricht. Die Datei wird im aktuellen Dateiverzeichnis angelegt, wenn kein absoluter Pfadname angegeben ist. Existiert sie dort, wird sie erweitert.

datei nicht angegeben:
mail nimmt *mbox* an.

w[␣datei] (write)
wie *s*, jedoch wird der Nachrichtenkopf nicht in die Datei geschrieben.

datei nicht angegeben:
mail nimmt *mbox* an.

u[*n*]
Löschen der Nachricht mit Nummer *n* wieder rückgängig machen.

n nicht angegeben:
Löschen der zuletzt gelesenen Nachricht rückgängig machen.

x (exit)
mail beenden. Der Briefkasten bleibt unverändert. Das heißt, gelöschte Nachrichten bleiben erhalten.

y[␣datei...]
Entspricht Option w.

?
Zeigt eine Übersicht aller Kommandos.

Die Funktionsweise von *mail* läßt sich beeinflussen, wenn Sie die Zugriffsrechte des Briefkastens durch *chmod* verändern. Mit den Zugriffsrechten für den Briefkasten legen Sie fest, ob er gelesen und beschrieben (0666), nur gelesen (0664) oder weder gelesen noch beschrieben (0660) werden darf. Wird die Voreinstellung (0660) verändert, dann bleibt die Datei erhalten, auch wenn sie leer ist, um die gewünschten Zugriffsrechte beibehalten zu können. Der Systemverwalter kann diesen Dateischutz mit der Option DEL_EMPTY_MAILFILE in *mailcnfg* überschreiben. Die Gruppennummer des Briefkastens muß *mail* sein, damit neue Nachrichten gesendet werden können. Zudem muß der Briefkasten von der Gruppe *mail* beschreibbar sein.

ARBEITSWEISE IM SENDEMODUS

mail liest den zu sendenden Text von der Standard-Eingabe. Den Textabschluß bildet das Dateiende (EOF, Taste END) oder ein Punkt . in der ersten Spalte einer Zeile. Der eingelesene *Brief* wird dann an die *Briefkästen* der angegebenen Empfänger angehängt. Beachten Sie dabei, daß eine Nachricht größer 100 kByte unter Umständen nicht korrekt übertragen werden kann.

Als Empfänger gilt im allgemeinen jeder Benutzername, der *login* bekannt ist. Ein *Brief* setzt sich zusammen aus einigen *Kopfzeilen*, einer Leerzeile und dem *Nachrichteninhalt*. Die Kopfzeilen wiederum enthalten eine oder mehrere UNIX-Postmarks:

FROM *absender datum_und_zeit* [remote from *ferner_rechner_name*]

und eine oder mehrere Zeilen der Form:

schlüsselwort: [*druckbarer text*].

Ein Schlüsselwort besteht aus beliebigen druckbaren Zeichen mit Ausnahme von Doppelpunkt : und Zwischenraumzeichen.
Immer vorhanden ist die Kopfzeile *Content-Length:*, die die Größe des Nachrichteninhalts in Bytes angibt. Die Kopfzeile *Content-Type:*, die den Typ des Nachrichteninhalts (wie text, binary, zusammengesetzt etc.) beschreibt, ist nur vorhanden, wenn ein Nachrichteninhalt vorhanden ist. Eine Kopfzeile kann über mehrere Zeilen gehen, wenn die Folgezeilen mit einem Zwischenraumzeichen beginnen.

Mit folgenden Kopfzeilen beeinflussen Sie den Sendevorgang:

Transport-Options:[␣*/option...*]
Default-Options:[␣*/option...*]
>To:␣*empfänger*[␣*/option...*]

Bei Optionen, die sich widersprechen, verwendet *mail* die zuerst gefundene.

/option kann sein:

/delivery
Den Absender informieren, daß die Nachricht im Briefkasten von *empfänger* erfolgreich angekommen ist.

/nodelivery
Den Absender nicht über erfolgreiches Ankommen informieren.

/ignore
Den Absender nicht über mißlungenes Ankommen informieren, aber Nachricht nicht zurücksenden.

return
Den Absender über mißlungenes Ankommen informieren und Nachricht zum Absender zurückschicken.

/report
Wie */return*, die Nachricht wird jedoch nicht zurückgeschickt.

Keine Optionen angegeben
Voreinstellung ist */nodelivery/return*.

Ein Brief, der nicht zugestellt werden kann, wird an den Absender mit aussagekräftigen Fehlermeldungen zurückgeschickt. Wenn während der Texteingabe das Signal *SIGINT* (Taste DEL) eintrifft sichert *mail* den Text in der Datei *dead.letter*. *dead.letter* wird im aktuellen Dateiverzeichnis oder, falls dies nicht möglich, im Login-Verzeichnis des Benutzers angelegt. Kann sie auch dort nicht angelegt werden, dann unterbleibt das Sichern des Textes. Die Datei *dead.login* wird prinzipiell nicht überschrieben, sondern jedesmal erweitert.

Sie können auch Netzadressen als Empfänger angeben. Die Art der Adressierung hängt dabei von den im lokalen System verfügbaren Transportmechanismen ab. Am häufigsten eingesetzt werden die beiden folgende Methoden: Bei der an UUCP angelehnten Adressierung geben Sie den Namen des fernen Rechners gefolgt von einem Ausrufezeichen und dem Namen des Empfängers an (z.B. sysa!user). Mit einer ganzen Folge von Systemnamen können Sie einen Brief durch ein weitläufiges Netz dirigieren (z.B. sysa!sysb!sysc!user). Bei der Domänen-orientierten Adressierung werden an den Namen des Empfängers ein '@' und Angaben über Domäne bzw. Teildomäne angehängt (z.B. user@sf.att.com). Informieren Sie sich beim lokalen Systemverwalter über die im lokalen System üblichen Adressierungsmethoden.

ARBEITSWEISE IM NACHSENDEMODUS

Mit der Option *-F* können Sie ankommende Post anderen Empfängern nachsenden.

Befindet sich in der Empfängerliste ein Pipe-Zeichen mit nachfolgendem Kommando, wird die Nachricht durch eine Pipe gesendet. Dieses Kommando wird in der Umgebung des Nachrichtenempfängers (dem Basisnamen des Briefkastens) ausgeführt. Heißt der Briefkasten beispielsweise *var/mail/foo*, wird *foo* in */etc/passwd* gesucht, um dazugehörige Gruppennummer, Benutzernummer und das Home-Dateiverzeichnis zu ermitteln. Die Umgebung für das Kommando enthält lediglich die Variablen HOME, LOGNAME, TZ, PATH (=*/usr/usr/bin:*) und SHELL (=*/usr/bin/sh*). Das Kommando wird dann im Home-Dateiverzeichnis des Empfängers gestartet. Ist der Empfänger in /etc/passwd nicht vorhanden, wird das Kommando nicht ausgeführt und der Absender der Nachricht erhält eine entsprechende Fehlermeldung.

Nach dem Pipe-Zeichen müssen Optionen und Parameter, die Zwischenraumzeichen enthalten und die zum Kommando gehören, von entwerteten Anführungszeichen \" eingeschlossen sein. Sonderzeichen der Shell sollten Sie nur verwenden, wenn das Kommando die Shell ist.

Innerhalb des Kommandos sind folgende Attribute möglich, die ersetzt werden, bevor das Kommando ausgeführt wird:

%R
Pfad zurück zum Absender der Nachricht.

%c
Wert der Kopfzeile *Content-Type:*, falls vorhanden.

%S
Wert der Kopfzeile *Subject:*, falls vorhanden.

Gibt das Kommando einen Wert ungleich 0 zurück, nimmt *mail* an, daß die Nachricht nicht empfangen werden konnte und erzeugt eine entsprechende Fehlermeldung an den Absender.

Die Nachsendung von Nachrichten kann zu Endlosschleifen führen, wenn beispielsweise *benutzera* an *benutzerb* nachsendet, dieser wieder an *benutzera* usw.. Lokale Schleifen werden sofort erkannt. Schleifen im Netz werden erst erkannt, wenn eine eingebaute Schranke von 20 überschritten ist. In beiden Fällen gilt die Nachricht als nicht empfangen.

Aus Sicherheitsgründen wird bei Aktivierung der Nachsendung automatisch *chmod s+g* an *briefkasten* durchgeführt, und dementsprechend beim Abschalten *chmod s-g* beim Abschalten. Ist das setGID-Bit nicht gesetzt, wenn *mail* eine Nachricht an ein Kommando weiterleiten will, kann diese nicht empfangen werden, und eine entsprechende Fehlermeldung wird an den Absender übermittelt.

Um einen korrekten Ablauf sicherzustellen, sollte der Briefkasten auf read-write gesetzt sein und die Gruppennummer *mail* haben.

Für zusätzliche Funktionalität können Sie die beiden UNIX-Kommandos *notify(1)* und *vacation(1)* einsetzen. Mit *notify(1)* können neu eintreffende Nachrichten asynchron gemeldet werden. Mit *vacation(1)* lassen sich Nachrichten automatisch beantworten, wenn der Empfänger für einen längeren Zeitraum nicht erreichbar ist.

ARBEITSWEISE IM FEHLERSUCHEMODUS

Um die Wirkungsweise einer Testdatei zu prüfen, geben Sie "*mail -T testdatei empfänger*" ein und danach irgendeine einfache Nachricht, die Sie mit CTRL d abschließen. *mail* zeigt sowohl die Eingaben an, als auch wie sich die Eingabe verändert, wenn die Testdatei für den angegebenen Empfänger abgearbeitet wird.

ENDE-STATUS

Beim Senden bzw. Nachsenden ist der Ende-Status immer gleich 0. Beim Empfangen gilt:

0 Nachrichten liegen vor.
1 Nachrichten liegen nicht vor.

FEHLERMELDUNGEN

mail gibt Fehlermeldungen aus. Diese sind in der Regel selbsterklärend.

DATEIEN

$MAIL
Der Name des Standard-Briefkastens steht in der Variablen *$MAIL*. Darin sucht *mail* ankommende Nachrichten.

dead.letter
In dieser Datei sichert *mail* Nachrichten, die z.B. wegen eines Fehlers nicht abgeschickt werden können oder für die die Texteingabe mit DEL abgebrochen wurde. Die Datei wird im aktuellen Dateiverzeichnis bzw. im Login-Verzeichnis des Benutzers angelegt und prinzipiell nicht überschrieben, sondern erweitert.

$HOME/mbox
Standard-Dateiname, wenn Sie Nachrichten mit *s* oder *w* sichern.

/etc/passwd
Zur Identifikation des Absenders und Lokalisierung des Empfängers.

/etc/mail/mailsurr
System-Testdatei zur Fehlersuche.

/etc/mail/mailcnfg
Informationsdatei zur Initialisierung.

*/tmp/ma**
Temporäre Datei.

*/tmp/MLDBG**
Datei für Informationen im Fehlersuchemodus.

/var/mail/.lock*
Sperre für Post-Verzeichnis. Unter bestimmten Bedingungen kann es gelegentlich vorkommen, daß eine Sperre nicht aufgehoben werden kann.

/var/mail/:saved
Verzeichnis mit temporären Dateien, in denen Daten für den Fall eines Systemzusammenbruchs gesichert werden.

/var/mail/user
Eingehende Post für *user*, also der *Briefkasten*.

BEISPIELE

1. Nachricht senden:

   Der Inhalt der Datei *winni.brf* soll an den Benutzer *winni* geschickt werden.

```
$ mail winni < winni.brf
```

2. Nachrichten lesen:

```
$ mail
From mecky Fri Apr  7 08:26:26 1989
Date: Fri, 7 Apr 89 08:26:23  +0100
From: mecky
To: hans

Deine letzte Nachricht habe ich erhalten. Bin aber gerade
aus dem Urlaub zurueck und kann nichts damit anfangen.

Bring mich mal auf den neuesten Stand.

Gruss Mecky

? ↵
From werner Thu Apr  6 13:31:24 1989
Date: Thu, 6 Apr 89 13:31:22  +0100
From: werner
To: hans

Hallo,
Danke fuer die Korrekturen.

Werner

? q
```

SIEHE AUCH

chmod, *login*, *mailx*, *notify*, *sh*, *write*, *vacation*
mail_pipe, *mailsurr*, *mailcnfg* [5]

Leitfaden für Benutzer [2]

mailalias
Umsetzen von Aliasnamen für Mailadressen

Da Adressen für die elektronische Postverwaltung sehr kompliziert aufgebaut sein können, gibt es die Möglichkeit, vereinfachte Aliasnamen für komplizierte Adressen einzurichten.

mailalias hilft Ihnen, zu den vereinfachten Aliasnamen wieder die ursprünglichen Postadressen herauszufinden.

mailalias wird von *mail* aufgerufen.

```
mailalias [option] ... name ...
```

Keine Option angegeben
: *mailalias* gibt eine Liste von Postadressen aus, die dem angegebenen Aliasnamen *name* zugeordnet sind. Die Ausgabe erfolgt auf die Standardausgabe.

option

-s
: (s - silent) Wenn mehrere Adressen gesucht werden, stellt *mailalias* jeder Ausgabezeile den Suchnamen *name* voran. Mit der Option *-s* unterbleibt diese Voranstellung.

-v
: (v - verbose) *mailalias* gibt zusätzliche Informationen auf die Standardausgabe aus, über die der Benutzer den Ablauf des Programms verfolgen kann.

Arbeitsweise

mailalias führt die Suche nach den Postadressen in folgenden Schritten durch:

- Die Datei /var/mail/*name* wird gesucht. Ist sie vorhanden, wird *name* ausgegeben und *mailalias* beendet sich.
- Die lokale Aliasdatei *$HOME/lib/names* wird nach passenden Postadressen durchsucht. Wird eine Zeile gefunden, die mit *name* beginnt, so wird der Rest der Zeile ausgegeben, und *mailalias* beendet sich.
- Die Aliasdateien des Systems, die in der Datei */etc/mail/namefiles* aufgelistet sind, werden nach übereinstimmenden Postadressen durchsucht. Wird eine Zeile gefunden, die mit *name* beginnt, so wird der Rest der Zeile ausgegeben, und *mailalias* beendet sich.

Standardmäßig enthält die Systemdatei */etc/mail/namefiles* den Pfad */etc/mail/lists* und die Datei */etc/mail/names*.

Falls der angegebene Pfad ein Dateiverzeichnis *dvz* beschreibt, so sucht *mailalias* nach einer Datei namens *dvz/name*.

- Falls in den bisherigen Schritten keine Postadressen gefunden wurden, so gibt *mailalias* den Aliasnamen *name* aus und beendet sich.

Format der Aliasdateien

Die Aliasdateien sind aus Zeilen mit folgendem Format aufgebaut:

```
aliasname     liste-von-adressen
```

Die einzelnen Adressen werden durch Leerzeichen oder Tabulatoren voneinander getrennt. Fortsetzungszeilen werden mit einem Gegenschrägstrich \ am Ende der Zeile gekennzeichnet. Kommentarzeilen beginnen mit einem Nummernzeichen #.

DATEIEN

$HOME/lib/names
Datei mit privaten Aliasnamen.

/etc/mail/namefiles
Liste von Dateien, die durchsucht werden sollen.

/etc/mail/names
Datei, die prinzipiell durchsucht wird.

SIEHE AUCH

uucp, *mail*
smtp, *smtpd*, *smtpqer*, *smtpsched*, *tosmtp* [5]

Format 1: Lesemodus

mailx[␣option]...

Keine Option angegeben
mailx verhält sich wie im Abschnitt *Arbeitsweise von mailx* beschrieben.

option

Optionen sind einzeln, durch Leerzeichen getrennt, anzugeben.

-e
Nur prüfen, ob Nachrichten vorliegen. Wenn Nachrichten vorliegen, gibt *mailx* den Ende-Status 0 zurück, sonst 1. Dann beendet sich *mailx*.
mailx durchläuft keine Startdatei (siehe *Kommandodateien*).

-f[␣datei]
gibt den Briefkasten an, aus dem *mailx* Nachrichten lesen soll.

datei
bezeichnet den Briefkasten, d.h. die Datei, aus der *mailx* Nachrichten lesen soll.

datei nicht angegeben:
mailx liest die Nachrichten aus dem benutzereigenen Briefkasten, das ist *$HOME/mbox*.

-f datei nicht angegeben:
mailx liest die Nachrichten aus dem Standard-Briefkasten, das ist */var/mail/$USER*.

-H
mailx gibt nur Übersichtszeilen aus und beendet sich. Der Ende-Status ist 0 wie bei Option *-e*, wenn Nachrichten vorliegen, sonst 1. Der Aufbau der Übersichtszeilen ist weiter unten beschrieben (siehe *mailx*-Kommando *headers*).

-l
Kopfzeilen mit Nachrichtengruppen- und Artikelnummer werden mit ausgegeben. Die Option *-f* muß gesetzt sein.

-n
Die globale Startdatei */etc/mail/mailx.rc* soll nicht durchlaufen werden (siehe *Kommandodateien*).

-N
Ausgabe der Meldungszeile und der Übersichtszeilen nach dem *mailx*-Aufruf unterdrücken.

-T␣datei
Nachdem die Nachricht gelesen wurde, werden Nachrichten- und Artikelnummer in *datei* aufgezeichnet. Diese Option setzt die Option *-l*.

mailx
Nachrichten interaktiv bearbeiten (mail extended)

Mit *mailx* können Sie elektronische Post auf komfortable Weise senden und empfangen. Empfangene Post können Sie ablegen, löschen und beantworten, zu sendende Post editieren, überprüfen etc.. Wenn Ihr Rechner an ein Netz angeschlossen ist, können Sie auch mit Benutzern an anderen Rechnern Nachrichten austauschen.

mailx ist eine Erweiterung des Kommandos *mail*. Nachrichten enthalten einen Nachrichtenkopf mit Informationen, die zum Weiterleiten der Nachricht dienen und die zum Teil implementierungsabhängig sind. Auf den Nachrichtenkopf folgt eine Leerzeile und anschließend der Text der Nachricht.

Mit *mailx* können Sie

- prüfen, ob Nachrichten vorliegen (Format 1, Option *-e*)
- Nachrichten lesen (Lesemodus, Format 1)

  mailx-Kommandos im Lesemodus unterstützen das

  - Weiterverarbeiten von Nachrichten mit SINIX-Kommandos
- Nachrichten senden (Sendemodus, Format 2).

  mailx-Kommandos im Sendemodus (Tilde-Kommandos) ermöglichen das

  - Zusammenstellen von Nachrichten aus vorhandenen Nachrichten, Dateien und Werten von Umgebungsvariablen,
  - Bearbeiten von Nachrichten mit einem Editor während des *mailx*-Dialogs.

Im Unterschied zu *mail* sammelt *mailx* gelesene Nachrichten automatisch in einem benutzereigenen Briefkasten (standardmäßig *$HOME/mbox*).
Die Arbeitsweise und die Kommandos sind nach den Optionen beschrieben.

```
mailx[ option]...                                          Format 1
mailx[ option]... empfänger...                             Format 2
```

-u␣benutzerkennung
mailx liest die Nachrichten aus dem Standard-Briefkasten des angegebenen Benutzers, vorausgesetzt, Sie haben Leserecht.

-V
Die Versionsnummer von *mailx* wird ausgegeben und *mailx* beendet.

Format 2: Sendemodus

mailx[␣option]...␣empfänger␣...

Keine Option angegeben
mailx verhält sich wie im Abschnitt *Arbeitsweise von mailx* beschrieben.

option

Optionen sind einzeln, durch Leerzeichen getrennt, anzugeben.

-d
Die Fehlersuch-Ausgabe wird eingeschaltet.

-F
Alle gesendeten Nachrichten werden in einer Datei protokolliert. 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 Dateinamen in der *mailx*-Variablen *record*. Wenn diese nicht gesetzt ist, wird nichts protokolliert.

-h␣n
Die Nachricht soll höchstens *n*-mal von einer Station im Netz zu einer anderen übertragen werden. Damit können bei der Zustellung von Nachrichten Endlosschleifen im Netz vermieden werden.

-i
mailx ignoriert das Signal SIGINT (siehe *mailx-Variable ignore*).

-n
Die globale Startdatei */etc/mail/mailx.rc* soll nicht durchlaufen werden (siehe *Kommandodateien*).

-r␣adresse
Die angegebene Adresse wird an die Netz-Software weitergereicht. Alle Tilde-Kommandos werden ignoriert.

-s␣subject
subject ist eine Zeichenkette, die *mailx* in das Feld *Subject:* im Nachrichtenkopf einträgt. Damit können Sie der Nachricht einen Titel geben.

-U
Konvertiert UUCP-Adressen zu Internet-Standard. Überschreibt die Variable *conv*. Beachten Sie aber, daß der Internet-Standard noch nicht vollständig unterstützt wird.

empfänger
- Benutzerkennung im lokalen System oder ein Netz-Pfad, falls der Rechner an ein Netz angeschlossen ist.
- Alias-Namensgruppe (siehe *mailx*-Kommando *alias*).
- Shell-Kommando mit Pipe-Symbol davor.

Kann die Nachricht nicht angenommen werden, wird sie, wenn möglich, wieder im Briefkasten des Absenders abgelegt.
Beginnt *empfänger* mit einem Pipe-Symbol (|), wird der Rest der Zeile als Shell-Kommando interpretiert, zu dem die Nachricht durch eine Pipe gesendet wird. Dies ermöglicht eine Schnittstelle zu allen Programmen, die von der Standardeingabe lesen. Sie können so z.B. mit dem Kommando *lp(1)* abgeschickte Nachrichten ausdrucken.

Arbeitsweise von mailx

Im Lesemodus prüft *mailx* den Inhalt des Briefkastens. Dies ist standardmäßig die Datei */var/mail/$USER* (Standard-Briefkasten). Im Standard-Briefkasten treffen die ankommenden Nachrichten ein.

Dann arbeitet *mailx* die Startdateien ab, in denen Sie z.B. *mailx*-Variablen initialisieren können (siehe *Kommandodateien*).

Das Verhalten von *mailx* wird durch Umgebungsvariablen und *mailx*-Variablen beeinflußt (siehe *mailx-Variablen* und *UMGEBUNGSVARIABLEN*).

mailx meldet sich je nach Modus.

Format1: Lesemodus

Liegen keine Nachrichten vor, meldet *mailx*:

```
No mail for benutzerkennung
```

Liegen Nachrichten vor, meldet sich *mailx* mit einer Meldungszeile und einer Übersicht über die im Briefkasten vorhandenen Nachrichten.

Pro Nachricht gibt *mailx* eine Übersichtszeile aus. Der Aufbau der Übersichtszeilen ist beim *mailx*-Kommando *headers* beschrieben.

Das Größerzeichen > markiert die aktuelle Nachricht. Auf die aktuelle Nachricht beziehen sich *mailx*-Kommandos, wenn man keine Nachrichtenliste angibt.
Das Fragezeichen fordert zur Eingabe eines *mailx*-Kommandos auf.

Format2: Sendemodus

mailx gibt 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. Die Aufforderung entfällt, wenn der Titel in der Kommandozeile eingegeben wurde (Option *-s*). Ist der eingegebene Text zu lang, wird die Fehlermeldung *mail: ERROR signal 10* ausgegeben und die Nachricht wird nicht abgeschickt.

Anschließend befindet sich *mailx* im Sendemodus und Sie können den Nachrichtentext eingeben. Während der Texteingabe sind die *mailx*-Kommandos erlaubt, die mit dem Zeichen Tilde ~ beginnen (Tilde-Kommandos sind *mailx*-Kommandos im Sendemodus). Sie müssen ab Spalte 1 eingegeben werden.
Den Eingabetext speichert *mailx* in einer temporären Datei im Dateiverzeichnis */usr/tmp*.

Das Kommando ~ . oder die Taste END beendet die Texteingabe.

mailx-Kommandos, Eingabeformat

mailx-Kommandos haben das folgende Format:

```
[kommando][␣nachrichtenliste][␣argument] ...
```

kommando
: ist der Name des *mailx*-Kommandos. Diese sind im folgenden Abschnitt alphabetisch beschrieben.
Alle Kommandonamen können Sie in abgekürzter Form angeben. Die Kurzform ist durch Fettdruck hervorgehoben.

kommando nicht angegeben (nur ↵):
mailx führt das Kommando *print* aus.

nachrichtenliste
: Angabe einer oder mehrerer Nachrichten, die das Kommando bearbeiten soll. Mehrere Angaben trennt man durch Leerzeichen.
Nachrichten kann man dabei wie folgt spezifizieren:

n
: *n* gibt die Nachrichtennummer an
(siehe *mailx*-Kommando *headers*).

.
: die aktuelle Nachricht (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.

:c

c gibt den Typ der Nachricht an. Folgende Angaben sind möglich:

d gelöschte Nachrichten (deleted)

n neue Nachrichten (new)

o alte Nachrichten (old)

r gelesene Nachrichten (read)

u noch nicht gelesene Nachrichten (unread)

Beachten Sie, daß es vom jeweiligen *mailx*-Kommando abhängt, ob eine Nachrichtenauswahl sinnvoll ist oder nicht.

nachrichtenliste nicht angegeben:
mailx nimmt die aktuelle Nachricht an.

argument

Die Angabe von Argumenten ist beim jeweiligen Kommando beschrieben.
Wenn Dateinamen gefordert werden, kann man die üblichen Sonderzeichen der Shell verwenden. Zeichenketten, die Leerzeichen enthalten, sind in Anführungszeichen "..." einzuschließen, wenn sie als ein Argument interpretiert werden sollen.

mailx-Kommandos im Lesemodus

Zum Eingabeformat der folgenden Kommandos siehe vorhergehender Abschnitt. Für einige Kommandonamen gibt es Synonyme. Die synonyme Angabe ist beim alphabetisch ersten Kommando genannt. Beim alphabetisch zweiten Kommando wird nur auf das erste verwiesen.

Die meisten dieser *mailx*-Kommandos kann man sowohl im Dialog als auch in Kommandodateien benutzen (siehe *Kommandodateien*). Auf Ausnahmen davon ist beim jeweiligen Kommando verwiesen.

!shell-kommando
führt *shell-kommando* aus. Standardmäßig wird */usr/bin/sh* aufgerufen und der angegebene Kommandoaufruf übergeben.
Einen
anderen Kommandointerpreter kann man mit der *mailx*-Variablen *SHELL* angeben (siehe *mailx-Variablen*).
Wenn die *mailx*-Variable *bang* gesetzt ist, speichert *mailx* das zuletzt ausgeführte Kommando. Man kann es mit *!!* wiederholen.

Das Kommando darf nicht in einer Kommandodatei stehen.

#kommentar
Leeres Kommando. Damit kann man Kommentare in Kommandodateien (z.B. *.mailrc*) einfügen.

=
Nummer der aktuellen Nachricht ausgeben.

?
Übersicht über *mailx*-Kommandos ausgeben.

alias␣alias-name␣empfänger␣...
group␣alias-name␣empfänger␣...
definiert einen Alias-Namen für die angegebenen Empfänger.
mailx
setzt die definierten Empfänger ein, wenn man den Alias-Namen als Empfänger angibt.
Das Kommando ist nützlich in Kommandodateien.

alternates[␣name]...
definiert alternative Namen für die eigene Benutzerkennung. Beantworten Sie eine Nachricht mit *R* oder *r*, streicht *mailx* die alternativen Namen aus der Liste der Empfänger (siehe *mailx-Variablen, allnet*).

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 behandelt, das heißt sie werden beim Schließen des Standard-Briefkastens in den benutzereigenen Briefkasten geschrieben. Sie werden nicht als gesichert gekennzeichnet.

Ansonsten funktioniert *copy* wie *save*.

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 (siehe *DATEIEN*).

Die Nachrichten werden als gelesen behandelt, das heißt sie werden beim Schließen des Standard-Briefkastens in den benutzereigenen Briefkasten geschrieben. Sie werden nicht als gesichert gekennzeichnet.

Ansonsten funktioniert *Copy* wie *Save*.

Das Kommando darf nicht in einer Kommandodatei stehen.

delete[␣nachrichtenliste]
Nachricht aus dem Briefkasten löschen.
Wenn die *mailx*-Variable *autoprint* gesetzt ist, wird die nächste Nachricht nach der gelöschten ausgegeben (siehe *mailx-Variablen*).

Eine gelöschte Nachricht kann man innerhalb des *mailx*-Laufes mit *undelete* zurückholen.

discard[␣feld]...
ignore[␣feld]...
unterdrückt die angegebenen Felder des Nachrichtenkopfes bei der Ausgabe, z.B.: *Cc, Date, From, Status, Subject, To*

discard wirkt auf die *mailx*-Kommandos *next*, *pipe* (bzw. |), *print* und *type*. Es lassen sich beliebige Zeichenketten definieren (Groß- und Kleinschreibung wird nicht unterschieden). Sie werden entfernt, wenn sie im Nachrichtenkopf am Beginn einer Zeile stehen und mit Doppelpunkt enden.

Beim Sichern einer Nachricht werden die Felder jedoch ausgegeben, ebenso bei den Kommandos *Print* und *Type*.

Die Wirkung von *discard* läßt sich mit *undiscard* bzw. *unignore* rückgängig machen.

feld nicht angegeben:
discard gibt eine Liste der unterdrückten Felder aus.

dp[␣nachrichtenliste]
dt[␣nachrichtenliste]
Nachrichten aus dem Briefkasten löschen und die nächste Nachricht nach der zuletzt gelöschten ausgeben.

Eine gelöschte Nachricht kann man innerhalb des *mailx*-Laufes mit *undelete* zurückholen.

echo␣zeichenkette␣...
gibt *zeichenkette* auf die Standard-Ausgabe aus (wie das SINIX-Kommando *echo*). Mit *$name* kann man auf den Wert der Umgebungsvariablen *name* zugreifen. *echo* gibt immer deren Wert aus, auch wenn eine *mailx*-Variable gleichen Namens definiert ist.

Das Kommando ist nützlich in Kommandodateien.

edit[␣nachrichtenliste]
ruft den mit der *mailx*-Variablen *EDITOR* eingestellten Editor auf (standardmäßig *ed*) und lädt die angegebenen Nachrichten (siehe *mailx-Variablen*).

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, siehe *DATEIEN*).

Das Kommando 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 *mailx*-Kommando *quit*.

file[␣datei]
folder[␣datei]
Schließt den aktuell bearbeiteten Briefkasten (wie bei *quit*) und öffnet die angegebene Datei als Briefkasten. Dabei zeigt *mailx* die entsprechenden Übersichtszeilen an.

datei
datei gibt den zu bearbeitenden Briefkasten an. Für *datei* akzeptiert *mailx* folgene Sonderzeichen:

% der aktuelle Standard-Briefkasten (*/var/mail/$USER*)

%benutzerkennung
der Standard-Briefkasten des angegebenen Benutzers (*/var/mail/benutzerkennung*)

die vorher bearbeitete Datei

& der aktuelle benutzereigene Briefkasten (*$HOME/mbox* bzw. durch die *mailx*-Variable *$MBOX* festgelegt, siehe *mailx-Variablen*)

datei nicht angegeben:
mailx bleibt im aktuellen Standard-Briefkasten und meldet die Anzahl der darin enthaltenen Nachrichten.

Wenn man mit dem eigenen Standard-Briefkasten arbeitet, wird durch die Angabe *fi %* der Briefkasten geschlossen und wieder geöffnet. Dadurch kann man inzwischen neu eingetroffene Nachrichten erhalten.

folders
gibt mit *ls* 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.

Den Inhalt des Subject-Feldes entnimmt *mailx* dem Subject-Feld der Nachricht und stellt die Zeichenfolge *Re:* voran.

Wenn Sie die Texteingabe beendet haben, sendet *mailx* die Nachricht ab.

Im Unterschied zu *reply* protokolliert *followup* die Nachricht in eine Datei im aktuellen Dateiverzeichnis, deren Name gleich dem Namen des Empfängers gesetzt wird (Netz-Pfade werden abgetrennt). Den Namen holt *mailx* aus der ersten Zeile des Nachrichtenkopfs (From...).

Wenn die Variable *outfolder* nicht gesetzt ist, legt *mailx* die Datei im aktuellen Dateiverzeichnis an. Wenn die Variable *outfolder* gesetzt ist, legt *mailx* die Datei in dem Dateiverzeichnis ab, das in der Variablen *folder* festgelegt ist.

Die Datei wird erweitert, falls sie existiert (siehe *DATEIEN*). Die *mailx*-Variable *record* wird nicht ausgewertet.

Das Kommando darf nicht in einer Kommandodatei stehen.

Followup[␣nachrichtenliste]
beantwortet die erste in der Nachrichtenliste angegebene Nachricht wie das *mailx*-Kommando *Reply*.
Das ist die Nachricht mit der kleinsten Nummer, die sich aus der Nachrichtenliste ergibt.

Die Antwort geht an alle Absender aus *nachrichtenliste*. Die Absender entnimmt *Followup* jeweils der ersten Zeile des Nachrichtenkopfs (From:...).

mailx geht in den Sendemodus. Den Inhalt des Subject-Feldes entnimmt *mailx* dem Subject-Feld der ersten Nachricht und stellt die Zeichenfolge *Re:* voran.

Wenn Sie die Texteingabe beenden, sendet *mailx* die Nachricht ab.

Im Unterschied zu *Reply* protokolliert *Followup* die Nachricht in eine Datei im aktuellen Dateiverzeichnis, deren Name gleich dem Namen des Absenders der ersten Nachricht gesetzt wird (Netz-Pfade werden abgetrennt). Den Namen holt *mailx* aus der ersten Zeile des Nachrichtenkopfs (From...).

Wenn die Variable *outfolder* nicht gesetzt ist, legt *mailx* die Datei im aktuellen Dateiverzeichnis an. Wenn die Variable *outfolder* gesetzt ist, legt *mailx* die Datei in dem Dateiverzeichnis ab, das in der Variablen *folder* festgelegt ist.

Die Datei wird erweitert, falls sie existiert (siehe *DATEIEN*). Die *mailx*-Variable *record* wird nicht ausgewertet.

mailx schreibt die Nachricht standardmäßig in eine Datei im aktuellen Dateiverzeichnis, deren Name gleich dem Absender der ersten Nachricht gesetzt wird (Netz-Pfade werden abgetrennt).
Die Datei wird erweitert, falls sie existiert (siehe *DATEIEN*).

Das Kommando darf nicht in einer Kommandodatei stehen.

from [␣nachrichtenliste]
gibt die Übersichtszeile jeder angegebenen Nachricht auf die Standard-Ausgabe aus (siehe *mailx*-Kommando *headers*).

group␣alias-name empfänger...
siehe *alias*

headers[␣nachricht]
gibt diejenige Bildschirmseite mit Übersichtszeilen aus, die *nachricht* enthält.
Die Anzahl von Zeilen einer Seite entnimmt *mailx* der *mailx*-Variablen *screen*. Ist *screen* nicht gesetzt, gibt *mailx* 20 Zeilen aus.

Aufbau der Übersichtszeilen:

Eine Übersichtszeile besteht aus 9 Feldern.

```
mailx version 2.14 Type ? for help.
"/usr/mail/hans": 4 messages 1 new 4 unread
 U   1 guenter            Mon Nov  21 13:05     10/164  Test 1
 U   2 hans               Mon Nov  21 13:15     10/228  Freigabe
 U   3 gast               Mon Nov  21 14:00      9/176  Tools
>N   4 gast               Mon Nov  21 15:03      8/149  Tools
?
```

Feld 1
gibt den Bearbeitungsstatus der Nachricht an (diese Angaben können von der Implementierung abhängen).

O
Nachricht wurde gelesen. Diese Nachrichten werden in *$HOME/ mbox* gesichert (bei Beendigung von *mailx* oder bei Verlassen des Standard-Briefkastens).

U ungelesene Nachricht. Diese Nachrichten werden im aktuellen Briefkasten gehalten, wenn *mailx* mit quit verlassen wird.

N neue Nachricht, das heißt, die Nachricht ist seit dem letzten *mailx*-Aufruf oder seit dem letzten Briefkastenwechsel eingegangen.

M Die Nachricht wurde mit *mbox* gesichert. Sie wird beim Schließen des Standard-Briefkastens aus diesem gelöscht.

H Die Nachricht wurde mit *preserve* (*hold*) gesichert. Sie verbleibt im Standard-Briefkasten, wenn dieser geschlossen wird.

S Die Nachricht wurde mit *save* gesichert. Sie wird beim Schließen des Standard-Briefkastens aus diesem gelöscht.

> kennzeichnet die aktuelle Nachricht

Feld 2
Nachrichtennummer. Die Nachrichten numeriert *mailx* bei jedem Aufruf neu durch. Die älteste Nachricht erhält die Nummer 1.

Feld 3
Absender der Nachricht, wie in der ersten Zeile des Nachrichtenkopfs bei FROM angegeben.

Feld 4-7
Eingangsdatum und -Zeit

Feld 8
Größe der Nachricht in Zeilen/Zeichen

Feld 9
Titel (Subject). Es werden die ersten 25 Zeichen des Subject-Eintrages abgebildet.

help
gibt eine Übersicht der wichtigsten *mailx*-Kommandos auf die Standard-Ausgabe aus.

hold[␣nachrichtenliste]
preserve[␣nachrichtenliste]
hält die angegebenen Nachrichten im Briefkasten.
Die Nachrichten sind in der Übersichtszeile mit *P* gekennzeichnet und bleiben im Briefkasten, auch wenn sie gelesen oder gesichert wurden.

Mit *touch* läßt sich die Wirkung von *hold* aufheben und umgekehrt.

Das Kommando darf nicht in einer Kommandodatei stehen.

if modus
kommando-liste1
else
kommando-liste2
endif
Die Konstruktion bedingt abhängig vom Modus (senden oder lesen), welche Kommandoliste ausgeführt wird.

modus
s
kommando-liste1 wird ausgeführt, wenn *mailx* im Sendemodus (Format 2) aufgerufen wurde, sonst wird *kommando-liste2* ausgeführt.

r
kommando-liste1 wird ausgeführt, wenn *mailx* im Lesemodus (Format 1) aufgerufen wurde, sonst wird *kommando-liste2* ausgeführt.

In der Kommandoliste dürfen alle *mailx*-Kommandos stehen, die in einer Kommandodatei erlaubt sind (siehe *Kommandodateien*). *if*, *else*, *endif* und jedes Kommando müssen in je einer Zeile stehen.

Das Kommando ist in Kommandodateien nützlich.

ignore[␣feld]...
siehe *discard*

list
gibt die Namen aller *mailx*-Kommandos auf die Standard-Ausgabe aus.

mail ␣empfänger...
mailx geht in den Sendemodus und nimmt die angegebenen Namen als Empfänger.

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 (siehe *DATEIEN*).

Das Kommando darf nicht in einer Kommandodatei stehen.

Mail␣empfänger...
wie *mail*. *Mail* verschickt eine Nachricht an *empfänger* und schreibt eine Kopie in eine Datei gleichen Namens.

mbox[␣nachrichtenliste]
Die angegebenen Nachrichten werden beim Schließen des aktuell bearbeiteten Briefkastens in den benutzereigenen Briefkasten geschrieben und anschließend gelöscht, 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 (siehe *mailx-Variablen*).

next[␣nachrichtenliste]
gibt die nächstfolgende Nachricht aus, die auf die Nachrichtenliste paßt.
Das ist z.B. sinnvoll, wenn man die nächste Nachricht eines bestimmten Absenders ausgeben will (z.B. *n benutzerkennung*).

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 als gelesen 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]
siehe *hold*

print[␣nachrichtenliste]
type[␣nachrichtenliste]
gibt die angegebenen Nachrichten auf die Standard-Ausgabe aus.
Die Nachrichten werden als gelesen gekennzeichnet. Sie werden beim Schließen des Briefkastens in den benutzereigenen Briefkasten geschrieben und anschließend gelöscht. Der benutzereigene Briefkasten ist *$HOME/mbox* bzw. die in der *mailx*-Variablen *MBOX* festgelegte Datei (siehe *mailx-Variablen*).

Wenn die *mailx*-Variable *crt* gesetzt ist, übergibt *mailx* Ausgaben, die mehr Zeilen haben als dort festgelegt, an das SINIX-Kommando *pg*. Mit der *mailx*-Variablen *PAGER* kann man ein anderes SINIX-Kommando als *pg* festlegen.

Print
Type
wirkt wie *print*, gibt aber den ganzen Nachrichtenkopf aus. Das heißt, *Print* unterdrückt die Wirkung von *discard*.

quit
beendet *mailx*. Der aktuell bearbeitete Briefkasten wird geschlossen.

Wurde der Standard-Briefkasten bearbeitet, gilt folgendes:

- Gelesene Nachrichten und mit *M* gekennzeichnete Nachrichten werden in den benutzereigenen Briefkasten geschrieben und anschließend gelöscht. Der benutzereigene Briefkasten ist *$HOME/mbox* bzw. die in der *mailx*-Variablen *MBOX* festgelegte Datei (siehe *mailx-Variablen*).
- Ungelesene Nachrichten (U) und mit *hold* gekennzeichnete Nachrichten (P) bleiben im Standard-Briefkasten.
- Gesicherte Nachrichten (*) werden aus dem Standard-Briefkasten gelöscht.

Siehe *mailx*-Kommando *exit*.

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.

Den Inhalt des Subject-Feldes entnimmt *mailx* dem Subject-Feld der Nachricht und stellt die Zeichenfolge *Re:* voran.

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 (siehe *DATEIEN*).

Das Kommando darf nicht in einer Kommandodatei stehen.

Reply[␣nachrichtenliste]
Respond[␣nachrichtenliste]
beantwortet die erste in der Nachrichtenliste angegebene Nachricht. Das ist die Nachricht mit der kleinsten Nummer, die sich aus der Nachrichtenliste ergibt.

Die Antwort geht an alle Absender aus *nachrichtenliste*. Die Absender entnimmt *Reply* jeweils der ersten Zeile des Nachrichtenkopfs (nicht dem From-Feld weiter unten im Nachrichtenkopf).

mailx geht in den Sendemodus. Den Inhalt des Subject-Feldes entnimmt *mailx* dem Subject-Feld der ersten Nachricht und stellt die Zeichenfolge *Re:* voran.

Wenn Sie die Texteingabe beenden, 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 (siehe *DATEIEN*).

Das Kommando darf nicht in einer Kommandodatei stehen.

save[[␣nachrichtenliste]␣datei]
schreibt die angegebenen Nachrichten in die Datei *datei*. Die Datei wird erweitert, falls sie existiert (siehe *DATEIEN*).

Die Nachrichten werden als gesichert gekennzeichnet (* in Feld 1 der Übersichtszeilen). Das heißt, sie werden aus dem Standard-Briefkasten gelöscht, sobald *mailx* beendet wird (außer die *mailx*-Variable *keepsave* ist gesetzt). Siehe *mailx*-Kommandos *exit* und *quit*.
Ansonsten funktioniert *save* wie *copy*.

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 der Liste gesetzt wird (From-Eintrag, Netz-Pfade werden abgetrennt). Die Datei wird erweitert, falls sie existiert (siehe *DATEIEN*).

Die Nachrichten werden als gesichert gekennzeichnet (* in Feld 1 der Übersichtszeilen). Das heißt, sie werden aus dem Standard-Briefkasten gelöscht, sobald *mailx* beendet wird (außer die *mailx*-Variable *keepsave* ist gesetzt).
Siehe *mailx*-Kommandos *exit* und *quit*.
Ansonsten funktioniert *Save* wie *Copy*.

set[␣name[=eintrag]]
setzt die Variable *name*.

name
eine interne *mailx*-Variable (siehe *mailx-Variablen*) oder eine frei definierte Variable.

eintrag
eine Zeichenkette (auch die leere) oder ein numerischer Wert.

eintrag nicht angegeben:
die Variable 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 kann man nicht verändern. Definiert man eine gleichlautende Variable, dann nimmt *mailx* jedoch deren Wert (außer bei *echo*).

Auf Variablen kann man mit dem Tilde-Kommando *~i name* zugreifen.

Die Angabe *\n* innerhalb eines Variableninhalts wird als Neue-Zeile-Zeichen interpretiert, *\t* als Tabulatorzeichen.

Variablen löschen kann man mit *unset*.

shell
ruft standardmäßig */usr/bin/sh* auf.
Einen anderen Kommandointerpreter kann man mit der *mailx*-Variablen *SHELL* angeben (siehe *mailx-Variablen*).

Das Kommando 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 *Kommandodateien*).

top[␣nachrichtenliste]
gibt die ersten 5 Zeilen des Nachrichtenkopfes für jede angegebene Nachricht auf die Standard-Ausgabe aus. Die Anzahl der ausgegebenen Zeilen kann man mit der *mailx*-Variablen *toplines* verändern.

touch[␣nachrichtenliste]
Die angegebenen Nachrichten werden als gelesen behandelt, d.h. sie werden beim Schließen des Briefkastens in den benutzereigenen Briefkasten geschrieben und anschließend gelöscht. Der benutzereigene Briefkasten ist *$HOME/mbox* bzw. die in der *mailx*-Variablen *MBOX* festgelegte Datei (siehe *mailx-Variablen*).

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]
siehe *print*

Type[␣nachrichtenliste]
siehe *Print*

undelete[␣nachrichtenliste]
holt die angegebenen Nachrichten zurück, wenn sie in dieser Sitzung gelöscht wurden. Die Nachrichten werden als gelesen behandelt.

Wenn die *mailx*-Variable *autoprint* gesetzt ist, wird die letzte der zurückgeholten Nachrichten ausgegeben (siehe *mailx-Variablen*).

undiscard[␣feld]
unignore[␣feld]
Hebt die Wirkung von *discard* bzw. *ignore* auf. Die angegebenen Felder des Nachrichtenkopfes werden bei der Ausgabe nicht mehr unterdrückt.

undiscard␣feld...
unignore␣feld...
löscht die angegebenen Felder des Nachrichtenkopfes.

unset[␣name...]
löscht die angegebenen Variablen. Umgebungsvariable lassen sich nicht löschen.

Löscht man eine *mailx*-Variable, deren Name gleichlautend mit einer Umgebungsvariablen ist, kann man 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 (siehe *mailx-Variablen*).

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, siehe *DATEIEN*).

Das Kommando darf nicht in einer Kommandodatei stehen.

write[␣nachrichtenliste]␣datei
schreibt die angegebenen Nachrichten in die Datei *datei*. Die Datei wird erweitert, falls sie existiert (siehe *DATEIEN*).

write läßt den Nachrichtenkopf und die letzte Leerzeile weg, ansonsten funktioniert *write* wie *save*.

Die Nachrichten werden mit * in Feld 1 der Übersichtszeilen gekennzeichnet. Sie werden aus dem Briefkasten gelöscht, sobald *mailx* beendet wird (außer die *mailx*-Variable *keepsave* ist gesetzt). Siehe *mailx*-Kommandos *exit* und *quit*.

xit
siehe *exit*

z[±]
zeigt die nächste (*z+*) bzw. vorhergehende Seite (*z-*) der Übersichtszeilen (siehe *mailx*-Kommando *headers*).
Die Anzahl von Zeilen einer Seite entnimmt *mailx* der *mailx*-Variablen *screen*. Ist *screen* nicht gesetzt, gibt *mailx* 20 Zeilen aus.

Kein Argument angegeben:
wie *z+*.

mailx-Kommandos im Sendemodus (Tilde-Kommandos)

mailx-Kommandos im Sendemodus stehen am Zeilenanfang und beginnen mit dem Fluchtsymbol Tilde ~. Das Fluchtsymbol kann man mit der *mailx*-Variablen *escape* umdefinieren (siehe *mailx-Variablen*).

Tilde-Kommandos dürfen nicht in einer Kommandodatei stehen.

~!shell-kommando
führt *shell-kommando* aus. Standardmäßig wird */usr/bin/sh* aufgerufen und der angegebene Kommandoaufruf übergeben.
Einen anderen Kommandointerpreter kann man mit der *mailx*-Variablen SHELL angeben (siehe *mailx-Variablen*).

~.
beendet die Texteingabe.

Vorsicht

Wenn Sie mit *rlogin* an einem fernen Rechner arbeiten, interpretiert REMOS sowohl ~! als auch ~. und bricht die Sitzung ab.
Abhilfe: Beenden Sie die Texteingabe mit der Taste END oder definieren Sie das Fluchtsymbol mit der *mailx*-Variablen *escape* um oder definieren Sie das Beenden der Texteingabe mit der *mailx*-Variablen *dot* um.

~:mailx-kommando
~␣mailx-kommando
führt das angegebene *mailx*-Kommando 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*.

~?
Übersicht über die Tilde-Kommandos ausgeben.

~a
fügt den Wert der *mailx*-Variablen *sign* in den Text ein (siehe *mailx-Variablen*).

~A
fügt den Wert der *mailx*-Variablen *Sign* in den Text ein (siehe *mailx-Variablen*). Damit kann man z.B. eine weitere Briefsignatur definieren.

~b␣name...
fügt einen oder mehrere Namen zur Bcc-Liste hinzu.
Die Bcc-Liste (blind carbon copy, verdeckter Verteiler) enthält die Namen weiterer Empfänger der Nachricht. Diese Namen werden nicht in den Nachrichtenkopf eingefügt.

~c␣name...
fügt einen oder mehrere Namen zur Cc-Liste hinzu.
Die Cc-Liste (carbon copy, offener Verteiler) enthält die Namen weiterer Empfänger der Nachricht. Diese Namen werden in den Nachrichtenkopf zur Information eingefügt (Cc-Eintrag).

~d
fügt den Inhalt der Datei *$HOME/dead.letter* in den Text ein. Diese Datei enthält Nachrichten, die *mailx* nicht absenden konnte (siehe *mailx-Variablen, DEAD*).

~e
ruft den mit der *mailx*-Variablen *EDITOR* eingestellten Editor auf (standardmäßig *ed*) und lädt den bisher eingegebenen Text (siehe *mailx-Variablen, EDITOR*).
Nach Beenden der Editorsitzung kann der bearbeitete Text wie vorher weitergeschrieben werden.

~f[␣nachrichtenliste]
fügt die angegebenen Nachrichten unverändert in den Text ein.
Das Kommando wird nur ausgeführt, wenn *mailx* im Lesemodus (Format 1) aufgerufen wurde.

~h
fordert 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
fügt den Wert von *variable* in den Text ein. *variable* kann eine *mailx*-Variable oder eine Umgebungsvariable sein.

~m[␣nachrichtenliste]
fügt die angegebenen Nachrichten in den Text ein. Der Text wird um eine Tabulatorposition nach rechts verschoben.
Das Kommando wird nur ausgeführt, wenn *mailx* im Lesemodus (Format 1) aufgerufen wurde.

~p
zeigt den bisher eingegebenen Text am Bildschirm.

~q
bricht die Texteingabe ab. Der bisher eingegebene Text wird nicht abgeschickt, sondern in die Datei *$HOME/dead.letter* geschrieben (siehe *mailx-Variablen, DEAD*). *~q* wirkt wie die Taste DEL, läßt sich aber nicht mit der Variablen *ignore* unterdrücken.

~r␣datei
~<␣datei
~<!␣shell-kommando
fügt den Inhalt der angegebenen Datei in den Text ein.
Ist *!shell-kommando* (anstelle von *datei*) angegeben, wird *shell-kommando* ausgeführt und dessen Ausgabe in den Text eingefügt.

~s␣zeichenkette...
ersetzt den Inhalt des Subject-Feldes (Titel) im Nachrichtenkopf durch den Inhalt der Zeichenkette. Die Wiederholung (...) bedeutet, daß Sie Zeichenketten mit Leerzeichen nicht in Anführungszeichen zu setzen brauchen.

~t␣empfänger...
fügt die Namen eines oder mehrerer Empfänger hinzu (To-Liste). Mehrere Namen sind durch Leerzeichen zu trennen.

~v
ruft den mit der *mailx*-Variablen VISUAL eingestellten Editor auf (standardmäßig *vi*) und lädt den bisher eingegebenen Text (siehe *mailx-Variablen, VISUAL*).
Nach Beenden der Editorsitzung kann der bearbeitete Text wie vorher weitergeschrieben werden.

~w␣datei
schreibt den bisher eingegebenen Text in die angegebene Datei. Der Kopf wird nicht mitgeschrieben.

~x
bricht die Texteingabe ab. Der bisher eingegebene Text wird nicht abgeschickt und nicht gesichert.

~|␣shell-kommando
Der bisher eingegebene Text wird an die Standard-Eingabe von *shell-Kommandos* übergeben.
Ist der Ende-Status dieses Kommandos 0, dann wird der bisher eingegebene Text durch die Standard-Ausgabe des Kommandos ersetzt.

Kommandodateien

Kommandodateien sind Dateien mit *mailx*-Kommandos. Jedes *mailx*-Kommando muß in einer eigenen Zeile stehen. Kommandodateien kann man mit dem Kommando *source* ausführen oder man verwendet sie als Startdateien (siehe unten).

In einer Kommandodatei dürfen *mailx*-Kommandos stehen, außer den Tilde-Kommandos und den Kommandos:
!, Copy, edit, followup, Followup, hold, mail, preserve, reply, Reply, shell, visual.

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 (außer bei Option *-e*).

mailx arbeitet erst die globale Startdatei */etc/mail/mail.rc* ab, dann die benutzereigene Startdatei *$HOME/.mailrc*, soweit diese vorhanden sind.
Den Pfadnamen der benutzereigenen Startdatei kann man mit der Umgebungsvariablen *MAILRC* neu festlegen (siehe *UMGEBUNGSVARIABLEN*).

MAILX-VARIABLEN

mailx verwendet Umgebungsvariablen und interne *mailx*-Variablen.

mailx-Variablen kann man mit den *mailx*-Kommandos *set* und *unset* verändern, Umgebungsvariablen nicht.

Außer den im folgenden beschriebenen *mailx*-Variablen kann man frei definierte Variablen verwenden. Man kann sie importieren (Umgebungsvariable) oder mit *set* definieren.

Wenn eine Variable importiert wird, kann man ihren Wert mit *set* verändern; *unset* setzt sie wieder auf den ursprünglichen Wert. D.h. eine importierte Variable kann man nicht löschen.

Die folgenden Variablen sind interne *mailx*-Variablen. Man kann sie mit dem *mailx*-Kommando *set* setzen oder aus der Umgebung importieren.
Mit dem *mailx*-Kommando *unset* kann man diese Variablen löschen. Wenn eine Variable gelöscht wird, setzt *mailx* den jeweils beschriebenen Standard-Wert ein.

allnet
mailx behandelt alle Netz-Pfadnamen als gleich, die auf denselben Benutzernamen enden. Als Netz-Pfadnamen gelten Adressen der Form
...rechner!rechner!benutzerkennung.

Siehe Kommando *alternates* und Variable *metoo*.

Standard-Wert: Die Variable ist nicht gesetzt.

append
Nachrichten, die in den benutzereigenen Briefkasten (standardmäßig *$HOME/mbox*) geschrieben werden, werden an den Dateianfang und nicht an das Dateiende geschrieben.

Standard-Wert: Die Variable ist nicht gesetzt.

askbcc
Nach Eingabe des Subject-Feldes fordert *mailx* die Eingabe der Bcc-Liste.

Standard-Wert: Die Variable ist nicht gesetzt.

askcc
mailx fordert beim Aufruf die Eingabe der Cc-Liste an
(siehe
~c und *~h*).

Standard-Wert: Die Variable ist nicht gesetzt.

askbcc
wie *askcc* für *Bcc*-Liste.

asksub
mailx fordert beim Aufruf die Eingabe für das Subject-Feld an.
(siehe Kommandos *~s*, *~h*, option *-s*).

Standard-Wert: Die Variable ist gesetzt.

autoprint
Nach dem Kommando *delete* wird die nächste Nachricht und nach dem Kommando *undelete* die letzte der gelöschten Nachrichten ausgegeben.

Standard-Wert: Die Variable ist nicht gesetzt.

bang
Wenn *bang* gesetzt ist, speichert *mailx* das zuletzt mit *!kommando* ausgeführte SINIX-Kommando. Man kann es mit *!!* wiederholen.

Standard-Wert: Die Variable ist nicht gesetzt.

cmd=shell-kommando
mailx führt *shell-kommando* beim Kommando *pipe* aus, falls dort kein Kommando angegeben ist.

Standard-Wert: keiner.

conv=konversion
Für *konversion* können Sie *internet* angeben. *mailx* konvertiert dann UUCP-Adressen in Internet-Adressen.

Standard-Wert: keine Konversion.

crt=anzahl
Wenn eine Ausgabe mit *print* oder verwandten Kommandos mehr als *anzahl* Zeilen enthält, übergibt *mailx* sie an das Kommando, das in der Variablen *PAGER* festgelegt ist (Standard-Wert *PAGER=pg*).

Standard-Wert: Die Variable ist nicht gesetzt.

DEAD=datei
In die Datei *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 jedesmal überschrieben, falls sie existiert.

Standard-Wert: *$HOME/dead.letter*

debug
mailx gibt zusätzliche Meldungen zur Fehlerdiagnose aus.

Standard-Wert: Die Variable ist nicht gesetzt.

dot
Ein Punkt . in einer eigenen Zeile beendet die Texteingabe (anstelle des Kommandos ~.).

Standard-Wert: Die Variable ist nicht gesetzt.

EDITOR=shell-kommando
mailx führt *shell-kommando* aus, wenn man das Kommando *edit* oder *~e* gibt.

Standard-Wert: *ed*

escape=c
Das Fluchtsymbol Tilde ~ für die Tilde-Kommandos wird durch das Zeichen *c* ersetzt.

Standard-Wert: ~

folder=dateiverzeichnis
Wenn man vor einen anzugebenden Dateinamen ein Pluszeichen + schreibt, bezieht sich *mailx* auf *dateiverzeichnis* und nicht auf das aktuelle Dateiverzeichnis. Wenn *dateiverzeichnis* kein absoluter Pfadname ist, nimmt *mailx* *$HOME/dateiverzeichnis* an.

Die Angabe *+datei* gilt für alle *mailx*-Kommandos, die Dateinamen erwarten.

folder wird auch von der Variablen *outfolder* benötigt.

Man kann *+datei* auch beim Aufruf von *mailx* in der Kommandozeile angeben.

Standard-Wert: Die Variable ist nicht gesetzt.

header
mailx gibt nach dem Aufruf die erste Seite mit den Übersichtszeilen aus (siehe Kommando *headers*), sowie die aktuelle *mailx*-Version und die Anzahl der Nachrichten.
Wenn *header* nicht gesetzt ist, unterbleiben diese Ausgaben.

Standard-Wert: Die Variable ist gesetzt.

hold
Gelesene Nachrichten verbleiben im Standard-Briefkasten und werden nicht in den benutzereigenen Briefkasten gesichert (siehe *mailx*-Kommando *quit*).

Standard-Wert: Die Variable ist nicht gesetzt.

ignore
Das Signal SIGINT soll bei der Texteingabe ignoriert werden.

Standard-Wert: Die Variable ist nicht gesetzt.

ignoreeof
Die Texteingabe kann nur mit einem Punkt in einer eigenen Zeile oder dem Kommando ~. beendet werden. Dateiende (EOF, Taste END) wird ignoriert (siehe Variable *dot*).

Standard-Wert: Die Variable ist nicht gesetzt.

keep
Wenn der Standard-Briefkasten leer ist, soll er nicht gelöscht werden.

Standard-Wert: Die Variable ist nicht gesetzt.

keepsave
Dateien, die als gesichert gekennzeichnet sind, sollen nicht aus dem Standard-Briefkasten gelöscht werden.

Standard-Wert: Die Variable ist nicht gesetzt.

MBOX=datei
datei bezeichnet den benutzereigenen Briefkasten. Dorthin werden gelesene Nachrichten geschrieben, bevor sie *mailx* aus dem Standard-Briefkasten entfernt. Die Datei wird erweitert.

Standard-Wert: *$HOME/mbox*

metoo
Wenn die eigene Benutzerkennung in der Empfängerliste (To-Liste) erscheint, soll sie nicht daraus gestrichen werden.

Standard-Wert: Die Variable ist nicht gesetzt.

LISTER=shell-kommando
mailx benutzt *shell-kommando* zum Auflisten der Dateien des in *folder* genannten Dateiverzeichnisses (siehe Kommando *folders*).

Standard-Wert: *ls*

onehop
Wenn man mit *reply* oder *followup* auf eine Nachricht antwortet, bildet *mailx* alle Empfänger-Adressen relativ zum Rechner des Absenders.
Wenn *onehop* gesetzt ist, unterläßt *mailx* diese Adressenergänzung. Das ist sinnvoll, wenn in einem Netz alle Rechner direkt erreichbar sind.

Standard-Wert: Die Variable ist nicht gesetzt.

outfolder
Wenn die Variable *outfolder* gesetzt ist, legt *mailx* Protokolldateien, die von den Kommandos *followup* bzw. *Followup* erzeugt werden, nicht im aktuellen Dateiverzeichnis ab. Stattdessen legt die Variable *folder* dieses Dateiverzeichnis fest. Wenn die Variable *folder* nicht gesetzt ist, stellt *mailx* ein Pluszeichen + vor den Dateinamen und legt die Datei im aktuellen Dateiverzeichnis ab.

Standard-Wert: Die Variable ist nicht gesetzt.

page
mailx fügt beim Kommando *pipe* nach jeder Nachricht ein Zeichen Formularvorschub (FF = CTRL L = X'0C') ein.

Standard-Wert: Die Variable ist nicht gesetzt.

PAGER=shell-kommando
mailx übergibt an *shell-kommando* Ausgaben, die länger sind, als die in der Variablen *crt* festgelegte Anzahl von Zeilen.

Standard-Wert: *pg*

prompt=zeichenkette
Setzt das Bereitzeichen für *mailx*-Kommandos 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, wenn die Variable *record* gesetzt ist. Sind die Variablen *record* und *outfolder* gesetzt, nicht jedoch *folder*, so werden die Nachrichten in *+datei* und nicht in *datei* gespeichert (siehe *outfolder*).
Dies gilt nicht für die Kommandos *followup* und *Followup*, Die Datei wird erweitert.

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 mit DEL abgebrochen wurde. Gesichert wird in die Datei, die in der Variablen *DEAD* festgelegt ist.

Standard-Wert: Die Variable ist gesetzt.

screen=anzahl
Das Kommando *header* nimmt für eine Seite *anzahl* Zeilen an.

Standard-Wert: 20

sendmail=shell-kommando
mailx benutzt *shell-Kommando* zum Absenden von Nachrichten.

Standard-Wert: */usr/bin/rmail*

sendwait
mailx kehrt nach einem Sende-Kommando erst in den Kommandomodus zurück, wenn die Nachricht abgeschickt wurde.

Standard-Wert: Die Variable ist nicht gesetzt.

SHELL=shell-kommando
shell-kommando legt den Kommando-Interpreter fest, den *mailx* benutzt, um SINIX-Kommandos auszuführen (siehe Kommandos *!shell-kommando*, *shell*).

Standard-Wert: *sh*

showto
Das Kommando *header* zeigt den ersten Eintrag aus der Empfängerliste (To-Liste) anstelle des Namens des Absenders, wenn der Absender gleich der eigenen Benutzerkennung ist. Den Absender ermittelt *header* immer aus der ersten Zeile der Nachricht (From ...).

Standard-Wert: Die Variable ist nicht gesetzt.

sign=zeichenfolge
sign ist vorgesehen für eine Briefsignatur. Man kann diese mit *~a* bei der Texteingabe einfügen.

Standard-Wert: Die Variable ist nicht gesetzt.

Sign=zeichenfolge
Sign ist vorgesehen für eine Briefsignatur. Man kann diese mit *~A* bei der Texteingabe einfügen.

Standard-Wert: Die Variable ist nicht gesetzt.

toplines=anzahl
anzahl ist die Anzahl der Zeilen aus dem Nachrichtenkopf, die das Kommando *top* ausgibt.

Standard-Wert: 5

VISUAL=shell-kommando
mailx führt *shell-kommando* aus, wenn man das Kommando *visual* oder *~v* gibt.

Standard-Wert: *vi*

FEHLERMELDUNGEN

mailx gibt Fehlermeldungen aus. Diese sind in der Regel selbsterklärend.

DATEIEN

$HOME/.mailrc
Benutzereigene Startdatei. Diese muß eine Kommandodatei sein und wird von *mailx* bei jedem Aufruf durchlaufen, falls sie existiert (siehe *Startdateien*).

$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.

/etc/mail/mailx.rc
Globale Startdatei. Diese muß eine Kommandodatei sein und wird von *mailx* bei jedem Aufruf durchlaufen, falls sie existiert (siehe *Startdateien*).

$HOME/mbox
Der benutzereigene Briefkasten. Dorthin sichert *mailx* Nachrichten, die gelesen wurden.

/var/mail/$USER
Der Standard-Briefkasten. Darin sucht *mailx* ankommende Nachrichten.

./$USER
Dateien im aktuellen Dateiverzeichnis, deren Name gleich einer Benutzerkennung lautet, legt *mailx* bei folgenden Kommandos an:
Copy, followup, Followup, Save.

Anstelle des aktuellen Dateiverzeichnisses kann man dabei auch ein anderes Dateiverzeichnis wählen
(siehe *UMGEBUNGSVARIABLEN 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.

UMGEBUNGSVARIABLEN

mailx kopiert den Wert einer Umgebungsvariablen in eine entsprechende *mailx*-Variable. Wird diese geändert, nimmt *mailx* den geänderten Wert. Wird diese gelöscht, setzt sie *mailx* wieder auf den Wert der Umgebungsvariablen. *echo* greift nur auf die originalen Umgebungsvariablen zu.

Auf Umgebungsvariablen kann man mit *echo* und *~i* zugreifen.

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.

MAILRC=datei
datei definiert den Namen der benutzereigenen Startdatei (Standard: *$HOME/.mailrc*, siehe *Kommandodateien*).

$USER
Der Variablen *$USER* entnimmt *mailx* den Benutzernamen, um z.B. den Standard-Briefkasten zu finden.

BEISPIELE

1. Variable dot, autoprint und cmd setzen und *mailx* aufrufen.

```
$ dot=; autoprint=; cmd=lpr; export dot autoprint cmd
$ mailx
```

2. Beispiel für eine Startdatei

   Es sollen Variablen gesetzt und, falls *mailx* im Lesemodus aufgerufen wird, alle Nachrichten von *winni* ausgedruckt werden (mit *lpr*).

```
# Variablen fuer Bearbeitung
set page crt=24 cmd=lpr VISUAL=maxed
set sign="\n\tManualredaktion D ST QM2\n\t8 Muenchen 83"

# Verteiler: Systemverwalter im Netz
alias sys root@muenchen root@nuernberg root@frankfurt

# Bestimmte Post ausdrucken
if r
pipe winni lpr
from winni
endif
```

   Beachten Sie, daß *mailx* die Prozedur abbricht, wenn ein Kommando nicht ausgeführt werden kann. Dies ist hier beim Kommamndo *pipe* der Fall, weil keine Nachrichten von *winni* vorliegen. *from* (und eventuelle weitere Kommandos) führt *mailx* dann nicht aus.

3. Beispiel zum Dialog im Sendemodus

```
$ mailx sys
Subject: mailx-Beschreibung
Eine überabeitete mailx-Beschreibung
ist ab sofort erhältlich.
~a

        Manualredaktion DI ST QM2
        8 Muenchen 83
~.
```

Das Beispiel verwendet die Startdatei aus Beispiel 1. Der Empfänger der Nachricht ist *sys*. *~a* fügt die Unterschrift ein, *~.* beendet die Texteingabe.

4. Beispiel zum Dialog im Lesedemodus

```
$ mailx
mailx version 2.14 Type? for help.
"/usr/mail/hans": 4 messages 1 new 4 unread
 U  1 guenter            Mon Nov  21 13:05    10/164  Test 1
 U  2 hans               Mon Nov  21 13:15    10/228  Freigabe
 U  3 root               Mon Nov  21 14:00     9/176  Tools
>N  4 root               Mon Nov  21 15:03     8/149  Tools
? p
Message 4:
From root Thu Jun 15 14:20:00 1989
Date: Thu, 15 Jun 89 14:19:58  +0100
From: Superuser <root>
To: hans
Subject: Tools
Cc: winni
Status: RO

Tools zur Textbearbeitung sind freigegeben.
Installationsdiskette und Beschreibung
sind bei winni abzuholen.

? mb 1-3
? q
Saved 4 messages in /usr1/hans/mbox
```

Die aktuelle Nachricht ist Nachricht 1 als erste neue Nachricht. Diese wird mit *p* am Bildschirm ausgegeben. Die anderen Nachrichten sollen ungelesen in *$HOME/mbox* abgelegt werden.

SIEHE AUCH

ls, *mail*, *pg*, *sh*

makekey
Code für Verschlüsselung festlegen

makekey ist ein Verschlüsselungsprogramm. Es wird vor allem zum Verschlüsseln von Passwörtern verwendet (*passwd*).

Das Kommando liest von der Standard-Eingabe und wird normalerweise in Pipes verwendet.

```
/usr/lib/makekey
```

Arbeitsweise

makekey verarbeitet eine Eingabe von 10 Zeichen. Die letzten beiden Zeichen (Auswahl-Zeichen) legen den Verschlüsselungs-Algorithmus fest, mit dem die ersten 8 Zeichen verschlüsselt werden.
Die ersten acht Zeichen dürfen beliebige ASCII-Zeichen sein. Die letzten beiden Zeichen sollten Buchstaben, Ziffern oder die Zeichen Punkt . oder Schrägstrich / sein.

makekey gibt zunächst die beiden Auswahl-Zeichen aus. Dann folgen die 11 Bytes, die die Verschlüsselung darstellen.

Vorsicht

makekey kann unterschiedliche Ergebnisse liefern, abhängig davon, ob die Eingabe am Bildschirm oder von einer Datei aus erfolgt.

SIEHE AUCH

ed, *crypt*, *vi*
passwd [5]

man
Online-Dokumentation nutzen (manual pages)

Mit dem Kommamdo *man* können Sie die SINIX-Online-Dokumentation nutzen, das heißt:
- die Beschreibung eines Kommandos oder einer Funktion auf die Standard-Ausgabe oder einen Drucker ausgeben,
- eigene Dokumentation editieren und auf die Standard-Ausgabe oder einen Drucker ausgeben.

Die Online-Dokumentation ist zum Teil deutsch-, zum Teil englischsprachig. Sie ist nicht formatiert, d.h. Sie können sie mit jedem Editor editieren und auf jeden Drucker ausgeben.
Für die Ausgabe auf den Bildschirm ist es ratsam, den *man*-Aufruf mit dem Kommando *pg* zu einer Pipe zu verbinden. Dann können Sie auf dem Bildschirm vorwärts und rückwärts blättern oder an gewünschte Stellen springen.

```
man␣[option]...[␣kapitelnummer]␣name
```

Keine Option angegeben
Die Dokumentation zu dem Eintrag *name* wird auf die Standard-Ausgabe ausgegeben.

option

Sie können mehrere Optionen gleichzeitig angeben. Die Reihenfolge der Optionen ist beliebig. Die Optionen müssen Sie einzeln, d.h. durch Leerzeichen getrennt, angeben.

-p[druckergruppe]
Die Dokumentation wird auf den Drucker ausgegeben und nicht auf die Standard-Ausgabe.
Wenn Sie die Ausgabe auf einen bestimmten Drucker leiten wollen, so müssen Sie dessen Druckergruppe hier angeben. Die Druckergruppe kann Ihnen Ihr Systemverwalter nennen.

druckergruppe nicht angegeben:
Die Ausgabe erfolgt auf den ersten freien Drucker der Druckergruppe, die in /usr/spool/spooler/CONFIG definiert ist.

-e[dateiverzeichnis]
Die Dokumentation wird in das angegebene Dateiverzeichnis kopiert. Wenn Sie kein Dateiverzeichnis angeben, wird die Dokumentation in das aktuelle Dateiverzeichnis kopiert.
Der Pfadname muß relativ angegeben werden, und darf nicht absolut angegeben

werden.
Die Dokumentation wird nicht auf die Standard-Ausgabe ausgegeben.
Anschließend können Sie die Dokumentation z.B. editieren. Wenn Sie danach die editierte Dokumentation mit dem *man*-Kommando weiterverarbeiten wollen, müssen Sie beim Aufruf die Option *-d* angeben.
-e und *-d* zusammen angeben, bewirkt nichts.

-d

man sucht die Dokumentation zu *name* im aktuellen Dateiverzeichnis.
Mit der Option *-e* können Sie die "offizielle" Dokumentation in das aktuelle Dateiverzeichnis kopieren, aber dort auch Dokumentation für eigene Kommandos oder Funktionen ablegen. Anschließend können Sie diese Dokumentation mit *man* weiterverarbeiten.
Beim *man*-Aufruf können Sie die Option *-d* mit anderen Optionen kombinieren.
Wenn Sie *-e* und *-d* zusammen angeben, bewirken Sie nichts.

-d nicht angegeben:
man sucht im Dateiverzeichnis */usr/catman/?_man*. Das Fragezeichen steht für den Abschnitt der UNIX-Standard-Dokumentation: 1 für Kommandos, 3 für C-Bibliotheksfunktionen, ... Die Nummern stimmen mit den Kapitelnummern der Original-UNIX-Dokumentation überein.

-w

man gibt nur den Pfadnamen relativ zum Dateiverzeichnis */usr/catman* aus. Der Name der Datei, die die Dokumentation enthält, endet häufig mit *.z*. Dieses Suffix zeigt an, daß die Datei mit dem Kommando *pack* komprimiert wurde, um Speicherplatz zu sparen.

kapitelnummer

kapitelnummer ist eine ganze Zahl zwischen 1 und 8. Die Kapitelnumerierung ist aus der Original-UNIX-Dokumentation übernommen.
Die Angabe der Kapitelnummer dient zum schnelleren und eindeutigen Auffinden der Dokumentation:
- schneller, weil weniger Einträge durchsucht werden müssen, und
- eindeutig, weil es gleichnamige Einträge in verschiedenen Dateiverzeichnissen unter */usr/catman* gibt: z.B. *chmod* heißt ein Kommando und ein Systemaufruf.

Alle Kommandos stehen in Kapitel 1.

kapitelnummer nicht angegeben:
man gibt alle Dokumentationen aus, die zum angegeben Namen passen.

name

Name des Kommandos oder der Funktion, etc., dessen bzw. deren Dokumentation Sie verarbeiten wollen.

DATEIEN

*/usr/catman/?_man/man[1-8]/**
Vorformatierte Handbuch-Einträge

BEISPIEL

```
$ man man | pg
```

Die Beschreibung von *man* wird auf den Bildschirm ausgegeben. Mit *pg* können Sie beliebig in der Beschreibung vor- und zurückblättern.

SIEHE AUCH

pg

mesg
Nachrichtenempfang verbieten oder erlauben

mesg regelt den Empfang von Nachrichten auf einer Datensichtstation. Mit *mesg* können Sie entweder abfragen, ob Ihre Datensichtstation Nachrichten empfangen kann (Format 1), oder festlegen, daß andere Benutzer mit *write* Nachrichten auf Ihren Bildschirm schreiben dürfen bzw. nicht schreiben dürfen (Format 2).

```
mesg                                                    Format 1
mesg␣option                                             Format 2
```

Format 1: Einstellung der Datensichtstation abfragen

mesg

mesg gibt die aktuelle Einstellung für Ihren Bildschirm aus und liefert den Ende-Status 0, falls der Empfang von Nachrichten erlaubt ist, sonst 1.

Format 2: Einstellung festlegen

mesg␣option

option
: Angabe, ob der Empfang von Nachrichten erlaubt oder verboten wird.

-y
: Andere Benutzer dürfen Nachrichten an die Datensichtstation des aufrufenden Benutzers senden.

-n
: Andere Benutzer dürfen keine Nachrichten an die Datensichtstation des aufrufenden Benutzers senden. Dieses Verbot gilt nicht für Nachrichten, die der Systemverwalter mit *wall* verschickt. Diese Nachrichten werden trotzdem empfangen.

ENDE-STATUS

0 Senden von Nachrichten erlaubt
1 Senden von Nachrichten nicht erlaubt
2 Fehler

DATEIEN

*/dev/tty**
Bildschirmdateien

BEISPIELE

1. Die Benutzerin *herta* ist an der Datensichtstation *tty01* angemeldet und fragt die Einstellung der Datensichtstation ab:

```
$ mesg
is n
```

Die Datensichtstation *tty01* ist für den Empfang von Nachrichten gesperrt. Sie können daher an *herta* an der Datensichtstation *tty01* keine Nachricht schicken.

```
$ write herta
Permission denied.
```

2. Einstellung an der Datensichtstation *tty01* ändern:

```
$ mesg y
```

SIEHE AUCH

tty, *write*
wall [5]

mkdir
Dateiverzeichnis erzeugen
(make a directory)

Mit *mkdir* können Sie ein neues Dateiverzeichnis einrichten.

mkdir trägt im neuen 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 Schreibrecht haben, das dem einzurichtenden Dateiverzeichnis übergeordnet ist.

```
mkdir[␣option]␣dateiverzeichnis␣...
```

Keine Option angegeben
: *mkdir* legt die in *dateiverzeichnis* angegebenen Dateiverzeichnisse mit den Zugriffsrechten *777* (siehe *chmod*) an. Mit dem *umask* Befehl kann dieser Wert umgesetzt werden.

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 für *dateiverzeichnis* sowohl den relativen als auch den absoluten Pfadnamen angeben.

  Sie können mehrere Dateiverzeichnisse angeben.

  Das neue Dateiverzeichnis erhält die realen Benutzer- und Gruppennummern des aufrufenden Prozesses.

ENDE-STATUS

0 *mkdir* konnte alle angegebenen Dateiverzeichnisse einrichten
≠0 Ein Fehler ist aufgetreten. *mkdir* gibt eine Fehlermeldung aus.

BEISPIEL

Anlegen des Dateiverzeichnisses *briefe* im Dateiverzeichnis */usr/sysiphus/sonstiges*:

Stellen Sie zunächst fest, in welchem Dateiverzeichnis Sie sich befinden. Die Zugriffsrechte sind auf den Wert 777 gesetzt.

```
$ pwd
/usr/sysiphus
```

Lassen Sie 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
```

Legen Sie das neue Dateiverzeichnis an.

```
$ mkdir sonstiges/briefe
```

Prüfen Sie, ob das Dateiverzeichnis *briefe* angelegt wurde.

```
$ cd sonstiges
$ ls -l
total 5
drwx--x--x   2 sysiphus   gruppe1      520   Jan 22 16:21  briefe
                  .
                  .
                  .
```

SIEHE AUCH

rm, rmdir, sh, umask
intro(), mkdir() [14]

mkmsgs
Meldungsdateien für gettxt erstellen (make messages)

mkmsgs erstellt eine Datei mit Zeichenketten, auf die die Textsuch-Kommandos *gettxt, srchtxt, exstr* und die C-Funktion *gettxt()* zugreifen können. Die Eingabe für *mkmsgs* ist eine editierbare Datei mit Zeichenketten, die in einer bestimmten Landessprache geschrieben sind (siehe *setlocale()* [14]). Die Ausabedatei enthält dieselben Zeichenketten, aber in einem Format, das den Zugriff durch *gettxt* und *gettxt()* ermöglicht. *mkmsgs* kopiert die Zeichenketten aus der Eingabedatei in der dort vorliegenden Reihenfolge in die Ausgabedatei.

```
mkmsgs [-i lokale] [-o] eingabe_datei meldungsdatei
```

Keine Option angegeben
meldungsdatei wird im aktuellen Dateiverzeichnis angelegt.

-i lokale
Für *lokale* geben Sie die Landessprache an, in der die Zeichenketten in *eingabedatei* geschrieben sind. Der Wert, den man *lokale* gibt, soll später der Umgebungsvariablen *LC_MESSAGES* bzw. *LANG* zugewiesen werden, damit nachher mit *gettxt()* auf diese Datei zugegriffen werden kann (siehe *BEISPIELE*). Die für *gettxt* und *gettxt()* lesbare *meldungsdatei* wird dann im Dateiverzeichnis */usr/lib/locale/lokale/LC_Messages* angelegt, wobei Sie mit Ihrer Angabe für *lokale* den einfachen Namen des Dateiverzeichnisses */usr/lib/locale/lokale* bestimmen. Nur der Systemverwalter und Mitglieder der Benutzergruppe *bin* können in diesem Dateiverzeichnis Dateien anlegen oder überschreiben. Wenn es unter */usr/bin/locale* das von Ihnen angegebene Dateiverzeichnis *lokale* noch nicht gibt, wird es angelegt.

Die Option *-i* ist nur für den Systemverwalter bestimmt. Ein normaler Benutzer hat weder das Recht, unter */usr/lib/locale* ein Dateiverzeichnis zu erzeugen, noch unter */usr/lib/locale/etwas/LC_MESSAGES* eine Datei anzulegen.

-o
Wenn es *meldungsdatei* schon gibt, wird sie überschrieben.

eingabedatei
Datei mit den ursprünglichen Zeichenketten, die *mkmsgs* in ein anderes Format bringt. Jede Zeichenkette muß in einer neuen Zeile stehen und nicht druckbare Zeichen müssen durch Escape-Folgen dargestellt werden, z.B. \t für das Tabulator-Zeichen, \n für das Neue-Zeile-Zeichen usw. (wie bei der C-Funktion *printf()*). Außerdem sind auch Oktaldarstellungen in der Form \nnn erlaubt. Eine leere Zeichenkette stellen Sie durch eine leere Zeile dar.

Eine bereits bestehende *eingabedatei* können Sie mit einem beliebigen Editor verändern: vorhandene Zeichenketten können Sie überschreiben, neue Zeichenketten können Sie nur am Ende der Datei anfügen. Aus einer geänderten *eingabedatei* können Sie mit *mkmsgs* eine neue *meldungsdatei* erzeugen und an der richtigen Stelle im Dateibaum anlegen, nämlich unter */usr/lib/locale/etwas/LC_MESSAGES*, so daß die Textsuch-Funktion *gettxt()*, bzw. das Kommando *gettxt*, darauf zugreifen kann, wenn *LC_MESSAGES* (oder *LANG*, wenn *LC_MESSAGES* nicht gesetzt ist) auf den Wert *etwas* gesetzt ist. Sonst findet die Textsuch-Funktion *gettxt()* später nicht die richtige Zeichenkette und es besteht keine Kompatibilität mehr zwischen den verschiedenen Software-Versionen.

DATEI

/usr/lib/locale//LC_MESSAGES/**
Von *mkmsgs* erzeugte Meldungsdateien. Unter */usr/lib/locale* gibt es einige Dateiverzeichnisse, mindestens eines für jede unterstützte Landessprache.

BEISPIELE

1. Wir nehmen an, die Eingabedatei *C.str* enthalte folgende Zeichenketten:

```
Datei %s:\t kann nicht geoeffnet werden\n
%s: Unbekanntes Dateiverzeichnis\n
   .
   .
   .
Schreibfehler\n
   .
   .
```

Der folgende Aufruf von *mkmsgs* kopiert die Zeichenketten aus *C.str* im richtigen Format in die Meldungsdatei *UX* und legt diese im aktuellen Dateiverzeichnis an:

```
$ mkmsgs C.str UX
```

Der folgende Aufruf von *mkmsgs* kopiert die Zeichenketten aus der Eingabedatei *FR.str* im richtigen Format in die Meldungsdatei *UX* und legt diese im Dateiverzeichnis */usr/lib/locale/french/LC_MESSAGES* an:

```
$ mkmsgs -i french FR.str UX
```

Auf die Zeichenketten in */usr/lib/locale/french/LC_MESSAGES/UX* wird von den o.g. Kommandos und der C-Funktion *gettxt* zugegriffen, falls Sie die Umgebungsvariable *LC_MESSAGES* auf den Wert *LC_MESSAGES=french* gesetzt haben.

2. Vorhandene Dateien:

   Inhalt von *Input_datei1*:

```
Hallo !\n\007
Gute\t... Nacht !\n
Guten Morgen !\n
```

   Inhalt von *Input_datei2*:

```
Salut !\n
Bonne nuit !\n
```

   Erzeugungsverfahren und Installation der Meldungsdateien:

   Nehmen wir an, daß die Dateiverzeichnisse */usr/lib/locale/german/LC_MESSAGES* und */usr/lib/locale/french/LC_MESSAGES* nicht existieren. Der Aufruf durch den Systemverwalter

```
$ mkmsgs -i german Input_datei1 hallo
```

   wird erst das Dateiverzeichnis */usr/lib/locale/german* anlegen, darunter das Dateiverzeichnis *LC_MESSAGES*, und darunter eine Datei namens *hallo* erzeugen. Der Aufruf

```
$ mkmsgs Input_datei2 hallo
```

   wird eine Datei *hallo* im aktuellen Dateiverzeichnis erzeugen. Um diese danach mit *gettxt* in der Lokale *french* benützen zu können, muß der Systemverwalter folgende Aktionen durchführen:

```
$ mkdir /usr/lib/locale/french
$ mkdir /usr/lib/locale/french/LC_MESSAGES
$ cp hallo /usr/lib/locale/french/LC_MESSAGES/hallo
```

Zugriff auf diese Dateien:

Nehmen wir an, *LC_MESSAGES* ist leer und *LANG* hat den Wert *german*:

```
$ gettxt hallo:1
Hallo !
(gleichzeitig piepst es am Bildschirm, \007 ist das Klingelzeichen)
$ gettxt hallo:2
Gute    ... Nacht !
$ gettxt hallo:3
Guten Morgen !
$ LC_MESSAGES=french
$ export LC_MESSAGES
$ gettxt hallo:1
Salut !
$ gettxt hallo:2
Bonne nuit !\n
$ gettxt hallo:3
Message not found!!
(weil es nur 2 Meldungen in /usr/lib/locale/french/LC_MESSAGES/hallo gibt)
$ LC_MESSAGES=invalid
$ gettxt hallo:1
Message not found!!
$ gettxt hallo:2
Message not found!!
(weil es keine Datei /usr/lib/locale/invalid/LC_MESSAGES/hallo gibt)
```

SIEHE AUCH

exstr, gettxt, srchtxt
gettxt(), setlocale() [14]

3 *Internationale Umgebung - NLS (Native Language System)*

more
Bildschirmausgabe steuern

Die Kommandos *more* und *page* erlauben das Durchblättern von Dateien an der Datensichtstation.
Während *more* die Ausgabe durch Hochschieben der Bildschirmzeilen realisiert, löscht *page* den Bildschirm, bevor es eine neue Seite ausgibt. Das Kommando *more* ist weitgehend mit *page* identisch.

```
more[␣option][␣-zeilen][␣+zeilennummer][␣+/muster][␣datei]
```

option

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 das Bereitzeichen --MORE(..%)--. Drücken Sie die Leertaste, dann blättert *more* eine ganze Bildschirmseite weiter, mit ↵ 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 (bei *page* nur um eine Zeile). 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.

-c
Löscht vor Ausgabe einer neuen Seite den Bildschirm und ermöglicht damit ein schnelleres Durchblättern. **-c** wird nur ausgeführt, wenn in der Datensichtstation eine Funktion zum Löschen des Bildschirms vorhanden ist.

-d
Es werden Kommentare und Fehlermeldungen ausgegeben. Ohne *-d* weist *more* lediglich durch einen Klingelton auf Fehler hin.

-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.

-l
Das Formularvorschub-Zeichen CTRL l wird nicht wie Seitenumbruch behandelt.
Keine Angabe: nach jeder Zeile, die ein CTRL l enthält, unterbricht *more* die Ausgabe,

um ein *more*-Kommando entgegenzunehmen. Wenn eine Datei mit dem Formularvorschub-Zeichen beginnt, wird der Bildschirm vor Ausgabe der Datei gelöscht.

-r

Bei Angabe der Option *-r* verarbeitet *more* im Dateitext angegebene Steuerzeichen.
Keine Angabe: Steuerzeichen werden von *more* nicht ausgewertet.

-s

Die Ausgabedatei wird komprimiert, indem mehrere Leerzeichen zu einem zusammengefaßt werden.

-u

Die Hervorhebung von Escape-Sequenzen wird unterdrückt.
Keine Angabe: 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.

-w

Wenn kein Input mehr folgt, also der gesamte Text vollständig ausgegeben wurde, blendet *more* folgendes Bereitzeichen ein: --no more--. Damit *more* sich beendet, müssen Sie eine beliebige Taste drücken.
Keine Angabe: *more* beendet sich selbst, sobald kein Input mehr folgt.

-zeilen

Mit dem Wert für *zeilen* geben Sie die Anzahl der Zeilen an, die bei der Ausgabe am Bildschrim 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.

dateiname

Der Name der Datei, die ausgegeben werden soll. Sie können auch mehrere Dateinamen 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:

i Leertaste
more gibt *i* weitere Zeilen aus. Wenn Sie nur die Leertaste drücken, wird eine neue Bildschirmseite ausgegeben.

i ↵
more gibt *i* weitere Zeilen aus. Wenn Sie nur ↵ drücken, wird die nächste Zeile ausgegeben.

i CTRL d oder id
more gibt *i* weitere Zeilen aus. Gleichzeitig wird die Zeilenanzahl, um die weitergeblättert werden soll, auf *i* gesetzt.

iz
more setzt die Bildschirmgröße auf *i* und gibt den nächsten Bildschirm aus.

is
more überspringt bei der Ausgabe *i* Zeilen und gibt den darauf folgenden Text aus. Die Auslassung wird an der übersprungenen Stelle kommentiert.

if
Mit *i* geben Sie die Anzahl der Bildschirmseiten an, die ausgelassen werden sollen. An der übersprungenen Stelle wird die Auslassung der Bildschirmseiten in Zeilen angegeben.

i CTRL b oder ib
i legt fest, um wieviel Bildschirmseiten zurückgeblättert werden soll.

q oder Q oder :q oder :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
HELP-Taste.
Auf dem Bildschirm erscheint eine Übersichtstabelle über alle *more*-Kommandos mit einer kurzen Funktionsbeschreibung.

i/muster
more sucht nach dem i-ten Auftreten des regulären Ausdrucks *muster*. Tritt dieses Muster weniger als i-mal auf und liest *more* von einer Datei und nicht von einer Pipeline, bleibt die Position in der Datei unverändert. Sonst wird ein neuer Bildschirm ausgegeben, der zwei Zeilen vor der Zeile beginnt, in der der reguläre Ausdruck das i-te Mal auftritt.

in
more sucht nach dem i-ten Auftreten des zuletzt eingegebenen regulären Ausdrucks.

(Hochkomma) '
more springt zu dem Punkt, an dem die letzte Suche nach einem regulären Ausdruck begann. Alle folgenden '-Eingaben werden ignoriert. Falls noch keine Suchfunktion aufgerufen wurde, geht *more* an den Anfang der Datei.

!kommando
more startet eine neue Shell, die das 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 Shellkommando expandiert;

Wollen Sie ein % oder ein ! in das Kommando einfügen, müssen Sie das Zeichen (% oder !) durch einen vorangestellten Gegenschrägstrich \ entwerten.

i:n
more springt zur i-ten Datei aller in der Kommandozeile angegebenen Dateien. Sind weniger als *i* Dateien aufgeführt, springt *more* zur letzten Datei. Wenn Sie dagegen kein *i* angeben, springt *more* zu der in der Reihenfolge nächsten Datei. Befinden Sie sich bereits in der letzten Datei, beendet sich *more*.

i:p
more springt zur i-ten vorhergehenden Datei aller in der Kommandozeile angegebenen Dateien. Sind in der Kommandozeile weniger als *i* 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

Die Kommandos *more* und *page* setzen 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 und *page* arbeiten 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* und *page* ist langsamer als Vorwärtsspringen; deshalb ist der Rückwärtsgang zum Durchsuchen großer Dateien weniger geeignet.
Die Kommandos *more* und *page* holen sich die Daten zur Bildschirmsteuerung aus der Datei *termcap*.

DATEIEN

/usr/share/lib/termcap
Diese Datei enthält Daten zur Bildschirmsteuerung.

/usr/lib/more.help
HELP-Datei

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, *csh*, *ed*, *man*, *pg*, *script*, *sh*
environ, *termcap* [5]
nroff, *ul* [1-2]

mt
Magnetband bearbeiten (magnetic tape)

Mit *mt* können Sie den Treiber eines Magnetbandkassetten-Laufwerkes oder eines Magnetbandlaufwerkes ansprechen und:

- das Band nachspannen,
- das Band an den Bandanfang zurückspulen,
- auf eine bestimmte Bandstelle positionieren,
- eine Archiv-Endemarke schreiben oder
- Informationen über das angesprochene Magnetbandlaufwerk bzw. Magnetbandkassetten-Laufwerk abrufen.

Weitere Informationen zur Bedienung der Laufwerke finden Sie in der Betriebsanleitung zu dem entsprechenden Laufwerk oder in der Betriebsanleitung zu Ihrem Rechner.

Vor dem Aufruf beachten

Bevor Sie *mt* aufrufen, sollten Sie folgendes erledigen:

1. Legen Sie das Magnetband bzw. die Magnetbandkassette in das entsprechende Laufwerk.
2. Prüfen Sie, ob die Stromversorgung des Laufwerkes eingeschaltet ist.

Eine Gerätedatei voreinstellen

Das Kommando *mt* braucht den Namen einer Gerätedatei, damit es den entsprechenden Treiber findet. Wenn Sie beim Aufruf von *mt* keine Gerätedatei angeben (siehe Option *-f*), greift *mt* auf die Umgebungsvariable TAPE zu.
Wenn Sie der Variablen TAPE keinen Wert zugewiesen haben, greift *mt* auf die Gerätedatei */dev/rmt12* zu. Mit diesem Namen sprechen Sie das 1/2 Zoll-Magnetbandlaufwerk MG16 an; das Band wird vor dem ersten Zugriff nicht automatisch zurückgespult.

Beispiel

Wenn Sie häufig auf das Magnetbandkassetten-Laufwerk zugreifen, können Sie der Umgebungsvariablen TAPE den absoluten Pfadnamen der Gerätedatei */dev/rmt32* zuweisen. Dazu tragen Sie die folgenden Zeilen in die Datei *$HOME/.profile* ein:

```
TAPE=/dev/rmt32
export TAPE
```

Wenn Sie sich das nächste Mal am System anmelden, brauchen Sie diese Gerätedatei beim Aufruf nicht mehr angeben. Wollen Sie zwischendurch auf eine andere Gerätedatei zugreifen, so rufen Sie *mt* mit der Option *-f* und der entsprechenden Gerätedatei auf (siehe Option *-f*).

```
mt[␣-f␣geraetedatei]␣kommando[␣n]
```

-f␣geraetedatei
den Treiber des bei *geraetedatei* angegebenen Laufwerkes bearbeiten.

geraetedatei
Für *geraetedatei* geben Sie die gewünschte Gerätedatei an (siehe nachfolgende Tabelle). Die Gerätedatei muß eine raw-Gerätedatei und darf keine block-Gerätedatei sein.

Der Name der entsprechenden Gerätedatei entscheidet darüber, ob das Magnetband nach dem Zugriff automatisch an den Bandanfang zurückgespult wird oder nicht.

Sie erkennen den Namen einer Gerätedatei, bei der das Magnetband nicht automatisch zurückgespult wird, wie folgt:

- die Nummer im Namen ist um 4 höher als bei der entsprechenden Gerätedatei, bei der das Magnetband automatisch zurückgespult wird.

Beispiel

/dev/rmt0
automatisch zurückspulen

/dev/rmt4
nicht automatisch zurückspulen

Wenn Sie mit *mt* auf eine bestimmte Bandstelle positionieren wollen, müssen Sie für das entsprechende Laufwerk die Gerätedatei angeben, bei der das Band nicht automatisch zurückgespult wird.

Mögliche Angaben für *geraetedatei* entnehmen Sie bitte A.3 *Gerätedateien für Datenträger*. Dort sind die Gerätedateien mit der zugehörigen Schreibdichte und Schreibgeschwindigkeit aufgeführt.

Die folgende Tabelle enthält nur einige wichtige Gerätedateien:

Gerätedatei	Laufwerktyp	automatisch zurückspulen
/dev/rmt/c0s0	Kassette	ja
/dev/rmt/c0s0n	Kassette	nein
/dev/rmt0	Band	ja
/dev/rmt4	Band	nein
/dev/rmt8	Band	ja
/dev/rmt12	Band	nein

Kassette: Magnetbandkassetten-Laufwerk
Band: Magnetbandlaufwerk

Bei Rechnern des Typs MX 300 I sind diese Gerätedateien standardmäßig eingerichtet. Bei den übrigen Rechnern sind nur die Gerätedateien für das Magnetbandkassetten-Laufwerk standardmäßig eingerichtet (siehe [5], [6], [7]).

-f nicht angegeben:
mt wertet die Umgebungsvariable TAPE aus.
Wenn Sie der Variablen TAPE keinen Wert zugewiesen haben, greift *mt* auf die Gerätedatei */dev/rmt12* zu. Mit diesem Namen sprechen Sie das 1/2 Zoll-Magnetbandlaufwerk MG16 an; das Band wird vor dem ersten Zugriff nicht automatisch zurückgespult.

kommando
das Kommando, das der der Treiber des entsprechenden Magnetband- bzw. des Magnetbandkassetten-Laufwerkes ausführen soll. Sie können immer nur ein *kommando* angeben. Für einige Kommandos gibt es zwei unterschiedliche Schreibweisen; diese Schreibweisen sind jeweils vor der Beschreibung des entsprechenden Kommandos angegeben.

Einige der folgenden Kommandos sind nur für Magnetbandkassetten-Laufwerke, andere nur für Magnetbandlaufwerke erlaubt. Manche Kommandos sind nur bei bestimmten Rechnern möglich. Beachten Sie bitte die jeweiligen Hinweise in der Beschreibung.

kommando kann sein:

fsf
vorwärts an den Anfang des nächsten Archivs positionieren.
Ein Archiv beinhaltet alle Dateien zwischen:

- der Band-Anfangsmarke und der ersten Archiv-Endemarke bzw.
- zwei Archiv-Endemarken.

Das letzte Archiv auf dem Band wird durch zwei Archiv-Endemarken begrenzt.

Kommandos wie *cp* und *tar* schreiben automatisch eine Archiv-Endemarke auf das Band, wenn sie nach Abschluß der Schreiboperation die entsprechende Gerätedatei schließen. Deshalb enthält ein Archiv immer die Dateien, die Sie bei einem Zugriff auf Magnetband bzw. Magnetbandkassette geschrieben haben.

Verwenden Sie *fsf* nur bei Gerätedateien, bei denen das Band nicht automatisch nach dem Zugriff zurückgespult wird
(siehe Option *-f*).

bsf
können Sie nur bei einem Magnetbandlaufwerk angeben, nicht bei einem Magnetbandkassetten-Laufwerk.

bsf positioniert zurück an den Anfang des vorausgehenden Archivs. Weitere Informationen zum Begriff Archiv finden Sie in der Beschreibung zu *fsf*.

Verwenden Sie *bsf* nur bei Gerätedateien, bei denen das Band nicht automatisch nach dem Zugriff zurückgespult wird
(siehe Option *-f*).

fsr
können Sie nur bei einem Magnetbandlaufwerk angeben, nicht bei einem Magnetbandkassetten-Laufwerk.

fsr positioniert vorwärts an den Anfang des nächsten Satzes (record). Dazu sucht *mt* die nächste Satzlücke (record gap).

Mit *write()* können Sie Daten direkt auf ein Band schreiben. Mit jedem Aufruf von *write()* schreiben Sie einen Satz, das Satzende wird automatisch durch eine Satzlücke gekennzeichnet. Bei Magnetbandlaufwerken sind Sätze der Länge 1 byte bis 32 Kbyte erlaubt. Das erste *write()* legt die Satzlänge für das Band fest; Sie können aber auch Sätze variabler Länge auf ein Band schreiben.

Verwenden Sie *fsr* nur bei Gerätedateien, bei denen das Band nicht automatisch nach dem Zugriff zurückgespult wird (siehe Option *-f*).

bsr
können Sie nur bei einem Magnetbandlaufwerk angeben, nicht bei einem Magnetbandkassetten-Laufwerk.

bsr positioniert zurück zum Anfang des vorausgehenden Satzes (record). Dazu sucht *mt* die nächste Satzlücke (record gap). Weitere Informationen zum Begriff Satz finden Sie in der Beschreibung zu *fsr*.

Verwenden Sie *bsr* nur bei Gerätedateien, bei denen das Band nicht automatisch nach dem Zugriff zurückgespult wird (siehe Option *-f*).

eof
weof
Archiv-Endemarke an die aktuelle Band-Position schreiben. Auf diese Weise können Sie ein "leeres" Archiv erzeugen. Eine Shell-Prozedur könnte dann solange Dateien vom Band einlesen, bis sie ein leeres Archiv findet.

Kommandos wie *cp* und *tar* schreiben automatisch eine Archiv-Endemarke, bevor sie die entsprechende Gerätedatei schließen.

rewind
das Magnetband an den Bandanfang zurückspulen. Das Argument *n* wird ignoriert.

Verwenden Sie *rewind* nur bei Gerätedateien, bei denen das Band automatisch nach dem Zugriff zurückgespult wird (siehe Option *-f*).

offline
rewoffl
können Sie nur bei einem Magnetbandlaufwerk angeben, nicht bei einem Magnetbandkassetten-Laufwerk.

offline und *rewoffl* spulen das Magnetband an den Bandanfang zurück und bringen das Magnetbandgerät in den "off line"-Modus. Das Argument *n* wird ignoriert.
Weitere Informationen hierzu finden Sie in der Betriebsanleitung zu Ihrem Magnetbandlaufwerk.

ret
retension
können Sie nur bei Magnetbandkassetten-Laufwerken angeben.

ret und *retension* spannen das Magnetband nach. Das Argument *n* wird ignoriert. In den folgenden Fällen sollten Sie das Magnetband nachspannen:

- Wenn Sie eine Magnetbandkassette in das Laufwerk einlegen.
  Bei Rechnern des Typs MX 300 I wird das Magnetband automatisch vor dem ersten Zugriff nachgespannt.
- Wenn Sie das Magnetband einmal vollständig beschrieben oder gelesen haben.

noret
können Sie nur bei Magnetbandkassetten-Laufwerken angeben.

noret spannt das Magnetband nicht nach. Das Argument *n* wird ignoriert.

Bei Rechnern des Typs MX 300 I wird das Magnetband automatisch vor dem ersten Zugriff nachgespannt.

erase
den Inhalt des Magnetbandes ab der aktuellen Position bis Bandende löschen. Das Argument *n* wird ignoriert.

bufsiz[␣groesse]
können Sie nur bei Magnetbandkassetten-Laufwerken angeben.

bufsiz verändert die Größe des Puffers, in dem die Daten bei Lese- oder Schreib-Operationen gesammelt werden. Erst wenn dieser Puffer voll ist, wird der Inhalt auf das Magnetband geschrieben oder in die entsprechenden Dateiverzeichnisse eingelesen.
Das Argument *n* wird ignoriert.

groesse
eine Zahl zwischen 40 und 1000. Diese Zahl legt die Größe des Puffers in kbyte fest. Ein größerer Puffer beschleunigt nachfolgende Lese- und Schreib-Operationen. Die Änderung gilt bis zum nächsten Aufruf von *bufsiz*.

Nach der Installation ist dieser Puffer standardmäßig 1 MB groß.

groesse nicht angegeben:
Die Puffergröße wird auf den kleinsten Wert 40 kbyte gesetzt.

status
Statusinformationen über das Magnetbandgerät ausgeben. Diese Informationen unterscheiden sich je nach Band- und Laufwerk-Typ.
Das Argument *n* wird ignoriert.

Die Ausgabe enthält immer den verwendeten Geräte-Typ. Zusätzlich wird die aktuell eingestellte Puffergröße ausgegeben.

Beispiel

Für ein Magnetbandkassetten-Laufwerk am MX 300 I erhalten Sie folgende Ausgabe:

```
Tandberg Cartridge Streamer tape drive:
  residual=2000 ds=0 er=0
```

n
eine ganze Zahl; das zuvor angegebene *kommando* wird *n*-mal ausgeführt. Bei den folgenden Kommandos wird das Argument *n* ignoriert:

bufsiz, *erase*, *noret*, *offline*, *ret*, *retension*, *rewind*, *rewoffl*, *status*

n nicht angegeben:
Das angegebene *kommando* wird einmal ausgeführt.

ENDE-STATUS

0 Der Treiber hat das Kommando *mt* erfolgreich ausgeführt.
1 Der Treiber hat das beim Aufruf von *mt* angegebene *kommando* nicht erkannt.
2 Bei der Ausführung des Kommandos *mt* ist ein Fehler aufgetreten.

DATEIEN

*/dev/rmt**
Gerätedateien für Magnetbandlaufwerke; die Nummer im Namen legt fest, ob das Magnetband automatisch zurückgespult wird oder nicht (siehe Option *-f*).

*/dev/rmt/**
Gerätedateien für die Magnetbandkassetten-Laufwerke
(siehe Option *-f*).

UMGEBUNGSVARIABLE

TAPE
Dieser Variablen können Sie als Wert die Gerätedatei zuweisen, auf die das Kommando *mt* zugreifen soll, wenn die Option *-f* nicht angegeben ist.

BEISPIELE

1. Die Datei *termine/kritisch* soll aus dem fünften Archiv einer Magnetbandkassette eingelesen werden. Die Magnetbandkassette ist in das Laufwerk eingelegt und die aktuelle Bandposition ist der Bandanfang:

   a) An den Anfang des fünften Archivs positionieren:

   ```
   $ mt -f /dev/rmt/c0s0 fsf 4
   ```

   b) Die gewünschte Datei einlesen:

   ```
   $ tar xvf /dev/rmt/c0s0 termine/kritisch
   x termine/kritisch
   ```

   Nach Ausführung von *tar* wird das Magnetband automatisch an den Bandanfang zurückgespult, da als Gerätedatei */dev/rmt/c0s0* angegeben ist.

2. Das Magnetband soll nach dem Einlegen der Kassette nicht automatisch vor dem ersten Zugriff nachgespannt werden:

   a) Die Magnetbandkassette in das Laufwerk legen.

   b) Das Kommando eingeben:

   ```
   $ mt -f /dev/rmt/c0s0 noret
   ```

   c) Anschließend mit weiteren Kommandos auf das Band zugreifen.
   Die aktuelle Bandposition ist jetzt der Bandanfang, da als Gerätedatei */dev/rmt/c0s0* angegeben ist.

   Wenn das nächste Mal eine Magnetbandkassette in das Laufwerk eingelegt wird, wird das Band wie üblich wieder automatisch nachgespannt.

3. Mit dem Kommando *tar* soll ein neues Archiv hinter die bereits auf dem Band enthaltenen Archive angehängt werden:

   a) Die Magnetbandkassette in das Laufwerk legen.

   b) Auf die letzte Archiv-Endemarke positionieren:

   ```
   $ mt -f /dev/rmt/c0s0 eod
   ```

   c) Das neue Archiv ab dieser Position auf das Band schreiben:

   ```
   $ tar cvf /dev/rmt/c0s0 cprog
   ```

   Nach Ausführung von *tar* wird das Magnetband nicht automatisch an den Bandanfang zurückgespult, da als Gerätedatei */dev/rmt/c0s0* angegeben ist. Die aktuelle Bandposition ist die letzte Archiv-Endemarke auf dem Band. Sie können also ohne nochmaliges Positionieren mit *tar* ein weiteres Archiv anhängen.

SIEHE AUCH

cp, *dd*, *env*, *set*, *tar*
ioctl(), *write()* [14]

mv
Dateien versetzen oder umbenennen (move)

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 Kopie der versetzten oder umbenannten Datei.

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.

```
mv[␣option][␣--]␣datei␣dateineu                    Format 1
mv␣datei␣...␣dvz                                   Format 2
mv␣dvz␣dvzneu                                      Format 3
```

Format 1: Datei umbenennen

mv[␣option][␣--]␣datei␣dateineu

Keine Option angegeben
: Wenn Sie für *dateineu* eine bestehende Datei angeben, für die Sie kein Schreibrecht haben, werden Sie gefragt, ob *mv* ausgeführt werden soll.
Ist die Standard-Eingabe keine Datensichtstation und geben Sie für *dateineu* eine bestehende Datei an, dann überschreibt *mv* den Inhalt von *dateineu* sofort.

option

-f
: Wenn Sie für *dateineu* eine bestehende Datei angeben, für die Sie kein Schreibrecht haben, werden Sie gefragt, ob *mv* wirklich ausgeführt werden soll.
Ist die Option *-f* angegeben, unterbleibt diese Frage.
Die Option *-f* überlagert die Option *-i*.

-i
: (i - interactive) Wenn Sie für *dateineu* eine bestehende Datei angeben, werden Sie gefragt, ob *mv* wirklich ausgeführt werden soll.

--
: Beginnt der Name der Datei, die Sie umbenennen wollen mit Bindestrich -, dann kennzeichnen Sie das Ende der Optionen mit --.

datei
Name der Datei, die Sie umbenennen wollen.
Wenn das übergeordnete Dateiverzeichnis von *datei* schreibbar ist, und das t-Bit (sticky-Bit) gesetzt hat, muß eine der folgenden Bedingungen erfüllt sein:

- 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

dateineu
Neuer Name der Datei, der sich von *datei* unterscheiden muß. Wenn schon eine Datei mit dem Namen *dateineu* existiert, wird sie mit dem Inhalt von *datei* überschrieben, wenn Sie für *dateineu* Schreibrecht haben (siehe Option *-i*).

Wenn Sie für eine bestehende *dateineu* kein Schreibrecht haben, so wird der Modus der Zugriffsberechtigung ausgegeben, und Sie werden zur Beantwortung einer Frage aufgefordert. Wenn Sie die Frage mit *y* bejahen, wird *mv* ausgeführt.
Ist die Option *-f* angegeben oder ist die Standard-Eingabe keine Datensichtstation, unterbleibt diese Frage.

Wenn *datei* eine einfache Datei ist und *dateineu* ein Verweis auf eine andere Datei mit Verweisen, so bleiben diese Verweise erhalten, und eine neue Datei mit dem Namen *dateineu* wird angelegt.

Format 2: Dateien in ein anderes Dateiverzeichnis versetzen

mv␣datei␣...␣dvz

datei
Namen der Dateien, die in das Dateiverzeichnis *dvz* übertragen werden soll. Ein Dateiverzeichnis kann nicht übertragen werden.

dvz
Name des Dateiverzeichnisses, in das die Datei übertragen werden soll. Sie brauchen für das Dateiverzeichnis Schreibrecht.

Format 3: Dateiverzeichnis umbenennen

mv␣dvz␣dvzneu

dvz
Name des Dateiverzeichnisses, das Sie umbenennen wollen.

dvzneu
Neuer Name des Dateiverzeichnisses. Falls es schon ein Dateiverzeichnis mit dem Namen *dvzneu* gibt, meldet *mv* einen Fehler. *dvz* und *dvzneu* müssen zum gleichen physikalischen Dateisystem gehören. Die den Dateiverzeichnissen direkt übergeordneten Dateiverzeichnisse .. müssen nicht identisch sein.

BEISPIELE

1. Die Datei *lieder* im aktuellen Dateiverzeichnis soll in *popsongs* umbenannt und in das Dateiverzeichnis */usr/petra/kunst/musik* übertragen werden.

```
$ mv lieder /usr/petra/kunst/musik/popsongs
```

2. Die Dateien *efeu*, *papyrus* und *flieder* im aktuellen Dateiverzeichnis sollen ihre Namen behalten und ins Dateiverzeichnis */usr/petra/pflanzen* übertragen werden.

```
$ mv efeu papyrus flieder /usr/petra/pflanzen
```

SIEHE AUCH

chmod, cp, cpio, find, ln, rm

newform Format einer Textdatei ändern (new format)

Mit *newform* können Sie Textdateien oder Text, den *newform* von der Standard-Eingabe liest, neu formatieren.
newform formatiert den Eingabetext zeilenweise entsprechend den angegebenen Optionen. Das Ergebnis schreibt *newform* auf die Standard-Ausgabe.

```
newform[␣option]...[␣datei]...
```

Keine Option angegeben
newform schreibt die eingelesenen Zeilen unverändert auf die Standard-Ausgabe.

option

Die Optionen müssen einzeln, von anderen Optionen durch Leerzeichen getrennt, eingegeben werden (siehe *BEISPIELE*).
Die Reihenfolge der Optionen ist entscheidend für die Wirkungsweise von *newform*. Die Optionen werden der Reihe nach abgearbeitet. Zum Beispiel liefert die Optionenfolge *-e15 -l60* ein anderes Ergebnis als *-l60 -e15*.

Mit Ausnahme von *-s* können Sie die Optionen mehrmals angeben.
Die Optionen dürfen auch zwischen oder nach den Dateinamen stehen; dies hat dieselbe Wirkung, wie wenn alle Optionen vor den Dateinamen stünden.
Alle angegebenen Optionen gelten für alle angegebenen Dateien.

Tabulatorzeichen in Leerzeichen umwandeln

-i[tabspec]
(i - input tab specification) Mit dieser Option können Sie Tabulatorzeichen in Leerzeichen umwandeln. Mit Hilfe von *tabspec* definieren Sie eine Liste von Tabulatorstops. *newform* wandelt dann jedes Tabulatorzeichen der Eingabezeile in so viele Leerzeichen um, daß das darauffolgende Zeichen auf einem Tabulatorstop steht (siehe *Beispiel 1*).

tabspec nicht angegeben:
tabspec hat den Standard-Wert -8; d.h. *newform* nimmt Tabulatorstops in den Spalten 1, 9, 17, 25 usw. an.

tabspec angegeben:
tabspec kann die folgenden Werte annehmen:

-n, **-0**;
n1,n2,...;
-a, **-a2**, **-c**, **-c2**, **-c3**, **-f**, **-p**, **-s**, **-u**;
--, **--**name.

-n
newform nimmt Tabulatorstops in den Spalten 1, 1+*n*, 1+2**n*, 1+3**n* usw. an. *n* ist eine ganze Zahl größer gleich 1.

-0
newform erwartet keine Tabulatorzeichen. Enthält die Eingabezeile dennoch Tabulatorzeichen, so werden diese Tabulatorzeichen in je ein Leerzeichen umgewandelt. Die Angabe von -i-0 liefert also das gleiche Ergebnis wie die Angabe von -i-1.

n1,n2,...
newform nimmt Tabulatorstops in den Spalten *n1, n2* usw. an.
n1, n2 usw. sind ganze Zahlen größer gleich 1. Schreiben Sie vor eine der Zahlen ein Pluszeichen +, wird der nachfolgende Wert zur zuletzt angegebenen Zahl addiert und die Summe als neuer Tabulatorstop eingesetzt. Die Zahlen müssen in aufsteigender Reihenfolge und durch Kommata getrennt angegeben werden. Sie können eine oder mehrere, höchstens aber 40 Zahlen angeben.

-a
entspricht der Tabulatorstopliste 1,10,16,36,72
(für Assembler, anwendbar bei einigen Großrechnern)

-a2
entspricht der Tabulatorstopliste 1,10,16,40,72
(für Assembler, anwendbar bei einigen Großrechnern)

-c
entspricht der Tabulatorstopliste 1,8,12,16,20,55
(für COBOL, normales Format)

-c2
entspricht der Tabulatorstopliste 1,6,10,14,49
(für COBOL, kompaktes Format)

-c3
entspricht der Tabulatorstopliste
1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
(für COBOL, kompaktes Format, mit mehr Tabulatorstops als bei *-c2*)

-f
entspricht der Tabulatorstopliste 1,7,11,15,19,23
(für FORTRAN)

-p
entspricht der Tabulatorstopliste
1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
(für PL/1)

-s
entspricht der Tabulatorstopliste 1,10,55
(für SNOBOL)

-u
entspricht der Tabulatorstopliste 1,12,20,44
(für Assembler, anwendbar bei einigen Großrechnern)

--
newform sucht die Tabulatorspezifikation in der ersten Zeile, die es von der Standard-Eingabe liest. Die Tabulatorspezifikation muß das folgende Format haben:

<:ttabspec **d:>**

--name
newform sucht die Tabulatorspezifikation in der ersten Zeile der Datei *name*. Die Tabulatorspezifikation muß das folgende Format haben:

<:ttabspec **d:>**

Leerzeichen in Tabulatorzeichen umwandeln

-o[tabspec]
(o - output tab specification) Mit dieser Option können Sie Leerzeichen in Tabulatorzeichen umwandeln. Mit Hilfe von *tabspec* definieren Sie eine Liste von Tabulatorstops. Stehen unmittelbar vor einem dieser Tabulatorstops ein oder mehrere Leerzeichen, dann wandelt *newform* diese Leerzeichenfolge in ein Tabulatorzeichen um (siehe *Beispiel 2*).

tabspec nicht angegeben:
tabspec hat den Standard-Wert -8; d.h. *newform* nimmt Tabulatorstops in den Spalten 1, 9, 17, 25 usw. an.

tabspec angegeben:
Für *tabspec* können Sie die gleichen Werte wie bei Option *-i* angeben. Geben Sie den Wert -0 an, dann wandelt *newform* keine Leerzeichen in Tabulatorzeichen um.

Tabulatorspezifikation ausgeben

-f
(f - format line) *newform* gibt vor allen anderen Ausgabezeilen die Tabulatorspezifikation aus.
Die Tabulatorspezifikation hat das folgende Format:

<:ttabspec **d:>**

Für *tabspec* gibt *newform* den Wert aus, den Sie im selben *newform*-Aufruf bei der letzten *-o*-Option angegeben haben. Haben Sie bei *-o* keinen Wert angegeben oder haben Sie überhaupt keine *-o*-Option angegeben, dann gibt *newform* für *tabspec*

den Standard-Wert -8 aus.

Vorsicht

Wenn Sie *newform* mit der Option *-f* aufrufen und im selben Aufruf für die letzte *-o*-Option *-o--* angegeben haben und wenn Sie vor dieser *-o*-Option *-o--* oder *-i--* angegeben haben, dann gibt *newform* eine falsche Tabulatorspezifikation aus.
Beispiel: newform ... -i-- ... -o-- -f datei

Erstes Feld ans Zeilenende setzen

-s

(s - shear off)
newform schneidet alle Zeichen der Eingabezeile bis zum ersten Tabulatorzeichen einschließlich ab.
Werden nur 8 Zeichen oder weniger abgeschnitten (das erste Tabulatorzeichen nicht mitgezählt), dann werden mit Ausnahme des ersten Tabulatorzeichens alle abgeschnittenen Zeichen an das Zeilenende gesetzt.
Werden mehr als 8 Zeichen abgeschnitten (das erste Tabulatorzeichen nicht mitgezählt), dann wird das achte Zeichen durch ein * ersetzt; die darauffolgenden Zeichen werden gelöscht. Das erste Tabulatorzeichen wird in jedem Fall gelöscht.

Zusammenwirken mit anderen Optionen:
Wird die Option *-s* abgearbeitet, dann schneidet *newform* die Zeichen am Beginn der Eingabezeile ab und speichert die abgeschnittenen Zeichen intern, bis alle restlichen Optionen auf die Eingabezeile angewandt worden sind. Erst dann setzt *newform* die Zeichen ans Ende der bearbeiteten Zeile.

Vorsicht

Wenn Sie die Option *-s* angeben, dann erwartet *newform*, daß alle Eingabezeilen ein Tabulatorzeichen enthalten.

Zeilen kürzen bzw. verlängern

-l[n]

(l - line length) Die effektive Zeilenlänge wird auf *n* Zeichen festgelegt. *n* ist eine ganze Zahl größer gleich 1.
Die Option *-l* ist nur zusammen mit Option *-b, -e, -p* oder *-a* sinnvoll.

n nicht angegeben:
Die effektive Zeilenlänge wird auf 72 Zeichen festgelegt.

-l[n] nicht angegeben:
Die effektive Zeilenlänge beträgt standardmäßig 80 Zeichen.

Vorsicht

Tabulator- und Rücksetz-Zeichen (tabs und backspaces) werden als jeweils ein Zeichen interpretiert werden; mit der Option *-i* können Sie die Tabulatorzeichen jedoch in Leerzeichen umwandeln.

-b[n]

(b - beginning of the line) Ist die Länge der Eingabezeile größer als die effektive Zeilenlänge (siehe Option *-l*), dann kürzt *newform* die Zeile um die ersten *n* Zeichen.
n ist eine ganze Zahl größer gleich 1.

n nicht angegeben:
newform kürzt die Zeile am Zeilenanfang gerade so weit, daß sie die effektive Zeilenlänge erreicht.

-e[n]

(e - end of the line) Ist die Länge der Eingabezeile größer als die effektive Zeilenlänge (siehe Option *-l*), dann kürzt *newform* die Zeile um die letzten *n* Zeichen.
n ist eine ganze Zahl größer gleich 1.

n nicht angegeben:
newform kürzt die Zeile am Zeilenende gerade so weit, daß sie die effektive Zeilenlänge erreicht.

-ck

(c - character) Als Füllzeichen wird *k* definiert. *k* ist ein beliebiges druckbares ASCII-Zeichen.
Die Option *-c* ist nur zusammen mit Option *-p* oder *-a* sinnvoll.

-c nicht angegeben:
Standardmäßig ist das Leerzeichen Füllzeichen.

-p[n]

(p - prefix) Ist die Länge der Eingabezeile kleiner als die effektive Zeilenlänge (siehe Option *-l*), dann stellt *newform* der Zeile *n* Füllzeichen (siehe Option *-c*) voran.
n ist eine ganze Zahl größer gleich 1.

n nicht angegeben:
newform stellt der Eingabezeile gerade so viele Füllzeichen voran, daß sie die effektive Zeilenlänge erreicht.

-a[n]

(a - append) Ist die Länge der Eingabezeile kleiner als die effektive Zeilenlänge (siehe Option *-l*), dann fügt *newform* am Zeilenende *n* Füllzeichen an (siehe Option *-c*).
n ist eine ganze Zahl größer gleich 1.

n nicht angegeben:
newform fügt am Ende der Eingabezeile gerade so viele Füllzeichen an, daß sie die effektive Zeilenlänge erreicht.

datei
Name der Textdatei, die neu formatiert werden soll.
Sie können mehrere Dateien angeben. Die angegebenen Optionen gelten dann für alle Dateien.

datei nicht angegeben:
newform liest von der Standard-Eingabe.

FEHLERMELDUNGEN

Alle hier beschriebenen Fehler führen zum Abbruch.

`not -s format`

Sie haben *newform* mit der Option *-s* aufgerufen, aber eine der Eingabezeilen enthielt kein Tabulatorzeichen.

`can't open datei`

Die Eingabedatei *datei* kann nicht geöffnet werden, da sie nicht existiert oder Sie auf die Datei nicht zugreifen dürfen etc.

`internal line too long`

Eine Zeile enthält nach ihrer Erweiterung im internen Arbeitspuffer mehr als 512 Zeichen.

`tabspec in error`

Das Argument tabspec, das Sie der Option *-i* oder *-o* mitgegeben haben, hatte nicht das richtige Format.

`tabspec indirection illegal`

Sie haben newform mit der Option *-i--*, *-o--*, *-i--datei* oder *-o--datei* aufgerufen, und die Tabulatorspezifikation, die *newform* daraufhin von der Standard-Eingabe oder aus *datei* gelesen hat, enthielt wiederum ein *tabspec* in Form von *-i--*, *-o--*, *-i--datei* oder *-o--datei*. Dies ist nicht zulässig.

BEISPIELE

Tabulatorzeichen in Leerzeichen umwandeln: Option -i

1. Die Datei *datei* hat folgenden Inhalt:

```
rot ⇥ gruen
```

Zwischen den Worten *rot* und *gruen* steht nur ein Tabulatorzeichen. Mit dem Kommando *od* können Sie den Dateiinhalt überprüfen - *od* macht auch das Tabulator- und das Neue-Zeile-Zeichen sichtbar:

```
$ od -c datei
0000000   r o t \t g r u e n \n
0000012
```

0000012 gibt oktal die Anzahl der in *datei* enthaltenen Zeichen an.

Vergleichen Sie die Ausgabe von *cat* und von *newform -i*:

```
$ cat datei
rot     gruen
$ newform -i datei
rot     gruen
```

In beiden Fällen beginnt das Wort *gruen* auf Spalte 9. *newform -i* läßt den Dateiinhalt jedoch nur scheinbar unverändert. Das Kommando *od* zeigt, daß es das Tabulatorzeichen in 5 Leerzeichen umwandelt:

```
$ newform -i datei | od -c
0000000   r o t           g r u e n \n
0000016
```

Geben Sie der Option *-i* ein Argument *tabspec* mit. Beachten Sie: Zwischen *-i* und dem Wert für *tabspec* darf kein Leerzeichen stehen.

```
$ newform -i-4 datei
rot gruen
```

Hier hat *newform* einen Tabulatorstop in Spalte 5 angenommen; folglich beginnt das Wort *gruen* auf Spalte 5. Das Tabulatorzeichen wurde in nur ein einziges Leerzeichen umgewandelt.

Definieren Sie eine Liste von Tabulatorstops. Dazu legen Sie eine Datei an, die in einer Zeile mehrere Tabulatorzeichen enthält:

```
$ cat > datei1
rot ⇥ gruen ⇥ blau ⇥ gelb
END
$ newform -i5,13,18 datei1
rot gruen   blau gelb
```

Hier hat *newform* Tabulatorstops in den Spalten 5, 13 und 18 angenommen. Folglich beginnt das Wort *gruen* auf Spalte 5, d.h. das erste Tabulatorzeichen wurde in 1 Leerzeichen umgewandelt; das Wort "blau" beginnt auf Spalte 13, d.h. das zweite Tabulatorzeichen wurde in 3 Leerzeichen umgewandelt usw.

Leerzeichen in Tabulatorzeichen umwandeln: Option -o

2. Die Datei *datei* hat folgenden Inhalt:

```
rot␣␣␣␣␣gruen
```

Zwischen den Worten *rot* und *gruen* stehen 5 Leerzeichen. Wandeln Sie die Leerzeichen in ein Tabulatorzeichen um:

```
$ newform -o datei | od -c
0000000   r o t \t g r u e n \n
0000012
```

newform hat in Spalte 9 einen Tabulatorstop angenommen und alle Leerzeichen, die unmittelbar vor Spalte 9 standen, in 1 Tabulatorzeichen umgewandelt.

Geben Sie nun für *tabspec* den Wert -6 an:

```
$ newform -o-6 datei | od -c
0000000   r o t \t     g r u e n \n
0000014
```

Hier hat *newform* alle Leerzeichen, die unmittelbar vor Spalte 7 standen, in 1 Tabulatorzeichen umgewandelt. Auf dieses Tabulatorzeichen folgen die restlichen zwei Leerzeichen.

Zeilen kürzen und verlängern

3. Die Datei *datei* hat folgenden Inhalt:

```
montag
dienstag
mittwoch
donnerstag
freitag
samstag
sonntag
```

Alle Zeilen aus *datei*, die länger als 6 Zeichen sind, sollen rechts soweit gekürzt werden, bis sie gerade 6 Zeichen lang sind:

```
$ newform -l6 -e datei
montag
dienst
mittwo
donner
freita
samsta
sonnta
```

Alle Zeilen aus *datei* sollen rechtsbündig formatiert werden; die Zeilen sollen auf Spalte 20 enden:

```
$ newform -l20 -p datei
              montag
            dienstag
            mittwoch
          donnerstag
             freitag
             samstag
             sonntag
```

Der folgende *newform*-Aufruf löscht die Folgenummern aus einem COBOL-Programm; die Folgenummern stehen in den Spalten 1-6.

```
$ newform -l1 -b6 programm
.
.
.
```

Die Option *-l1* ist notwendig, damit *jede* Zeile um 6 Zeichen gekürzt wird (siehe *Beispiel 5*).

Zeilenabschnitte umordnen: Option -s

4. Die Datei *datei* hat folgenden Inhalt:

```
Meier ⇥ Eduard
Mueller ⇥ Hugo

$ cat datei
Meier   Eduard
Mueller Hugo
```

Sie möchten nun die Zeilen so umstrukturieren, daß zuerst der Vorname kommt und dann, nach einem Leerzeichen, der Nachname.

```
$ newform -a1 -s datei
Eduard Meier
Hugo Mueller
```

Zunächst (Option *-a1*) wurde am Ende der Zeilen ein Leerzeichen angefügt. Anschließend (Option *-s*) wurden der Nachname und das Tabulatorzeichen abgeschnitten und der Nachname hinten angefügt.

In diesem Fall hätten Sie auch *newform -s -a1 datei* eingeben können; *newform* hätte dann zuerst Nachnamen und Tabulatorzeichen abgeschnitten, die Nachnamen intern gespeichert, ein Leerzeichen am Zeilenende angefügt und dann erst die Nachnamen angefügt (siehe *Beispiel 6*).

Reihenfolge der Optionen und mehrfache Angabe derselben Option

5. Die Datei *farben* hat folgenden Inhalt:

```
rot
giftgruen
gelb
lilablassblau
```

Kürzen Sie alle Zeilen, die länger als 4 Zeichen sind, am Ende um 2 Zeichen:

```
$ newform -l4 -e2 farben
rot
giftgru
gelb
lilablassbl
```

Vertauschen Sie die beiden Optionen, so erhalten Sie ein anderes Ergebnis:

```
$ newform -e2 -l4 farben
rot
giftgruen
gelb
lilablassblau
```

Hier wird die effektive Zeilenlänge zunächst auf den Standard-Wert 80 gesetzt; es werden also nur die Zeilen gekürzt, die länger als 80 Zeichen sind - das ist hier keine einzige. Dann erst erhält die effektive Zeilenlänge den Wert 4; das hat aber keine Auswirkung mehr.

Nun soll allen Zeilen der Datei *farben*, die höchstens 5 Zeichen enthalten, ein Stern vorangestellt werden. Anschließend soll in allen Zeilen, die länger als 10 Zeichen sind, das letzte Zeichen abgeschnitten werden.

```
$ newform -l5 -c* -p1 -l10 -e1 farben
*rot
giftgruen
*gelb
lilablassbla
```

6. Die Datei *bestellung* hat folgenden Inhalt:

```
1000 ⇥ Meier ⇥ Frankfurt
 150 ⇥ Mueller ⇥ Muenchen
 750 ⇥ Huber ⇥ Augsburg
2500 ⇥ Schmidt ⇥ Koeln ⇥ ***
```

Die Zeilen der Datei *bestellung* sollen nun folgendermaßen umformatiert werden: Die Zahlen und das erste Tabulatorzeichen sollen abgeschnitten werden, so daß die Namen am Anfang stehen. Die übrigen Tabulatorzeichen sollen in Leerzeichen umgewandelt werden, und zwar soll *newform* alle 15 Spalten einen Tabulatorstop annehmen (d.h. in den Spalten 16, 31, 46 usw.). Anschließend sollen alle Zeilen bis Spalte 30 mit Leerzeichen aufgefüllt bzw. ab Spalte 31 abgeschnitten werden. Die abgeschnittenen Zahlen sollen ab Spalte 31 stehen.
Dies leistet der folgende *newform*-Aufruf:

```
$ newform -s -i-15 -130 -a -e bestellung
Meier           Frankfurt       1000
Mueller         Muenchen         150
Huber           Augsburg         750
Schmidt         Koeln           2500
```

Beachten Sie, daß die Optionen von links nach rechts abgearbeitet werden: Nachdem die Zahlen und das Tabulatorzeichen abgeschnitten wurden (Option *-s*), beginnen die Namen auf Spalte 1. Nach der Umwandlung der restlichen Tabulatorzeichen in Leerzeichen (Option *-i-15*) sind nur die ersten 3 Zeilen kürzer als 30 Zeichen und werden folglich mit Leerzeichen aufgefüllt (Option *-a*); die letzte Zeile ist länger als 30 Zeichen und wird deshalb am Ende gekürzt (Option *-e*). Erst jetzt werden die abgeschnittenen Zahlen ans Ende der Zeilen gesetzt (Option *-s*, 2.Teil).

SIEHE AUCH

csplit, tabs
fspec [5]

newgrp
Gruppenzugehörigkeit ändern (new group)

Das in die Bourne-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.

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-Variable, 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. Ist für diese Gruppe in der Datei */etc/group* ein Kennwort vereinbart, erwartet *newgrp* die Eingabe dieses Kennwortes, bevor der Wechsel stattfindet.

In eine Gruppe, in der Sie kein Mitglied sind, können Sie nur wechseln, wenn für diese Gruppe ein Kennwort vereinbart ist. Andernfalls bricht *newgrp* mit einer Fehlermeldung ab.

```
newgrp [␣-] [␣gruppe]
```

Kein Argument angegeben
: Sie wechseln zurück in die Gruppe, deren Gruppen-Nummer (GID) in der Datei */etc/passwd* für Ihre Benutzerkennung eingetragen ist.

\-
: dürfen Sie nur angeben, wenn für Ihre Benutzerkennung in der Datei */etc/passwd* als Programm */bin/sh* eingetragen ist.
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.

- 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 in der Datei */etc/passwd* 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 in der Datei */etc/passwd* 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 Gruppe stattfindet.

Wenn Sie wieder in die Benutzergruppe wechseln wollen, die für Sie in der Datei */etc/passwd* eingetragen ist, rufen Sie *newgrp* ohne Angabe eines Gruppennamens auf.

gruppe nicht angegeben:
Sie wechseln zurück in die Gruppe, deren Gruppen-Nummer (GID) in der Datei */etc/passwd* für Ihre Benutzerkennung eingetragen ist.

Ein Kennwort für eine Gruppe einrichten

Damit Sie auch in eine Gruppe wechseln können, in der Sie nicht Mitglied sind, muß der Systemverwalter für diese Gruppe ein Kennwort einrichten.

Nur für den Systemverwalter

Für eine Gruppe können Sie ein Kennwort nur mit "Tricks" definieren, denn es gibt kein Kommando dafür. Sie haben zwei Möglichkeiten:

- Sie können ein verschlüsseltes Kennwort aus der Datei */etc/passwd* abschreiben, dessen Bedeutung Sie im Klartext kennen.

- Sie richten in der Datei */etc/passwd* eine "Dummy-Kennung" ein und definieren mit dem Kommando *passwd* ein Kennwort. Dieses verschlüsselte Kennwort schreiben Sie dann in die Datei */etc/group*.

ENDE-STATUS

immer 0

FEHLERMELDUNGEN

`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.

DATEIEN

/etc/group
legt für die in der Datei */etc/passwd* eingetragenen Gruppennummern einen Namen fest, sowie alle Mitglieder dieser Gruppe.

/etc/passwd
enthält alle eingerichteten Benutzerkennungen und Gruppennummern.

BEISPIELE

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, login, sh
group, passwd, environ [5]

news
Ausgabe von Nachrichten

news gibt Nachrichten aus dem Systemnachrichtenpool */var/news* aus, wobei die aktuelleren Nachrichten zuerst ausgegeben werden.

```
news[␣option]...[␣nachrichten]...
```

Keine Option angegeben
: *news* gibt die Inhalte der Dateien aus dem Dateiverzeichnis */var/news* aus, die aktueller sind als zum Zeitpunkt des letzten Aufrufs von *news*.

  Bei der Ausgabe werden die aktuelleren Nachrichten zuerst ausgegeben (nach dem *last in - first out* Prinzip).

  Jede Nachricht wird mit einem entsprechenden Kopfteil versehen.

  Beim erstmaligen Aufruf legt *news* eine Datei *.news_time* im HOME-Dateiverzeichnis (bestimmt durch die Variable $HOME) des aufrufenden Benutzers an. Bei zukünftigen Aufrufen wählt *news* anhand der Zugriffszeit dieser Datei die Nachrichten aus, die dem Benutzer angezeigt werden: Es werden nur die Nachrichten ausgegeben, die aktueller sind als die Zugriffszeit der Datei, also aktueller als zum letzten Aufruf von *news*.

  Bei jedem Aufruf von *news* wird die Zugriffszeit der Datei entsprechend der aktuellen Systemzeit modifiziert.

option

-a
: (a - all) *news* gibt, unabhängig von der Aktualität, alle Nachrichten aus. Ist diese Option eingeschaltet, wird die Zugriffszeit der Datei *.news_time* nicht verändert.

-n
: (n - name) *news* gibt nur die Namen der aktuellen Nachrichtendateien aus, nicht jedoch deren Inhalt. Die Zugriffszeit der Datei *.news_time* wird nicht verändert.

-s
: (s - statistics) *news* gibt die Anzahl der aktuellen Nachrichten aus, ohne jedoch deren Dateinamen oder Inhalte aufzulisten. Auch die Zugriffszeit der Datei *.news_time* wird nicht modifiziert.

  Eine sinnvolle Anwendung dieser Option ist es, einen solchen Aufruf von *news* in die *.profile* Datei eines jeden Benutzers einzutragen, oder sogar in */etc/profile*.

nachrichten

Mit *nachrichten* werden einzelne Dateien aus dem Dateiverzeichnis */var/news* spezifiziert, deren Inhalt angezeigt wird. Es können mehrere *nachrichten* angegeben werden.

nachrichten nicht angegeben:
news gibt die Inhalte der Dateien aus dem Dateiverzeichnis */var/news* aus, die aktueller sind als zum Zeitpunkt des letzten Aufrufs von *news*.

Arbeitsweise

news zeigt die einzelnen Nachrichten hintereinander am Bildschirm an.

Die Anzeige einer Nachricht kann mit DEL abgebrochen werden. *news* bricht daraufhin die Anzeige der aktuellen Nachricht ab, und zeigt die nächste Nachricht an. Zweimaliges Drücken von DEL kurz hintereinander bricht *news* komplett ab.

DATEIEN

/etc/profile
Wird von jeder Login-Shell ausgeführt. Wird vom Systemverwalter erstellt.

*/var/news/**
Nachrichtendateien

$HOME/.news_time
Informationsdatei für den letzten Aufruf von *news*.

SIEHE AUCH

profile, environ [5]

nice
Priorität von Kommandos ändern

Mit dem Kommando *nice* können Sie die Priorität, mit der ein Kommando ausgeführt wird, beeinflussen. Die Priorität eines Prozesses ist maßgeblich dafür, wann der Prozeß dem freiwerdenden Prozessor zugeteilt wird. Hohe Prioritätswerte entsprechen niedriger Priorität und umgekehrt. Der Prioritätswert wird durch verschiedene Parameter bestimmt:

- Wie lange ein Prozeß bereits dem Prozessor zugeteilt war.
- Wie lange ein Prozeß bereits auf den Prozessor gewartet hat.
- Der Wert, den Sie beim *nice*-Aufruf angeben.

Der aufrufende Prozeß muß in der Klasse der im Zeitscheibenverfahren abzuarbeitenden Prozesse (time-sharing class) eingetragen sein (siehe *priocntl*). Der aufrufende Prozeß ist normalerweise die Shell des Benutzers. Das Kommando *kommando* wird in der Klasse der im Zeitscheibenverfahren abzuarbeitenden Prozesse ausgeführt.

Wenn Sie nicht Systemverwalter sind, können Sie mit *nice* den Prioritätswert nur erhöhen, d.h. die Priorität des Prozesses erniedrigen (Sie sind "nice" zu anderen Benutzern).

Als Systemverwalter können Sie den Prioritätswert auch vermindern und damit die Priorität erhöhen.

```
nice [-zahl] kommando
```

zahl
: *zahl* ist eine ganze Zahl.

  Für normale Benutzer:
  : *zahl* liegt im Bereich von 1 bis 19. Der Prioritätswert wird vergrößert, das Kommando wird also mit niedrigerer Priorität ausgeführt. Alle größeren Angaben als 19 wirken wie 19.

  Nur für den Systemverwalter:
  : *zahl* liegt im Bereich von -20 bis 19. Negative Angaben für *zahl* bewirken eine Verkleinerung des Prioritätswertes und damit eine Erhöhung der Priorität. Alle kleineren Angaben als -20 wirken wie -20, alle größeren Angaben als 19 wirken wie 19.

Das Kommando *ps -l* gibt unter anderem die Spalte NI aus. Dort erscheint ein Zahlenwert zwischen 0 und 39: 0 entspricht *nice --20*
19 entspricht *nice -39*

zahl nicht angegeben:
für *zahl* wird 10 angenommen

kommando
Beliebiges Kommando oder Shell-Prozedur.

ENDE-STATUS

nice liefert den Ende-Status des ausgeführten Kommandos *kommando* zurück.

BEISPIELE

1. Das Kommando *ps -ale* mit verminderter Priorität ablaufen lassen:

```
$ nice -15 ps -ale
```

2. Der Systemverwalter läßt das gleiche Kommando mit erhöhter Priorität ablaufen:

```
$ nice --5 ps -ale
```

SIEHE AUCH

ps, nohup, priocntl
nice() [14]

nl
Textzeilen numerieren (number lines)

nl liest Zeilen aus einer Datei oder von der Standard-Eingabe und schreibt sie numeriert 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. Zu Beginn einer logischen Seite wird der Zeilenzähler zurückgesetzt (Ausnahme: Option *-p*). Für die Zeilennumerierung 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 numeriert werden sollen.

Der Beginn des Kopf-, Haupt- und Fußteils einer logischen Seite wird normalerweise durch eine Eingabezeile angezeigt, die ausschließlich eine der folgenden Zeichenketten enthält:

Zeile	Beginn von
\:\:\:	Kopfteil
\:\:	Hauptteil
\:	Fußteil

Sind im Eingabetext keine Begrenzungszeichenketten enthalten, dann interpretiert *nl* den Text vollständig als Hauptteil einer logischen Seite.

```
nl[␣option]  [␣datei]
```

Keine Option angegeben

nl numeriert im Hauptteil alle Zeilen, die druckbare Zeichen enthalten; die Zeilen im Kopf- und Fußteil werden nicht numeriert.
Zu Beginn einer logischen Seite wird der Zeilenzähler auf 1 gesetzt.
nl numeriert 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 numeriert werden sollen

-b[typ]
(b - body) Mit dieser Option geben Sie an, welche Zeilen im Hauptteil einer logischen Seite numeriert werden sollen.

typ kann sein: **a**, **n**, pregulärer_ausdruck oder **t**.

a
(a - all lines) Alle Zeilen werden numeriert.

n
(n - no lines) Keine Zeile wird numeriert.

pregulärer_ausdruck
Alle Zeilen werden numeriert, die eine zu *regulärer_ausdruck* passende Zeichenfolge enthalten.
regulärer_ausdruck ist ein einfacher (nicht internationalisierter) 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 numeriert.

typ nicht angegeben:
typ = *t*, d.h. nur die Zeilen werden numeriert, die druckbare Zeichen enthalten.

-b nicht angegeben:
Nur die Zeilen werden numeriert, die druckbare Zeichen enthalten. (D.h. Option *-b* mit Argument *t* ist der Standard-Fall.)

-h[typ]
(h - header) Wie *-b[typ]*, nur für den Kopfteil.

typ nicht angegeben:
typ = *n*, d.h. die Zeilen im Kopfteil werden nicht numeriert.

-h nicht angegeben:
Die Zeilen im Kopfteil werden nicht numeriert. (D.h. Option *-h* mit Argument *n* ist der Standard-Fall.)

-f[typ]
(f - footer) Wie *-b[typ]*, nur für den Fußteil.

typ nicht angegeben:
typ = *n*, d.h. die Zeilen im Fußteil werden nicht numeriert.

Option *-f* nicht angegeben:
Die Zeilen im Fußteil werden nicht numeriert. (D.h. Option *-f* mit Argument *n* ist der Standard-Fall.)

Zeilenzähler zurücksetzen / nicht zurücksetzen

-p
Der Zeilenzähler wird zu Beginn einer logischen Seite nicht zurückgesetzt.

-vstartnummer
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

-idiff
Die Differenz zwischen zwei Zeilennummern ist *diff*.

-i nicht angegeben:
Die Differenz ist 1.

Ausgabeformat festlegen

-nformat
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.

trenner nicht angegeben:
Der Text folgt direkt auf die Zeilennummer.

-s nicht angegeben:
Zeilennummer und Text werden durch ein Tabulatorzeichen voneinander getrennt.

-wn
Die Zeilennummern haben bis zu *n* Stellen.

-w nicht angegeben:
Die Zeilennummern haben bis zu 6 Stellen.

Numerierung von Leerzeilen festlegen

-ln
nl interpretiert *n* aufeinanderfolgende Leerzeilen als eine einzige Leerzeile.

Beispiel

```
$ nl -ba -l2
```

bewirkt, daß im Hauptteil bei mehreren aufeinanderfolgenden Leerzeilen nur jede zweite Leerzeile numeriert wird (im Kopf- und Fußteil werden überhaupt keine Zeilen numeriert).

-l nicht angegeben:
Jede Leerzeile wird als vollständige Zeile interpretiert (n = 1).

Trennzeichen von Kopf-, Haupt- und Fußteil definieren

-dx[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.

BEISPIELE

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
```

Alle Zeilen im Hauptteil numerieren

```
$ nl -ba gedichte
```

```
      Das aesthetische Wiesel

 1        Ein Wiesel
 2   sass auf einem Kiesel
 3   inmitten Bachgeriesel.
 4
 5        Wisst ihr,
 6         weshalb?
 7
 8       Das Mondkalb
 9      verriet es mir
10       im stillen:
11
12      Das raffinier-
13         te Tier
14   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
```

Zeilenzähler nicht zurücksetzen

```
$ nl -p 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

12  Eine Frau aus Clausthal-Zellerfeld,
13  die taeglich in den Keller faellt,
14  haelt den Rekord
15  in diesem Sport,
16  weil sie von Tag zu Tag schneller faellt.

      Verfasser unbekannt
```

In Zehner-Schritten numerieren

```
$ nl -v10 -i10 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*

nohup
Kommando ausführen und dabei Signale ignorieren (no hangup)

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.

```
nohup kommando
```

kommando
: 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. Andernfalls wird die Ausgabe hinten angehängt.

FEHLERMELDUNG

```
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.

DATEIEN

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).

UMGEBUNGSVARIABLE

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.

BEISPIEL

Sie starten folgenden Auftrag:

```
$ 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() [14]

notify
Meldung über die Ankunft neuer Post

notify gibt dem Benutzer Meldung über die Ankunft neuer Nachrichten. Bei den Nachrichten handelt es sich um elektronische Post, die mit dem *mail* Kommando versendet wurde.

Bei der Ankunft neuer Nachrichten überprüft *mail*, ob diese zu einem anderen Empfänger oder als Eingabe für ein Kommando weitergeleitet werden sollen. Mit *notify* wird die Weiterleitung zum Briefkasten des Benutzers eingerichtet.

```
notify␣-y[␣-m␣datei]                                    Format 1
notify[␣-n]                                             Format 2
```

Keine Option angegeben
: *notify* informiert über den aktuellen Status des automatischen Nachrichten-Meldewesens. Es gibt an, ob es aktiviert ist oder nicht.

option

-m␣postdatei
: (m - mailfile) Mit dieser Option kann die Datei angegeben werden, in der die Nachrichten abgespeichert werden, wenn das automatische Nachrichten-Meldewesen aktiviert ist.

  Option nicht angegeben:
  Die voreingestellte Datei für die Abspeicherung der Nachrichten ist *$HOME/.mailfile*.

-n
: Löschen der Einrichtungen des Nachrichten-Meldewesens.

-y
: Installation des Nachrichten-Meldeswesens.

Arbeitsweise

Die Meldung über eingehende Nachrichten funktioniert so, daß zuerst in der Systemdatei */var/adm/utmp* nachgeschaut wird, ob der Empfänger am System angemeldet ist. Wenn ja, wird das entsprechende Bildschirmterminal, dessen Geräteeintrag ebenfalls der Datei */var/adm/utmp* entnommen wird, zum Schreiben geöffnet. Jetzt wird der Benutzer über die erhaltene Nachricht informiert. Die Meldung beinhaltet Informationen über den Sender der Nachricht. Falls die Nachricht eine "Subject:"-Kopfzeile enthält, so wird auch diese mit angegeben (aus Sicherheitsgründen werden alle undruckbaren Zeichen im Kopf zu Ausrufezeichen konvertiert).

Falls der Benutzer mehr als einmal am System angemeldet ist, erhält sie oder er auch mehrere Meldungen, je eine pro Bildschirm. Um die Meldung für eine bestimmte Login-Sitzung zu sperren, kann das *mesg* Kommando angewendet werden. Es sperrt die Schreibberechtigung für das Bildschirmgerät.

Falls mehrere Rechner über RFS (Remote File Sharing) oder NFS (Network File System) verbunden sind, überprüft *notify* die */var/adm/utmp* Dateien aller angeschlossenen Rechner.

Falls es zu Problemen beim Zuliefern der Post an die entsprechende Briefkastendatei kommt, schaut *notify* das Dateiverzeichnis der Briefkastendatei in der Systemdatei */etc/mail/notify.fsys* nach. Falls das Dateiverzeichnis in der ersten Spalte der Datei gefunden wird, wird die Nachricht an das System weitergeleitet, das in der zweiten Spalte der Datei aufgelistet ist. Ansonsten würde die Nachricht an den Absender zurückgeschickt.

DATEIEN

*/tmp/notif**
Temporäre Datei.

*/var/mail/**
Standardbriefkasten der Benutzer.

/usr/lib/mail/notify2
Programm, das die Ankunft neuer Post meldet.

/etc/mail/notify.fsys
Liste der Dateisysteme.

/var/adm/utmp
Liste der angemeldeten Benutzer.

SIEHE AUCH

mail, *mesg*

od
Inhalt einer Datei oktal ausgeben (octal dump)

od schreibt den Inhalt einer Datei auf die Standard-Ausgabe, wobei Sie für die Ausgabe über Optionen ein oder mehrere Ausgabe-Formate angeben können.

Die erste Spalte jeder Ausgabe-Zeile gibt die Position des ersten in dieser Zeile enthaltenen Zeichens in der Datei an. Je nachdem welches Ausgabeformat Sie gewählt haben, ist diese Angabe oktal, dezimal oder hexadezimal.

```
od[␣option][␣datei][␣[+]offset]
```

Keine Option angegeben
2-byte Worte werden als vorzeichenlose Oktalzahlen 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
Bytes werden als Oktalzahlen interpretiert.

-c
Bytes werden als ASCII-Zeichen interpretiert. Einige nicht-druckbare Zeichen werden gemäß den in der Sprache "C" gültigen Konventionen als Fluchtsymbol-Folgen ausgegeben:

```
Null-Byte (alle Bits sind 0)              \0
Rücksetzzeichen (backspace)               \b
Seitenvorschub-Zeichen (form feed)        \f
Neue-Zeile-Zeichen (newline)              \n
Wagenrücklauf-Zeichen (carriage return)   \r
Tabulatorzeichen (horizontal tabulation)  \t
```

Die übrigen nicht-druckbaren Zeichen werden als dreistellige Oktalzahlen ausgegeben.

-d
2-byte Worte werden als vorzeichenlose Dezimalzahlen interpretiert.

-D
4-byte Worte werden als vorzeichenlose Dezimalzahlen interpretiert.

-f
4-byte Worte werden als Gleitkommazahlen interpretiert.

-F
8-byte Worte werden als Zahlen mit erweiterter Genauigkeit interpretiert.

-o
2-byte Worte werden als vorzeichenlose Oktalzahlen interpretiert.

-O
4-byte Worte werden als vorzeichenlose Oktalzahlen interpretiert.

-s
2-byte Worte werden als Dezimalzahlen mit Vorzeichen interpretiert.

-S
4-byte Worte werden als Dezimalzahlen mit Vorzeichen interpretiert.

-v
(v - verbose) Alle Daten werden angezeigt.

-x
2-byte Worte werden als vorzeichenlose Hexadezimalzahlen interpretiert.

-X
4-byte Worte werden als vorzeichenlose Hexadezimalzahlen 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.

BEISPIELE

1. Inhalt der Datei *text* oktal ohne Vorzeichen ausgeben:

```
$ cat text
Noch kann man alles verstehen.
$ od text
0000000 067516 064143 065440 067141 020156 060555 020156 066141
0000020 062554 020163 062566 071562 062564 062550 027156 000012
0000037
```

2. Inhalt der Datei *text* vom sechsten Byte an oktal und als ASCII-Zeichen ausgeben:

```
$ od -bc text 5.
0000005 153 141 156 156 040 155 141 156 040 141 154 154 145 163 040 166
          k   a   n   n       m   a   n       a   l   l   e   s       v
0000021 145 162 163 164 145 150 145 156 056 012
          e   r   s   t   e   h   e   n   .  \n
0000031
```

SIEHE AUCH

Tabellen und Verzeichnisse, Zeichensatz ISO 646

pack
Dateien komprimieren

Mit *pack* können Sie Dateien komprimieren, eine Komprimierung ist jedoch nicht in jedem Falle möglich (siehe *FEHLERMELDUNGEN*).

Vorsicht

Komprimierte Dateien sind nicht unbedingt auf andere Systeme portierbar.

Der Umfang der Komprimierung hängt von der Größe der Eingabedatei und der Verteilung der Häufigkeit des Auftretens einzelner Zeichen ab. Da der erste Teil einer komprimierten Datei aus Codierungsregeln bestehen kann, lohnt es sich gewöhnlich nicht, kleine Dateien zu komprimieren, es sei denn, die Datei hat eine sehr ungleichmäßige Zeichendichte (z.B. wenn sie Graphiken oder Abbildungen enthält).

Die Dateien werden in der Regel um 35-40% komprimiert.
Die komprimierten Versionen von Objektdateien, bei denen ein umfangreicherer Zeichensatz verwendet wird und die eine gleichmäßigere Zeichendichte aufweisen, sind lediglich um 10% kleiner als die Originaldateien.

Mit *unpack* können Sie den Originalzustand einer komprimierten Datei wiederherstellen. Mit *pcat* können Sie komprimierte Dateien im Originalzustand auf die Standard-Ausgabe ausgeben.

```
pack[␣-][␣-f]␣datei␣...
```

-

Auf der Standard-Ausgabe werden ausführliche statistische Informationen über die erfolgte Komprimierung ausgegeben.

Die Option - wirkt als Umschalter: Wenn Sie beim Aufruf von *pack* mehrere Dateien angeben, können Sie die Option - vor jeder Datei setzen. Jedes zweite Auftreten der Option schaltet dabei die Ausgabe ausführlicher statistischer Informationen über die Komprimierung nachfolgend angegebener Dateien wieder aus.

Beispiel

```
$ pack - datei1 datei2 - datei3 - datei4
  pack: datei1: 40.6% Compression
          from 31960 to 18984 bytes
          Huffman tree has 15 levels below root
          89 distinct bytes in input
          dictionary overhead = 111 bytes
          effective  entropy  = 4.75 bits/byte
          asymptotic entropy  = 4.72 bits/byte
  pack: datei2: 41.7% Compression
          from 268716 to 156692 bytes
          Huffman tree has 14 levels below root
          94 distinct bytes in input
          dictionary overhead = 115 bytes
          effective  entropy  = 4.66 bits/byte
          asymptotic entropy  = 4.66 bits/byte
  pack: datei3: 40.6% Compression
  pack: datei4: 41.6% Compression
          from 249569 to 145840 bytes
          Huffman tree has 14 levels below root
          94 distinct bytes in input
          dictionary overhead = 115 bytes
          effective  entropy  = 4.67 bits/byte
          asymptotic entropy  = 4.67 bits/byte
```

Für *datei1* und *datei2* werden ausführliche statistische Informationen über die Komprimierung ausgegeben, da die Option – gesetzt ist. Für *datei3* wird Ausgabe ausführlicherer Informationen wieder ausgeschaltet, da - ein zweites Mal angegeben wird, und für *datei4* werden schließlich wieder ausführlichere Informationen ausgegeben.

– nicht angegeben:
Es wird lediglich der Prozentsatz der Komprimierung ausgegeben.

-f

Die Komprimierung von *datei* wird erzwungen; auf diese Weise kann *pack* auf ein ganzes Dateiverzeichnis angewandt werden, selbst wenn bei einigen der darin enthaltenen Dateien kein Speicherplatz gespart wird (siehe *FEHLERMELDUNGEN*).

-f nicht angegeben:
Es werden lediglich die Dateien komprimiert, bei denen dadurch Speicherplatz gespart werden kann.

datei

Name der Datei, die komprimiert werden soll. Sie können mehrere Dateien angeben.

datei darf kein Dateiverzeichnis sein. Die komprimierte Datei erhält den Namen *datei.z*, *datei* wird nach erfolgreicher Komprimierung gelöscht. *datei.z* hat dieselben Zugriffsrechte, dasselbe Änderungsdatum und denselben Eigentümer wie *datei*. *datei* darf höchstens 12 Zeichen lang sein, damit die Namenserweiterung auf *datei.z* noch möglich ist.

FEHLERMELDUNGEN

Bei allen im folgenden beschriebenen Fehlern wird das Kommando *pack* nicht ausgeführt, d.h., die Datei bleibt in ihrem Originalzustand und es wird keine komprimierte Datei angelegt.

```
pack: ganz_langer_dateiname: file name too long
```

Der Name der zu komprimierenden Datei darf aus höchstens 12 Zeichen bestehen.

```
pack: datei.z: already packed
```

Eine bereits mit *pack* komprimierte Datei *datei.z* kann nicht noch einmal komprimiert werden.

```
pack: datei: has links
```

Auf Dateien, die mit *pack* komprimiert werden sollen, dürfen keine Verweise eingetragen sein.

```
pack: dir: cannot pack a directory
```

Sie dürfen das Kommando *pack* nicht auf Dateiverzeichnisse anwenden.

```
pack: datei: cannot open
```

Sie haben kein Leserecht für die zu komprimierende Datei oder *datei* existiert nicht.

```
pack: datei.z: already exists
```

Im aktuellen Dateiverzeichnis gibt es bereits eine Datei mit dem Namen *datei.z*. Es spielt keine Rolle, ob diese Datei tatsächlich das Ergebnis eines *pack*-Aufrufs ist.

```
pack: datei.z: cannot create
```

Die komprimierte Datei kann nicht angelegt werden, weil Sie z.B. kein Schreibrecht für das aktuelle Dateiverzeichnis haben.

```
pack: datei: no saving - file unchanged
```

Eine Komprimierung von *datei* würde keine Einsparung an Speicherplatz bringen. Sie können die Komprimierung erzwingen, wenn Sie die Option *-f* angeben. Diese Fehlermeldung wird auch dann ausgegeben, wenn *datei* das umbenannte Ergebnis eines *pack*-Aufrufs ist.

```
pack: datei: read error pack: datei.z: write error
```

Beim Lesen der Originaldatei oder beim Schreiben der komprimierten Datei trat ein Fehler auf.

ENDE-STATUS

Der Ende-Status von *pack* ist die Anzahl der Dateien, für die das Kommando nicht erfolgreich ausgeführt werden konnte.

BEISPIEL

Die Datei *dat*, die in unkomprimiertem Zustand 5226 Byte belegt, wird komprimiert.

```
$ ls -l
  total 8
  -rw-------  1  fritz    gruppe1    5226 Aug 19 09:27 dat

$ pack dat
pack: dat: 37.8% Compression

$ ls -l
  total 8
  -rw-------  1  fritz    gruppe1    2473 Aug 19 09:27 dat
```

SIEHE AUCH

cat, *compress*, *pcat*, *uncompress*, *unpack*, *zcat*

page
Bildschirmausgabe steuern

Die Kommandos *page* und *more* erlauben das Durchblättern von Dateien an der Datensichtstation.
Während die Ausgabe mit *more* durch Hochschieben der Bildschirmzeilen realisiert wird, löscht *page* den Bildschirm, bevor es eine neue Seite ausgibt.

```
more[␣option][␣-zeilen][␣+zeilennummer][␣+/muster][␣datei]
```

Das Kommando *page* ist mit *more* weitgehend identisch.

passwd
Login-Kennwort und Kennwortattribute eintragen oder ändern (password)

passwd trägt für eine Benutzerkennung ein Kennwort in die Datei */etc/passwd* ein oder ändert ein vorhandenes Kennwort. Zusätzlich können bestimmte Attribute des Kennworts ausgegeben oder geändert werden.

Jeder Benutzer kann sich die Attribute seines Kennworts ausgeben lassen. Benutzer ohne Systemverwalterrechte können nur ihr eigenes Kennwort und dessen Attribute ändern. Der Systemverwalter kann die Kennwörter und deren Attribute für alle lokalen Benutzerkennungen ändern.

Vom Benutzer ohne Systemverwalterrechte erwartet *passwd* die Eingabe des alten Kennworts, falls vorhanden.
Dann fordert *passwd* die Eingabe des neuen Kennworts. Das neue Kennwort muß zweimal eingegeben werden, um Fehler zu vermeiden. Die eingegebenen Kennwörter werden am Bildschirm nicht angezeigt.

Wenn das alte Kennwort eingegeben wurde, überprüft *passwd*, ob die zeitlichen Bedingungen für Kennwörter erfüllt sind. Falls dies nicht der Fall ist, beendet sich *passwd* (vgl. *shadow*). Wenn für das Kennwort des Benutzers keine speziellen zeitlichen Bedingungen gesetzt wurden (Optionen *-n* und *-x*), werden die Standardwerte *MAXWEEK* und *MINWEEK* aus der Datei */etc/default/passwd* verwendet. Falls zeitliche Bedingungen gesetzt wurden, wird die Information darüber in der Datei */etc/shadow* nicht verändert.

Wenn die zeitlichen Bedingungen erfüllt sind, überprüft *passwd*, ob das neue Kennwort das korrekte Format hat.

Ein Kennwort

- muß mindestens *PASSLENGTH* Zeichen lang sein, wobei der Wert *PASSLENGTH* der Datei */etc/default/passwd* entnommen wird. *PASSLENGTH* hat mindestens den Wert 6. *passwd* wertet immer nur die ersten 8 Zeichen eines Kennworts aus, auch wenn PASSLENGTH größer als 8 ist.
- muß mindestens zwei Buchstaben und ein numerisches oder Sonderzeichen enthalten. Die Buchstaben können groß oder klein geschrieben sein.
- darf nicht gleich dem vorwärts- oder rückwärts geschriebenen Benutzernamen sein, oder aus einer zyklischen Vertauschung der Buchstaben des Benutzernamens bestehen. Bei diesen Vergleichen wird nicht zwischen Groß- und Kleinschreibung unterschieden.
- muß sich vom alten Kennwort um mindestens drei Zeichen unterscheiden. Bei diesem Vergleich wird nicht zwischen Groß- und Kleinschreibung unterschieden.

passwd wertet nur die ersten 8 Zeichen eines Kennwortes aus.

Der Systemverwalter braucht das alte Kennwort nicht einzugeben, und das neue Kennwort wird nicht auf das korrekte Format überprüft. Auch die zeitlichen Bedingungen für Kennwörter werden nicht überprüft. Der Systemverwalter kann ein leeres Kennwort erzeugen, indem er nur ↵ eingibt, wenn er nach dem neuen Kennwort gefragt wird. Im Gegensatz zu *passwd -d* erfolgt in diesem Fall eine Abfrage nach dem Kennwort.

```
passwd[ option][ benutzerkennung]
```

Keine Option angegeben
: Es wird das Kennwort der Benutzerkennung geändert, unter der Sie sich angemeldet haben. Wechseln Sie mit *su* in eine andere Benutzerkennung, gibt *passwd* die Fehlermeldung: `permission denied.`

option

-s
: (s - show) Die Attribute des Kennworts für die Kennung *name* werden in folgender Form angezeigt:

  Benutzerkennung Zustand MM/TT/JJ Minimum Maximum Warnung

  oder, falls keine zeitlichen Bedingungen eingestellt sind:

  Benutzerkennung Zustand

  Dabei bedeutet

  Benutzerkennung
  : Benutzerkennung, unter der sich der Benutzer angemeldet hat.

  Zustand
  : Der augenblickliche Zustand des Kennworts des Benutzers: *PS* bedeutet, daß die Benutzerkennung mit einem Kennwort versehen oder gesperrt ist; *LK* bedeutet, daß die Kennung gesperrt ist; *NP* bedeutet, daß kein Kennwort eingetragen ist.

  MM/TT/JJ
  : Datum der letzten Änderung des Kennworts für die *Benutzerkennung*. Diese Angaben werden in Greenwich Mean Time ausgedrückt und können daher um bis zu einen Tag von anderen Zeitzonen abweichen.

  Minimum
  : gibt an, wieviele Tage zwischen zwei Änderungen des Kennworts für die *Benutzerkennung* mindestens verstreichen müssen. Standard ist der Wert *MINWEEKS* aus der Datei */etc/default/passwd*.

Maximum
gibt an, wieviele Tage das Kennwort höchstens gelten soll. Standard ist der Wert *MAXWEEKS* aus der Datei */etc/default/passwd*.

Warnung
gibt an, wieviele Tage vor dem Auslaufen des Kennworts der Benutzer gewarnt werden soll.

Die folgenden Optionen können nur vom Systemverwalter benutzt werden:

-l
(l - lock) Das Kennwort für *benutzerkennung* wird gesperrt.

-d
(d - delete) Das Kennwort für *benutzerkennung* wird gelöscht. Es wird beim Anmelden kein Kennwort abgefragt.

Vorsicht
Wenn der Systemverwalter ein Kennwort für einen Benutzer mit der Option *-d* gelöscht hat und die zeitliche Überprüfung für diesen Benutzer eingeschaltet ist, kann der Benutzer kein neues Kennwort mehr eingeben. Dies gilt auch, wenn der Eintrag *PASSREQ* in der Datei */etc/login/default* den Wert *YES* hat. Der Benutzer hat daher kein Kennwort mehr. Deshalb sollten Sie die Option *-d* immer mit der Option *-f* anwenden. Damit wird der Benutzer gezwungen, sein Kennwort bei der nächsten Anmeldung zu ändern.

-n min
Zwischen zwei Änderungen des Kennworts müssen mindestens *min* Tage verstreichen. Ist *min* größer als *max*, kann der Benutzer das Kennwort nicht ändern. *-n* sollte immer zusammen mit der Option *-x* verwendet werden, es sei denn, *max* wurde auf den Wert -1 gesetzt (keine zeitliche Überprüfung). In diesem Fall muß *min* nicht gesetzt werden.

-x max
Das Kennwort für *benutzerkennung* ist höchstens *max* Tage gültig. Falls *max* den Wert -1 hat, wird die zeitliche Überprüfung für *benutzerkennung* sofort ausgeschaltet. Falls *max* den Wert 0 hat, wird die zeitliche Überprüfung ausgeschaltet und der Benutzer muß beim nächsten Anmelden sein Kennwort ändern.

-w warn
Der Benutzer wird *warn* Tage, bevor sein Kennwort ungültig wird, gewarnt.

-a
(a - all) Die Eigenschaften der Kennwörter werden für alle Einträge aufgelistet. Verwenden Sie *-a* nur zusammen mit *-s*. Die *benutzerkennung* muß in diesem Fall nicht angegeben werden.

-f
(f - force) Das Kennwort für *benutzerkennung* wird ungültig. Der Benutzer muß beim nächsten Anmelden das Kennwort ändern.

benutzerkennung
ist die Benutzerkennung, der das neue Kennwort zugeordnet werden soll. Dieser Operand ist nur für den Systemverwalter sinnvoll. Wenn Sie nicht der Systemverwalter sind, können Sie nur Ihre eigene Benutzerkennung angeben.

benutzerkennung nicht angegeben:
passwd ändert das Kennwort für die aktuelle Benutzerkennung.

ENDE-STATUS

0 bei Erfolg
1 Zugriff verweigert
2 Unzulässige Kombination der Optionen
3 Ein unerwarteter Fehler ist aufgetreten. Die Kennwortdatei wurde nicht verändert
4 Ein unerwarteter Fehler ist aufgetreten. Die Kennwortdatei wurde nicht gefunden.
5 Die Kennwortdatei wird gerade bearbeitet. Versuchen Sie es später noch einmal.
6 Für eine Option wurde ein unzulässiges Argument übergeben.

FEHLERMELDUNGEN

```
passwd: xxx does not exist.
```

Es gibt für diese Benutzerkennung keinen Eintrag in der Datei */etc/passwd*.

```
passwd: Permission denied.
```

Sie dürfen das Kennwort für die angegebene Benutzerkennung nicht ändern. Nur der Systemverwalter darf Kennwörter für anderere Benutzerkennungen als die eigene ändern.

```
Sorry.
```

Sie haben das angeforderte alte Kennwort falsch eingegeben.

```
Password unchanged.
```

Das Kennwort konnte nicht geändert werden.

`Password must contain at least two alphabetic characters and at least one numeric or special character.`

Das Kennwort muß mindestens ein nicht numerisches Zeichen enthalten.

`Password is too short - must be at least 6 characters.`

Sie werden aufgefordert, ein längeres Kennwort einzugeben.

`Too many failures - try later.`

Ihr Kennwort genügt den Anforderungen nicht. Geben Sie ein neues Kennwort ein.

`Password must differ by at least 3 positions.`

Ihr Kennwort muß sich an mindestens drei Stellen unterscheiden.

`password file busy - try again.`

Der Systemverwalter bearbeitet gerade die Datei */etc/passwd*. Um Inkonsistenzen zu vermeiden, müssen Sie warten, bis die Datei wieder freigegeben wird.

DATEIEN

/etc/passwd
Datei, in die die Kennwörter eingetragen werden.

/etc/shadow,
"versteckte" Kennwort-Datei. Enthält dieselben Einträge wie */etc/passwd*, aber verschlüsselte Kennwörter. Kein allgemeines Leserecht.

BEISPIEL

Der Benutzer *aladin*, der am Rechner *alibaba* arbeitet, möchte sein Kennwort *simsalabim* ändern. Das neue Kennwort soll *bimsalaba* heißen:

```
$ passwd
Changing local password for aladin on alibaba
Old password:           Blinde Eingabe von simsalabim
New password:           Blinde Eingabe von bimsalaba
Retype new password:    Nochmal blinde Eingabe von bimsalaba
```

SIEHE AUCH

login
crypt() [14]
useradd, usermod, userdel, id, passws, passmgmt, pwconf, shadow, su [5]

paste
Zeilen zusammenfügen

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.

```
paste[␣-dliste]␣datei␣...                    Format 1
paste␣-s[␣-dliste]␣datei␣...                 Format 2
```

Format 1: Die n-ten Zeilen von mehreren Dateien zusammenfügen

paste[␣-dliste]␣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.

-dliste
: (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[␣-dliste]␣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.

-dliste
(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.

ENDE-STATUS

nicht definiert

FEHLERMELDUNGEN

`paste: line too long`

Ausgabezeilen dürfen nicht länger als 511 Zeichen werden.

`paste: too many files - limit 12`

Es dürfen nicht mehr als 12 Eingabedateien angegeben werden
(außer bei der Option *-s*).

```
paste: no delimiters
```

Bei Option *-d* fehlt *liste*.

```
paste: cannot open datei
```

datei ist nicht vorhanden, oder der Benutzer hat kein Leserecht.

BEISPIELE

Beispiele zu Format 1

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
```

2. Das aktuelle Dateiverzeichnis enthält die folgenden Dateien:

```
$ ls
korr
namen
plan
probe
prog.c
tst
verwalt
witze
```

Das folgende Kommando numeriert 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* !

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.

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

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*

pcat
Komprimierte Dateien ausgeben

pcat gibt den Original-Zustand einer mit *pack* komprimierten Datei auf die Standard-Ausgabe aus. Die komprimierte Datei bleibt unverändert. In ihren Original-Zustand bringen können Sie eine mit *pack* komprimierte Datei mit dem Kommando *unpack*. Im Gegensatz zu *cat* ist *pcat* kein Filter, da es nicht von der Standard-Eingabe liest.

```
pcat 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 *pcat* nach der entsprechenden *.z*-Datei.

ENDE-STATUS

Der Ende-Status von *pcat* ist die Differenz zwischen der Anzahl der beim Aufruf angegebenen Dateien und der Anzahl der Dateien, deren Originalzustand ausgegeben wurde. In welchen Fällen der Originalzustand einer komprimierten Datei nicht mit *pcat* ausgegeben werden kann, erfahren Sie im Abschnitt *Fehlermeldungen*.

FEHLERMELDUNGEN

Bei allen im folgenden beschriebenen Fehlern wird das Kommando *pcat* nicht ausgeführt.

```
pcat: ganz_langer_dateiname: file name too long
```

Der Name der komprimierten Datei ohne *.z* darf höchstens 12 Zeichen lang sein.

```
pcat: datei: cannot open
```

Sie haben kein Leserecht für *datei*, oder *datei* existiert nicht.

```
pcat: unpack.z: not in packed format
```

unpack.z hat zwar das Suffix *.z*, ist aber nicht das Ergebnis eines *pack*-Aufrufs.

BEISPIELE

1. Die Datei *liesmich* wird mit *cat* ausgegeben, dann mit *pack* komprimiert und anschließend wird die komprimierte Datei mit *pcat* ausgegeben. Da die Datei *liesmich* so klein ist, daß eine Komprimierung nicht sinnvoll ist, wird *pack* mit der Option *-f* aufgerufen:

```
$ cat liesmich
Noch bin ich nicht komprimiert!

$ pack -f liesmich
pack: liesmich: -36.4% Compression

$ pcat liesmich.z
Noch bin ich nicht komprimiert!
```

2. Von der komprimierten Datei *komp.z* soll eine Kopie in Originalgröße mit dem Namen *dekomp* gemacht werden:

```
$ pcat komp >dekomp
```

SIEHE AUCH

cat, compress, pack, uncompress, unpack, zcat

pg
Dateien seitenweise ausgeben (page)

pg gibt Dateien auf die Standard-Ausgabe aus. Ist die Standard-Ausgabe dem Bildschirm zugeordnet, dann können Sie die Dateien seitenweise vorwärts und rückwärts durchblättern.

Nach jeder Seite gibt *pg* standardmäßig einen Doppelpunkt : als Bereitzeichen aus. Auf dieses Bereitzeichen antworten Sie mit einem Kommando. Vom jeweiligen Kommando hängt es ab, welche Seite *pg* als nächste anzeigt (siehe *KOMMANDOS WÄHREND DES ABLAUFS VON PG*).

pg gibt nach einem Tabulatorzeichen das folgende Zeichen auf der nächsten Tabulatorposition aus. Als Abstand zwischen zwei Tabulatorpositonen nimmt *pg* acht Zeichen an.

Ist die Standard-Ausgabe nicht dem Bildschirm zugeordnet, dann findet keine Seitenaufteilung statt und es können keine Kommandos eingegeben werden. *pg* verhält sich in diesem Fall so wie das Kommando *cat*. Wenn Sie allerdings mehrere Dateien angeben, stellt *pg* jeder Datei einen Kopfteil voran.

pg entnimmt die Eigenschaften der Datensichtstation der Datenbasis *terminfo*, wobei der Typ der Datensichtstation durch die Umgebungsvariable *TERM* festgelegt wird. Falls *TERM* nicht definiert ist, wird als Typ der Datensichtstation *dumb* angenommen.

```
pg[␣-zahl][␣-p␣zeichenkette][␣option] ... [␣datei] ...
```

Kein Argument angegeben
: *pg* liest von der Standard-Eingabe und gibt das Gelesene seitenweise auf der Standard-Ausgabe aus.

-zahl
: Sie legen fest, wieviele Zeilen eine Seite am Bildschirm haben soll. Bei aufeinanderfolgenden Seiten steht die letzte Zeile der Vorgängerseite am Anfang.
Ist Option *-f* nicht angegeben, macht *pg* einen Zeilenumbruch, wenn eine Zeile über den rechten Bildschirmrand hinausgeht. Jede Zeile am Bildschirm ist auch für *pg* eine Zeile.

  zahl
  : Anzahl der Zeilen.
  zahl=1: wie *zahl*=2; eine Seite besteht also immer aus mindestens zwei Zeilen.

Während des Ablaufs von *pg* ändern Sie die Größe einer Seite mit dem Kommando *w* (siehe *Weitere Kommandos*).

-zahl nicht angegeben:
Eine Seite am Bildschirm hat 23 Zeilen.

-p␣zeichenkette
(p - prompt) ändert das Bereitzeichen (bzw. die Bereitzeichenkette) für Kommandos (siehe *KOMMANDOS WÄHREND DES ABLAUFS VON PG*).

zeichenkette
ist die neue Bereitzeichenkette. Steht in *zeichenkette* die Folge *%d*, so enthält die Bereitzeichenkette, die *pg* ausgibt, statt *%d* die aktuelle Seitennummer. Dies gilt nur für das erste Auftreten von *%d*.
Die aktuelle Seite ist die Seite, deren erste Zeile am Bildschirm zu sehen ist.

-p␣zeichenkette nicht angegeben:
pg verwendet einen Doppelpunkt : als Bereitzeichen.

option

-c
(c - clear) *pg* löscht den Bildschirm, bevor es eine neue Seite ausgibt.

-c nicht angegeben:
pg schiebt die vorherige Seite zeilenweise nach oben, bis die erste Zeile der neuen Seite auf der zweiten Bildschirmzeile steht.

-e
pg zeigt am Ende einer Datei sofort die nächste Datei. Falls keine Datei mehr folgt, beendet sich *pg*.

-e nicht angegeben:
Nach der letzten Zeile einer Datei gibt *pg* nochmals das Bereitzeichen aus. Vor dem Bereitzeichen steht `(EOF)`.
Wenn Sie die Taste ↵ drücken, zeigt *pg* die nächste Datei an oder beendet sich.

-f
pg macht keinen Zeilenumbruch, wenn eine Zeile über den rechten Bildschirmrand hinausgeht.

Wenn der Bildschirm hardwaremäßig automatisch Zeilen umbricht, kann *-f* diesen Zeilenumbruch nicht verhindern. Trotzdem zählt *pg* die Zeilen, als ob kein Umbruch stattgefunden hätte: Der Anfang der betroffenen Seite ist am Bildschirm nicht mehr sichtbar.

-f nicht angegeben:
pg macht einen Zeilenumbruch, wenn eine Zeile über den rechten Bildschirmrand hinausgeht. Jede Zeile am Bildschirm ist auch für *pg* eine Zeile.

-n

(n - newline) *pg* führt ein Kommando sofort aus, wenn Sie nach dem Bereitzeichen den Buchstaben des Kommandos eingeben (siehe *KOMMANDOS WÄHREND DES ABLAUFS VON PG*). Kommt in einem Kommando kein Buchstabe vor, dann drücken Sie die Leertaste oder die Taste ↵.

-n nicht angegeben:
Sie müssen jedes Kommando mit der Taste ↵ abschließen.

-s

(s - standout) *pg* stellt alle Nachrichten und das Bereitzeichen in inverser Schrift dar.

+zahl

pg gibt die Datei erst ab einer angegebenen Zeile aus.

zahl
Nummer der Zeile, ab der *pg* ausgibt.

-r

(r - restricted) Shell-Kommandos sind nicht erlaubt. *pg* gibt eine Fehlermeldung aus, beendet sich jedoch nicht.

+/rA

pg beginnt mit der Ausgabe bei der Zeile, in der zum erstenmal eine Zeichenkette vorkommt, die zum regulären Ausdruck *rA* paßt. *rA* ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*)

Diese Option hat nur dann eine Wirkung, wenn die Datei mehr Zeilen hat als eine Seite am Bildschirm (siehe oben *-zahl*).

Sie können auch während des Ablaufs von *pg* nach Zeichenketten suchen (siehe *Kommandos zum Suchen von Zeichenketten*).

datei

Name der Datei, die Sie durchblättern wollen. Sie können mehrere Dateien angeben; in diesem Fall gibt *pg* vor jeder neuen Datei folgendes aus:
`(Next file: datei).`

Wenn Sie für *datei* einen Bindestrich - angeben, liest *pg* von der Standard-Eingabe.

Mit den Kommandos *n* und *p* können Sie während des Ablaufs von *pg* zwischen mehreren Dateien wechseln (siehe *Weitere Kommandos*).

datei nicht angegeben:
pg liest von der Standard-Eingabe.

KOMMANDOS WÄHREND DES ABLAUFS VON PG

Ein Kommando schreiben Sie in die letzte Zeile hinter das Bereitzeichen. Dann schließen Sie es mit der Taste ↵ ab, es sei denn, Sie haben durch die Option *-n* vereinbart, daß Kommandos sofort ausgeführt werden.

pg kann gerade eine Seite anzeigen oder auf die Eingabe eines Kommandos warten. Während diesen unterschiedlichen Zuständen reagiert *pg* auf die Tasten DEL oder CTRL \ unterschiedlich:

- *pg* beendet sich, wenn es auf die Eingabe eines Kommandos gewartet hatte.
- *pg* bricht die Anzeige einer Seite ab und gibt sofort das Bereitzeichen aus. Sie können neue Kommandos eingeben.

Wenn Sie *pg* als Kommando in einer Pipeline verwenden, dann wirken die Tasten DEL oder CTRL \ auch auf die anderen Kommandos dieser Pipeline.
Angenommen das Kommando *grep* schreibt in die Pipeline und *pg* liest aus der Pipeline. Wenn Sie die Taste DEL drücken, während *pg* eine Seite ausgibt, geschieht folgendes: *grep* wird abgebrochen, *pg* aber nicht. *pg* kann noch das aus der Pipeline lesen, was *grep* bis zu seinem Abbruch hineingeschrieben hat.

Kommandos zum Blättern

↵

␣
ohne Kommando blättert *pg* um eine Seite weiter.

zahl
pg blättert auf die Seite, die Sie bei *zahl* angeben. Dabei spielt es keine Rolle, ob diese Seite vor oder nach der aktuellen Seite liegt.

+[zahl]
pg blättert weiter.

zahl
Anzahl der Seiten, um die weitergeblättert wird. Maximal blättert *pg* bis zur letzten Seite.

zahl nicht angegeben:
pg blättert um eine Seite weiter.

-[zahl]
pg blättert zurück.

zahl
Anzahl der Seiten, um die zurückgeblättert wird. Maximal blättert *pg* bis zur ersten Seite zurück.

zahl nicht angegeben:
pg blättert um eine Seite zurück.

[zahl]l
(l - line) *pg* positioniert absolut auf Zeilen.

zahl
Die Nummer der Zeile, die in der ersten Bildschirmzeile stehen soll. Dabei spielt es keine Rolle, ob diese Zeile vor oder nach der aktuellen Seite liegt.

zahl nicht angegeben:
pg blättert um eine Zeile weiter.

+[zahl]l
pg blättert um Zeilen weiter.

zahl
Anzahl der Zeilen, um die *pg* weiterblättern soll.

zahl nicht angegeben:
pg blättert um eine Zeile weiter.

-[zahl]l
pg blättert um Zeilen zurück.

zahl
Anzahl der Zeilen, um die *pg* zurückblättern soll.

zahl nicht angegeben:
pg blättert eine Zeile zurück.

[+]d
[+]END
pg blättert um eine halbe Seite am Bildschirm nach vorn. + können Sie voranstellen, müssen es aber nicht.

-d
-END
pg blättert um eine halbe Seite zurück.

zahlf
zahl Bildschirmseiten werden übersprungen.

zahlz
Wie ↵. *zahl* gibt, falls angegeben, die neue Anzahl der Zeilen pro Bildschirmseite an.

.
CTRL l
pg wiederholt die Ausgabe der aktuellen Seite.

$
pg blättert auf die letzte Seite der aktuellen Datei. *$* sollte mit Vorsicht benutzt werden, wenn von Pipe gelesen wird.

Kommandos zum Suchen von Zeichenketten

[k]/rA/[ort]

pg sucht vorwärts nach einer zu *rA* passenden Zeichenkette. Die Stelle, an der die Zeichenkette vorkommt, wird am Bildschirm ausgegeben.
pg beginnt mit der Suche unmittelbar nach der aktuellen Seite. Läßt sich keine passende Zeichenkette finden, dann endet die Suche am Ende der aktuellen Datei.

k

pg sucht nach dem *k*-ten Auftreten einer passenden Zeichenkette.

k nicht angegeben:
pg beginnt bei der ersten passenden Zeichenkette mit der Ausgabe.

rA

Regulärer Ausdruck, der die gesuchte Zeichenkette beschreibt. *rA* ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*).

ort

Ort an dem die passende Zeichenkette am Bildschirm angezeigt wird. Sie können angeben:

m (m - middle) Die Zeichenkette wird in der Mitte des Bildschirms angezeigt.
b (b - bottom) Die Zeichenkette wird in der letzten Bildschirmzeile angezeigt.
t (t - top) Die Zeichenkette wird in der ersten Bildschirmzeile angezeigt.

ort nicht angegeben:
Die passende Zeichenkette wird in der ersten Bildschirmzeile ausgegeben.

[k]?rA?[ort]
[k]^rA^[ort]

pg sucht rückwärts nach einer zu *rA* passenden Zeichenkette. Die Stelle, an der die Zeichenkette vorkommt, wird am Bildschirm ausgegeben.
pg beginnt mit der Suche unmittelbar vor der aktuellen Seite. Läßt sich keine passende Zeichenkette finden, dann endet die Suche am Anfang der aktuellen Datei.

k

pg sucht nach dem *k*-ten Auftreten einer passenden Zeichenkette.

k nicht angegeben:
pg beginnt bei der ersten passenden Zeichenkette mit der Ausgabe.

rA

Regulärer Ausdruck, der die gesuchte Zeichenkette beschreibt. *rA* ist ein einfacher regulärer Ausdruck (siehe *Tabellen und Verzeichnisse, Reguläre Ausdrücke*).

ort
Ort an dem die passende Zeichenkette am Bildschirm angezeigt wird. Sie können angeben:

m (m - middle) Die Zeichenkette wird in der Mitte des Bildschirms angezeigt.
b (b - bottom) Die Zeichenkette wird in der letzten Bildschirmzeile angezeigt.
t (t - top) Die Zeichenkette wird in der ersten Bildschirmzeile angezeigt.

ort nicht angegeben:
Die passende Zeichenkette wird in der ersten Bildschirmzeile ausgegeben.

Weitere Kommandos

h
(h - help) *pg* gibt eine Kurzübersicht zu den vorhandenen Kommandos.

[i]n
(n - next) *pg* macht die *i*-te nachfolgende Datei aus der Aufrufzeile zur aktuellen Datei. Hat die Aufrufzeile nicht genügend Dateien, dann wird die letzte Datei die aktuelle Datei.

i nicht angegeben:
pg macht die nächste Datei aus der Aufrufzeile zur aktuellen Datei.

[i]p
(p - previous) *pg* macht die *i*-te vorhergehende Datei aus der Aufrufzeile zur aktuellen Datei. Hat die Aufrufzeile nicht genügend Dateien, dann wird die erste Datei die aktuelle Datei.

i nicht angegeben:
pg macht die vorhergehende Datei aus der Aufrufzeile zur aktuellen Datei.

iw
(w - window) Eine Seite am Bildschirm hat ab sofort *i* Zeilen.

Standardmäßig hat eine Seite am Bildschirm 23 Zeilen. Diese Einstellung können Sie auch beim Aufruf von *pg* ändern (siehe oben *-zahl*).

sdatei
(s - save) *pg* sichert die aktuelle Datei unter dem Namen, den Sie bei *datei* angeben.

!kommando
pg übergibt *kommando* an den Kommandointerpreter, mit dem die Umgebungsvariable SHELL belegt ist. Ist diese nicht definiert, dann übergibt *pg* an die Standard-Shell.
Dieses Kommando müssen Sie immer mit der Taste ↵ abschließen. Dies gilt auch dann, wenn Sie *pg* mit *-n* aufgerufen haben.

q
Q

(q - quit) *pg* wird beendet.

Am Ende der letzten Datei der Aufrufzeile wird *pg* auch durch ↵ beendet.

UMGEBUNGSVARIABLE

SHELL

Sie bestimmen mit dieser Umgebungsvariablen den Kommadointerpreter, dem *pg* während des Ablaufs Kommandos übergibt (siehe Kommando *!*).

BEISPIEL

Sie wollen sich in der Datei *text* die Stelle ansehen, an der das Wort *Beispiel* zum ersten Mal vorkommt:

```
$ pg +/Beispiel text
```

SIEHE AUCH

cat, *ed*, *grep*, *more*
terminfo [5]

pr
Dateien für Ausgabe aufbereiten (print)

pr teilt Dateien in Seiten auf und gibt diese auf die Standard-Ausgabe aus.

```
pr[␣option][␣datei]...
```

Keine Option angegeben
: Jede Seite hat einen Kopfteil mit vier Leerzeilen und einer Kopfzeile. Die Kopfzeile enthält das Datum und die Uhrzeit der letzten Dateiänderung, den Dateinamen und die Seitennummer.
Am Ende hat jede Seite einen Fußteil mit fünf Leerzeilen. Die letzte Seite wird bis zum Seitenende aufgefüllt. Wenn die Standardausgabe auf eine Datensichtstation geleitet wird, werden andere Ausgaben auf diese Datensichtstation so lange verzögert, bis die Standardausgabe beendet ist. Damit wird verhindert, daß Fehlermeldungen mit der Ausgabe vermischt werden.

option

Die folgende Übersicht sagt Ihnen, mit welcher Option Sie die Ausgabe in welcher Weise beeinflussen:

Aufgabe	Option
Anfangsseite k festlegen	+k
Aufteilen des Textes auf k Spalten	-k
Festlegen der Auffüllreihenfolge bei Spalten	-a
Festlegen der Gesamtbreite bei Spalten	-w
Abschneiden von Zeilen bei Spalten verhindern	-s
Dateien in Spalten nebeneinander ausgeben	-m
Verdoppeln des Zeilenabstandes	-d
Umwandeln von Tabulatorzeichen in Leerzeichen	-e
Umwandeln von Leerzeichen in Tabulatorzeichen	-i
Durchnumerieren der Zeilen	-n
Text nach rechts einrücken	-o
Ändern der Seitenlänge	-l
Ändern des Dateinamens in der Kopfzeile	-h
Seitenweise Ausgabe auf dem Bildschirm	-p
Seitenvorschub statt Leerzeilen am Seitenende	-f
Zeilen nicht abschneiden	-F
Unterdrücken von Meldungen	-r
Unterdrücken von Kopf- und Fußteil einer Seite	-t

Mehrere Optionen können Sie ohne Leerzeichen hintereinander schreiben. Der Bindestrich - steht dann nur vor der ersten Option.
Die Reihenfolge der Optionen spielt keine Rolle.

+k
Für *k* geben Sie eine Seitennummer an. Ab dieser Seite beginnt *pr* mit der Ausgabe.

+k nicht angegeben:
Die Ausgabe beginnt mit der ersten Seite.

-k
pr gibt die Datei in Spalten aus. Für *k* geben Sie die Anzahl der Spalten an.

Die Spalten einer Seite füllt *pr* nacheinander von oben nach unten. Diese Einstellung können Sie mit der Option *-a* ändern.

Die Breite einer Seite ist bei mehrspaltiger Ausgabe 72 Zeichen. Diese Einstellung können Sie mit der Option *-w* ändern.

Die Breite einer Spalte errechnet *pr*, indem es die Seitenbreite durch die Spaltenanzahl dividiert. Ist eine Zeile zu lang, dann schneidet *pr* diese rechts ab. Sie verhindern das Abschneiden mit der Option *-s*.

-k nicht angegeben:
Die Ausgabe einer Datei ist einspaltig.

-a
pr füllt die Spalten einer Seite von links nach rechts. Die Option *-a* ist nur zusammen mit der Option *-k* sinnvoll.

-a nicht angegeben:
pr füllt die Spalten von oben nach unten.

-wk
legt bei spaltenweiser Ausgabe die Breite einer Seite fest. *-w* ist nur zusammen mit den Optionen *-k* oder *-m* sinnvoll.

k
Seitenbreite in Zeichen.

-w nicht angegeben:
Die Seitenbreite bei spaltenweiser Ausgabe ist 72.

-s[c]
Die Option verhindert bei spaltenweiser Ausgabe, daß *pr* überlange Zeilen rechts abschneidet.

c

Zeichen, das zwei Spalten trennt.

c nicht angegeben:
Das Trennzeichen ist das Horizontal-Tabulatorzeichen (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*).

-m

pr gibt alle angegebenen Dateien in Spalten nebeneinander aus. Es gelten dieselben Regeln, wie bei der Option *-k* (eine Datei mehrspaltig ausgeben). Diese Option darf nicht gemeinsam mit *-k* verwendet werden.

-m nicht angegeben:
pr gibt mehrere Dateien hintereinander aus.

-d

pr gibt nach jeder Zeile eine Leerzeile aus.

-e[c][k]

pr ersetzt jedes Tabulatorzeichen durch so viele Leerzeichen, daß das nachfolgende Zeichen auf einer Tabulatorposition steht.

c

Zeichen, das *pr* als Tabulatorzeichen interpretiert. Sie können ein beliebiges nicht-numerisches Zeichen angeben.

c nicht angegeben:
pr interpretiert das Horizontal-Tabulatorzeichen als Tabulatorzeichen (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*).

k

Abstand zwischen zwei Tabulatorpositionen. Die erste Tabulatorposition einer Zeile ist immer die erste Spalte. Wenn Sie 0 für *k* angeben, ist der Abstand 8.

k nicht angegeben:
Der Abstand zwischen zwei Tabulatorpositionen ist 8.

-i[c][k]

pr ersetzt Leerzeichen zwischen zwei Tabulatorpositionen dann durch ein Tabulatorzeichen, wenn die Leerzeichen aufeinanderfolgen und das letzte Leerzeichen vor einer Tabulatorposition steht.

c

Zeichen, das *pr* als Tabulatorzeichen setzt. Sie können ein beliebiges nicht-numerisches Zeichen angeben.

c nicht angegeben:
pr setzt das Horizontal-Tabulatorzeichen als Tabulatorzeichen (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*).

k

Abstand zwischen zwei Tabulatorpositionen. Die erste Tabulatorposition einer Zeile ist immer die erste Spalte. Wenn Sie 0 für *k* angeben, ist der Abstand 8.

k nicht angegeben:
Der Abstand zwischen zwei Tabulatorpositionen ist 8.

-n[c][k]

pr schreibt vor jede Zeile die Zeilennummer.

Wenn Sie *-n* bei einer spaltenweisen Ausgabe verwenden, numeriert *pr* die Zeilen vor der spaltenweisen Ausgabe.

c

Zeichen, das Zeilennummer und Beginn der Zeile trennt. Sie können ein beliebiges nicht-numerisches Zeichen angeben.

c nicht angegeben:
Als Trennzeichen dient das Horizontal-Tabulatorzeichen (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*).

k

Anzahl der Stellen für eine Zeilennummer.

k nicht angegeben:
Die Anzahl der Stellen für eine Zeilennummer ist 5.

-ok

pr verschiebt jede Ausgabezeile um *k* Stellen nach rechts.

-lk

pr gibt jede Seite mit *k* Zeilen aus. Wenn die Option *-t* nicht gesetzt ist, gehören der jeweils fünfzeilige Kopf- und Fußteil mit zur Seite.

Geben Sie für *k* 10 oder weniger an, dann gibt *pr* Kopf- und Fußteil nicht aus.

-l nicht angegeben:
Eine Seite hat 66 Zeilen.

-h␣name

pr schreibt in die Kopfzeile anstelle des Dateinamens Ihre Angabe für *name*.

-p

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.

-f

Für den Fußteil einer Seite schreibt *pr* das Zeichen Formularvorschub (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*).

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:
Der Fußteil einer Seite besteht aus Leerzeichen.

-r

pr gibt keine Fehlermeldung aus, wenn es 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.

-t

pr gibt keinen Kopf- und keinen Fußteil aus. Auch die letzte Seite wird nicht mit Leerzeichen aufgefüllt.

-F

(F - fold) Die Zeilen der Eingabedatei werden nicht abgeschnitten, sondern gefaltet, wenn sie länger als 80 Zeichen sind bzw. länger als die Spaltenbreite im Mehrspaltenmodus (Optionen *-a* und *-m*).

datei

Name der Datei, die Sie zur Ausgabe aufbereiten möchten. Wenn Sie mehrere Dateien angeben, gibt *pr* diese hintereinander aus.
Wenn Sie für *datei* einen Bindestrich - angeben, liest *pr* von der Standard-Eingabe.

datei nicht angegeben:
pr liest von der Standard-Eingabe.

DATEI

*/dev/tty**

Wenn die Standardausgabe auf eine Datensichtstation */dev/tty** geleitet wird, werden andere Ausgaben auf diese Datensichtstation so lange verzögert, bis die Standardausgabe beendet ist. Damit wird verhindert, daß Fehlermeldungen mit der Ausgabe vermischt werden.

BEISPIELE

1. Sie wollen hintereinander *datei1* und *datei2* dreispaltig ausdrucken. Damit Sie vom Druckertyp unabhängig sind, soll *pr* am Ende einer Seite das Zeichen für den Seitenvorschub benutzen:

```
$ pr -3f datei1 datei2 | lp
```

2. Sie wollen den Tabulatorabstand 8 in *datei1* für *datei2* auf 6 ändern. Dazu wandeln Sie zunächst die Tabulatorzeichen aus *datei1* in Leerzeichen um (Tabulatorabstand 8); anschließend wandeln Sie die Leerzeichen wieder in Tabulatorzeichen um (Tabulatorabstand 6); das Ergebnis schreiben Sie in *datei2*; mit *-t* unterdrücken Sie bei beiden *pr*-Kommandos den Kopf- und den Fußteil:

```
$ pr -e -t datei1 | pr -i6 -t > datei2
```

3. In der Datei *monate* steht in jeder Zeile ein Monatsname. Diese Datei soll *pr* dreispaltig mit einer zweistelligen Numerierung ausgeben. Die Spalten sollen von links nach rechts gefüllt werden:

```
$ pr -3a -n2 monate

Oct 07 11:09 1988   monate Page 1

 1      Januar          2      Februar         3      Maerz
 4      April           5      Mai             6      Juni
 7      Juli            8      August          9      September
10      Oktober        11      November       12      Dezember
```

SIEHE AUCH

cat, fmt, fold, more, pg, page

printf
Formatierte Ausgabe

printf gibt die von Ihnen angegebenen Argumente in formatierter Form aus. Sie können für verschiedene Argumente auch verschiedene Formate angeben. *printf* unterstützt alle Formatangaben für Zeichenketten wie bei der C-Funktion *printf()*. Für die übrigen Formatangaben sind die Ergebnisse nicht definiert.

```
printf 'format' [arg] ...
```

'format'
Zeichenfolge, die 3 verschiedene Objekttypen enthalten kann:

- Zeichen, die unverändert ausgegeben werden sollen.
- Umwandlungsausdrücke, von denen jeder höchstens eines der angegebenen Argumente *arg* bearbeitet.
- Escape-Folgen der C-Sprache, die in der Ausgabe in entsprechende Zeichen umgewandelt werden, z.B. wird *\n* in ein Neue-Zeile-Zeichen umgewandelt.

Ein Umwandlungsausdruck besteht aus:

%[arg_nr**$**][feldbreite][.genauigkeit]zeichen

%
steht immer am Anfang des Umwandlungsausdrucks. Soll das %-Zeichen nicht Bestandteil des Umwandlungsausdrucks sein, sondern ein auszugebendes Zeichen, muß es durch ein zusätzliches %-Zeichen entwertet werden (%%).

arg_nr**$**
dezimale Ganzzahl, mit der Sie angeben, das wievielte Argument bearbeitet werden soll. Der Zahl muß ein *$*-Zeichen folgen.

arg_nr nicht angegeben:
Das nächste, auf das zuletzt umgewandelte Argument folgende Argument wird bearbeitet.

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. 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 *feldbreite* und *genauigkeit* auch einen Stern * einsetzen. In diesem Fall bestimmt dann jeweils ein ganzzahliges Argument die Feldbreite und Genauigkeit. Diese Argumente müssen vor der umzuwandelnden Zeichenkette stehen. Beginnt das Argument für die Feldbreite mit einem Bindestrich -, wird die Zeichenkette linksbündig ausgegeben. Ist das Argument für die Genauigkeit Null oder negativ, wird nichts ausgegeben.

Ist die Zeichenkette länger als die Feldbreite, wird das Feld automatisch erweitert.

zeichen
Der Buchstabe *s* gibt an, daß das Argument *arg* als Zeichenkette aufgefaßt wird. Alle Zeichen der Zeichenkette werden ausgegeben bis \0 oder die bei *genauigkeit* angegebene Zeichenzahl erreicht wird. Ist *genauigkeit* nicht angegeben, wird die gesamte Zeichenkette ausgegeben.

arg
Zeichenfolge, die im durch *format* bestimmten Format auf der Standard-Ausgabe ausgegeben wird. Sind weniger Argumente vorhanden als *format* verlangt, ist das Ergebnis nicht definiert. Sind mehr Argumente vorhanden als *format* verlangt, werden die überzähligen Argumente ignoriert.

arg nicht angegeben:
Das Ergebnis ist nicht definiert.

BEISPIELE

1. Sie wollen auf dem Bildschirm "Guten Morgen Chef" ausgeben.

```
$ printf '%s %s %s\n' Guten Morgen Chef
```

   Folgendes Kommando erzielt die gleiche Wirkung:

```
$ printf '%2$s %s %1$s\n' Chef Guten Morgen
```

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.
```

3. Sie wollen eine 4-stellige Zahl in eine 8-stelliges Feld rechtsbündig und anschließend linksbündig eintragen.

```
$ printf '%8s\n' 0374
  0374
$ printf '%-8\n' 0374
0374␣␣␣␣
```

SIEHE AUCH

printf() [14]

priocntl
Zeitscheibenverteilung und Prioritäten einstellen (priority control)

Mit *priocntl* können Sie

- Parameter für die Zeitscheiben-Verteilung (also die Zuweisung von Rechenzeit) der angegebenen Prozesse anzeigen oder einstellen,
- Informationen über die augenblickliche Konfiguration des *Schedulers* anzeigen. Der *Scheduler* ist der Prozeß, der die Zeitscheiben-Verteilung für alle Prozesse regelt.
- ein Kommando mit bestimmten Parametern für die Zeitscheiben-Verteilung ausführen.

Es gibt verschiedene Klassen von Prozessen mit jeweils unterschiedlichen Zielen und Strategien zur Zeitscheiben-Verteilung, wobei jeder Prozeß genau einer Klasse angehört. Zur Zeit werden die beiden Klassen *Echtzeit* (Realtime) und *Time-sharing* unterstützt. Die Eigenschaften dieser Klassen sowie die jeweils zulässigen Optionen werden in den Abschnitten *Die Klasse "Echtzeit" (Realtime)* und *Die Klasse "Time-sharing"* erläutert. Vorausgesetzt, Sie haben ausreichende Berechtigung, können Sie mit *priocntl* die Klasse eines laufenden Prozesses sowie andere Parameter zur Zeitscheiben-Verteilung ändern.

Vorsicht

Einem ablauffähigen Prozeß der Klasse *Echtzeit* wird vor allen anderen Prozessen Rechenzeit zugeteilt. Unsachgemäßer Einsatz von *Echtzeit*-Prozessen kann das Laufzeitverhalten des Systems daher dramatisch verschlechtern!

```
priocntl[ -l]                                                    Format 1
priocntl -d[ -i typ][ liste]                                     Format 2
priocntl -s[ -c klasse][ option] ... [ -i typ][ liste]           Format 3
priocntl -e[ -c klasse][ option] ... kommando[ argumente]        Format 4
```

Format 1: Informationen über alle Klassen ausgeben

priocntl[-l]

-l

(l - list) Es werden eine Liste aller augenblicklich im System konfigurierten Klassen sowie Informationen über die einzelnen Klassen ausgegeben. Format und Bedeutung dieser Informationen werden unten bei den jeweiligen Klassen erläutert.

Format 2: Klasse und Parameter für Prozesse ausgeben

priocntl␣-d[␣-i␣typ][␣liste]

-d

(d - display) Es werden die Klasse und klassenabhängige Parameter zur Zeitscheiben-Verteilung für die angegebenen Prozesse angezeigt.

-i␣typ

(i - identifier) Mit *-i typ* und einer eventuell angegebenen *liste* können ein oder mehrere Prozesse spezifiziert werden, auf die *priocntl* angewandt werden soll. Die Interpretation der *liste* hängt vom angegebenen *typ* ab. Im folgenden sind die zulässigen Werte für *typ* sowie die zugehörige Interpretation der *liste* angegeben:

-i␣pid

liste ist eine Liste von Prozeßnummern. *priocntl* wird auf alle Prozesse angewandt, deren Prozeßnummer in der Liste steht.

-i␣ppid

liste ist eine Liste von Vater-Prozeßnummern. *priocntl* wird auf alle Prozesse angewandt, deren Vater-Prozeßnummer in der Liste steht.

-i␣pgid

liste ist eine Liste von Prozeßgruppen-Nummern. *priocntl* wird auf alle Prozesse angewandt, die zu einer Prozeßgruppe in der Liste gehören.

-i␣sid

Liste ist eine Liste von Sitzungsnummern. *priocntl* wird auf alle Prozesse angewandt, die zu einer der Sitzungen in der Liste gehören.

-i␣class

Liste ist der Name einer einzelnen Klasse. Sie können entweder *RT* für die Klasse *Echtzeit* oder *TS* für die Klasse *Time-sharing* angeben. *priocntl* wird auf alle Prozesse der angegebenen Klasse angewandt.

-i␣uid

Liste ist eine Liste von Benutzernummern. *priocntl* wird auf alle Prozesse angewandt, deren effektive Benutzernummer in der Liste steht.

-i␣gid

Liste ist eine Liste von Gruppennummern. *priocntl* wird auf alle Prozesse angewandt, deren effektive Gruppennummer in der Liste steht.

-i␣all

priocntl wird auf alle existierenden Prozesse angewandt. In diesem Fall sollte *liste* nicht angegeben werden. Falls *liste* doch angegeben wird, wird sie ignoriert. Die Liste der Prozesse kann jedoch eingeschränkt werden, wenn Sie nicht über ausreichende Berechtigung verfügen.

-i␣typ nicht angegeben:

Als Identifikationstyp wird *pid* verwendet.

liste

liste spezifiziert die Prozesse, auf die *priocntl* angewandt werden soll. Die Bedeutung der Listenelemente ist bei der Option *-i* beschrieben. Wenn eine *liste* angegeben wird, muß sie am Ende der Kommandozeile stehen. Die einzelnen Listenelemente müssen durch Leerzeichen oder Tabulatorzeichen voneinander getrennt werden.

liste nicht angegeben:

Wurde bei der Option *-i* für das Argument *typ* der Wert *pid*, *ppid*, *pgid*, *sid*, *class*, *uid* oder *gid* angegeben, so wird jeweils die eigene Prozeßnummer, Vater-Prozeßnummer, Prozeßgruppen-Nummer, Sitzungsnummer, Klasse, Benutzernummer oder Gruppennummer des Kommandos *priocntl* verwendet.

Format 3: Klasse und Parameter für Prozesse einstellen

priocntl␣-s[␣**-c**␣klasse][␣option]...[␣**-i**␣typ][␣liste]

-s

(s - set) Die Klasse und klassenspezifische Parameter werden für die angegebenen Prozesse auf die in der Kommandozeile angegebenen Werte eingestellt.

-c␣klasse

(c - class) *klasse* gibt an, welche Klasse für die Zeitscheiben-Verteilung eingestellt werden soll. Zulässige Werte für *klasse* sind *RT* für *Echtzeit* und *TS* für *Time-sharing*.

-c␣klasse nicht angegeben:

typ und *liste* müssen Prozesse bezeichnen, die alle zur selben Klasse gehören. Anderenfalls beendet sich *priocntl* mit einer Fehlermeldung.

option

Hier geben Sie klassenspezifische Optionen an. Mögliche Optionen für die einzelnen Klassen werden in den Abschnitten *Die Klasse "Echtzeit" (Realtime)* und *Die Klasse "Time-sharing"* beschrieben.

Keine Optionen angegeben

Die klassenabhängigen Parameter werden auf die Standardwerte für die bei *-c klasse* angegebene Klasse gesetzt. Falls die Option *-c klasse* auch nicht angegeben wurde, werden die Standardwerte für die augenblickliche Klasse des Prozesses eingestellt.

-i␣typ

Mit dieser Option können die Prozesse spezifiziert werden, deren Parameter eingestellt werden sollen. Die möglichen Werte sind unter *Format 2* beschrieben.

liste

Zusammen mit *-i␣typ* spezifiziert *liste* die Prozesse, deren Parameter eingestellt werden sollen. Einzelheiten sind unter *Format 2* beschrieben.

Damit Sie für einen bestimmten Prozeß die Parameter für die Zeitscheibenverteilung mit *priocntl* ändern können, muß Ihre reale oder effektive Benutzernummer gleich der realen oder effektiven Benutzernummer dieses Prozesses sein, es sei denn, Sie haben als effektive Benutzernummer die des Systemverwalters. Dies sind die Mindestvoraussetzungen, die für alle Klassen gelten. Für einzelne Klassen können zusätzliche Beschränkungen gelten, wenn ein Prozeß diese Klasse erhalten soll oder spezielle Parameter für diese Klasse eingestellt werden sollen.

Wenn mit *-i typ* und *liste* eine Liste von Prozessen angegeben wurde, behandelt *priocntl* die Prozesse in einer implementierungsabhängigen Reihenfolge. Falls bei einem oder mehreren der angegebenen Prozesse ein Fehler auftritt, fährt *priocntl* je nach Art des Fehlers mit den anderen Prozessen in der Liste fort oder beendet sich. Wenn der Fehler auf mangelnde Berechtigung zurückzuführen ist, gibt *priocntl* eine Fehlermeldung aus und behandelt dann die anderen Prozesse in der Liste. Die angegebenen Parameter werden dann für alle Prozesse eingestellt, für die Sie ausreichende Berechtigung haben. Wenn der Fehler eine andere Ursache hat, werden die anderen Prozesse in der Liste nicht mehr behandelt. *priocntl* gibt in diesem Fall eine Fehlermeldung aus und beendet sich dann sofort.

Für die Zeitscheiben-Verteilung bei der Ausführung gewisser Systemprozesse gibt es die besondere Klasse *sys* (der *swapper*-Prozeß zur Ein- und Auslagerung auf Platte gehört zum Beispiel dieser Klasse an). Ein Prozeß kann niemals nachträglich die Klasse *sys* erhalten. Außerdem werden Prozesse der Klasse *sys* ignoriert, wenn sie mit *-i typ* und *liste* angegeben werden.

Beispiel

Für *typ* wurde der Wert *uid* angegeben und *liste* besteht nur aus dem Wert 0. Dann werden alle Prozesse mit der Benutzernummer 0 spezifiziert außer denen der Klasse *sys* und (falls die Parameter mit der Option -s geändert werden sollen) der Prozeß *init*.

Der Prozeß *init* mit der Prozeßnummer 1 stellt einen Sonderfall dar. Wenn Sie mit *priocntl* die Klasse oder andere Parameter für die Zeitscheibenverteilung ändern wollen, müssen Sie für *typ* den Wert *pid* angeben und *liste* darf nur den Wert 1 enthalten. Sie können dem Prozeß *init* zwar jede beliebige Klasse zuweisen, die auf dem System konfiguriert ist, die Klasse *Time-sharing* ist jedoch fast immer angemessen. Andere Klassen können sich sehr nachteilig auswirken.

Format 4: Ein Kommando mit bestimmten Parametern ausführen

priocntl␣-e[␣-c␣klasse][␣option]...kommando[␣argumente]

-e
(e - execute) Das *kommando* wird in der angegebenen Klasse mit den angegebenen Optionen für die Zeitscheiben-Verteilung ausgeführt.

-c klasse
klasse gibt an, welche Klasse das auszuführende Kommando erhalten soll. Zulässige Werte sind *RT* für die Klasse *Echtzeit* und *TS* für die Klasse *Time-sharing*.

-c klasse nicht angegeben:
Das Kommando wird mit Ihrer augenblicklichen Klasse ausgeführt.

option
Hier geben Sie klassenspezifische Optionen an. Mögliche Optionen für die einzelnen Klassen werden in den Abschnitten *Die Klasse "Echtzeit" (Realtime)* und *Die Klasse "Time-sharing"* beschrieben.

Keine Optionen angegeben:
Das Kommando wird mit den Standardwerten für die bei *-c klasse* angegebene Klasse ausgeführt. Falls die Option *-c klasse* auch nicht angegeben wurde, wird das Kommando mit den Standardwerten für Ihre augenblickliche Klasse ausgeführt.

kommando
Das Kommando, das in der angegebenen Klasse mit den angegebenen Parametern ausgeführt werden soll.

argumente
Argumente für das auszuführende Kommando.

Die Klasse "Echtzeit" (Realtime)

In der Klasse *Echtzeit* erfolgt die Verteilung der Zeitscheiben für die Zentraleinheit (CPU) nach festen Prioritäten und unter der Maßgabe, daß ein Prozeß einen anderen von der CPU "verdrängen" kann (preemptive scheduling). Dies ist für solche Prozesse sinnvoll, die kurze und vorhersehbare Antwortzeiten sowie die vollständige Steuerung der Prioritäten durch den Benutzer benötigen. Falls die Klasse *Echtzeit* auf dem System konfiguriert ist, sollte sie vollständige Kontrolle über den höchsten Bereich der Prioritäten für die Zeitscheiben-Verteilung haben. Damit wird gewährleistet, daß ein ablauffähiger Prozeß der Klasse *Echtzeit* vor jedem anderen Prozeß jeder anderen Klasse Rechenzeit erhält.

Einem Prozeß der Klasse *Echtzeit* kann eine Echtzeit-Priorität (real-time priority, *rtpri*) zugeordnet werden. Dieser Wert liegt zwischen 0 und einem Maximalwert *x*, wobei der Wert von *x* konfiguriert werden kann. Er kann für eine spezielle Konfiguration mit dem Kommando *priocntl -l* angezeigt werden.

Die Verteilung der Zeitscheiben für Prozesse der Klasse *Echtzeit* erfolgt nach festen Prioritäten. Die Priorität eines Echtzeit-Prozesses wird niemals verändert, es sei denn, der Benutzer oder ein Anwendungs-Programm wollen die Priorität explizit verändern.

Für einen Prozeß der Klasse *Echtzeit* ist der Wert *rtpri* praktisch gleich der internen Priorität für die Zuteilung von Rechenzeit an diesen Prozeß. Dieser Wert bestimmt die Priorität eines Echtzeit-Prozesses im Verhältnis zu anderen Prozessen dieser Klasse. Ein höherer Wert von *rtpri* steht für eine höhere Priorität. Da die Klasse *Echtzeit* den höchsten Bereich der Prioritäten für die Zuteilung von Rechenzeit steuert, ist gewährleistet, daß ein ablauffähiger Prozeß mit dem höchsten Wert von *rtpri* vor jedem anderen Prozeß im System Rechenzeit erhält.

Zusätzlich zur Steuerung der Prioritäten können Sie mit *priocntl* die Länge der Zeitscheiben beeinflussen, die einem Prozeß der Klasse *Echtzeit* zugeteilt werden. Die Länge der Zeitscheibe gibt an, wie lange ein Prozeß höchstens Rechenzeit erhält, vorausgesetzt, er wartet nicht auf ein externes Ereignis (*sleep*) wie z.B. Durchführung eines Festplattenzugriffs. Ein Prozeß kann jedoch vor Ablauf seiner Zeitscheibe unterbrochen werden, wenn ein anderer Prozeß mit höherer Priorität ablauffähig wird.

Sie können die Priorität und die Länge der Zeitscheibe für einen Prozeß der Klasse *Echtzeit* mit dem Kommando

priocntl -d[␣-i␣typ][␣liste]

anzeigen lassen. Die Länge der Zeitscheibe wird dabei in Millisekunden ausgegeben. Die Werte werden für jeden spezifizierten Prozeß der Klasse *Echtzeit* ausgegeben.

Zulässige Optionen für das Setzen von Echtzeit-Parametern (Optionen *-s* und *-e*):

-p␣rtpri
Die Echtzeit-Priorität des angegebenen Prozesses bzw. der angegebenen Prozesse wird auf *rtpri* gesetzt.

Option *-p* nicht angegeben:
Wenn der Prozess bereits der Klasse *Echtzeit* angehört, bleibt die Priorität unverändert. Wenn ein Prozeß einer anderen Klasse die Klasse *Echtzeit* erhält, wird die Priorität auf den Standardwert 0 gesetzt.

-t␣zeitdauer
Die Länge der Zeitscheibe für den angegebenen Prozeß bzw. die angegebenen Prozesse wird auf *zeitdauer* gesetzt. Wird als Zeitdauer 0 angegeben oder ein Wert, der über dem Maximalwert für die Dauer von Zeitscheiben liegt, wird mit einer Fehlermeldung abgebrochen. Der Maximalwert hängt von der Implementierung ab und ist normalerweise sehr groß.

Option *-t* nicht angegeben:
Wenn der Prozeß bereits der Klasse *Echtzeit* angehört, bleibt die Länge der Zeitscheibe unverändert. Wenn ein Prozeß einer anderen Klasse die Klasse *Echtzeit* erhält, wird die Länge der Zeitscheibe auf einen Standardwert gesetzt, der von der Priorität und der Konfiguration des Systems abhängt (siehe *rt_dptbl* [5]).

-r␣auflösung
(r - resolution) *auflösung* gibt an, in welcher Einheit bei der Option *-t* die Länge der Zeitscheibe angegeben wird. *auflösung* ist eine positive ganze Zahl zwischen 1 und 1000000000 (einschließlich). Der reziproke Wert von *auflösung* gibt die benutzte Einheit in Sekunden an.

Beispiel

-t␣10␣-r␣100 gibt als Zeiteinheit 1/100 Sekunde an und die Zeitdauer wird auf 10/100 Sekunden, also 1/10 Sekunde gesetzt.

Obwohl eine sehr hohe Auflösung angegeben werden kann (nämlich eine Nanosekunde), wird die angegebene Zeitdauer immer auf das nächste ganzzahlige Vielfache der Auflösung der Systemuhr aufgerundet.

Beispiel

Die höchste Auflösung auf einem System beträgt 10 Millisekunden. Wird mit den Optionen *-t* und *-r* eine Zeitdauer von 34 Millisekunden angegeben, so wird bei diesem System auf 40 Millisekunden aufgerundet.

Die Option *-r* kann nur zusammen mit der Option *-t* angegeben werden.

Option *-r* nicht angegeben:
Es wird 1 Millisekunde als Einheit für die Länge der Zeitscheiben bei *-t* verwendet.

Jede Kombination der Optionen *-p* und *-t* (und eventuell *-r*) kann mit *-s* oder *-e* verwendet werden.

Nur der Systemverwalter kann einem Prozeß einer anderen Klasse die Klasse *Echtzeit* geben. Wenn Sie die Priorität oder die Länge der Zeitscheibe für einen Echtzeit-Prozeß ändern wollen, müssen Sie entweder Systemverwalterberechtigung haben oder bereits in der Klasse *Echtzeit* sein (Ihr Kommandointerpreter läuft als Echtzeit-Prozeß) und die gleiche reale oder effektive Benutzernummer haben wie der Prozeß, dessen Parameter geändert werden sollen.
Die Echtzeit-Priorität und die Länge der Zeitscheibe werden bei den Systemaufrufen *fork* und *exec* unverändert weitergegeben.

Die Klasse "Time-sharing"

Die Strategie für die Verteilung der Zeitscheiben in der Klasse *Time-sharing* ist auf eine gleichmäßige und effektive Zuteilung der CPU-Rechenzeit für Prozesse mit unterschiedlicher CPU-Ausnutzung ausgelegt. Die Ziele dieser Strategie sind gute Antwortzeiten bei interaktiven Prozessen und guter Durchsatz bei rechenintensiven Anwendungen. Die Verteilung der Zeitscheiben soll zu einem gewissen Maß vom Benutzer beeinflußbar sein.

Einem Prozeß der Klasse *Time-sharing* kann eine Time-sharing-Priorität (time-sharing user priority, *tsupri*) zugeordnet werden. Dieser Wert liegt zwischen *-x* und *x*, wobei der Wert von *x* konfiguriert werden kann. Er kann für eine spezielle Konfiguration mit dem Kommando *priocntl -l* angezeigt werden.

Mit der vom Benutzer einstellbaren Time-sharing-Priorität kann die Zuteilung von Rechenzeit an Prozesse der Klasse *Time-sharing* zu einem gewissen Maß beeinflußt werden. Wird der Wert *tsupri* für einen Prozeß der Klasse *Time-sharing* erhöht oder vermindert, so wird auch die interne Priorität dieses Prozesses bei der Zeitscheiben-Verteilung erhöht oder vermindert. Es ist jedoch nicht gewährleistet, daß ein Prozeß der Klasse *Time-sharing* mit einem höheren *tsupri*-Wert vor einem Prozeß mit niedrigerem *tsupri*-Wert Rechenzeit erhält. Der Wert *tsupri* ist nämlich nur eine Komponente bei der Bestimmung der internen Priorität für die Verteilung der Zeitscheiben. Das System kann die interne Priorität eines Time-sharing-Prozesses dynamisch anpassen, wobei andere Faktoren eine Rolle spielen (z.B. bisherige CPU-Ausnutzung).

Zusätzlich zu den systemspezifischen Grenzwerten für die Benutzer-Priorität, die mit *priocntl -l* angezeigt werden können, gibt es für jeden Prozeß einen Grenzwert für die Benutzer-Priorität, die dieser Prozeß erhalten kann (time-sharing user priority limit, *tsuprilim*)

Sie können die Benutzer-Priorität und den Grenzwert für die Benutzer-Priorität für einen Prozeß der Klasse *Time-sharing* mit dem Kommando

priocntl -d[␣-i␣typ][␣liste]

anzeigen lassen. Die Werte werden für jeden spezifizierten Prozeß der Klasse *Time-sharing* ausgegeben.

Zulässige Optionen für das Setzen von Time-sharing-Parametern (Optionen *-s* und *-e*):

-m tsuprilim
Darf nur gemeinsam mit *-e* gesetzt werden. Der Grenzwert für die Benutzer-Priorität wird für den angegebenen Prozeß bzw. die angegebenen Prozesse auf den Wert *tsuprilim* gesetzt.

Jeder Prozeß der Klasse *Time-sharing* kann den Grenzwert für die Benutzer-Priorität für sich selbst und für jeden anderen Prozeß mit der selben Benutzernummer vermindern. Der Grenzwert darf darf jedoch nur von einem Prozeß mit Systemver-

walter-Berechtigung erhöht werden. Wenn ein Prozeß einer anderen Klasse die Klasse *Time-sharing* erhält, darf ebenfalls nur der Systemverwalter einen Grenzwert einstellen, der größer als Null ist.

Option *-m* nicht angegeben:
Wenn der Prozess der Klasse *Time-sharing* angehört, bleibt der Grenzwert unverändert. Wenn ein Prozeß einer anderen Klasse die Klasse *Time-sharing* erhält, wird der Grenzwert auf den Standardwert 0 gesetzt.

-p␣tsupri

Darf nur gemeinsam mit *-e* gesetzt werden. Die Benutzer-Priorität des angegebenen Prozesses bzw. der angegebenen Prozesse wird auf den Wert *tsupri* gesetzt.

Jeder Prozeß der Klasse *Time-sharing* kann seine eigene Benutzer-Priorität oder die eines anderen Prozesses mit der selben Benutzernummer auf einen Wert einstellen, der kleiner oder gleich dem Grenzwert für diesen Prozeß ist. Wenn Sie einen größeren Wert einstellen wollen oder den Grenzwert unter die aktuelle Benutzer-Priorität senken, wird als Benutzer-Priorität der Grenzwert eingestellt.

Option *-p* nicht angegeben:
Wenn der Prozess der Klasse *Time-sharing* angehört, bleibt die Benutzer-Priorität normalerweise unverändert. Wenn ein Prozeß einer anderen Klasse die Klasse *Time-sharing* erhält, wird die Benutzer-Priorität auf den eingestellten Grenzwert für die Benutzer-Priorität gesetzt.

Ausnahme

Die Option *-p* wurde nicht angegeben und mit *-m* wurde ein Grenzwert eingestellt, der unter der augenblicklichen Benutzer-Priorität *tsupri* liegt. In diesem Fall wird die Benutzer-Priorität auf den eingestellten Grenzwert gesetzt.

Jede Kombination der Optionen *-m* und *-p* kann mit *-s* oder *-e* verwendet werden.

Die Benutzer-Priorität und der Grenzwert für die Benutzer-Priorität werden bei den Systemaufrufen *fork* und *exec* unverändert weitergegeben.

FEHLERMELDUNGEN

`Process(es) not found:`

Keiner der angegebenen Prozesse existiert.

`Specified processes from different classes:`

Die Option *-s* zum Einstellen von Parametern wurde angegeben, die Option *-c klasse* wurde nicht angegeben, und die spezifizierten Prozesse gehören verschiedenen Klassen an.

`Invalid option or argument:`

Eine Option oder ein Argument für eine Option ist unbekannt oder ungültig.

BEISPIELE

1. Mit dem Kommando

   ```
   priocntl -s -c RT -t 1 -r 10 -i pid 1234 2345 3456
   ```

   erhalten die Prozesse mit den angegebenen Prozeßnummern die Klasse *Echtzeit*, sofern sie noch nicht zu dieser Klasse gehören. Ihre Echtzeit-Priorität wird auf den Standardwert 0 gesetzt. Die Echtzeit-Parameter anderer Echtzeit-Prozesse bleiben unverändert. Die Länge der Zeitscheibe wird für alle angegebenen Prozesse auf 1/10 Sekunde gestellt (eine Einheit bei einer Auflösung von 1/10 Sekunde).

2. Die Aufrufzeile

   ```
   priocntl -e -c RT -p 15 -t 20 kommando
   ```

   startet das angegebene Kommando als Echtzeit-Prozeß mit einer Echtzeit-Priorität von 15 und einer Zeitscheibe von 20 Millisekunden.

3. Mit dem Kommando

   ```
   priocntl -s -c TS -i uid 17
   ```

   erhalten alle Prozesse mit der Benutzernummer 17 die Klasse *Time-sharing*, sofern noch nicht zu dieser Klasse gehören. Der Grenzwert für die Benutzer-Priorität und die Benutzer-Priorität werden auf den Standardwert 0 gesetzt. Die Time-sharing-Parameter von Prozessen, die bereits der Klasse Time-sharing angehören, bleiben unverändert.

4. Die Aufrufzeile

   ```
   priocntl -e -c TS -m 0 -p -15 kommando argumente
   ```

   startet das angegebene Kommando in der Klasse *Time-sharing* mit dem Grenzwert 0 für die Benutzer-Priorität und der Benutzer-Priorität -15.

SIEHE AUCH

ps, *nice*
priocntl() [14]
rt_dptbl [5]

ps
Prozeßdaten abfragen (process status)

ps gibt Informationen über Prozesse aus. Dabei wird ein "Schnappschuß" 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.

```
ps [ -adeflcj ] [ -g grplist ] [ -p proclist ] [ -t termlist ] [ -u uidlist ]
   [ -s sesslist ]
```

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 und
- dem Kommandonamen CMD.

Die Bedeutung der Ausgabespalten ist im Abschnitt *Ausgabe* genauer erläutert.

-a
Es werden Informationen über alle Prozesse ausgegeben, außer über Prozesse, die Prozeßgruppenführer sind oder die nicht mit einer Datensichtstation verbunden sind.

-c
Es werden Informationen über den Steuerungsprozeß *scheduler* ausgegeben (vgl. *priocntl*). *-c* beeinflußt die Ausgabe bei den Optionen *-f* und *-l*.

-d
Es werden Informationen über alle Prozesse ausgegeben, außer über Prozesse, die Prozeß-Sitzungsführer sind (siehe *setsid()* [8]).

-e
Es werden Informationen über alle Prozesse ausgegeben.

-f
Es werden ausführliche Informationen ausgegeben. Dazu untersucht *ps* den Speicher oder den Swap-Bereich. *ps* versucht festzustellen, welcher Kommandoname und welche Argumente angegeben wurden, als der Prozeß erzeugt wurde, und gibt den Kommandonamen und seine Argumente aus. Die Argumente werden jedoch nur angezeigt, wenn der Prozeß dem Aufrufer des *ps* gehört bzw. *ps* von *root* aufgerufen wird. Falls dies nicht gelingt, z.B. wenn der Prozeß ausgelagert (swapped) ist,

wird der Kommandoname eingeschlossen in eckigen Klammern [...] ausgegeben.

-g␣grplist
Es werden nur Informationen über Prozesse ausgegeben, deren Prozeß-Sitzungsführer in *grplist* angegeben werden.

grplist
grplist ist eine Liste von Nummern von Prozeß-Sitzungsführern. *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 wird die Kenn-Nummer der Sitzung und Kenn-Nummer der Prozeßgruppe ausgegeben.

-l
gibt eine lange Liste aus.

-p␣proclist
Es werden nur Informationen ausgegeben zu Prozessen, deren Prozeßnummern in *proclist* angegeben sind.

proclist
proclist ist eine Liste von Prozeßummern.

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 Prozeß-Sitzungsführer ausgegeben, die in sesslist angegeben wurden.

sesslist
sesslist ist eine Liste von Nummern von Prozeß-Sitzungsführern. *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. *tty04*) oder, falls der Gerätename mit *tty* beginnt, nur durch die Nummer (z.B. *04*).

Die Datensichtstationen werden durch ein Komma getrennt, oder die Liste wird in Anführungszeichen eingeschlossen "...", wobei die Datensichtstationen dann durch Komma und/oder Leerzeichen getrennt werden können.

-u␣uidlist

Es werden nur Informationen über Prozesse ausgegeben, für die die Benutzernummer UID oder die Benutzerkennung des Prozeßeigentümers in *uidlist* angegeben sind. Es wird die Benutzernummer ausgegeben, außer wenn zusätzlich die Option -*f* angegeben wird, dann wird die Benutzerkennung ausgegeben.

Ausgabe

Im folgenden werden die Überschriften der Spalten und die Bedeutung der Spalten in der Ausgabe von *ps* erläutert. Die Buchstaben *f* und *l* bezeichnen die Option, bei der die entsprechende Spalte in der Ausgabe erscheint, *alle* bedeutet, daß die Spalte bei allen Optionen erscheint. Beachten Sie bitte, daß Sie mit den Optionen *f* und *l* nur bestimmen, welche Informationen Sie zu einem Prozeß erhalten, und nicht, zu welchem Prozeß Sie Informationen erhalten.

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 Prozeß schläft, wartet auf ein Ereignis
- R Prozeß ist bereit zum Laufen
- I Zwischenstatus: Prozeß ist im Entstehen
- Z Zwischenstatus: Prozeß terminiert
- T getraceter Prozeß ist gestoppt
- X Prozeß wartet auf mehr Speicherplatz

UID (*f, l*)

(UID - user ID) Benutzernummer des Prozeß-Eigentümers. 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 Prozeß-Nummer. Unter dieser Nummer kann z.B. der Prozeß mit *kill* beendet werden.

PPID (*f, l*)
(PPID - parent process ID) Prozeß-Nummer des Vaterprozesses.

C (*f, l*)
Prozessor-Auslastung für das Scheduling, wird bei *-c* nicht ausgegeben.

CLS (*f, l*)
Scheduling-Gruppe (Prozesse, die im Zeitscheibenverfahren verarbeitet werden), wird nur bei *-c* ausgegeben.

PRI (*l*)
Prioritätswert des Prozesses. Höhere Zahlen bedeuten normalerweise niedrigere Priorität. Wurde *-c* angegeben, so bedeuten höhere Zahlen höhere Priorität.

NI (*l*)
Wert, mit dem die Prozeß-Priorität verändert wurde (siehe *nice*). Dieser Wert wird bei *-c* nicht ausgegeben. Nur Prozeße der Klasse *time-sharing* haben einen solchen Wert.

ADDR (*l*)
Adresse (physikalische Seiten-Frame-Nummer)
der user-area, falls resident, andernfalls die Platten-Adresse des ausgelagerten Prozesses.

SZ (*l*)
Die Größe in Blocks des Speicher-Abbildes (core-image) des Prozesses.

WCHAN (*l*)
Nummer des Ereignisses, auf das der Prozeß wartet oder schläft. 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.

COMMANDD (*alle*)
Der Name des Kommandos. Der vollständige Kommandoname und seine Argumente werden ausgegeben, wenn die Option *-f* gesetzt ist.

Ein Prozeß, der sich beendet hat und der einen Vaterprozeß hat, der bisher noch nicht auf ihn gewartet hat, wird mit `<defunct>` bezeichnet.

Wenn *termlist*, *proclist*, *uidlist* oder *grplist* nicht angegeben wurde, überprüft *ps* Standard-Ein/Ausgabe und die Standardfehlerausgabe in dieser Reihenfolge, um die kontrollierende Datensichtsichtstation 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.
Auf einem stark belasteten System kann *ps* einen *lseek*-Fehler melden und sich beenden: Nachdem die Adresse des Benutzerbereichs eines Prozesses bestimmt wurde, kann *ps* unter Umständen erst dann auf diesen Bereich positionieren, wenn der Prozeß schon beendet und die Adresse damit ungültig ist.

DATEIEN

/unix
enthält die Symboltabelle des Systems

*/dev/sxt/**
*/dev/tty**
*/dev/xt/**
Suchdateien für die Namen der Datensichtstationen

/dev/kmem
Virtueller Speicher des Kerns

/dev/swap
Standardgerät für die Auslagerung von Prozessen

/dev/mem
Speicher

/etc/passwd
Informationen über Benutzerkennungen

/etc/ps_data
enthält interne Datenstrukturen

SIEHE AUCH

kill, *nice*, *priocntl*, *sh*
getty [5]

pwd
Pfadnamen des aktuellen Dateiverzeichnisses ausgeben (print working directory)

Das in die Bourne-Shell *sh* eingebaute Kommando *pwd* schreibt den absoluten Pfadnamen des aktuellen Dateiverzeichnisses auf die Standard-Ausgabe.

```
pwd
```

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. angewendet, wenn ein Benutzer mit Dateien, die sich auf einem fernen Rechner befinden, weiterarbeiten möchte.

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 (mit *-r*) von einem Rechner in ein Dateiverzeichnis eines anderen Rechners.

Das Kommando *rcp* kann nur von Benutzern angewendet werden, die auf dem fernen Rechner zugriffsberechtigt sind. Ein Benutzer ist dann auf dem fernen Rechner zugriffsberechtigt, wenn

- der Name des lokalen Rechners und die Benutzerkennung in der Datei *.rhosts* eingetragen sind, die sich im $HOME-Dateiverzeichnis des Benutzer am fernen Rechner befindet, oder
- der lokale Rechner in der Datei */etc/hosts.equiv* des fernen Rechners eingetragen ist und der Benutzer am fernen Rechner mit der gleichen Benutzerkennung arbeitet wie am lokalen Rechner.

```
rcp[␣-p]␣datei1␣datei2                                    Format 1
rcp[␣-pr]␣datei␣ ...␣dvz                                  Format 2
```

Format 1: Einzelne Dateien kopieren

rcp[␣-p]␣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 die selbe Änderungszeit, Zugriffszeit und die selben Zugriffsrechte wie die Originaldatei.

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*. Ausgangspunkt ist das aktuelle Dateiverzeichnis.

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.
login ist die Benutzerkennung am fernen Rechner.
pfad ist der Pfadname von *datei1*.
domäne ist der Name der Domäne, der der ferne Rechner angehört.

Pfadnamen können die üblichen Metazeichen (z.B. *, ?) zur Dateinamenerzeugung enthalten. Diese werden auf dem lokalen Rechner ausgewertet.

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]␣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 die selbe Änderungszeit, Zugriffszeit und Zugriffsrechte wie die Originaldatei.

-r

Es werden rekursiv alle Unterverzeichnisse von *dvz* kopiert. In diesem Fall muß Datei ein Dateiverzeichnis sein.

datei

kann sein

- eine Datei
- mehrere Dateien
- ein Dateiverzeichnis (bei *-r*)

am lokalen oder an einem fernen Rechner.

Am lokalen Rechner gilt folgende Form:

datei_name ... oder
dateiverzeichnis.

datei_name 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 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

DATEIEN

/etc/hosts.equiv
Liste von Rechnernamen; Benutzerkennungen eines fernen Rechners, dessen Name in dieser Liste steht, haben Zugang zum lokalen Rechner, wenn sie auch dort bekannt sind

$HOME/.rhosts
Liste der Rechnernamen mit Benutzerkennungen, die sich unter Ihrer Kennung anmelden dürfen

BEISPIELE

1. Der Benutzer *hans* möchte aus seinem aktuellen Dateiverzeichnis die Datei *test* an den Benutzer *franz* am Rechner *bremen* kopieren. Die Datei soll am Rechner *bremen* den Namen *hans_test* bekommen. Die Zugriffsrechte sind gegeben.

```
$ rcp test franz@bremen:hans_test
```

2. Der Benutzer *hans* am Rechner *grafing* will die Dateiverzeichnisse *test* und *uebung* und alle darin enthaltenenen Unterverzeichnisse kopieren.
Die Dateiverzeichnisse sollen von der Benutzerkennung *hans* am Rechner *bremen* auf den Rechner *athen* in das Dateiverzeichnis */usr/mecky* kopieren. Die Benutzerkennung *hans* ist auf den Rechnern *bremen* und *athen* vorhanden und zugriffsberechtigt.

```
$ rcp -r bremen:test bremen:uebung athen:/usr/mecky
```

SIEHE AUCH

ftp, rlogin, rsh

read
Argumente von der Standard-Eingabe lesen und Shell-Variablen zuweisen

Das in die Bourne-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 die Prozedur fortgesetzt (siehe auch *BEISPIELE*).

```
read name ...
```

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.

ENDE-STATUS

0 *read* wurde erfolgreich ausgeführt
1 *read* hat keine Eingabe erhalten, also nur Datei-Ende (EOF) gelesen.

FEHLERMELDUNGEN

```
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.

UMGEBUNGSVARIABLE

IFS
: Argument-Trennzeichen. Standardmäßig sind der Variablen IFS zugewiesen: Leerzeichen, Tabulator-Zeichen, Neue-Zeile-Zeichen.

BEISPIELE

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 Kunden-Namen.

   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.

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 immmer wieder die erste Zeile der Datei */etc/group*.

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, *sh*

readonly
Shell-Variablen schützen

Das in die Bourne-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.

```
readonly[␣name]...
```

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:

```
readonly name
.
.
.
```

FEHLERMELDUNG

```
name: is read only
```

Diese Fehlermeldung erhalten Sie, wenn Sie versuchen, einer geschützten Shell-Variablen einen Wert zuzuweisen.

BEISPIEL

Die Shell-Variable *HOME* in der aktuellen Shell schützen:

```
$ readonly HOME
$ readonly
readonly HOME
$ sh
$ readonly
$ END
$ readonly
readonly HOME
```

In der Subshell ist die Variable *HOME* nicht geschützt.

SIEHE AUCH

sh, *ksh*

red
Eingeschränkter zeilenorientierter Editor im Dialogbetrieb

red funktioniert wie *ed* mit folgenden Einschränkungen: Sie können nur Dateien bearbeiten, die sich im aktuellen Dateiverzeichnis befinden. Außerdem können Sie keine !-Kommandos ausführen lassen.

```
red[ -s][ -p zeichenkette][ -x][ -C][ datei]
```

Beschreibung der Parameter siehe *ed*.

SIEHE AUCH

ed

relogin
Aktuelles Shell-Fenster als login-Eintrag definieren

relogin definiert dasjenige Shell-Fenster, das mit der Standard-Eingabe verbunden ist, als aktuelle Datensichtstation des Benutzers. Dazu wird der Eintrag *Leitung* in der Datei */var/adm/utmp* geändert. Dies bewirkt, daß Nachrichten, die mit *write* an den Benutzer geschickt werden, an dieses Shell-Fenster geleitet werden. Außerdem zeigt das Kommando *who* dieses Shell-Fenster als aktuelle Datensichtstation für den Benutzer an. *relogin* kann nur unter *layers* verwendet werden.

relogin wird von *layers* automatisch aufgerufen. Beim Start wird das zuerst erzeugte Shell-Fenster als aktive *Leitung* in */usr/adm/utmp* eingetragen. Bei Beendigung von *layers* wird der Eintrag auf die ursprüngliche Leitung zurückgesetzt.

relogin kann von einem Benutzer dazu verwendet werden, ein anderes Shell-Fenster für den Empfang von mit *write* geschriebenen Nachrichten zu bestimmen.

Wenn die Standard-Eingabe keine Datensichtstation ist, liefert *relogin* einen Fehler.

```
/usr/lib/layersys/relogin[␣-s][␣leitung]
```

-s
Fehlermeldungen werden unterdrückt.

leitung
Gibt an, welcher Eintrag in */usr/adm/utmp* geändert werden soll. Die Datei wird nach einem Eintrag mit dem Feld *Leitung* durchsucht. Dieses Feld wird durch die Leitung ersetzt, die mit der Standard-Eingabe verbunden ist. Um festzustellen, welche Leitungen zur Benutzerkennung *name* gehören, verwenden Sie das Kommando *ps -f -u name*. In dem Feld *Leitung* werden alle aktiven Leitungen für diesen Benutzer angezeigt (siehe *ps*). *relogin* liefert einen Fehler, wenn die *Leitung* nicht zu dem Benutzer gehört, der das Kommando aufgerufen hat.

DATEI

```
/var/adm/utmp
```

SIEHE AUCH

layers, *mesg*, *ps*, *who*, *write*
utmp [5], [14]

rksh
Eingeschränkte Korn-Shell

rksh ist die eingeschränkte Version der Korn-Shell. Sie wird für spezielle Logins und Arbeitsumgebungen benutzt, deren Leistungsfähigkeit stärker kontrolliert wird, als dies bei einer Standardumgebung notwendig ist.

```
rksh[␣option]...[␣datei][␣argument]...
```

option
Siehe *ksh*.

datei
Siehe *ksh*.

argument
Siehe *ksh*.

Arbeitsweise

Die Arbeitsweise entspricht der Korn-Shell *ksh* mit den folgenden Einschränkungen:

- Das eingebaute *ksh*-Kommando *cd* ist gesperrt, d.h. Sie können Ihr aktuelles Dateiverzeichnis nicht verlassen.
- Sie können den Variablen ENV, PATH und SHELL keine neuen Werte zuweisen.
- Kommandos, in deren Name beim Aufruf ein Schrägstrich / enthalten ist, werden nicht ausgeführt. Sie können also nur Kommandos ausführen, die in Ihrem aktuellen Dateiverzeichnis oder in einem der Dateiverzeichnisse, deren Pfade der Shell-Variablen PATH zugewiesen wurden, stehen.
- Die Umlenkung in Dateien ist nicht erlaubt. Kommandos werden nicht ausgeführt, wenn die Kommandozeile die Zeichen >, >|, <> oder >> enthält.

Diese Einschränkungen treten nach der Ausführung der Datei *$HOME/.profile* und der durch die Variabale *ENV* festgelegten Datei in Kraft.

Wird als Kommando eine Shell-Prozedur aufgerufen, dann ruft die eingeschränkte Korn-Shell dazu die normale Korn-Shell *ksh* auf! Innerhalb von Prozeduren steht also der volle Funktionsumfang der Korn-Shell zur Verfügung - während dem Benutzer nur eine eingeschränkte Arbeitsumgebung zur Verfügung steht.

Nur für den Systemverwalter

Als Systemverwalter können Sie die Arbeitsumgebung für Benutzer einrichten, die nur mit der eingeschränkten Korn-Shell arbeiten sollen. Sie sollten folgendes festlegen:

- In welchem Dateiverzeichnis darf gearbeitet werden?
  Die Benutzer sollten nicht im Login-Dateiverzeichnis arbeiten, sie sollten für das Login-Dateiverzeichnis nur Leserecht haben.
- Welche Kommandos dürfen ausgeführt werden?
  Dazu können Sie entweder ein Unterdateiverzeichnis im Login-Dateiverzeichnis anlegen oder ein allgemeineres Verzeichnis, wie z.B */usr/rbin*, und die erlaubten und mit *rksh* sicher ausführbaren Kommandos dorthin kopieren.
  Auch für diese Dateiverzeichnisse sollte der Benutzer kein Schreibrecht haben.
- Welche Korn-Shell-Skripten dürfen ausgeführt werden?
  Es sollte z.B. nicht möglich sein, eine Subshell aufzurufen, denn diese Subshell ist nicht eingeschränkt.
- Wie sind die Werte der Variablen *ENV*, *PATH* und *SHELL* zu setzen?
  Es sollte z.B. nicht möglich sein, über die *!*-Anweisung der Kommandos *ex* oder *vi* eine nicht eingeschränkte Shell aufzurufen.

In die Datei *$HOME/.profile* und *$ENV* tragen Sie die entsprechenden Kommandos und Wertzuweisungen ein. Das *cd*-Kommando, das in das Arbeitsdateiverzeichnis wechselt, in dem der Benutzer mit der eingeschränkten Korn-Shell arbeiten soll, wird eines der letzten Kommandos sein. Für diese Dateien und die Verzeichnisse in denen sie stehen (z.B. *$HOME*) darf der Benutzer ebenfalls kein Schreibrecht haben.

SIEHE AUCH

ksh

rlogin
An einem fernen Rechner anmelden (remote login)

Mit *rlogin* können Sie sich vom lokalen Rechner an einem fernen Rechner anmelden. Nach dem Anmelden können Sie am fernen Rechner arbeiten, als ob es der lokale Rechner wäre.

Das Kommando kann auch mehrfach verwendet werden. Nach dem Anmelden an einem fernen Rechner ist ein Anmelden an einem weiteren fernen Rechner möglich.

Das Kommando wird z.B. angewendet, wenn Sie

- mit Programmen arbeiten möchten, die nur an einem anderen Rechner zur Verfügung stehen (z.B. Editoren, dann müssen Sie ggf. Ihre Dateien mit *rcp* oder *ftp* zum fernen Rechner kopieren).
- an Dateien arbeiten, die aus organisatorischen Gründen auf einem anderen Rechner gespeichert sind (z.B. Datenbanken),
- Kommunikationsfunktionen nutzen wollen, die an einem anderen Rechner zur Verfügung stehen.

rlogin kann nur angewendet werden, wenn Sie auf dem fernen Rechner zugriffsberechtigt sind. Zugriffsberechtigt sind Sie dann, wenn Sie am fernen Rechner

- eine Benutzerkennung mit dem dazugehörenden Kennwort kennen oder
- in der Datei *.rhosts* der Name Ihres lokalen Rechners und die Benutzerkennung, unter der Sie arbeiten, eingetragen sind oder
- mit der gleichen Benutzerkennung arbeiten wie am lokalen Rechner und der lokale Rechner in der Datei */etc/hosts.equiv* des fernen Rechners eingetragen ist.

Aus Sicherheitsgründen muß der Eigentümer der Datei *.rhosts* entweder der Benutzer auf dem fernem Rechner oder der Systemverwalter sein.

rlogin kann nur dann ausgeführt werden, wenn am fernen Rechner der Dämon *rlogind* aktiv ist. *rlogind* wird vom Dämon *inetd* gestartet.

rlogin baut eine Verbindung zum Dämon *rlogind* des fernen Rechners auf. Der Dämon *rlogind* des fernen Rechners eröffnet ein Pseudo-TTY und startet den Kommandointerpreter.

Mit Hilfe des Pseudo-TTYs werden die Ausgaben der Shell am fernen Rechner auf den lokalen Rechner umgeleitet.

Wird die Shell beendet (z.B. mit der Taste END), so ist damit auch die Sitzung am fernen Rechner beendet.

Die Umgebungsvariable TERM hat für die Arbeit am fernen Rechner den selben Wert wie auf dem lokalen Rechner. Die Größe der Datensichtstation oder des Fensters wird auch zum fernen Rechner übermittelt, falls der Server dies unterstützt. Die Steuerung der Ausgabe mit CTRL s und CTRL q sowie das Leeren der Puffer bei Unterbrechungen werden korrekt durchgeführt.

Sie können diese Version von *rlogin* nur mit dem Netzwerk-Protokoll TCP verwenden.

```
rlogin[ -8][ -ex][ -l login] host
```

-8

Alle Zeichen zwischen dem fernen Rechner und dem lokalen Bildschirm werden mit 8 Bit übertragen. *-8* müssen Sie z.B. angeben, wenn Sie eine Datensichtstation mit Fenstereigenschaften verwenden und *layers* auf dem fernen Rechner laufen lassen wollen.

-8 nicht angegeben:
Voreinstellung ist die Übertragung der Zeichen mit 7 Bit.

-esymbol

Sie definieren ein Fluchtsymbol *symbol*, mit dem Sie zum Ausgangspunkt des *rlogin*-Kommandos zurückkommen.

Als Fluchtsymbol kann jedes druckbare Zeichen angegeben werden.

Zwischen *-e* und dem Symbol *symbol* darf kein Leerzeichen sein.

Wird das Fluchtsymbol gefolgt von einem Punkt (.) am Anfang einer Zeile eingegeben, wird die Sitzung am fernen Rechner sofort abgebrochen und zur Ausgangssitzung am lokalen Rechner zurückgekehrt.

Die Tastenfolge

↵ *symbol* .

bricht auf jeden Fall das Arbeiten am fernen Rechner ab und kehrt zu dem Rechner zurück, von dem aus *rlogin* aufgerufen wurde. Geöffnete Dateien werden nicht gesichert.

*-e*symbol nicht angegeben:
~ ist ein Fluchtsymbol, unabhängig davon, ob Sie ein zusätzliches Symbol angeben oder nicht. Bei deutscher Tastaturbelegung ist auch ß als Fluchtsymbol voreingestellt.

-l␣login
ist die Benutzerkennung, mit der sich der Benutzer am fernen Rechner anmelden kann. Sie muß angegeben werden, wenn die Benutzerkennung, unter der am lokalen Rechner gearbeitet wird, sich von der am fernen Rechner unterscheidet.

-l login nicht angegeben:
rlogin verwendet am fernen Rechner die gleiche Benutzerkennung wie am lokalen Rechner.

host
ist der Name des fernen Rechners, an dem Sie sich anmelden wollen.

DATEIEN

/etc/passwd
Paßwortdatei

*/usr/hosts/**
enthält die Version des Kommandos, die speziell für den durch *host* bestimmten Rechner vorgesehen ist

/etc/hosts.equiv
Liste von Rechnernamen; Benutzerkennungen eines fernen Rechners, dessen Name in dieser Liste steht, haben Zugang zum lokalen Rechner, wenn sie auch dort bekannt sind

$HOME/.rhosts
Liste der Rechnernamen mit Benutzerkennungen, die sich unter Ihrer Kennung anmelden dürfen

UMGEBUNGSVARIABLE

TERM
Typ der verwendeten Datensichtstation.

BEISPIELE

1. Ein Benutzer am lokalen Rechner *athen* möchte am fernen Rechner *bremen* mit der gleichen Benutzerkennung arbeiten.

```
$ rlogin bremen
Kennwort:
```

   Vom Benutzer wird ein Kennwort verlangt. Gibt er das Kennwort richtig ein, kann er wie gewohnt arbeiten.

2. Der Benutzer *hans* möchte am fernen Rechner *athen* unter der Benutzerkennung *franz* arbeiten. Er ist zugriffsberechtigt und definiert sich als Fluchtsymbol *&*.

```
$ rlogin athen -e\& -l franz
```

   Der Benutzer kann jederzeit seine Sitzung am Rechner *athen* mit der Tastenfolge ↵ *&* . beenden.

SIEHE AUCH

rsh, *stty*, *tty*, *login*
named, *hosts*, *hosts.equiv* [5]

rm
Dateien löschen
(remove files)

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.

```
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 und ein Fragezeichen ? ausgegeben. Wenn Sie diese Frage mit einem Wort, das mit *y* beginnt, oder mit *y* bejahen, wird der Eintrag gelöscht; anderenfalls bleibt er erhalten. Wenn die Standard-Eingabe keine Datensichtstation ist, wird der Eintrag ohne Rückfrage gelöscht.

option

-f
: Die Einträge werden ohne Rückfrage gelöscht. Wenn Sie kein Schreibrecht für das Dateiverzeichnis haben, werden Dateien nicht gelöscht.

-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 nichtleeren schreibgeschützten Dateiverzeichnisses ist nicht möglich (auch nicht mit der Option *-f*). In diesem Fall wird eine Fehlermeldung ausgegeben.

-i
: *rm* fragt für jede schreibgeschützte 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 Standardeingabe keine Datensichtstation ist.

--
: Mit -- kann das Ende der Optionen gekennzeichnet werden. Dies kann dazu benutzt werden, Dateinamen anzugeben, die mit - beginnen.

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.

BEISPIELE

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.

2. Löschen des Dateiverzeichnisses *norm* mit allen Dateien und Unterdateiverzeichnissen.

```
$ rm -r norm
```

SIEHE AUCH

rmdir
unlink [14]

rmail
Nachrichten senden

rmail ist eine eingeschränkte Version von *mail*, mit der Sie nur Nachrichten senden können. *uucp* verwendet *rmail* als Sicherheitsvorkehrung. Wenn Sie eine Applikation schreiben, die selbst Nachrichten erzeugt, sollten Sie statt *mail* in jedem Fall *rmail* verwenden.

```
rmail [-tw] [-m nachrichten_typ] empfänger ...
```

Die Beschreibung ist identisch mit der Beschreibung von *mail* im Sendemodus.

SIEHE AUCH

mail

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.

```
rmdir [-p] [-s] dateiverzeichnis...
```

Keine Option angegeben
rmdir löscht die angegebenen Dateiverzeichnisse.

-p
(p - parents) Das angebene Dateiverzeichnis wird gelöscht sowie alle übergeordneten Dateiverzeichnisse, die dadurch leer werden. Eine Meldung auf der Standardausgabe gibt an, ob der komplette Pfad entfernt wurde, oder ob ein Teil des Pfads erhalten blieb.

-s
Die Meldung bei der Option *-p* wird unterdrückt.

dateiverzeichnis
Name des zu löschenden Dateiverzeichnisses.
Sie können mehrere Dateiverzeichnisse angeben.

FEHLERMELDUNGEN

```
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: ../dv1: Can't remove current directory or ..
```

Das aktuelle bzw. übergeordnete Dateiverzeichnis kann nicht gelöscht werden.

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 root     root        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 root     root        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
unlink() [14]

rsh Shell-Kommando am fernen Rechner ausführen (remote shell)

Mit *rsh* können Sie ein Kommando an einem fernen Rechner ausführen.

Das Kommando wenden Sie z.B. an, wenn Sie

- sich über den Inhalt eines Dateiverzeichnisses am fernen Rechner informieren wollen,
- eine Datei am fernen Rechner ausdrucken wollen.

Das Kommando kann nur angewendet werden, wenn der Benutzer auf dem fernen Rechner zugriffsberechtigt ist. Zugriffsberechtigt ist der Benutzer dann, wenn

- für seine Benutzerkennung und seinen lokalen Rechner ein Eintrag in der Datei *$HOME/.rhosts* des Benutzers auf dem fernen Rechner vorliegt oder
- auf dem fernen Rechner die gleiche Benutzerkennung existiert, unter der er am lokalen Rechner arbeitet und der lokale Rechner in der Datei */etc/hosts.equiv* des fernen Rechners eingetragen ist.

Das Kommando kann nur dann ausgeführt werden, wenn am lokalen und 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 des Namens des fernen Rechners aufrufen.

Stop-Signal unterbrechen nur den *rsh*-Prozeß auf dem lokalen Rechner.

Die Umgebungsvariablen des lokalen Rechners werden nicht an den Kommandointerpreter auf der fernen Maschine weitergegeben.

```
rsh␣host[␣-n][␣-l␣login][␣cmd][␣args]
```

host
: ist der Name des fernen Rechners, an dem ein Shell-Kommando ausgeführt wird.

  Der Name des fernen Rechners muß in der Datei */etc/hosts* eingetragen sein oder in der Datenbasis für Namen in der Internet-Domäne (oder in beiden). Jeder Rechner hat einen offiziellen Namen (der erste Eintrag in der Datenbasis) und wahlweise einen oder mehrere Spitznamen. Sie können alle eingetragenen Namen verwenden.

-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 Standardeingabe liest (siehe *Beispiel 1*).

-n wird auch in einigen Fällen benötigt, wo es nicht ganz offensichtlich ist. *-n* wird u.a. dann benötigt, wenn ein *rsh*-Kommando in den Hintergrund gestartet wird.

Beispiel:

```
rsh host dd if=/dev/nrmt0 bs=20b | tar xvpBf -
```

versetzt Ihren Kommandointerpreter in einen merkwürdigen Zustand, da 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.

Das passiert nur, wenn *rsh* am Anfang einer Pipe steht und nicht von der Standard-Eingabe liest. Verwenden Sie *-n* nicht, wenn *rsh* tatsächlich von der Standard-Eingabe lesen soll. Wenn Sie in dem Kommando

```
tar cf - . | rsh host dd of=/dev/rmt0 obs=20b
```

die Option *-n* angeben, liest *rsh* fälschlicherweise von */dev/null* statt von der Pipe.

-l login

ist die Benutzerkennung, mit der sich der Benutzer am fernen Rechner anmelden kann. Sie muß angegeben werden, wenn die Benutzerkennung, unter der am fernen Rechner gearbeitet werden soll, anders ist, als die Benutzerkennung am lokalen Rechner.

cmd

ist das SINIX-Kommando, das am fernen Rechner ausgeführt werden soll.

cmd nicht angegeben:
rsh benutzt *rlogin*, um Sie am fernen Rechner anzumelden.

args

sind die Argumente, die mit dem Kommando am fernen Rechner verwendet werden.

Arbeitsweise

Das Kommando *rsh* überträgt

- seine Standardeingabe dem Kommando am fernen Rechner,
- die Standardausgabe des Kommandos am fernen Rechner auf die Standardausgabe des Kommandos *rsh*,
- die Standardfehlerausgabe des Kommandos am fernen Rechner auf die Standardfehlerausgabe 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, multiplan* usw. können nicht mit dem Kommando *rsh* aktiviert werden. Dafür ist das Kommando *rlogin* zu verwenden.

DATEIEN

/etc/hosts.equiv
: Liste von Rechnernamen; Benutzerkennungen eines fernen Rechners, dessen Name in dieser Liste steht, haben Zugang zum lokalen Rechner, wenn sie auch dort bekannt sind.

/$HOME/.rhosts
: Liste der Rechnernamen mit Benutzerkennungen, die sich unter Ihrer Kennung anmelden dürfen.

BEISPIELE

1. Der Benutzer *hans* möchte sich den Inhalt des Dateiverzeichnisses */usr/hans/test* auf dem fernen Rechner *athen* anschauen. Die Ausgabe am Bildschirm soll überlaufgesteuert sein. Die Benutzerkennung *hans* ist auch am fernen Rechner *athen* vorhanden.

```
$ rsh athen -n ls -l /usr/hans/test | more
oder
$ rsh athen -n ls -l /usr/hans/test | pg

insgesamt 120 KB
```

```
-rw------  1  hans        4 Jan    7   14:12   hallo
-rw------  1  hans       19 Dec    8   08:05   hallo_1
drwx-x--x  2  hans       23 Nov   10   09:20   test
                          .
                          .
                          .
-- More --
```

2. Der Benutzer *hans* möchte den Inhalt der fernen Datei */usr/hans/hallo_1* am fernen Rechner *athen* an die lokale Datei *test_1* anhängen. Die Benutzerkennung *hans* ist auch am fernen Rechner *athen* vorhanden.

   ```
   $ rsh athen cat /usr/hans/hallo_1 >> test_1
   ```

3. Der Benutzer *hans* möchte die ferne Datei */usr/hans/reise_1* an die ferne Datei */usr/hans/holland* anhängen. Beide Dateien befinden sich im Rechner *athen*. Die Benutzerkennung *hans* ist auch auf dem Rechner *athen* vorhanden.

   ```
   $ rsh athen cat /usr/hans/reise_1 ">>" /usr/hans/holland
   ```

4. Ein Benutzer möchte die ferne Datei */usr/emil/test* im Rechner *athen* am Rechner *athen* ausdrucken. Dem Benutzer steht am Rechner *athen* die Benutzerkennung *emil* zur Verfügung.

   ```
   $ rsh athen -l emil lpr /usr/emil/test
   ```

5. Ein Benutzer möchte die lokale Datei */usr/hans/dat* am fernen Rechner *athen* ausdrucken.

   ```
   $ rsh athen lpr < /usr/hans/dat
   ```

SIEHE AUCH

rlogin, *vi*
named, *hosts*, *hosts.equiv* [5]

ruptime
Zustand der Rechner im lokalen Netz anzeigen

Mit *ruptime* können Sie sich über den Zustand der Rechner im lokalen Netz informieren. Zu jedem Rechner werden folgende Informationen ausgegeben:

- der Name des Rechners,
- der Zustand (up / down),
- die Ein- bzw. Ausschaltzeit,
- die Anzahl der Benutzer,
- die Belastung.

Das Kommando kann z.B. angewendet werden, wenn ein Benutzer an einem überlasteten Rechner arbeitet und prüfen möchte, welcher Rechner weniger ausgelastet ist.

Das Kommando *ruptime* kann nur für solche Rechner eine Statuszeile ausgeben, für die es eine Rechner-Status-Datei im Dateiverzeichnis */var/spool/rwho* gibt.

Ferne Rechner können nur dann eine Status-Meldung liefern, wenn auf ihnen der Dämon *in.rwhod* aktiv ist. Der lokale Rechner kann die Status-Meldungen nur dann empfangen, wenn auch auf dem lokalen Rechner der Dämon *in.rwhod* aktiv ist.

```
ruptime[␣-a][␣-l][␣-r][␣-t][␣-u]
```

Keine Option angegeben
: Es wird eine Statuszeile für alle fernen Rechner im lokalen Netz ausgegeben. Die Ausgabezeilen werden nach den Rechnernamen sortiert.

-a
: kann angegeben werden, wenn angemeldete Benutzer, die länger als eine Stunde nicht mehr aktiv waren, in der Belastungsstatistik aufgeführt werden sollen.

-l
: kann angegeben werden, wenn die Ausgabeliste nach relativer Auslastung der Rechner sortiert sein soll.

-r

Die Sortierreihenfolge wird umgedreht. Dies betrifft die Optionen *-l*, *-t* und *u* oder, falls keiner dieser Schalter angegeben ist, die Sortierung nach Rechnernamen.

-r nicht angegeben:

Es wird nach aufsteigender Anordnung sortiert.

-t

kann angegeben werden, wenn die Ausgabeliste nach der Anschaltdauer der Rechner sortiert sein soll.

-u

kann angegeben werden, wenn die Ausgabeliste nach der Anzahl der Benutzer sortiert sein soll.

Arbeitsweise

Die Statuszeilen werden aus den Nachrichtenpaketen aufgebaut, die von jedem Rechner einmal in der Minute gesendet werden.

Rechner, von denen seit fünf Minuten kein Nachrichtenpaket empfangen wurde, werden als *nicht aktiv* vermerkt.

Angemeldete Benutzer, die seit mehr als einer Stunde nicht mehr aktiv waren, werden in der Belastungsstatistik nur angezeigt, wenn die Option *-a* gesetzt ist.

Die Aktivitäten des Dämon *in.rwhod* können in großen Netzen zu einer starken Systemauslastung führen, da von jedem Rechner einmal pro Minute eine Status-Meldung empfangen wird und auf Platte abgelegt werden muß. In diesem Fall empfiehlt es sich, den Dämon mit dem Kommando *kill* zu beenden und in der Datei */etc/inet/rc.inet* die Variable *RWHOD* zurückzusetzen. Denn *in.rwhod* wird nur dann gestartet, wenn *RWHOD* gesetzt ist.

DATEI

*/var/spool/rwho/whod.**
Dateien mit Informationen, die *ruptime* verarbeitet

BEISPIEL

Ein Benutzer möchte an einem Rechner arbeiten, der weniger ausgelastet ist.

```
$ ruptime

Rottach     down     2:27
amsterdam     up     2:01,     0 users,  load  8.00,  8.12,  8.23
athen         up 18+18:17,     5 users,  load  0.06,  0.26,  0.50
augsburg      up     0:39,     1 user,   load  0.00,  0.00,  0.00
babel         up  6+21:48,     3 users,  load  0.17,  0.16,  0.00
baccardi      up  1+21:02,     0 users,  load  0.07,  0.07,  0.00
beaverton     up 28+20:29,     2 users,  load  0.13,  0.12,  0.00
berlin      down    13:49
bern          up  2+02:46,     1 user,   load  6.96,  6.74,  6.80
bremen        up  1+00:18,     2 users,  load 13.43,  9.04,  7.94
```

Im ersten Ausgabefeld ist der Name des Rechners angegeben.

Im zweiten Feld ist verzeichnet, ob der Rechner aktiv, *up*, oder nicht aktiv, *down* ist.

Im dritten Feld ist die Zeit angegeben, seit wann der Rechner läuft bzw. nicht mehr läuft. Das Format der Zeitangabe ist:

Tag+Stunde:Minute

Im vierten Feld ist die Anzahl der angemeldeten Benutzer eingetragen.

Im fünften Feld ist die durchschnittliche Belastung des Rechners angezeigt. Die Anzeige gilt für die letzten 5, 10 und 15 Minuten. Die Zahlen sind die durchschnittliche Anzahl der Prozesse, die ausführbereit sind und um den Prozessor konkurrieren.

SIEHE AUCH

rwho, *rup*, *rusers*
rwhod [5], [10], [11]

rwho
Aktive Benutzerkennungen im Netz anzeigen

Mit *rwho* können Sie sich über Benutzer informieren, die an den Rechnern im lokalen Netz angemeldet sind.

Die Ausgabe enthält für jeden Benutzer

- die Benutzerkennung, unter der er angemeldet ist,
- den Namen des Rechners, an dem er arbeitet,
- den Namen des Bildschirms,
- Datum und Uhrzeit wann sich der Benutzer angemeldet hat,
- die Zeit, die seit der letzten Eingabe vergangen ist.

Das Kommando kann z.B. dann angewendet werden, wenn ein Benutzer eine Nachricht an andere Benutzer des Netzes senden will. Er kann vorher feststellen, ob die Benutzer aktiv sind.

Das Kommando *rwho* kann nur über die Benutzer der Rechner informieren, für die es eine Rechner-Status-Datei im Dateiverzeichnis */var/spool/rwho* gibt.

Ferne Rechner können nur dann eine Status-Meldung liefern, wenn auf ihnen der Dämon *in.rwhod* aktiv ist. Der lokale Rechner kann die Statusmeldungen nur dann empfangen, wenn auch auf dem lokalen Rechner der Dämon *in.rwhod* läuft.

```
rwho [-a]
```

Keine Option angegeben
: Es werden alle aktiven Benutzerkennungen angezeigt.

-a
: kann angegeben werden, wenn auch die Benutzerkennungen angezeigt werden sollen, die seit mehr als einer Stunde keine Eingabe an ihrem Bildschirm gemacht haben.

Arbeitsweise

Die Informationen, die das Kommando *rwho* ausgibt, werden aus den Statuszeilen aufgebaut, die in den Rechner-Status-Dateien */var/spool/rwho* hinterlegt sind.

Die Statuszeilen werden aus den Nachrichtenpaketen aufgebaut, die von jedem Rechner einmal in der Minute gesendet werden.

Rechner, von denen seit fünf Minuten kein Nachrichtenpaket empfangen wurde, werden als *nicht aktiv* vermerkt.

Die Zeitspanne, die verstreicht, wenn ein Benutzer eine Minute oder länger keine Eingabe mehr macht, wird auch in der Statuszeile ausgegeben.

rwho arbeitet nicht über Gateway-Rechner.

Die Aktivitäten des Dämon *in.rwhod* können in großen Netzen zu einer starken Systemauslastung führen, da von jedem Rechner einmal pro Minute eine Status-Meldung empfangen wird und auf Platte abgelegt werden muß. In diesem Fall empfiehlt es sich, den Dämon mit dem Kommando *kill* zu beenden und in der Datei */etc/inet/rc.inet* die Variable *RWHOD* zurückzusetzen. Denn *in.rwhod* wird nur dann gestartet, wenn *RWHOD* gesetzt ist.

DATEI

/var/spool/rwho/whod.*
Dateien mit Informationen, die *rwho* verarbeitet

BEISPIEL

Ein Benutzer möchte wissen, welche Benutzer an den Rechnern im Netz aktiv sind.

```
$ rwho

admin    baccardi:tty00     Oct 28 10:47
admin    tahiti:tty00       Oct 28 10:33 :14
and      kempten:tty04      Oct 28 08:11
anna     troja:ttyp2        Oct 28 10:54 :02
anna     wuerzburg:console  Oct 28 08:48 :29
baerbel  wuerzburg:tty03    Oct 28 09:08 :18
bogi     neuhaus:console    Oct 28 07:54
claus    babel:ttypa        Oct 27 10:50
claus    babel:ttypc        Oct 27 10:51
claus    tokio:ttyp0        Oct 28 16:06 :43
daemon   athen:ttyp0        Oct 28 08:47 :29
gast     oslo:console       Oct 26 23:46 :03
julia    muenchen:tty04     Oct 28 11:18 :30
root     athen:ttyh5        Oct 28 09:02 :02
root     hadern:tty00       Oct 28 10:37
stef     portland:ttyp0     Oct 28 09:04
```

Im ersten Feld der Ausgabeliste ist die Benutzerkennung angegeben.

Im zweiten Feld ist der Rechnername und nach dem Doppelpunkt der Name des Bildschirms verzeichnet.

Im dritten Feld steht das Datum und die Uhrzeit, zu der sich der Benutzer am Rechner angemeldet hat.

Im vierten Feld ist die Zeitspanne in Minuten verzeichnet, seit der keine Eingabe mehr am Bildschirm gemacht wurde.

SIEHE AUCH

finger, *ruptime*, *rusers*
rwhod [5], [10], [11]

sag
Systemaktivität graphisch anzeigen (system activity graph)

sag wertet die Binärdaten aus einem vorhergehenden *sar* Aufruf aus und erstellt daraus eine Grafik über die Systemaktivitäten. Die Graphik wird am Bildschirm dargestellt. Jede der Dateneinheiten des *sar* Kommandos kann einzeln oder kombiniert aufgezeigt werden. Einfache Rechenarithmetik kann angegeben werden.

sag ruft *sar* auf und erhält die gewünschten Daten durch Zeichenkettenvergleich der Spaltenüberschriften. Diese sind beim *sar* Kommando beschrieben.

```
sag [␣option] ...
```

Keine Option angegeben
sag stellt die Tagesauslastung des Systems graphisch dar.

option

Die im folgenden beschriebenen Optionen werden von *sag* an das Kommando *sar* weitergegeben:

Das Argument *zeit* für die folgenden Optionen muß im Format hh[:mm] angegeben werden, wobei *hh* für Stunden und *mm* für Minuten steht.

-s[␣zeit]
Nur die Daten, die später als zum Zeitpunkt *zeit* entstanden sind, werden für die Grafik ausgewertet.

zeit nicht angegeben:
Die Voreinstellung für *zeit* ist 08:00 Uhr.

-e[␣zeit]
Nur die Daten, die bis zum Zeitpunkt *zeit* entstanden sind, werden für die Grafik ausgewertet.

zeit nicht angegeben:
Die Voreinstellung für *zeit* ist 18:00 Uhr.

-i␣sekunden
Nur die Daten, die in *sekunden*-Intervallen entstehen, werden für die Grafik ausgewertet. Ist keine Selektion im exakten *sekunden* Intervall möglich, wird der nächstmögliche Wert für Sekunden als Intervall gewählt.

-f␣datei
Die angegebene Datei *datei* wird als Datenquelle für *sar* benutzt.

-f␣datei nicht angegeben:
Die Voreinstellung ist die aktuelle Tagesdatei */usr/adm/sa/sa<dd>*, wobei <dd> für den jeweils aktuellen Tag steht.

Weitere Optionen

-T␣terminal
Die Ausgabe wird für einen Bildschirm des Typs *terminal* aufbereitet.
Vorgehensweise: *sag* ruft die Shell-Prozedur */bin/tplot* auf, die wiederum zu Terminal-spezifischen Plot-Programmen verzweigt.
Folgende Terminaltypen werden unterstützt: 450, 300, 300s, ver, tek, T4014.
Für die Terminaltypen 97801 - 97808 sind keine Anpassungen vorhanden bzw. technisch realisierbar.

-T terminal nicht angegeben:
Die Voreinstellung für *terminal* ist der Wert der Umgebungsvariablen *$TERM*.

Die Argumente für die nächsten zwei Optionen müssen dem folgenden Format entsprechen:

"name[␣op␣name]...[lo␣hi]"

name
Kann eine Zeichenkette sein, die einer Spaltenüberschrift der Ausgabe des sar Kommandos entspricht. Dahinter kann ein optionaler Gerätenamen in eckigen Klammern angegeben werden (z.B. r+w/s[dsk-1]). Andernfalls kann für name auch ein ganzzahliger numerischer Wert angegeben werden.

op
Die Operatoren +, -, * und / sind zulässig. Sie müssen jeweils von Leerzeichen umgeben sein.

Es können bis zu 5 Namen angegeben werden. Klammerung wird nicht erkannt. Unüblicherweise haben + und - Vorrang vor * und /. Die Auswertung erfolgt von links nach rechts.

lo
hi
optionale Grenzwerte. Sind sie nicht angegeben, so werden sie von den Daten abgeleitet.

-x␣spezifikation
Die Spezifikation für die x-Achse kann angegeben werden. Es kann nur eine einzige Spezifikation angegeben werden.

Falls diese Option nicht angegeben ist:
Die Zeit *zeit* wird als Spezifikation für die x-Achse verwendet.

-y␣spezifikation

Die Spezifikation für die y-Achse kann angegeben werden. Hier können bis zu 5 Spezifikationen, durch Strichpunkte getrennt, angegeben werden.

Falls diese Option nicht angegeben ist:
Die voreingestellte Spezifikation für die y-Achse ist:

"%usr 0 100; %usr + %sys 0 100; %usr + %sys + %wio 0 100"

DATEIEN

/usr/adm/sa/sa<dd>

Datendatei für den Tag <dd>

bin/tplot

Shell-Skript, das je nach Terminal-Angabe zum speziellen Terminal-Plot-Programm verzweigt.

SIEHE AUCH

sar

sar
Über Systemtätigkeit berichten

sar sammelt die Werte, die die Aktivitätszähler des Betriebssystems in von Ihnen bestimmten Zeitabständen angesammelt haben. Das Ergebnis wird in einer Datei in Binärform gesichert, wenn Sie beim Aufruf eine Datei angeben (Format 1). Sie können sich Daten dieser Datei, die während eines von Ihnen bestimmten Zeitraums eingetragen wurden, ausgeben lassen (Format 2).

```
sar[␣option]...[␣-o␣datei]␣t[␣n]                                    Format 1
sar[␣option]...[␣-s␣zeit][␣-e␣zeit][␣-i␣sek][␣-f␣datei]             Format 2
```

Format 1: Über Systemtätigkeiten berichten

sar[␣option]...[␣-o␣datei]␣t[␣n]

option

Mit folgenden Optionen können Sie Teilmengen der Daten angeben, über die *sar* berichten soll.

-a

sar berichtet über die Anwendung von Systemroutinen für den Dateizugriff:

Ausgabe	Bedeutung
iget/s	Anzahl der in S5- und UFS-Dateisystemen pro Sekunde über den Inode-Eintrag ermittelten Dateien
namei/s	Anzahl, wie oft pro Sekunde ein Pfadname im Dateisystem gesucht wurde
dirbk/s	Anzahl der pro Sekunde gelesenen Dateiverzeichnis-Blöcke im S5-Dateisystem

-A

sar berichtet über alle Daten (entspricht der Angabe aller übrigen Optionen).

-b

sar berichtet über die Pufferauslastung.

Ausgabe	Bedeutung
bread/s,bwrit/s	Daten-Übertragungen pro Sekunde zwischen Systempuffer und Festplatte oder anderen blockorientierten Geräten
lread/s,lwrit/s	Zugriffe auf Systempuffer pro Sekunde
%rcache,%wcache	Cache-Speicher-Treffer-Verhältnis, d.h. %rcache 1-bread/lread (in Prozent) %wcache 1-bwrit/lwrit (in Prozent)
pread/s,pwrit/s	physikalische Datenübertragungen über rohe (ungepufferte) Geräte pro Sekunde

Zusammen mit Option *-D*:
Über die Cache-Puffer-Aktivitäten für am lokalen Rechner über Netz (RFS, Remote File Sharing) montierte Resourcen wird berichtet.

-c

sar berichtet über Systemaufrufe.

Ausgabe	Bedeutung
scall/s	alle Arten von Systemaufrufen pro Sekunde
sread/s,swrit/s, fork/s,exec/s	spezielle Systemaufrufe
rchar/s,wchar/s	Zeichen, die durch read()- und write()-Systemaufrufe übertragen wurden

Zusammen mit Option *-D*:
Systemaufrufe werden unterteilt in empfangene, gesendete und rein lokale Aufrufe. Bei *exec()* und *fork()* wird nicht über empfangene und gesendete Systemaufrufe berichtet.

-C

sar berichtet über die Cache-Ausnutzung (caching overhead) der über RFS montierten Dateisysteme.

Ausgabe	Bedeutung
snd-inv/s	Anzahl der "invalidation messages" (Informationen über Dateiänderungen an Clients), die von Ihrem Rechner als Server gesendet werden
snd-msg/s	Anzahl der RFS-Meldungen, die pro Sekunde gesendet werden
rcv-inv/s	Anzahl der "invalidation messages" (Informationen über mögliche Änderung der Cache-Puffer durch Schreibzugriffe am Server-Rechner), die pro Sekunde vom fernen Server empfangen werden
rcv-msg/s	Anzahl aller ankommenden RFS-Meldungen, die pro Sekunde empfangen werden
dis-bread/s	Anzahl der read messages, die für das Caching verfügbar wären, wenn es nicht durch eine invalidation message abgebrochen worden wäre. Es wird der Verlust angezeigt, der durch die invalidation message verursacht wird
blk-inv/s	Anzahl der Pages, die vom Client Cache als Antwort auf cache invalidation messages entfernt wurden

-d

sar berichtet über die Aktivität aller blockorientierten Geräte wie z.B. Festplatte, Magnetbandlaufwerk, mit Ausnahme von XDC-Festplatten und XDC-Magnetbandlaufwerken. Bei der Ausgabe von Daten wird gewöhnlich *dsk-* zur Geräteangabe eines Plattenlaufwerks verwendet. Die Angabe für ein Magnetbandlaufwerk ist maschinenabhängig.

Ausgabe	Bedeutung
%busy	Zeit, die das Gerät mit der Ausführung eines Übertragungsauftrags beschäftigt war
avque	durchschnittliche Anzahl der Aufträge, die während der in *%busy* angegebenen Zeit in der Auftragswarteschlange standen
r+w/s	Anzahl der Datenübertragungen vom oder zum Gerät pro Sekunde
blks/s	Anzahl der pro Sekunde übertragenen Bytes in 512-byte-Einheiten
avwait	durchschnittliche Zeit in Millisekunden, die Datenübertragungsaufträge untätig in der Warteschlange verbringen
avserv	durchschnittliche Zeit in Millisekunden, in der Datenübertragungsaufträge bearbeitet werden (bei Festplatten einschließlich der für Suche, Zugriff und Übertragung benötigten Zeit)

-D

sar berichtet über die Aktivitäten der über RFS am lokalen Rechner montierten Dateisysteme.

Zusammen mit *-u* oder *-c* angegeben:
sar erstellt einen Bericht für das RFS.

Weder *-u* noch *-c* angegeben:
Als Standard wird *-u* automatisch gesetzt.

-g

sar berichtet über Paging-Aktivitäten.

Ausgabe	Bedeutung
pgout/s	Anforderungen zum Auslagern von Pages pro Sekunde
ppgout/s	Auslagerungen von Pages pro Sekunde
pgfree/s	Pages, die pro Sekunde vom Seitenverwaltungsdämon (page stealing daemon) auf die freie Liste geschrieben werden
pgscan/s	Pages, die pro Sekunde vom Seitenverwaltungsdämon (page stealing daemon) gesucht werden
%s5ipf	Prozentsatz der von *iget* von der freien Liste genommenen S5-Dateisystem-Inodes, die mit wiederverwendbaren Pages verbunden waren. Diese Pages werden vernichtet und können von Prozessen nicht wieder angefordert werden.

-k

sar berichtet über Speicher-Reservierungsaktivitäten des Betriebssystemkerns (Kernel Memory Allocation - KMA).

Ausgabe	Bedeutung
sml_mem,alloc,fail	Informationen über den Hauptspeicher-Pool (memory pool), der für kleine Speicheranforderungen (< 256 byte) Speicherplatz reserviert und belegt: Speicherplatz in Bytes, den KMA für den kleinen Pool zur Verfügung hat, Anzahl Bytes, die für kleine Speicherplatzanforderungen belegt sind und Anzahl der kleinen Speicherplatzanforderungen, die nicht befriedigt wurden.
lg_mem,alloc,fail	Information über den großen Hauptspeicher-Pool (512 byte bis 4 Kbyte) (analog zum kleinen Hauptspeicher-Pool s.o.)
ovsz_alloc,fail	Speicherplatz, der für übergroße Anforderungen (> 4 Kbyte) belegt ist und Anzahl der übergrossen Anforderungen, die nicht befriedigt werden konnten. Da übergroßer Speicherplatz dynamisch reserviert und belegt wird, gibt es hierfür keinen Pool.

-m

sar berichtet über Message- und Semaphoraktivitäten.

Ausgabe	Bedeutung
msg/s	Anzahl der Sende- und Empfangs-operationen pro Sekunde
sema/s	Anzahl der Semaphoraktivitäten pro Sekunde

-p

sar berichtet über das Paging.

Ausgabe	Bedeutung
pgin/s	Anforderungen pro Sekunde zum Einlagern von Pages
ppgin/s	Pro Sekunde eingelagerte Pages
vflt/s	Adressfehler (gültige Page nicht im Hauptspeicher)
pflt/s	Zugriffsfehler (ungültiger Zugriff auf Page)
slock/s	Durch software-lock-Anforderungen, die physikalische Ein-Ausgabe verlangen, verursachte Fehler

-q

sar berichtet über die durchschnittliche Warteschlangenlänge und die Auslastung.

Ausgabe	Bedeutung
runq-sz,%runocc	Ablauf-Warteschlange der Prozesse, die ablauffähig im Hauptspeicher stehen
swpq-sz,%swpocc	Austausch-Warteschlange der Prozesse, die ablauffähig ausgelagert sind. Hierüber wird nicht mehr berichtet

-r

sar berichtet über unbenutzte Speicherplatzpages und Festplattenblöcke.

Ausgabe	Bedeutung
freemem	Durchschnittliche Anzahl von Pages, die für Benutzer-Prozesse zur Verfügung stehen
freeswap	Festplattenblöcke, die für die Auslagerung von Pages (page swapping) zur Verfügung stehen

-S

sar berichtet über den Server- und Auftrags-Warteschlangenstatus.

Ausgabe	Bedeutung
serv/lo-hi	Durchschnittliche Anzahl der RFS-Server am System (*lo* gibt die mindeste, *hi* die höchstmögliche Anzahl von Servern an).
request %busy	% der Zeit, während der Empfangsdeskriptoren in der Auftragswarteschlange stehen
request avg lgth	Durchschnittliche Anzahl von Empfangsdeskriptoren, die auf Bedienung warten, wenn die Warteschlange besetzt ist
server %avail	% der Zeit, in der die Server untätig sind
server avg avail	Durchschnittliche Anzahl untätiger Server

-u

sar berichtet über die CPU-Auslastung (= Standard).

Ausgabe	Bedeutung
%usr	Laufzeitanteil im Benutzermodus
%sys	Laufzeitanteil im Systemmodus
%wio	untätig, Prozeß wartet auf blockorientierte Ein-/Ausgabe
%idle	untätig

Zusammen mit *-D*:
%sys wird unterteilt in

%sys remote — Zeit, die für Aufträge von fernen Rechnern verbraucht wurde

%sys local — Zeit für lokale Systemtätigkeiten

-v

sar berichtet über den Prozeß-Status, die Indexeinträge (i-nodes) und die Dateitabellen (file tables).

Ausgabe	Bedeutung
proc-sz, inod-sz, file-sz,lock-sz	Einträge/Größe für jede Tabelle (Auswertung erfolgt einmal pro Intervall)
ov	Überläufe, die zwischen den Intervallen bei jeder Tabelle auftreten

-w

sar berichtet über Systemeinlagerungs- und Auslagerungstätigkeiten (swapping, switching).

Ausgabe	Bedeutung
swpin/s,swpot/s bswin/s,bswot/s	Anzahl der Übertragungen und Anzahl der 512-byte-Einheiten, die zum Einlagern und Auslagern übertragen wurden (einschließlich erstmaligen Ladens einiger Programme)
pswch/s	Prozeßauslagerungen (switches)

-x

sar berichtet über Remote File Sharing Operationen.

Ausgabe	Bedeutung
open/s,create/s, lookup/s,readdir/s, getpage/s,putpage/s, other/s	Anzahl folgender Operationen, die pro Sekunde empfangen und vom Server gesendet werden: open, create, lookup, readdir, getpage, putpage und andere

-y
sar berichtet über Aktivitäten an der Datensichtstation.

Ausgabe	Bedeutung
rawch/s	Eingabezeichenrate
canch/s	über die kanonische Warteschlange verarbeitete Zeichenrate
outch/s	Ausgabezeichenrate
rcvin/s	Empfangs-Unterbrechungsrate
xmtin/s	Übertragungs-Unterbrechungsrate
mdmin/s	Modem-Unterbrechungsrate

-o␣datei
Die Ausgabe von *sar* wird in binärer Form in die Datei *datei* geschrieben.

t
Zeit in Sekunden, in der der Aktivitätszähler arbeiten soll. *t* sollte grösser sein als 5, da sonst *sar* selbst mitgezählt wird.
Wird *t* zusammen mit mehr als einer Option angegeben, so wird die Ausgabe sehr unübersichtlich.

n
Anzahl der Intervalle, in denen der Aktivitätszähler *t* Sekunden arbeiten soll.

n nicht angegeben:
Für *n* wird standardmäßig 1 eingesetzt.

Format 2: In einer Datei aufgezeichnete Systemtätigkeiten ausgeben

sar[␣option]...[␣**-s**␣zeit][␣**-e**␣zeit][␣**-i**␣sek][␣**-f**␣datei]

option
siehe *Format 1*

-s␣zeit
Startzeit des Berichts
Angabe in der Form: hh[:mm[:ss]]

-e␣zeit
Endezeit des Berichts
Angabe in der Form: hh[:mm[:ss]]

-i␣sek
Datensätze werden in *sek* Sekundenabständen ausgewählt und ausgegeben.

-i nicht angegeben:
Alle in der Datei gefundenen Intervalle werden ausgegeben.

-f␣datei
Name der Datei, aus der *sar* die Daten entnimmt.

-f nicht angegeben:
sar liest die Daten von der Tagesdatendatei */usr/adm/sa/sa*dd, wobei *dd* für den aktuellen Tag steht.

DATEI

*/usr/adm/sa/sa*dd
Tagesdatendatei für Systemtätigkeiten.
dd steht für den aktuellen Tag.

BEISPIELE

1. Über CPU-Aktivitäten des Tages bis zum Zeitpunkt des Aufrufs von *sar* berichten:

```
$ sar
```

2. Über in 12 Minuten ablaufende CPU-Aktivitäten berichten und Daten in der Datei *cpu_akt* speichern:

```
$ sar -o cpu_akt 60 12
```

3. Später die Auslastung von Festplatte und Magnetband während dieses Zeitraums überprüfen:

```
$ sar -d -f cpu_akt
```

SIEHE AUCH

sag, sar [7]

script Sitzung protokollieren

script schreibt alles, was während Ihrer Sitzung auf den Bildschirm ausgegeben wurde, in eine Datei. *script* beendet das Protokoll, wenn die aktuelle Shell beendet ist oder wenn Sie die Taste END (CTRL d) drücken.

```
script[␣-a][␣datei]
```

-a
script fügt das Protokoll an den Inhalt der Protokolldatei an.

-a nicht angegeben: Die Protokolldatei wird neu angelegt.

datei
Datei, in die *script* das Protokoll schreibt.

datei nicht angegeben: *script* schreibt das Protokoll in die Datei *typescript*.

BEISPIEL

```
$ script protokoll
Script started, file is protokoll
$ cat >prog.c
main()
{
   printf("Guten Morgen!\n")
}
END
$ cc prog.c
c1:
prog.c    4: [syntax]:   ';' expected
c1: errors: 1, warnings: 0
$ END Script done, file is protokoll
$
```

script legt im aktuellen Dateiverzeichnis eine Datei *protokoll* an. Durch Drücken der Taste END beenden Sie zuerst nur die Eingabe für *cat*. Erst durch das zweite END beenden Sie *script*. Die Datei *protokoll* hat dann folgenden Inhalt:

```
Script started on Thu Sep  6 11:57:51 1990
$ cat >prog.c
main()
{
   printf("Guten Morgen!\n")
}
$ cc prog.c
c1:
prog.c    4: [syntax]:   ';' expected
c1: errors: 1, warnings: 0
$
script done on Thu Sep  6 11:58:43 1990
```

sdiff
Dateien vergleichen und nebeneinander ausgeben

sdiff gibt die Unterschiede zwischen zwei Dateien auf die Standard-Ausgabe aus. *sdiff* benutzt hierfür die Ausgabe des Kommandos *diff*. *sdiff* gibt die Zeilen der beiden Dateien nebeneinander auf die Standard-Ausgabe aus und kennzeichnet dabei unterschiedliche Zeilen. In der Ausgabe stehen links Zeilen aus *datei1*, rechts Zeilen aus *datei2*. An den Zeichen dazwischen erkennen Sie, welche Zeilen identisch und welche unterschiedlich sind. Zeilen, die nur in *datei1* vorhanden sind, werden durch ein Kleinerzeichen < , Zeilen, die nur in *datei2* vorhanden sind, werden durch ein Größerzeichen > gekennzeichnet. Unterschiedliche Zeilen, vor oder zwischen denen eine in beiden Dateien gleiche Anzahl entweder identischer oder mit | gekennzeichneter Zeilen liegt, werden durch ein | gekennzeichnet.

```
sdiff[␣option] ...␣datei1␣datei2
```

option

-l
Bei identischen Zeilen soll nur *datei1* ausgegeben werden.

-o␣ausgabe_datei
ausgabe_datei ist der Name einer dritten Datei, in die die Ausgabe von *sdiff* nach Kriterien, die Sie festlegen, geschrieben wird. Standardmäßig werden in *ausgabe_datei* identische Zeilen gespeichert. Unterschiede werden am Bildschirm wie bei *diff* in Gruppen ausgegeben, z.B. alle aufeinanderfolgenden Zeilen mit dem Kleinerzeichen < . Dann wird die Ausgabe am Bildschirm unterbrochen und *sdiff* fordert Sie mit dem Prozentzeichen % zu einer der folgenden Eingaben auf, die Sie mit ↵ abschließen müssen. Sie können auch mehrere dieser Angaben direkt aneinandergehängt eingeben.

e[␣]l
Aufruf des Editors *ed*. Im *ed*-Puffer befindet sich die linke Spalte der zuletzt ausgegebenen unterschiedlichen Zeilen.

e[␣]r
Aufruf des Editors *ed*. Im *ed*-Puffer befindet sich die rechte Spalte der zuletzt ausgegebenen unterschiedlichen Zeilen.

e[␣]b
Aufruf des Editors *ed*. Im *ed*-Puffer befinden sich hintereinander die linke und rechte Spalte der zuletzt ausgegebenen unterschiedlichen Zeilen.

e
Aufruf des Editors *ed*. Der *ed*-Puffer ist leer.

l
Die linke Spalte der bisher erfolgten Ausgabe von *sdiff* wird an den Inhalt der Datei *ausgabe_datei* angefügt.

q
Verlassen des Programms.

r
Die rechte Spalte der bisher erfolgten Ausgabe von *sdiff* wird an den Inhalt der Datei *ausgabe_datei* angefügt.

s
In der weiteren Ausgabe am Bildschirm werden identische Zeilen unterdrückt. Vor der nächsten Gruppe von unterschiedlichen Zeilen wird zusätzlich ein *ed-skript* (siehe *ed*) ausgegeben, das die *ed*-Kommandofolge angibt, mit der *datei1* in *datei2* umgewandelt werden kann.

v
Hebt *s* auf.

Beim Verlassen des Editors wird der *ed*-Puffer an den Inhalt der Datei *ausgabe_datei* angefügt.

-s
Identische Zeilen werden nicht ausgegeben.

-w[␣]n
Mit der Zahl *n* können Sie die Anzahl der Spalten bestimmen, die die Ausgabezeilen haben sollen.

-w␣n nicht angegeben:
Die Ausgabezeile ist 130 Zeichen breit.

datei1␣datei2
Namen der beiden Dateien, die *sdiff* vergleichen soll. Wenn Sie für eine der beiden Dateien ein Dateiverzeichnis angeben, vergleicht *sdiff* die zweite Datei mit einer Datei gleichen Namens in dem angegebenen Dateiverzeichnis. *sdiff* meldet einen Fehler, wenn es eine der Dateien nicht findet oder nicht darauf zugreifen kann. Für die Dateien müssen Sie das Leserecht, für das Dateiverzeichnis das Ausführrecht besitzen (siehe *chmod*).

Wenn Sie für *datei2* einen Bindestrich - angeben, liest *sdiff* von der Standard-Eingabe und vergleicht die von dort gelesenen Zeichen mit *datei1*.

BEISPIEL

Die beiden Dateien *datei1* und *datei2* haben folgenden Inhalt:

```
datei1:          datei2:
 x                y
 a                a
 b                d
 c                c
 d
```

Mit *sdiff* erhalten Sie folgende Ausgabe:

```
$ sdiff datei1 datei2
x       |        y
a                a
b       <
c       <
d                d
        >        c
```

SIEHE AUCH

diff, *ed*

sed
Editor im Prozedurbetrieb

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, müssen Sie die Standard-Ausgabe in eine Datei umlenken.

```
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: einmal wegen der standardmäßigen Ausgabe aller bearbeiteten Zeilen und einmal 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 es in Hochkommata 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 mit einer Kopie 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, die die zuletzt in den Musterspeicher kopierte Zeile adressieren, werden nacheinander auf den Inhalt des Musterspeichers angewandt. Dabei kann der Inhalt des Musterspeichers verändert werden oder auch nicht. Das hängt von den *sed*-Kommandos ab.
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, Intervallgrenzen einschließlich, 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

adresse	Bedeutung:
$	letzte Zeile der Datei
n	*n*-te Eingabezeile, wobei *n* eine positive ganze Zahl ist. Die Nummern ergeben sich, indem alle Eingabezeilen fortlaufend durchnumeriert 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 nicht einzugeben! Sie zeigen 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. Führende Tabulator- oder Leerzeichen in *text* werden von *sed* entfernt. Um das zu verhindern, können Sie sie mit Gegenschrägstrich \ davor 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, und *text* wird ausgegeben.

[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 Bildschirmzeile zeigt an, daß die Textzeile in der nächsten Bildschirmzeile fortgesetzt wird.

[bereich]**n**
Der Inhalt des Musterspeichers wird auf die Standard-Ausgabe ausgegeben und dann sofort durch die nächste Eingabezeile ersetzt. Es werden also nur der 2. und 3. Schritt des Zyklus
ausgeführt.

[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**/r*A*/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. Sie dürfen maximal 10 verschiedene Dateien für *wdatei* verwenden.

Vorsicht

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]tmarke
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. Sie dürfen maximal 10 verschiedene Dateien für *wdatei* verwenden.

Vorsicht

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

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. Die schließende Klammer } muß am Zeilenanfang stehen. Es dürfen ihr nur Leerzeichen und Tabulatorzeichen 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 der bei der Bearbeitung veränderten oder nicht veränderten Eingabezeilen unterdrückt. Der auf #*n* folgende Zeileninhalt wird als Kommentar gewertet und nicht als *sed*-Kommando interpretiert.

FEHLERMELDUNGEN

`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.

BEISPIELE

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.

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.

3. Alle Zeilen einer Datei ausgeben, die keine Leerzeilen sind:

   ```
   $ sed '/^$/d' datei
   ```

   Alle Leerzeilen werden mit /^$/ adressiert und mit *d* gelöscht.

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.

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.

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

Das in die Bourne-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, macht *set* folgendes:
  - 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.

```
set[␣option...][␣argument]...
```

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.

-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.

-e

(e - exit) Die aktuelle Shell wird sofort beendet, falls ein Kommando einen 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 Ende-Status 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.

-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 *sh, WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?*).
Mit *-n* können Sie feststellen, ob eine Shell-Prozedur Syntax-Fehler enthält.

Vorsicht

Wenn Sie *set -n* im Dialog eingeben, können Sie die Option *-n* nur dadurch zurücksetzen, daß Sie die Shell mit END beenden.
In einer Shell-Prozedur können Sie die Option *-n* nicht zurücksetzen. Sie wird zurückgesetzt, wenn sich die Shell, die diese Shell-Prozedur ausführt, beendet.

-n nicht angegeben und noch nicht gesetzt:
Alle Kommandos werden gelesen, interpretiert und ausgeführt.

-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 einen Ende-Staus $\neq 0$ 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.

--

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 4*).

Mit *set* -- ändern Sie nicht die aktuell gesetzten Optionen.

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.

BEISPIELE

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
```

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.

3. Das Band einer eingelegten Magnetband-Kassette soll vor dem ersten Zugriff nicht nachgespannt werden. Das Kommando *mt* entnimmt der Shell-Variablen TAPE den Namen der Gerätedatei.
   Die entsprechende Shell-Prozedur *mtno* hat folgenden Inhalt:

```
set -u
mt -f $TAPE noret
```

   Weil die Option *-u* gesetzt ist, wird das Kommando *mt* nur ausgeführt, wenn die Shell-Variable definiert ist. So verhindern Sie, daß *mt* auf die Standard-Gerätedatei */dev/rmt0* zugreift.

   Wenn die Variable TAPE nicht definiert ist, bricht die Shell-Prozedur sofort mit der folgenden Fehlermeldung ab:

```
$ sh mtno
mtno: TAPE: parameter not set
```

4. 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, getopt, sh

Wegweiser durch die Beschreibung der Bourne-Shell sh

Die Beschreibung der Shell gliedert sich in drei Teile:

- Das Kommando *sh*
  - Login-Shell und Subshell
  - die Syntax des Komandos *sh*

- Die Shell als Kommandointerpreter
  - Kommandos eingeben und verknüpfen
  - Sonderzeichen der Shell
  - Shell-Variablen
  - Umgebung
  - Shell-Parameter
  - Kommandos durch ihre Ausgabe ersetzen
  - Eingabe und Ausgabe eines Kommandos umlenken
  - Argumente durch passende Dateinamen ersetzen
  - Wie bearbeitet die Shell die Kommandozeile?

- Die Shell als Programmiersprache
  - Kommentare in Shell-Prozeduren
  - Shell-Funktionen
  - Ablaufanweisungen der Shell

sh
Kommandointerpreter und Programmiersprache Bourne-Shell
Bourne-Shell sh

Die Shell ist die Benutzerschnittstelle zum Betriebssystem-Kern. Sie ist als SINIX-Kommando implementiert und kein Bestandteil des Kerns; die Shell und alle anderen SINIX-Kommandos kommunizieren mit dem Kern über Systemaufrufe.
Die Shell *sh* ist eine Version der Bourne-Shell. Was macht sie?

- Sie liest Eingaben von der Datensichtstation oder aus einer Datei, interpretiert sie nach bestimmten Regeln und sorgt für die Ausführung. Eine Datei, die Eingaben für die Shell enthält, heißt Shell-Prozedur.
- Die Shell läßt sich wie eine Programmiersprache anwenden. Sie können also mit den vorhandenen SINIX-Kommandos eigene Programme erstellen und ohne vorheriges Übersetzen ausführen.

Login-Shell

Diese Shell wird standardmäßig von *login* gestartet, wenn Sie sich an einer Datensichtstation angemeldet haben. Die Login-Shell ist mit der aufrufenden Datensichtstation verbunden, d.h. die Dateikennzahlen 0 für Standard-Eingabe, 1 für Standard-Ausgabe und 2 für Standard-Fehlerausgabe beziehen sich auf diese Datensichtstation. Bevor die Login-Shell ihr Bereitzeichen ausgibt, passiert der Reihe nach folgendes:

- Die Login-Shell weist einigen Variablen wie IFS, MAILCHECK, PATH, PS1 und PS2 Standard-Werte zu (siehe *SHELL-VARIABLEN*).
- Die Datei */etc/profile* wird ausgeführt.
- Die Datei *$HOME/.profile* wird ausgeführt, falls Sie diese Datei angelegt haben.

Nur in einer Login-Shell können Sie sich mit dem eingebauten *sh*-Kommando *login* neu am System anmelden.

Woran erkennen Sie, daß eine Login-Shell gestartet ist?
Das Kommando *echo $0* gibt den Aufrufnamen der aktuellen Shell aus. Die Ausgabe *-sh* bezeichnet eine Login-Shell, die Ausgabe *sh* eine normale Subshell.

Eine Login-Shell wird als Subshell gestartet, wenn Sie sich mit */bin/login* neu am System anmelden (siehe *SUBSHELL*).

Die Login-Shell beenden Sie mit der Taste END. Wenn die Login-Shell keine Subshell war, wird anschließend der Begrüßungsbildschirm ausgegeben, Ihre Sitzung ist also beendet.

Subshell

Mit *sh* rufen Sie eine neue Shell auf, auch Subshell genannt. Diese Shell ist derselben Datensichtstation zugeordnet wie die aufrufende Shell; außerdem erbt sie alle Shell-Variablen, die Sie in einer übergeordneten Shell definiert und exportiert haben (siehe *export* und Option *-a*). Sie können Shell-Variablen von der aktuellen Shell nur an Subshells weiterreichen, aber nicht an die übergeordnete Shell.

Im Gegensatz zur Login-Shell führt die Subshell die Dateien */etc/profile* und *$HOME/.profile* nicht aus. Sie können in einer Subshell beliebig Shell-Variablen definieren oder ändern bzw. beliebig das Dateiverzeichnis wechseln, ohne daß dies die übergeordnete Shell beeinflußt. Wenn sich die Subshell beendet, meldet sich die vorherige Shell in dem "Zustand" zurück, in dem sie die Subshell gestartet hat.

In einer Subshell wird das eingebaute *sh*-Kommando *login* abgewiesen. Sie können sich nur in einer Login-Shell mit *login* neu am System anmelden.

Mit *sh* können Sie den Ablauf der neuen Subshell steuern, entsprechend den Optionen, die Sie beim Aufruf angeben. Wenn Sie einen Dateinamen angeben, beendet sich die Subshell nach der Ausführung dieser Datei. Ist kein Dateiname angegeben, ist die Subshell interaktiv, d.h. sie liest solange die Eingaben von der Datensichtstation, bis Sie sie mit der Taste END beenden. Eine Shell, die eine Shell-Prozedur ausführt, ist nicht interaktiv (siehe auch Option *-i*).

```
sh[␣option...][␣datei][␣argument]...
```

Kein Operand angegeben
: Eine Subshell mit der Option *-s* wird aufgerufen. Der zuvor aktuelle Shell-Prozeß wird zum Vater der neuen Shell. Diese Shell können Sie mit END beenden.

Keine Option angegeben
: Wenn Sie *sh* mit Operanden aufrufen, die keine Optionen sind, müssen Sie in diesem Fall als ersten Operanden *datei* angeben. Dabei ist *datei* der Name einer Shell-Prozedur, für die Sie Leserecht haben müssen.
Eine Subshell wird aufgerufen, die die Kommandos der Shell-Prozedur der Reihe nach abarbeitet und sich anschließend wieder beendet.

option

steuert den Ablauf der aufgerufenen Subshell.
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.
Die Option *-c kommando* kann nur als letzte Option angegeben werden, bei den übrigen Optionen ist die Reihenfolge beliebig.

Wenn Sie den Ablauf der aktuellen Shell steuern wollen, verwenden Sie das eingebaute *sh*-Kommando *set*.

Es gibt zwei Arten von Optionen:

- Optionen, die Sie nur bei *sh* angeben können
- Optionen, die Sie auch beim eingebauten *sh*-Kommando *set* angeben können (siehe auch *set*).

Optionen nur für sh

-c␣kommando
(c - command) die Subshell führt *kommando* aus und beendet sich. Diese Option muß die letzte der beim Aufruf angegebenen Optionen sein. Auf *-c kommando* darf kein weiterer Operand folgen.

Mit der Option *-c kommando* können Sie Kommandos innerhalb eines C-Programms aufrufen.

Sie müssen *kommando* angeben, sonst erhalten Sie eine Fehlermeldung.

kommando
muß ein Aufrufargument sein. Enthält *kommando* Leer- oder Tabulatorzeichen, müssen Sie es in Anführungszeichen "..." oder Hochkommata '...' einschließen. Andernfalls interpretiert die Shell die Eingabe nur bis zum ersten Leerzeichen.

kommando ist das Kommando oder die ausführbare Shell-Prozedur, die in der aufgerufenen Subshell ausgeführt werden sollen. Für die Shell-Prozedur brauchen Sie Lese- und Schreibrecht.

Wenn Sie mehrere Kommandos oder Shell-Prozeduren angeben wollen, müssen Sie die entsprechenden Kommando-Trennzeichen entwerten.

Wenn Sie das Kommando oder die Shell-Prozedur mit dem einfachen Dateinamen aufrufen, sucht die Shell diese Datei in den Dateiverzeichnissen, deren absolute Pfadnamen der Variablen PATH zugewiesen sind.

-i
(i - interactive) eine interaktive Subshell wird aufgerufen. Diese Shell, auch Dialog-Shell genannt, liest Ihre Eingaben von der Standard-Eingabe, bearbeitet die Eingabe, startet die entsprechenden Kommandos und wartet auf deren Beendigung. Dann gibt sie ihr Bereitzeichen aus und bearbeitet weitere Eingaben. Auf die Signale 2 für SIGINT, 3 für SIGQUIT und 15 für SIGTERM reagiert eine interaktive Shell wie folgt:

Signal	erzeugt mit	interaktive Shell
SIGINT 2	DEL	Kommando läuft: ignorieren, d.h. eine interaktive Shell wird nicht beendet sonst: neues Bereitzeichen ausgeben
SIGQUIT 3	CTRL \	Kommando läuft: ignorieren sonst: ignorieren; eine Login-Shell gibt zusätzlich ein neues Bereitzeichen aus
SIGTERM 15	*kill -15 PID*	ignorieren; *PID* ist die Prozeß-Nummer der interaktiven Shell

Mit der Option *-i* können Sie innerhalb eines C-Programmes eine interaktive Shell aufrufen.

-i nicht angegeben:
Eine Shell ist standardmäßig interaktiv, wenn ihre Standard-Eingabe, -Ausgabe und -Fehlerausgabe einer Datensichtstation zugeordnet sind.
Eine Shell, die eine Shell-Prozedur ausführt, ist nicht interaktiv. Diese Prozedur-Shell reagiert auf die Signale SIGINT, SIGQUIT und SIGTERM wie folgt:

Signal	Prozedur-Shell	Prozedur-Shell &
SIGINT	auf Beendigung des laufenden Kommandos warten, dann abbrechen	ignorieren
SIGQUIT	ignorieren	ignorieren
SIGTERM	ignorieren	ignorieren

-p

(p - privileged) Die aufgerufene Shell setzt die effektive Benutzer- und Gruppennummer *nicht* auf die reale Benutzer- und Gruppennummer (also die des Aufrufers), sondern auf ihre eigene Benutzer- und Gruppennummer.

-r

(r - restricted shell) Eine eingeschränkte Subshell wird aufgerufen. 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 Kommando-Datei einen Schrägstrich / enthält. Sie können also nur solche Kommandos ausführen, die in Ihrem aktuellen Dateiverzeichnis stehen oder in den Dateiverzeich-

nissen, deren Pfade der Shell-Variablen PATH zugewiesen wurden.

- Kommandos werden abgewiesen, wenn der Aufruf die Zeichen > oder >> enthält. Die Ausgabe von Kommandos kann also nicht mit > oder >> in eine Datei umgelenkt werden.

Nur mit *sh -r* können Sie eine eingeschränkte Subshell aufrufen. Beachten Sie bitte, daß Sie mit *rsh* eine Shell an einem fernen Rechner aufrufen und nicht, wie in früheren Versionen von SINIX, eine eingeschränkte Shell.

Nur für den Systemverwalter

Als Systemverwalter können Sie eine Arbeitsumgebung für solche Benutzer einrichten, die nur mit der eingeschränkten Shell arbeiten sollen. Sie legen für diese Benutzer fest:

- in welchem Dateiverzeichnis sie arbeiten sollen.
  Die Benutzer sollten nicht im Login-Dateiverzeichnis arbeiten, sie sollten für das Login-Dateiverzeichnis nur Leserecht haben.
- welche Kommandos sie ausführen dürfen.
  Dazu können Sie ein Unterdateiverzeichnis im Login-Dateiverzeichnis anlegen und die erlaubten Kommandos dorthin kopieren.
  Es darf z.B. nicht möglich sein, eine Subshell aufzurufen, denn die Subshell einer eingeschränkten Shell ist nicht eingeschränkt.
- die Variable *PATH*

In der Datei *$HOME/.profile* tragen Sie die entsprechenden Kommandos ein und rufen die eingeschränkte Shell auf. Wenn die Login-Shell diese Datei ausgeführt hat, sollte der Benutzer in seinem Arbeitsdateiverzeichnis mit der eingeschränkten Shell arbeiten.

-r nicht angegeben:
Für die aufgerufene Subshell gelten die beschriebenen Einschränkungen nicht.

-s

(s - standard input) Die aufgerufene Subshell liest von der Standard-Eingabe und schreibt auf die Standard-Fehlerausgabe. Die eingebauten *sh*-Kommandos schreiben allerdings auf die Standard-Ausgabe.

Die aufgerufene Subshell versorgt die Stellungsparameter $1, $2, ... und $9 mit den ersten neun nach *-s* angegebenen Argumenten. Die Shell-Parameter $* und $@ versorgt die Subshell mit allen nach *-s* angegebenen Argumenten und den Shell-Parameter $# mit der Anzahl dieser Argumente. Sind weniger als neun Argumente angegeben, werden die restlichen Stellungsparameter mit der leeren Zeichenkette versorgt.

Diese Werte sind nur der aufgerufenen Subshell bekannt. Wenn Sie Stellungsparameter in der aktuellen Shell setzen wollen, verwenden Sie stattdessen das eingebaute *sh*-Kommando *set*.

Wenn Sie beim Aufruf außerdem *datei* angeben, versorgt die aufgerufene Subshell den Stellungsparameter $1 mit diesem Aufrufargument. Die in *datei* enthaltenen Kommandos werden also nicht ausgeführt. Wenn Sie mit *sh* eine Shell-Prozedur ausführen wollen, dürfen Sie die Option *-s* nicht angeben.

-s nicht angegeben:
-s wird standardmäßig gesetzt, wenn Sie *sh* ohne Operanden aufrufen.
Die Login-Shell wird immer mit *-s* aufgerufen.

Optionen für sh und set

-a

(a - automatic) In der aufgerufenen Subshell werden automatisch alle Shell-Variablen exportiert, die Sie neu definieren oder denen Sie einen anderen Wert zuweisen.

In dieser Subshell setzt *set +a* die Option *-a* zurück.

-a nicht angegeben:
Neu definierte oder geänderte Shell-Variablen werden in der aufgerufenen Subshell nicht automatisch exportiert. Nur wenn Sie eine Shell-Variable mit dem eingebauten *sh*-Kommando *export* exportieren, ist sie in einer weiteren Subshell bekannt.

-e

(e - exit) Die aufgerufene Subshell bzw. Shell-Prozedur wird sofort beendet, wenn ein Kommando einen Ende-Status ≠0 zurückgibt.

In dieser Subshell setzt *set +e* die Option *-e* zurück.

-e nicht angegeben:
Ist die aufgerufene Subshell interaktiv, können Sie sie mit END beenden. Ist die aufgerufene Shell eine Prozedur-Shell, wird sie nur dann vorzeitig beendet, wenn sie einen Syntax-Fehler entdeckt. Sonst wird die Prozedur-Shell beendet, wenn das letzte Kommando ausgeführt ist, unabhängig vom Ende-Status der einzelnen Kommandos.

-f

(f - file name) In der aufgerufenen Subshell bzw. Shell-Prozedur verlieren ab sofort die Zeichen Stern * sowie Fragezeichen ? sowie [...] ihre Sonderbedeutung. Das bedeutet, daß die entsprechenden Zeichenketten nicht durch passende Dateinamen ersetzt werden.

In dieser Subshell setzt *set +f* die Option *-f* zurück.

-f nicht angegeben:
Entsprechende Zeichenketten werden durch passende Dateinamen ersetzt.

-h

(h - hash) Die aufgerufene Subshell ändert ihr Verhalten bei der Definition von Shell-Funktionen (siehe *Beispiel 2*):
Wenn Sie in der aufgerufenen Subshell eine Shell-Funktion definieren, trägt die Shell bereits alle innerhalb der Funktion verwendeten Kommandos in die Hash-Tabelle ein (siehe *hash*).

In dieser Subshell setzt *set +h* die Option *-h* zurück.

-h nicht angegeben:
Die innerhalb einer Shell-Funktion verwendeten Kommandos werden erst dann in die Hash-Tabelle eingetragen, wenn die Shell-Funktion ausgeführt wird.

-k

(k - keyword) In der aufgerufenen Subshell können Sie Variablen-Zuweisungen, also Angaben der Form *name=wert*, beim Aufruf eines Kommandos an beliebiger Stelle in der Kommandozeile angeben. Die aufgerufene Subshell führt die Variablen-Zuweisung aus und trägt den entsprechenden Schlüsselwort-Parameter in die Ablauf-Umgebung dieses Kommandos ein.

In dieser Subshell setzt *set +k* die Option *-k* zurück.

-k nicht angegeben und noch nicht gesetzt:
Variablen-Zuweisungen müssen vor dem Kommandonamen stehen. Die Shell trägt die entsprechenden Schlüsselwort-Parameter in die Ablauf-Umgebung dieses Kommandos ein.

-n

(n - no execution) Die aufgerufene Subshell liest und interpretiert alle Kommandos, führt sie aber nicht aus; d.h. der letzte Schritt bei der Bearbeitung einer Kommandozeile entfällt (siehe *WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?*).
Mit *-n* können Sie feststellen, ob eine Shell-Prozedur Syntax-Fehler enthält (siehe *Beispiel 1*).

Vorsicht

Die Option *-n* kann nur dadurch zurückgesetzt werden, daß sich die Subshell beendet.

-n nicht angegeben:
Alle Kommandos werden gelesen, interpretiert und ausgeführt.

-t

(t - terminate) Die aufgerufene Subshell beendet sich, wenn die erste Kommandozeile ausgeführt ist.
Wenn Sie *sh -t* ohne den Operanden *datei* aufrufen, erwartet die Subshell in der nächsten Zeile das Kommando, das ausgeführt werden soll. Dabei wird kein Bereitzeichen ausgegeben.

-t nicht angegeben:
Eine Dialog-Shell beendet sich, wenn Sie die Taste END drücken.
Eine Prozedur-Shell beendet sich nur dann vorzeitig, wenn sie einen Syntax-Fehler entdeckt. Sonst beendet sie sich, wenn die Shell-Prozedur abgearbeitet ist.

Falls die Option *-e* gesetzt ist, beendet sich die Shell, sobald ein Kommando einen Ende-Status $\neq 0$ zurückgibt.

-u

(u - unset) Die aufgerufene Subshell gibt eine Fehlermeldung aus, wenn sie in einer Kommandozeile 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.

In dieser Subshell setzt *set +u* die Option *-u* zurück.

-u nicht angegeben:
Die Subshell gibt keine Fehlermeldung aus, wenn eine verwendete Shell-Variable nicht definiert ist. Standardmäßig wird die leere Zeichenkette als Wert eingesetzt.

-v

(v - verbose) Die aufgerufene Subshell schreibt ab sofort jede Eingabe ohne Änderungen auf die Standard-Fehlerausgabe. Erst dann wird das entsprechende Kommando ausgeführt (siehe *Beispiel 1*).
Zusammen mit der Option *-x* könnnen Sie so Shell-Prozeduren testen.

In dieser Subshell setzt *set +v* die Option *-v* zurück.

-v nicht angegeben:
Die aufgerufene Subshell protokolliert die Eingabe nicht.

-x

(x - execute) Die aufgerufene Subshell 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:

- Führen Sie die entsprechende Shell-Prozedur aus mit dem Kommando *sh -xv prozedur*, oder
- tragen Sie *set -xv* als erstes Kommando in die entsprechende Shell-Prozedur ein.

In dieser Subshell setzt *set +x* die Option *-x* zurück.

-x nicht angegeben:
Die aufgerufene Subshell protokolliert die bearbeitete Eingabe nicht.

--
kann nicht mit anderen Optionen kombiniert werden.
Das erste Argument, das Sie nach -- angeben, interpretiert die Shell nicht als Option, auch wenn es mit einem Bindestrich - beginnt. Auf diese Weise können Sie mit *sh* eine Shell-Prozedur ausführen, deren Namen mit einem Bindestrich beginnt.

datei
Name der Shell-Prozedur, die von der Subshell ausgeführt werden soll. Für diese Datei brauchen Sie Leserecht.

Auf die Option *-c kommando* darf kein weiterer Operand folgen, auch nicht *datei*.

Wenn Sie *sh* mit der Option *-s* aufrufen, versorgt die aufgerufene Subshell den Stellungsparameter $1 mit *datei*. Die in *datei* enthaltenen Kommandos werden also nicht ausgeführt.

datei nicht angegeben:
Die aufgerufene Subshell gibt ihr Bereitzeichen aus und liest Ihre Eingaben von der Standard-Eingabe. Diese Eingaben bearbeitet die Subshell entsprechend den gesetzten Optionen.

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.

Die aufgerufene Subshell versorgt die Stellungsparameter $1, $2, ... und $9 mit den ersten neun angegebenen Argumenten. Die Shell-Parameter $* und $@ versorgt die Subshell mit allen angegebenen Argumenten und den Shell-Parameter $# mit der Anzahl dieser Argumente. Sind weniger als neun Argumente angegeben, werden die restlichen Stellungsparameter mit der leeren Zeichenkette versorgt.
Mit *shift* können Sie das zehnte und weitere Aufrufargumente direkt ansprechen.

Diese Werte sind nur der aufgerufenen Subshell bekannt. Wenn Sie Stellungsparameter in der aktuellen Shell setzen wollen, verwenden Sie stattdessen das eingebaute *sh*-Kommando *set*.

argument nicht angegeben:
Die Stellungsparameter $1, $2, ..., $9 sowie die Shell-Parameter $@, $* und $# werden mit der leeren Zeichenkette versorgt.

ENDE-STATUS

≠0 Die Shell hat einen Syntax-Fehler in der Eingabe entdeckt.
Wenn die Shell beim Ausführen einer Shell-Prozedur einen Syntax-Fehler entdeckt, bricht sie die Ausführung sofort ab.

In allen anderen Fällen gibt die Shell den Ende-Status des zuletzt ausgeführten Kommandos zurück.

DATEIEN

/etc/profile
wird von jeder Login-Shell ausgeführt. Diese Datei erstellt der Systemverwalter.

$HOME/.profile
Shell-Prozedur, die jeder Benutzer in seinem HOME-Dateiverzeichnis einrichten kann. Die Datei *$HOME/.profile* wird von der Login-Shell nach */etc/profile* ausgeführt.

*/tmp/sh**
temporäre Dateien

*/dev/null**
leere Datei, auf die Standard-Eingabe gelenkt wird, z.B. bei Hintergrund-Kommandos

UMGEBUNGSVARIABLEN

Siehe *STANDARD-VARIABLEN DER SHELL.*

BEISPIELE

1. In einer neu erstellten Shell-Prozedur sollen Syntax-Fehler gesucht werden, ohne daß die Kommandos ausgeführt werden. Die Shell-Prozedur *syntax* hat folgenden Inhalt:

```
: # Aufruf mit sh syntax
set `echo *`
echo `pwd`:
ll {
ls -al $* | pg
}
```

Die Definition der Shell-Funktion *ll* ist fehlerhaft: die runden Klammern fehlen. Diesen Syntax-Fehler meldet die Shell bei folgendem Aufruf:

```
$ sh -nv syntax
set `echo *`
echo `pwd`:
ll {
ls -al $* | pg
}
syntax: syntax error at line 6: '}' unexpected
```

Da auch die Option *-v* gesetzt wird, gibt die Shell zusätzlich jede gelesene Kommandozeile aus. Die einzelnen Kommandos werden nicht ausgeführt.

2. Die Kommandos einer Shell-Funktion sollen bereits bei der Definition dieser Shell-Funktion in die Hash-Tabelle eingetragen werden:

```
$ sh -h
$ ll() {
ls -al $* | pg
}
$ hash
hits    cost    command
0       2       /usr/bin/pg
0       1       /bin/ls
.
.
.
```

In der aufgerufenen Subshell ist die Option *-h* gesetzt. Deshalb werden die Kommandos *ls* und *pg* gleich bei der Definition der Shell-Funktion *ll* in die Hash-Tabelle eingetragen.

SIEHE AUCH

cd, echo, eval, exec, exit, export, hash, pwd, read, readonly, set, shift, test, times, trap, type, ulimit, umask, unset, wait, :, ., [␣␣]
csh, jsh, ksh, rksh

exec(), exit(), signal(), ulimit(), umask(), wait() [14]

Tabellen und Verzeichnisse, Sonderzeichen der Bourne-Shell sh

Bach, F. et al.: UNIX Handbuch zur Programmentwicklung [20], *Kapitel 2, Benutzeroberflächen* und *Kapitel 4, Shell-Programmierung und Prototyping*

Kochan, Wood: UNIX Shell Programming [27]

Manis, Meyer: The UNIX Shell-Programming Language [30]

Die Shell als Kommandointerpreter

Dieser Abschnitt ist wie folgt gegliedert:

- Kommandos eingeben und verknüpfen
  - Pipelines
  - Kommandofolgen
  - Kommandofolgen klammern
  - Eingebaute Shell-Kommandos
- Sonderzeichen der Shell
- Shell-Variablen
  - Eine Shell-Variable definieren
  - Auf den Wert einer Variablen zugreifen
  - Standard-Variablen der Shell
- Die Umgebung
  - Die Umgebung eines Kommandos beim Start
  - Die Ablauf-Umgebung der aktuellen Shell
- Shell-Parameter
  - Shell-Parameter durch einen Wert ersetzen
  - Standard-Werte für Shell-Parameter vereinbaren
- Kommandos durch ihre Ausgabe ersetzen
- Eingabe und Ausgabe eines Kommandos umlenken
  - Die Standard-Eingabe für ein Kommando umlenken
  - Die Standard-Ausgabe für ein Kommando umlenken
  - Die Standard-Fehlerausgabe für ein Kommando umlenken
  - Mehrere Umlenkungen zu einem Kommando
- Argumente durch passende Dateinamen ersetzen
- Wie bearbeitet die Shell die Kommandozeile?

KOMMANDOS EINGEBEN UND VERKNÜPFEN

Kommandos können Sie entweder in einer Datei zusammenfassen und als Shell-Prozedur ausführen oder im Dialog der Reihe nach im Anschluß an das Bereitzeichen der Shell eingeben. Standardmäßig hat ein Kommando folgendes Format:

```
kommando␣parameter␣...␣↵
        |          |    |
        Trennzeichen  Abschluß
```

kommando
: Name des SINIX-Kommandos, der Shell-Prozedur, des eingebauten *sh*-Kommandos oder der Shell-Funktion, die ausgeführt werden sollen.

parameter
: Aufrufargument, das die Shell entweder selbst interpretiert oder an *kommando* übergibt. Abhängig vom jeweiligen Kommando können Sie auch mehrere Aufrufargumente angeben.

Der Kommandoname und die Aufrufargumente werden voneinander durch Tabulator- oder Leerzeichen getrennt; das letzte Aufrufargument und damit die Eingabe des Kommandos wird durch ein Neue-Zeile-Zeichen abgeschlossen. Sie können aber auch meherere Kommandos in eine Zeile schreiben. Die einzelnen Kommandos werden durch ein Kommando-Trennzeichen voneinander getrennt (siehe *Pipelines* und *Kommandofolgen*).

Der Shell-Variablen IFS sind standardmäßig als Argument-Trennzeichen Leer-, Tabulator- und Neue-Zeile-Zeichen zugewiesen. Der Inhalt von IFS wird in den folgenden Fällen ausgewertet:

- wenn das eingebaute *sh*-Kommando *read* die gelesene Eingabe-Zeile in Argumente zerlegt
- wenn die Shell die inzwischen bearbeitete Kommandozeile zum zweiten Mal in Argumente zerlegt (siehe *WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?, Schritt 8*).

Wenn Sie den Wert der Variablen IFS ändern, verstehen *read* und die Shell beim zweiten Zerlegen nur noch die jetzt gültigen Argument-Trennzeichen.

Wenn die Shell die eingegebene Kommandozeile zum ersten Mal in Argumente zerlegt (siehe *WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?, Schritt 3*), gelten Leer- und Tabulatorzeichen als Argument-Trennzeichen, unabhängig vom Inhalt der Umgebungsvariablen IFS.

Jedes SINIX-Kommando gibt an die Shell, in der es aufgerufen wurde, einen Wert zurück, seinen Ende-Status. Standardmäßig hat der Ende-Status den Wert:

0 bei fehlerfreiem Ablauf
≠0 bei fehlerhaftem Ablauf

Die Shell versteht aber nicht nur einfache Kommandos. Mehrere Kommandos können Sie wie folgt aneinanderhängen:

- Mit dem Zeichen | verbinden Sie zwei Kommandos bber eine Pipe zu einer Pipeline (siehe *Pipelines*).
- Mit Strichpunkt ; bzw. mit kommerziellem Und & bzw. mit && oder || verbinden Sie mehrere Kommandos oder Pipelines zu einer Kommandofolge (siehe *Kommandofolgen*).

Pipelines

Eine Pipeline ist eine Folge von zwei oder mehr Kommandos, die jeweils durch das Pipe-Zeichen | verbunden sind. Eine solche Verbindung heißt Pipe.

Eine Pipe zwischen zwei Kommandos lenkt die Standard-Ausgabe des ersten Kommandos um auf die Standard-Eingabe des zweiten Kommandos. Deshalb müssen die so verbundenen Kommandos folgende Bedingung erfüllen:

- Das erste Kommando einer Pipeline muß auf die Standard-Ausgabe schreiben.
- Das letzte Kommando einer Pipeline muß von der Standard-Eingabe lesen.
- Besteht die Pipeline aus mehr als zwei Kommandos, müssen alle Kommandos nach dem ersten und vor dem letzten Kommando der Pipeline von der Standard-Eingabe lesen und auf die Standard-Ausgabe schreiben. Kommandos, die von der Standard-Eingabe lesen und auf die Standard-Ausgabe schreiben, heißen auch Filter.

Die Shell startet alle Kommandos der Pipeline gleichzeitig als eigenständige parallele Prozesse; in einem Ein-Prozessor-System kann jedoch pro Zeiteinheit immer nur einer dieser Prozesse vom Prozessor bearbeitet werden. Die einzelnen Prozesse wechseln sich am Prozessor ab.

Jeder Prozeß, bis auf den, der zum ersten Kommando der Pipeline gehört, erhält seine Eingabe von einem anderen Prozeß. Außerdem leitet jeder Prozeß, bis auf den, der zum letzten Kommando der Pipeline gehört, seine Ausgabe an einen anderen Prozeß weiter.

Deshalb müssen die einzelnen Prozesse vom Betriebssystem-Kern synchronisiert werden. Jeweils die Prozesse, die zu zwei direkt aufeinander folgenden Kommandos in der Pipeline gehören, kommunizieren miteinander über einen Puffer. Der lesende Prozeß wartet, bis der schreibende Prozeß Daten in den Puffer geschrieben hat. Wenn der Puffer voll ist, wartet der schreibende Prozeß, bis der lesende Prozeß Daten aus dem Puffer gelesen hat.

Die Shell wartet, bis das letzte Kommando der Pipeline beendet ist. Erst dann bearbeitet sie weitere Eingaben.

Ende-Status einer Pipeline

Das ist der Ende-Status des letzten Kommandos der Pipeline.

Kommandofolgen

Mehrere Kommandos oder Pipelines können Sie zu einer Kommandofolge verbinden. Die einzelnen Glieder dieser Folge werden durch die Zeichen ; bzw. & bzw. && bzw. || voneinander getrennt. Das letzte Glied der Kommandofolge wird abgeschlossen durch ein Neue-Zeile-Zeichen, durch ; oder &. Ein Neue-Zeile-Zeichen, das nicht innerhalb von runden bzw. geschweiften Klammern steht, veranlaßt die Shell, die Kommandofolge auszuführen (siehe *Kommandofolgen klammern*).

So können Sie mehrere Kommandos auf einmal eingeben, ohne auf das Ende der einzelnen Kommandos zu warten. Erst wenn alle Kommandos der Kommandofolge ausgeführt sind, gibt die Shell ihr Bereitzeichen aus.

Im folgenden steht der Begriff Kommando für ein Kommando oder eine Pipeline. Die Kommando-Trennzeichen haben folgende Wirkung:

Zeichen	Wirkung
;	Die Shell bearbeitet das nächste Kommando erst, wenn das vorausgehende beendet ist. Sie gibt aber kein Bereitzeichen aus.
&	Die Shell startet das vorausgehende Kommando im Hintergrund und bearbeitet sofort das nachfolgende Kommando. Hintergrund-Prozesse ignorieren die Signale SIGINT und SIGQUIT (siehe auch *trap*). Die Shell lenkt die Standard-Eingabe aller Hintergrund-Kommandos um auf die leere Datei */dev/null*. Sie können aber die Standard-Eingabe eines Hintergrund-Kommandos umlenken auf eine andere Datei, z.B. mit < *datei* in der Kommandozeile.
&& (ANDIF)	Die Shell führt das nachfolgende Kommando nur aus, wenn das vorausgehende Kommando als Ende-Status den Wert 0 zurückgegeben hat.
|| (ORIF)	Die Shell führt das nachfolgende Kommando nur aus, wenn das vorausgehende Kommando als Ende-Status einen Wert ≠ 0 zurückgegeben hat.

Priorität der Kommando-Trennzeichen

Die Zeichen && und || haben gleiche Priorität, sie ist höher als die Priorität der Zeichen ; und &.
Die Zeichen ; und & haben ebenfalls gleiche Priorität, sie ist niedriger als die Priorität der Zeichen && und ||.

Beispiel

```
kommando1&kommando2||kommando3;kommando4↵
```

Die Shell startet *kommando1* im Hintergrund und bearbeitet sofort *kommando2*.
Wenn *kommando2* als Ende-Status den Wert 0 zurückgibt, führt die Shell *kommando4* aus.
Wenn *kommando2* als Ende-Status einen Wert ≠0 zurückgibt, führt die Shell nacheinander *kommando3* und *kommando4* aus.

Die Angabe || bindet stärker als & und ;.

Ende-Status einer Kommandofolge

Das ist der Ende-Status des letzten ausgeführten Kommandos der Kommandofolge.

Kommandofolgen klammern

Mit runden oder geschweiften Klammmern können Sie mehrere Kommandos zusammenfassen,

- um sie gemeinsam im Hintergrund zu starten; dabei bleibt die Reihenfolge der Ausführung erhalten
- um die Ausgaben aller Kommandos in eine Datei zu schreiben oder über eine Pipe an ein anderes Kommando weiterzuleiten

```
(kommandofolge)
```

kommandofolge
: Folge von Kommandos oder Pipelines, die durch ein Kommando-Trennzeichen voneinander getrennt sind.
Die Shell startet eine Subshell. Diese Subshell führt *kommandofolge* aus und beendet sich. Die vorher gültige Shell meldet sich zurück, die vorher gültigen Shell-Variablen und auch das vorher gültige Dateiverzeichnis haben sich nicht verändert.

Wenn Sie außerhalb der runden Klammern die Standard-Ausgabe in eine Datei umlenken, schreiben alle Kommandos bzw. Pipelines der Kommandofolge ihre Ausgaben nacheinander in die angegebene Datei. Das entsprechende gilt, wenn Sie die Standard-Fehlerausgabe in eine Datei umlenken.

Wenn Sie im Anschluß an die schließende runde Klammer ein Pipe-Zeichen | angeben, werden die Ausgaben aller Kommandos der Kommandofolge als Eingabe an das Kommando übergeben, das auf das Pipe-Zeichen folgt.

Wenn Sie im Anschluß an die schließende runde Klammer das Zeichen & angeben, wird eine Subshell im Hintergrund gestartet, die der Reihe nach alle Kommandos der Kommandofolge ausführt.

Beispiel

```
$ pwd;(cd /dev;pwd);pwd
/usr1/rosa
/dev
/usr1/rosa
```

In der aufgerufenen Subshell wird */dev* zum aktuellen Dateiverzeichnis. Dieser Wechsel des aktuellen Dateiverzeichnisses beeinflußt jedoch nicht die übergeordnete Shell.

Ende-Status

Das ist der Ende-Status des letzten ausgeführten Kommandos in *kommandofolge*.

```
{ kommandofolge; }
```

kommandofolge
Folge von Kommandos oder Pipelines, die durch ein Kommando-Trennzeichen voneinander getrennt sind.
Die aktuelle Shell führt *kommandofolge* aus. Die geschweiften Klammern { } sind Schlüsselwörter der Shell, d.h. vor { und } muß ein Kommando-Trennzeichen stehen und nach { und } ein Argument-Trennzeichen.

In den folgenden Fällen wird *kommandofolge* trotz der geschweiften Klammern in einer Subshell ausgeführt, d.h. die geschweiften Klammern haben die gleiche Wirkung wie die runden:

- Wenn Sie außerhalb der geschweiften Klammern die Standard-Ausgabe in eine Datei umlenken.
  Dabei schreiben alle Kommandos der Kommandofolge ihre Ausgaben nacheinander in die angegebene Datei.
  Das entsprechende gilt, wenn Sie die Standard-Fehlerausgabe in eine Datei umlenken.
- Wenn Sie im Anschluß an die schließende geschweifte Klammer ein Pipe-Zeichen | angeben.
  Dabei werden die Ausgaben aller Kommandos der Kommandofolge als Eingabe an das Kommando übergeben, das auf das Pipe-Zeichen folgt.
- Wenn Sie im Anschluß an die schließende geschweifte Klammer das Zeichen & angeben.
  Die Subshell wird im Hintergrund gestartet und führt der Reihe nach alle Kommandos der Kommandofolge aus.

Beispiele

1. Wenn die Datei *.profile* nicht vorhanden ist, soll eine entsprechende Meldung ausgegeben und die aktuelle Shell beendet werden:

```
[ -f .profile ] || {
   echo Datei .profile fehlt
   exit 1
}
```

   Da die Kommandos innerhalb der geschweiften Klammern von der aktuellen Shell ausgeführt werden, beendet *exit* die aktuelle Shell. Wegen der Verknüpfung mit ||, wird der Inhalt der geschweiften Klammern nur ausgeführt, wenn das eingebaute *sh*-Kommando *test* einen Ende-Status ≠0 zurückliefert.

2. Wird die Standard-Ausgabe der Kommandofolge auf eine Datei umgelenkt, so wird die Kommandofolge trotz der geschweiften Klammern in einer Subshell ausgeführt, wie bei Angabe von runden Klammern:

```
$ pwd;{ cd /dev; echo Datensichtstationen:; ls tty*;} > $HOME/ausgabe
/usr1/rosa
$ pwd
/usr1/rosa
$ cat ausgabe
Datensichtstationen:
tty00
tty01
tty03
tty10
```

   Das aktuelle Dateiverzeichnis hat sich nicht geändert, weil die Kommandofolge in einer Subshell ausgeführt wird. Die Shell hat die Ausgaben der Kommandos *echo* und *ls* zusammengefaßt und in die Datei *$HOME/ausgabe* geschrieben.

Ende-Status

Das ist der Ende-Status des letzten ausgeführten Kommandos in *kommandofolge*.

Eingebaute Shell-Komandos (sh)

Kommandos, die die Shell selbst ausführt, für deren Bearbeitung sie also keinen neuen Prozeß erzeugt, heißen eingebaute Shell-Kommandos (genauer: eingebaute *sh*-Kommandos). Sie unterscheiden sich von den übrigen "externen" Kommandos in den folgenden Punkten:

- Sie sind als Unterprogramme Bestandteil der Binär-Datei */bin/sh*. Deshalb können Sie die Namen nicht ändern. Ebenso können Sie die Zugriffsrechte nur für */bin/sh* ändern, aber nicht für ein einzelnes eingebautes *sh*-Kommando.
- Die Shell führt die eingebauten Kommandos bevorzugt aus (siehe *WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?*).
  Gibt es zu einem eingebauten ein gleichnamiges externes Kommando, so führt die Shell bei Angabe dieses Namens immer das eingebaute Kommando aus. Das externe Kommando wird ausgeführt, wenn der Aufruf-Name einen Schrägstrich / enthält, also z.B. bei Aufruf mit dem entsprechenden absoluten Pfadnamen.
- Die eingebauten Kommandos sind schneller, weil die Shell sie selbst ausführt.

Zu den meisten eingebauten *sh*-Kommandos gibt es keine externe Entsprechung. Zum eingebauten *sh*-Kommando *cd* kann es keine geben: Wenn zur Ausführung von *cd* ein neuer Prozeß erzeugt wird, ändert sich das aktuelle Dateiverzeichnis nur für diesen Prozeß. Nach der Ausführung meldet sich die aufrufende Shell zurück, das aktuelle Dateiverzeichnis hat sich nicht geändert.

Das Ausgabeformat von eingebauten *sh*-Kommandos ist nicht standardisiert. Deshalb sollten Sie sich bei Ihren Programmen oder Shell-Prozeduren nicht auf ein bestimmtes Ausgabeformat verlassen.

Da in diesem Handbuch die eingebauten *sh*-Kommandos zusammen mit den externen Kommandos in alphabetischer Reihenfolge beschrieben sind, folgt hier nur eine Liste der eingebauten *sh*-Kommandos:

eingebautes sh-Kommando	Funktion	externes Kommando
cd	aktuelles Dateiverzeichnis wechseln	
echo	Aufrufargumente ausgeben	/bin/echo
eval	Aufrufargumente bearbeiten und als Kommando ausführen	
exec	die aktuelle Shell überlagern	
exit	Shell-Prozedur beenden	
export	Shell-Variablen exportieren	
hash	Hash-Tabelle der Shell bearbeiten	
login	sich neu am System anmelden	/bin/login
newgrp	Gruppenzugehörigkeit ändern	/bin/newgrp
pwd	Pfadnamen des aktuellen Dateiverzeichnisses ausgeben	/bin/pwd
read	Argumente von Standard-Eingabe lesen und Shell-Variablen zuweisen	
readonly	Shell-Variablen schützen	
set	Shell-Optionen oder Stellungsparameter setzen	/bin/sh
shift	die Werte der Stellungsparameter nach links verschieben	
test	Bedingungen prüfen	
times	Gesamt-Laufzeit gestarteter Prozesse	
trap	Signalbehandlung ändern	
type	Typ eines Kommandos abfragen	
ulimit	Dateigröße für das Schreiben begrenzen oder aktuellen Grenzwert abfragen	
umask	Standard-Vergabe der Zugriffsrechte ändern	

Fortsetzung →

Fortsetzung

eingebautes sh-Kommando	Funktion	externes Kommando
unset	Shell-Variablen oder Shell-Funktionen aus der Umgebung löschen	
wait	auf die Beendigung von Hintergrund-Prozessen warten	
:	Ende-Status 0 zurückgeben	/bin/true
.	Shell-Prozeduren in der aktuellen Shell ausführen	
[ ]	Bedingungen prüfen, siehe *test*	

Die Beschreibungen der eingebauten *sh*-Kommandos enthalten folgende Informationen:

- im Kolumnentitel das Stichwort "eingebautes sh-Kommando"
- die Syntax dieser Kommandos
- einen Verweis auf ein entsprechendes externes Kommando, falls vorhanden
- Abweichungen des eingebauten Kommandos vom entsprechenden externen Kommando, falls es solche gibt

SONDERZEICHEN DER SHELL

Dieser Abschnitt enthält eine Liste der Zeichen und Zeichenkombinationen, die für die Shell eine Sonderbedeutung haben. Diese Sonderbedeutung ist kurz erklärt zusammen mit einem Verweis auf den Abschnitt, in dem dieses Zeichen ausführlich beschrieben ist (siehe auch *Tabellen und Verzeichnisse, Sonderzeichen der Bourne-Shell sh*).

Die Shell wertet die in einer Kommandozeile enthaltenen Sonderzeichen in mehreren Schritten aus (siehe auch *WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?*).

Sonderzeichen	Bedeutung	beschrieben in
Leerzeichen Neue-Zeile-Zeichen Tabulatorzeichen	Argument-Trennzeichen	Kommandos eingeben und verknüpfen
Neue-Zeile-Zeichen | ; & || &&	Kommando-Abschluß bzw. Kommando-Trennzeichen	Kommandofolgen
(kommandofolge) {␣kommandofolge;}	Kommandofolgen klammern	Kommandofolgen klammern
`kommando`	durch Ausgabe ersetzen	Kommando durch seine Ausgabe ersetzen
* ? [s] [c1-c2] [!s] [!c1-c2]	ersetzen, falls Dateinamen passen	Argumente durch passende Dateinamen ersetzen
>datei >>datei >&zahl >&-	Standard-Ausgabe umlenken Standard-Ausgabe schließen	Eingaben und Ausgaben eines Kommandos umlenken
<datei <<argument <<-argument <&zahl <&-	Standard-Eingabe umlenken Standard-Eingabe schließen	Eingaben und Ausgaben eines Kommandos umlenken

Fortsetzung →

Fortsetzung

Sonderzeichen	Bedeutung	beschrieben in
Leerzeichen Neue-Zeile-Zeichen Tabulatorzeichen	Argument-Trennzeichen	Kommandos eingeben und verknüpfen
name=wert	Wert zuweisen	Shell-Variablen
$name ${name}	Wert der Variablen ansprechen	Shell-Parameter
${name-wert} ${name?wert} ${name+wert}	bedingte Ersetzung, falls Variable definiert ist	Standard-Werte für Shell-Parameter vereinbaren
${name=wert} ${name:=wert}	Zuweisung, falls Variable nicht definiert ist Zuweisung, falls Variable nicht definiert oder ihr Wert die leere Zeichenkette ist	Standard-Werte für Shell-Parameter vereinbaren
${name:-wert} ${name:?wert} ${name:+wert}	bedingte Ersetzung, falls Variable definiert und ihr Wert nicht die leere Zeichenkette ist	Standard-Werte für Shell-Parameter vereinbaren
$0	1. Argument des Aufrufs	Shell-Parameter
$1, $2, ... , $9 **$*** **"$*"** **$@** **"$@"** **$#**	Stellungsparameter alle Aufrufargumente wie $*, aber ein Argument alle Aufrufargumente wie $@, aber $# Argumente Anzahl Aufrufargumente	Shell-Parameter
$$** **$! **$?** **$-**	Parameter der Shell, die nur gelesen werden können	Shell-Parameter
name() {␣kommandofolge;}	Shell-Funktion	Shell-Funktionen
**** **' ... '** **" ... "**	Entwertungszeichen	Sonderzeichen für die Shell entwerten
# **;;**	Kommentar-Zeichen Abschluß für Kommando-folgen bei *case*-Anweisung	Kommentare in Shell-Prozeduren Ablaufanweisungen der Shell

Sonderzeichen für die Shell entwerten

Sie können jedes Sonderzeichen für die Shell entwerten. Entsprechend dem Zeichen, das Sie entwerten wollen, verwenden Sie eines der nachfolgend beschriebenen Entwertungszeichen.

Die Entwertungszeichen selbst sind nicht Bestandteil der Aufrufargumente. Die Shell entfernt diese Zeichen, bevor sie die Argumente an das Kommando übergibt.

Entwertungszeichen

\
entwertet die Sonderbedeutung des nachfolgend angegebenen Zeichens. Mit dem Gegenschrägstrich \ können Sie alle Sonderzeichen der Shell entwerten.

Mit der Angabe \↵ verliert das Neue-Zeile-Zeichen seine Sonderbedeutung als Argument- oder Kommando-Trennzeichen. Die Shell ignoriert diese Angabe und eliminiert nicht nur \ sondern auch das Neue-Zeile-Zeichen aus der Kommandozeile.

'zeichenkette'
entwertet die Sonderbedeutung aller Zeichen der *zeichenkette*. Mit Hochkommata können Sie alle Sonderzeichen der Shell entwerten, bis auf das Hochkomma selbst.

Da innerhalb der Hochkommata auch Leer-, Tabulator- und Neue-Zeile-Zeichen ihre Sonderbedeutung als Argument-Trennzeichen verlieren, ist *zeichenkette* für die Shell ein einziges Argument.

"zeichenkette"
Nur die folgenden Zeichen behalten für die Shell ihre Sonderbedeutung:

$
gefolgt von einem Variablen-Namen, wird ersetzt durch den Wert dieser Shell-Variablen.

`kommando`
wird ersetzt durch die Ausgabe dieses Kommandos (siehe *KOMMANDOS DURCH IHRE AUSGABE ERSETZEN*).

\
entwertet die Sonderbedeutung eines nachfolgenden Sonderzeichens. Innerhalb der Anführungszeichen behalten nur die Zeichen $, '...' und \ ihre Sonderbedeutung. Folgt auf den Gegenschrägstrich keines dieser Sonderzeichen, ist er als Zeichen Bestandteil der Zeichenkette.

Alle übrigen Sonderzeichen der Shell innerhalb der Anführungszeichen sind, bis auf das Anführungszeichen selbst, entwertet.

Da innerhalb der Anführungszeichen auch Leer-, Tabulator- und Neue-Zeile-Zeichen ihre Sonderbedeutung als Argument-Trennzeichen verlieren, ist die Zeichenkette für die Shell ein einziges Argument. Allerdings gibt es hier eine Ausnahme: die Angabe "$@". Sie ist gleichbedeutend zu "$1" "$2"..., die einzelnen Aufrufargumente bleiben als Argumente erhalten.

Wirkung von Hochkommas und Anführungszeichen

entwerten	Argument-Trennzeichen	Kommando-Trennzeichen	\	$	*	?	[	'	"	'
'...'	ja	ja	ja	ja	ja	ja	ja	ja	ja	s
"..."	ja	ja	nein	nein	ja	ja	ja	nein	s	ja

s bedeutet: schließendes Zeichen

SHELL-VARIABLEN

Shell-Variablen sind immer vom Typ Zeichenkette. Sie müssen nicht deklariert werden, sondern sind definiert, sobald ihnen ein Wert zugewiesen ist.

Shell-Variablen benutzen Sie, um eine Zeichenkette zu speichern, die Sie häufig verwenden. Für den nötigen Speicherplatz sorgt die Shell selbst.

Im folgenden erfahren Sie,

- wie Sie Variablen definieren und ihnen einen Wert zuweisen
- wie Sie auf den Wert einer Variablen zugreifen
- mit welchen Standard-Variablen die Shell arbeitet.

Eine Shell-Variable definieren

Jede Shell-Variable besteht aus einem Namen und einem Wert. Sie definieren eine Shell-Variable wie folgt:

```
name=wert[ name=wert]...
```

name
: Name der Variablen. Ein Variablen-Name muß mit einem Buchstaben oder mit einem Unterstrich _ beginnen. Dem ersten Zeichen dürfen Buchstaben, Ziffern und Unterstriche in beliebiger Reihenfolge folgen.

  Eine Shell-Variable und eine Shell-Funktion dürfen in der aktuellen Shell nicht den gleichen Namen tragen.

  Wenn eine Variable dieses Namens bereits definiert ist, geht durch die neue Zuweisung ihr alter Wert in der aktuellen Shell verloren.

=
: der Zuweisungsoperator. Er darf von *name* und *wert* nicht durch Leerzeichen getrennt sein.

wert
: Wert der Variablen.
  Enthält *wert* Leer-, Tabulator- oder Neue-Zeile-Zeichen, müssen Sie diese entwerten. Die Shell weist der Variablen *name* die Zeichen *, $ und [...] zu, ohne sie zu interpretieren. *wert* wird also nicht ersetzt durch alle passenden Dateinamen (siehe *Argumente durch passende Dateinamen ersetzen*).

  Wenn Sie für *wert* "" oder '' oder nichts angeben, weisen Sie der Variablen *name* die leere Zeichenkette zu.

Die Zuweisung können Sie im Dialog eingeben oder in eine Shell-Prozedur schreiben. Sie können auch mehrere Zuweisungen in eine Zeile schreiben, jeweils getrennt durch ein Argument-Trennzeichen.

Die so definierten Variablen sind standardmäßig nur in der aktuellen Shell bekannt. Mit dem eingebauten *sh*-Kommando *export* machen Sie diese Variable auch jeder Subshell bekannt. Umgekehrt können Sie eine Variable, die Sie in einer Subshell definiert haben, nicht an die übergeordnete Shell zurückgeben. Weitere Informationen hierzu in *Die Umgebung*.

Mit dem eingebauten *sh*-Kommando *read* können Sie ebenfalls Shell-Variablen definieren.

Mit dem eingebauten *sh*-Kommando *unset* können Sie eine definierte Shell-Variable wieder löschen.

Auf den Wert einer Variablen zugreifen

Das eingebaute *sh*-Kommando *set* gibt alle Variablen und ihre Werte aus, die in der aktuellen Shell definiert sind. Auf den Wert einer Variablen können Sie aber auch direkt zugreifen wie folgt:

```
$name
```

name
: Name der Shell-Variablen, auf deren Wert Sie zugreifen wollen. Der Name muß ohne Leerzeichen auf das Dollar-Zeichen folgen.

name ist eine Shell-Variable, *$name* ist ein Schlüsselwort-Parameter. Einer Variablen können Sie einen Wert zuweisen, aber nicht einem Schlüsselwort-Parameter (siehe *SHELL-PARAMETER*).

Beispiel

Wenn Sie wissen wollen, welchen Wert die Variable HOME hat, geben Sie ein:

```
$ echo $HOME
```

Das eingebaute *sh*-Kommando *echo* schreibt den Wert der Variablen HOME auf die Standard-Ausgabe.

Standard-Variablen der Shell

An die Login-Shell werden bereits beim Start einige Standard-Variablen übergeben. Weitere Standard-Variablen definiert sie selbst, bevor sie sich mit dem ersten Bereitzeichen meldet.

Den meisten Standard-Variablen ist ein Standard-Wert zugewiesen, dieser Standard-Wert ist auch in jeder Subshell bekannt. Erst wenn Sie die Werte dieser Standard-Variablen ändern, müssen Sie sie für eine Subshell mit *export* exportieren. Sonst gilt in der Subshell wieder der Standard-Wert.

Wenn Sie Shell-Variablen dauerhaft definieren wollen, können Sie die entsprechenden Zuweisungen in die Datei *$HOME/.profile* eintragen. Die Login-Shell führt diese Datei aus, bevor sie ihr Bereitzeichen ausgibt.

Die Shell kennt die folgenden Standard-Variablen:

```
CDPATH=dateiverzeichnis[:dateiverzeichnis]...
```

Sie können der Variablen *CDPATH* die Pfadnamen der Dateiverzeichnisse zuweisen, die *cd* durchsuchen soll. Die einzelnen Pfadnamen müssen Sie durch einen Doppelpunkt : voneinander trennen. Das aktuelle Dateiverzeichnis weisen Sie zu, indem Sie einen Punkt . oder die leere Zeichenkette, also nichts angeben. Mit einem Doppelpunkt gleich nach dem Zuweisungsoperator = weisen Sie das aktuelle Dateiverzeichnis als erstes Dateiverzeichnis zu.

Das eingebaute *sh*-Kommando *cd* greift auf *CDPATH* zu, um das Dateiverzeichnis zu suchen, das zum aktuellen werden soll. Die Suche beginnt in dem Dateiverzeichnis, dessen Pfadname vor dem ersten Doppelpunkt steht. Von links nach rechts werden alle in *CDPATH* enthaltenen Pfadnamen abgearbeitet (siehe *cd*).

Standard-Wert: keiner, d.h. *CDPATH* ist standardmäßig nicht definiert.
Wenn *CDPATH* nicht definiert ist, sucht *cd* Dateiverzeichnisse, die Sie mit ihrem relativen Pfadnamen angeben, relativ zum aktuellen Dateiverzeichnis.

```
HOME=dateiverzeichnis
```

Der Variablen *HOME* wird der absolute Pfadname des *HOME*-Dateiverzeichnisses zugewiesen. Wenn Sie das eingebaute *sh*-Kommando *cd* ohne Argument aufrufen, greift *cd* auf *HOME* zu. Das *HOME*-Dateiverzeichnis wird zum aktuellen Dateiverzeichnis.

Die Login-Shell sucht die Datei *.profile* in dem Dateiverzeichnis, das der Variablen *HOME* zugewiesen ist.

Standard-Wert: Das Kommando *login* weist der Variablen *HOME* den absoluten Pfadnamen Ihres Login-Dateiverzeichnisses zu. Dieser Pfadname ist in der Datei */etc/passwd* für Ihre Benutzerkennung eingetragen.

IFS=c . . .

Der Variablen *IFS* werden Argument-Trennzeichen zugewiesen. Dabei ist *c* ein beliebiges Zeichen, Leer-, Tabulator- und Neue-Zeile-Zeichen müssen entwertet sein. Mehrere Zeichen werden ohne Leerzeichen nacheinander angegeben.

Die Shell wertet den Inhalt von *IFS* in den folgenden Fällen aus:

- wenn das eingebaute *sh*-Kommando *read* die gelesene Eingabe-Zeile in Argumente zerlegt
- wenn die Shell die inzwischen bearbeitete Kommandozeile zum zweiten Mal in Argumente zerlegt (siehe Abschnitt *WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?*, Schritt 8).

Die Variable *IFS* können Sie nicht mit *unset* aus der Umgebung löschen.

Standard-Wert: Leerzeichen Tabulatorzeichen Neue-Zeile-Zeichen
Die Login-Shell weist der Variablen *IFS* diesen Wert zu.

Wenn Sie den Wert der Variablen *IFS* ändern, verstehen *read* und die Shell beim zweiten Zerlegen nur noch die jetzt gültigen Argument-Trennzeichen.

LANG=sprache

Mit der NLS-Variablen *LANG* wird eine internationalisierte Shell-Umgebung definiert. Der Wert von *LANG* ist der Name einer existierenden Datenbasis. Die Datenbasen stehen unter dem Dateiverzeichnis */usr/lib/locale*. Ist die Variable *LC_ALL* nicht gesetzt, dann legt *LANG* sowohl die Landessprache fest als auch alles, was mit den von einem internationalisierten Programm benutzten NLS-Variablen *LC_COLLATE, LC_CTYPE, LC_MONETARY, LC_NUMERIC, LC_TIME* und *LC_MESSAGES* einzeln festgelegt werden kann.

Ist *LANG* nicht definiert oder leer oder hat es einen ungültigen Wert, dann verhält sich das System, als wäre es nicht internationalisiert.

```
LC_ALL=sprache
```

Die Variable *LC_ALL* überschreibt alle Variablen *LC_** und *LANG*.

```
LC_COLLATE=sprache
```

Ist die Variable *LC_ALL* nicht gesetzt, dann legt die NLS-Variable *LC_COLLATE* beim Sortieren von Zeichenketten die Sortierreihenfolge fest.

Ist *LC_COLLATE* nicht definiert oder ist ihr die leere Zeichenkette zugewiesen, dann gilt für diese Variable als Standard-Wert der Wert von *LANG*.
Hat *LC_COLLATE* einen ungültigen Wert, dann verhält sich ein internationalisiertes Programm, als wäre es nicht internationalisiert.

```
LC_CTYPE=sprache
```

Ist die Variable *LC_ALL* nicht gesetzt, dann legt die NLS-Variable *LC_CTYPE* für jedes Zeichen fest, zu welcher Zeichenklasse es gehört, und definiert, wie Klein- in Großbuchstaben und umgekehrt umgewandelt werden.

Ist *LC_CTYPE* nicht definiert oder ist ihr die leere Zeichenkette zugewiesen, dann gilt für diese Variable als Standard-Wert der Wert von *LANG*.
Hat *LC_CTYPE* einen ungültigen Wert, dann verhält sich ein internationalisiertes Programm, als wäre es nicht internationalisiert.

```
LC_MESSAGES=sprache
```

Ist die Variable *LC_ALL* nicht gesetzt, dann legt die NLS-Variable *LC_MESSAGES* die Sprache der Meldungen fest, die ein internationalisiertes Programm ausgibt. Hat *LC_MESSAGES* einen ungültigen Wert, dann verhält sich ein internationalisiertes Programm, als wäre es nicht internationalisiert.

```
LC_MONETARY=sprache
```

Ist die Variable *LC_ALL* nicht gesetzt, dann legt die NLS-Variable *LC_MONETARY* Währungszeichen und Währungsformate fest.

Ist *LC_MONETARY* nicht definiert oder ist ihr die leere Zeichenkette zugewiesen, dann gilt für diese Variable als Standard-Wert der Wert von *LANG*.
Hat *LC_MONETARY* einen ungültigen Wert, dann verhält sich ein internationalisiertes Programm, als wäre es nicht internationalisiert.

`LC_NUMERIC=sprache`

Ist die Variable *LC_ALL* nicht gesetzt, dann legt die NLS-Variable *LC_NUMERIC* Dezimalpunktdarstellung, Exponentzeichen und Tausendertrennzeichen fest.

Ist *LC_NUMERIC* nicht definiert oder ist ihr die leere Zeichenkette zugewiesen, dann gilt für diese Variable als Standard-Wert der Wert von *LANG*.
Hat *LC_NUMERIC* einen ungültigen Wert, dann verhält sich ein internationalisiertes Programm, als wäre es nicht internationalisiert.

`LC_TIME=sprache`

Ist die Variable *LC_ALL* nicht gesetzt, dann legt die NLS-Variable *LC_TIME* das Format von Datums- und Zeitangaben fest.

Ist *LC_TIME* nicht definiert oder ist ihr die leere Zeichenkette zugewiesen, dann gilt für diese Variable als Standard-Wert der Wert von *LANG*.
Hat *LC_TIME* einen ungültigen Wert, dann verhält sich ein internationalisiertes Programm, als wäre es nicht internationalisiert.

`LOGNAME=benutzerkennung`

Das Kommando *login* weist der Variablen *LOGNAME* die Benutzerkennung zu, unter der Sie sich am System angemeldet haben.

`MAIL=datei`

Der Variablen MAIL wird der absolute Pfadname der Datei zugewiesen, in die *mail* Ihre Post schreibt.

Wenn die Variable *MAILPATH* nicht definiert ist, benachrichtigt Sie die Shell, falls *mail* Post in diese Datei geschrieben hat, durch folgende Meldung:
`you have mail`

Standard-Wert: */var/mail/$USER*
Das Kommando *login* weist *MAIL* diesen Wert zu.

```
MAILCHECK=zahl
```

Der Variablen *MAILCHECK* wird eine ganze *zahl* zugewiesen. Die Shell greift auf die Variable *MAILCHECK* zu. Im Abstand von *zahl* Sekunden prüft sie, ob *mail* Post für Sie in die Datei *$MAIL* oder in eine der Dateien geschrieben hat, deren Pfadnamen der Variablen *MAILPATH* zugewiesen sind.

Wenn Sie *MAILCHECK* den Wert 0 zuweisen, prüft die Shell diese Dateien jedesmal, bevor sie ihr Bereitzeichen ausgibt.

Ist Post eingetroffen, erhalten Sie die Meldung: `you have mail`

Die Variable *MAILCHECK* können Sie nicht mit *unset* aus der Umgebung löschen.

Standard-Wert: 600, d.h die Shell prüft alle 600 Sekunden (10 Minuten), ob *mail* neue Post in eine der Dateien geschrieben hat, deren Pfadnamen den Variablen *MAIL* oder *MAILPATH* zugewiesen sind.
Die Login-Shell weist *MAILCHECK* diesen Wert zu.

```
MAILPATH=datei[%text][:datei[%text]]...
```

Sie weisen der Variablen *MAILPATH* die Pfadnamen der Dateien zu, in denen Post eintreffen kann. Die einzelnen Pfadnamen müssen Sie durch einen Doppelpunkt voneinander trennen. Sie brauchen Leserecht für diese Dateien.
Im Anschluß an jeden Pfadnamen können Sie *text* angeben; dieser Text muß durch ein Prozentzeichen % vom jeweiligen Pfadnamen getrennt sein. Die Shell gibt *text* aus, wenn Post in der entsprechenden Datei eingetroffen ist.

Die Shell greift auf die Variable *MAILPATH* zu, um festzustellen, ob in einer der angegebenen Dateien neue Post eingetroffen ist.

MAILPATH ist standardmäßig nicht definiert.

```
PATH=dateiverzeichnis[:dateiverzeichnis]...
```

Der Variablen *PATH* sind die absoluten Pfadnamen der Dateiverzeichnisse zugewiesen, in denen die Shell nach Kommandos suchen soll. Die einzelnen Dateiverzeichnisse werden durch Doppelpunkte voneinander getrennt.

Das aktuelle Dateiverzeichnis weisen Sie zu, indem Sie einen Punkt . oder die leere Zeichenkette, also nichts, für *dateiverzeichnis* angeben.
Aus Sicherheitsgründen sollten Sie der Variablen *PATH* das aktuelle Dateiverzeichnis immer als letztes zuweisen.

Die Shell greift auf die Variable *PATH* zu, wenn sie ein Kommando sucht, das Sie mit seinem einfachen Dateinamen aufgerufen haben, also ohne Schrägstrich / im Aufruf-Namen. Die Suche beginnt in dem Dateiverzeichnis, dessen Pfadname vor dem ersten Doppelpunkt steht. Von links nach rechts werden alle in *PATH* enthaltenen Pfadnamen abgearbeitet, bis die Shell die Datei gefunden hat oder feststellt, daß die Datei in keinem dieser Dateiverzeichnisse steht.

Die Variable *PATH* können Sie nicht mit *unset* aus der Umgebung löschen.

Standard-Wert: */bin:/usr/bin:*.
Die Login-Shell weist *PATH* diesen Wert standardmäßig zu.

```
PS1=zeichenkette
```

Der Variablen *PS1* wird die *zeichenkette* zugewiesen, die die Shell als Bereitzeichen ausgeben soll, wenn sie auf die nächste Eingabe wartet. Leer- und Tabulatorzeichen in *zeichenkette* müssen entwertet sein.

Die Variable PS1 können Sie nicht mit *unset* aus der Umgebung löschen.

Standard-Wert: *$␣*
Die Login-Shell weist *PS1* standardmäßig diesen Wert zu.

```
PS2=zeichenkette
```

Der Variablen PS2 wird die *zeichenkette* zugewiesen, die die Shell ausgeben soll, wenn sie nach der Eingabe eines Neue-Zeile-Zeichens noch weitere Eingaben erwartet oder wenn Sie das Neue-Zeile-Zeichen entwertet haben. Leer- und Tabulatorzeichen in *zeichenkette* müssen entwertet sein.

Die Variable PS2 können Sie nicht mit *unset* aus der Umgebung löschen.

Standard-Wert: >␣
Die Login-Shell weist *PS2* standardmäßig diesen Wert zu.

```
SHELL=kommandointerpreter
```

Der Variablen *SHELL* wird der absolute Pfadname des Kommandointerpreters zugewiesen. Jedesmal, wenn eine Subshell aufgerufen wird, prüft die aufrufende Shell, ob der Wert von *SHELL* das Zeichen *r* enthält. Wenn ja, ist die aufgerufene Subshell eingeschränkt.

Die Editoren *ed* und *vi* greifen auf die Variable *SHELL* zu, wenn Sie während der Arbeit in einem dieser Editoren die Shell oder ein Shell-Kommando aufrufen.

Standard-Wert: Das Kommando *login* weist dieser Variablen den absoluten Pfadnamen des Programms zu, das für Sie nach der Anmeldung am System gestartet werden soll. Dieser Pfadname ist in der Datei */etc/passwd* für Ihre Benutzerkennung eingetragen; standardmäßig ist das */bin/sh*.

```
TERM=typ
```

Der Systemprozeß *getty* weist der Variablen *TERM* den *typ* der Datensichtstation zu. *typ* ist der entsprechende Eintrag in der Datei */etc/gettytab*.

Editoren greifen auf die Variable *TERM* zu, um festzustellen, welche Eigenschaften die Datensichtstation hat.

```
TZ=std offset1 [sz [offset2] [,beginn[/zeit], ende[/zeit]]]
```

Die Variable *TZ* enthält Informationen über die Zeitzonen. Das Kommando *date* benutzt *TZ* zur Bestimmung der Zeitzone bzw. zur Umrechnung der Weltzeit *UTC* (Universal Time Coordinated) in die Ortszeit und umgekehrt.

Die einzelnen Komponenten des Wertes von *TZ* folgen ohne Leerzeichen aufeinander, z.B. MET-1. Sie haben folgende Bedeutung:

std
: Standard-Zeitzone. *std* besteht aus mindestens drei Zeichen. Erlaubt sind neben Groß- und Kleinbuchstaben alle Zeichen außer Ziffern, Komma, Minuszeichen -, Pluszeichen + und ASCII NUL. Das erste Zeichen darf kein Doppelpunkt sein.

offset1
: gibt den Wert an, den man zur Ortszeit hinzuaddieren muß, um die Weltzeit *UTC* (Universal Time Coordinated) zu erhalten.
offset1 hat das folgende Format:

  ```
  [±]hh[:mm[:ss]]
  ```

  ±
  : Steht vor der Zeitangabe ein Pluszeichen, befindet sich die Zeitzone westlich des Null-Meridians. Steht ein Minuszeichen davor, befindet sich die Zeitzone östlich des Null-Meridians.

    ± nicht angegeben:
    wirkt wie +.

hh
Stunden (0-24)

mm
Minuten (0-59)

ss
Sekunden (0-59)

sz
Sommerzeitzone (Format wie bei *std*).

sz nicht angegeben:
Es gibt keine Umstellung auf Sommerzeit.

offset2
gibt den Wert an, den man zur Orts-Sommerzeit hinzuaddieren muß, um die UTC zu erhalten.
offset2 hat das gleiche Format wie *offset1*.

offset2 nicht angegeben:
Sommerzeit = Standard-Zeit + 1 Stunde

beginn[/zeit],ende[/zeit]
beginn[/*zeit*] gibt an, wann von Standard-Zeit zu Sommerzeit gewechselt wird; *ende*[/*zeit*] gibt an, wann zurückgewechselt wird. Dabei geben *beginn* und *ende* den Tag, an dem gewechselt wird, an, und *zeit* gibt die Uhrzeit (in Ortszeit) an.

beginn und *ende* können sein:

Jn
steht für den n-ten Tag des Julianischen Kalenders, wobei $1 \leq n \leq 365$. Schalttage werden nicht mitgezählt. D.h. der 28. Februar ist immer der 59. Tag, und der 1. März ist immer der 60. Tag. Den 29. Februar können Sie in diesem Format nicht angeben.

n
steht für den n-ten Tag des Julianischen Kalenders, wobei $0 \leq n \leq 365$. Schalttage werden mitgezählt. In diesem Format können Sie also den 29. Februar angeben.

Mm.w.t
steht für den t-ten Tag der w-ten Woche des m-ten Monats ($0 \leq t \leq 6$, $1 \leq n \leq 5$, $1 \leq m \leq 12$).
Der 0-te Tag einer Woche ist der Sonntag. Die 1. Woche eines Monats ist die Woche, in die der erste Tag des Monats fällt. Ist für die Woche eine 5 angegeben (Mm.5.t), so ist der letzte t-Tag des Monats m gemeint. Zum Beispiel bezieht sich die Angabe M9.5.6 auf den letzten Samstag im September.

zeit hat das Format: `hh[:mm[:ss]]`
(siehe *offset1*)

zeit nicht angegeben:
zeit = 02:00:00

beginn[/*zeit*],*ende*[/*zeit*] nicht angegeben:
Der Wechsel von Standard- zu Sommerzeit findet am letzten Sonntag im April statt; von Sommerzeit zu Standard-Zeit wird am letzten Sonntag im Oktober gewechselt (jeweils um 2 Uhr morgens).

USER=benutzerkennung

Das Kommando *login* weist der Variablen *USER* die Benutzerkennung zu, unter der Sie sich am System angemeldet haben.
Die Variable *USER* ist nur noch aus Gründen der Kompatibilität zu früheren Versionen vorhanden. Ihre Rolle wird jetzt von der Variablen *LOGNAME* übernommen.

DIE UMGEBUNG

Jeder Prozeß - auch die Shell - erhält eine Reihe von Informationen vom aufrufenden Prozeß, wenn er gestartet wird:

- Informationen, die für den Ablauf des Prozesses wichtig sind, d.h. Informationen über Vater- und Sohn-Prozesse, über geöffnete Dateien, über das aktuelle Dateiverzeichnis, über die vereinbarte Signalbehandlung und ähnliches.
- Informationen über bestimmte Variablen und die Werte, die ihnen zugewiesen sind.

Alle diese Informationen zusammen sind in der Umgebung eines Prozesses enthalten. Der Prozeß selbst trägt während seines Ablaufs weitere lokale Variablen in seine Umgebung ein oder ändert lokal den Wert von bereits bekannten Variablen.

Nachfolgend ist beschrieben:

- welche Variablen die Shell beim Start eines Kommandos in die Umgebung dieses Kommandos einträgt (siehe *Die Umgebung eines Kommandos beim Start*)
- wie Sie die Umgebung der aktuellen Shell ändern können (siehe *Die Ablauf-Umgebung der aktuellen Shell*).

Die Umgebung eines Kommandos beim Start

Wenn Sie ein Kommando aufrufen, übergibt die Shell an dieses Kommando

- die Namen aller Shell-Variablen und ihre Werte, die Sie in der aktuellen oder einer übergeordneten Shell exportiert und nicht mit *unset* aus der Umgebung gelöscht haben, und
- die Standard-Variablen, deren Wert Sie nicht verändert und die Sie nicht mit *unset* gelöscht haben. Die Standard-Variablen *IFS, MAILCHECK, PATH, PS1* und *PS2* können Sie allerdings nicht löschen.

Diese Variablen und ihre Werte sind die Start-Umgebung des aufgerufenen Kommandos. Das Kommando erbt sie von der aufrufenden Shell. Mit *env* können Sie feststellen, welche Variablen die aktuelle Shell an jedes aufgerufene Kommando weiterreicht.

Die Start-Umgebung für ein Kommando erweitern

Wenn Sie ein Kommando aufrufen, können Sie seine Umgebung erweitern, ohne die Umgebung der aktuellen Shell und die Start-Umgebung der anderen Programme, die die Shell ausführt, zu beeinflussen. Dazu geben Sie vor dem Kommandonamen die entsprechenden Zuweisungen an.

Diese Zuweisungen werden nicht in der aktuellen Shell ausgeführt, sondern in dem Prozeß, den die Shell zur Ausführung des Kommandos startet. Die entsprechenden Variablen werden in die Umgebung des Kommandos eingetragen, deshalb kann das

Kommando darauf zugreifen. In der aktuellen Shell sind diese Variablen nicht vorhanden.

Ist in der aktuellen Shell die Option *-k* gesetzt (siehe *set,sh*), so können Sie diese Zuweisungen an einer beliebigen Stelle innerhalb des Kommando-Aufrufes angeben. Alle so definierten Variablen sind nur dem aufgerufenen Kommando bekannt.

Beispiel

Die beiden folgenden Kommandozeilen führen zum gleichen Ergebnis:

```
kunde=meier prozedur
(kunde=meier;export kunde; prozedur)
```

Die erste Zeile enthält eine Zuweisung vor dem Kommandonamen. Diese Zuweisung wird nicht in der aktuellen Shell ausgeführt, sondern in der Subshell, die *prozedur* ausführt. Die Subshell trägt die Variable *kunde* in die Umgebung für die Prozedur ein. Wenn die Prozedur ausgeführt ist, beendet sich die Subshell. In der übergeordneten Shell ist die Variable *kunde* nicht bekannt.

Die zweite Zeile enthält die runden Klammern; d.h. alle eingeschlossenen Kommandos werden in einer Subshell ausgeführt. Die Subshell trägt die Variable *kunde* in die Umgebung für *prozedur* ein. An die Prozedur wird diese Umgebung übergeben, also auch die Variable *kunde*. Wenn die Prozedur ausgeführt ist, beendet sich die Subshell. In der übergeordneten Shell ist die Variable *kunde* nicht bekannt.

Wenn Sie in der aktuellen Shell Variablen definieren und mit *export* exportieren, ändern Sie nicht nur die Umgebung der aktuellen Shell sondern auch die Start-Umgebung für alle anschließend aufgerufenen Kommandos (siehe nächster Abschnitt).

Die Ablauf-Umgebung der aktuellen Shell

Wenn Sie eine Shell aufrufen, untersucht sie die Umgebung, die ihr beim Aufruf übergeben wurde. Für jeden Variablen-Namen, den sie findet, erzeugt sie einen Parameter mit dem dazugehörigen Wert.

Die Login-Shell, die von *login* gestartet wird, erbt ihre Start-Umgebung von *login*. Die Variablen *IFS, MAILCHECK, PATH, PS1* und *PS2* initialisiert die Login-Shell selbst.

Jede Variable, die Sie in dieser Shell neu definieren oder deren Wert Sie ändern, wird als lokale Variable in die Ablauf-Umgebung dieser Shell eingetragen. Lokale Variablen verändern aber nicht die Umgebung der Kommandos, die die Shell startet. Das können Sie mit *env* überprüfen.

Das eingebaute *sh*-Kommando *set* gibt alle Variablen aus, die in der aktuellen Shell definiert sind, also auch alle lokal gültigen.

Sie können die Ablauf-Umgebung der aktuellen Shell und gleichzeitig die Start-Umgebung für alle anschließend aufgerufenen Kommandos ändern:

- Sie erweitern die Start-Umgebung für Kommandos mit dem eingebauten *sh*-Kommando *export*. Wenn Sie eine Shell-Variable exportieren, wird sie in die Start-Umgebung eingetragen. Exportierte Variablen sind der aktuellen Shell bekannt, allen Subshells und allen Kommandos, die sie aufruft.
- Sie löschen eine Variable aus der Start-Umgebung mit dem eingebauten *sh*-Kommando *unset*. Die Standard-Variablen IFS, MAILCHECK, PATH, PS1 und PS2 können Sie allerdings nicht löschen.
  Eine gelöschte Variable ist auch in der Ablauf-Umgebung der aktuellen Shell nicht mehr vorhanden.

Wenn in der aktuellen Shell die Option *-a* gesetzt ist, werden automatisch alle Shell-Variablen exportiert, die Sie neu definieren oder deren Wert Sie ändern (siehe *set* und *sh*).

Sie können aber auch nur die Start-Umgebung für ein bestimmtes Kommando ändern (siehe *Die Start-Umgebung für ein Kommando erweitern*). In diesem Fall ändert sich die Ablauf-Umgebung der aktuellen Shell nicht.

SHELL-PARAMETER

Ein Shell-Parameter wird eingeleitet durch das Dollar-Zeichen $. Die Shell ersetzt einen Parameter in der Kommandozeile durch den Wert, der aktuell für die entsprechende Shell-Variable gilt. Sie geben Shell-Parameter wie folgt an:

```
$parameter
```

parameter
Name einer Shell-Variablen oder eines der folgenden Zeichen:

0 1 2 3 4 5 6 7 8 9 * @ # - ? $!

Wenn auf den Namen eines Shell-Parameters ein Buchstabe, eine Ziffer oder ein Unterstrich _ folgt, interpretiert die Shell dieses Zeichen als Teil des Namens. Deshalb muß der Name von den nachfolgenden Zeichen abgegrenzt werden. Sie können den Namen beispielsweise in geschweifte Klammern einschließen:

```
${parameter}
```

Beispiel

```
$ datei=/usr1/rosa/tabellen/liste1
$ mv $datei ${datei}00
```

Wenn die geschweiften Klammern nicht angegeben sind, sucht die Shell eine Variable mit dem Namen *datei00*, die vielleicht nicht definiert ist.

Es gibt drei Arten von Parametern:

- Schlüsselwort-Parameter,
- Stellungsparameter und
- besondere Shell-Parameter.

Schlüsselwort-Parameter

Schlüsselwort-Parameter sind Parameter der Form *$name* bzw. *${name}*. Die Shell ersetzt Schlüsselwort-Parameter durch den Wert der entsprechenden Shell-Variablen. Ist diese Variable nicht definiert, so ersetzt die Shell den Schlüsselwort-Parameter durch die leere Zeichenkette.

Sie können neue Schlüsselwort-Parameter einrichten oder den Wert eines Schlüsselwort-Parameters ändern, indem Sie die entsprechende Shell-Variable neu definieren oder ihren Wert ändern. Das erreichen Sie mit der folgenden Zuweisung:

```
name=wert
```

name ist eine Shell-Variable, *$name* ist ein Shell-Parameter. Einer Variablen können Sie einen Wert zuweisen, aber nicht einem Shell-Parameter (siehe dazu *SHELL-VARIABLEN* und *DIE UMGEBUNG*).

Sie können einen Schlüsselwort-Parameter auch in jeder Subshell bekanntmachen, indem Sie die entsprechende Variable mit *export* in die Umgebung eintragen.

Stellungsparameter

Stellungsparameter sind die Parameter $1, $2, $3, $4, $5, $6, $7, $8 und $9. Diese Stellungsparameter versorgt die Shell mit den folgenden Argumenten aus der Kommandozeile zu:

$1 erstes Aufrufargument für dieses Kommando

.
.
.

$9 neuntes Aufrufargument für dieses Kommando

Auf die ersten neun Aufrufargumente können Sie also direkt zugreifen. Auf alle Aufrufargumente zusammen greifen Sie mit $* oder $@ zu. Mit dem eingebauten *sh*-Kommando *shift* können Sie das zehnte und weitere Aufrufargumente direkt ansprechen.

In einer Dialog-Shell können Sie nur indirekt mit dem eingebauten *sh*-Kommando *set* alle Stellungsparameter auf einmal mit Werten versorgen. Nach dem Aufruf von *set* können Sie in der aktuellen Shell auf diese Werte zugreifen.

Mit *sh -s* können Sie indirekt in der aufgerufenen Subshell alle Stellungsparameter auf einmal mit Werten versorgen.

Im Gegensatz zu den Schlüsselwort-Parametern können Sie den Wert eines einzelnen Stellungsparameters nicht direkt ändern und Sie können diesen Wert auch nicht mit *export* in die Umgebung eintragen.

Besondere Shell-Parameter

Die besonderen Shell-Parameter und ihre Werte:

$0 Aufruf-Name des Kommandos bzw. der Shell-Prozedur; bei Shell-Funktionen ist $0 der Aufruf-Name der aktuellen Shell oder der Name der Shell-Prozedur, je nachdem wo Sie die Funktion aufgerufen haben.

$* alle Aufrufargumente, also die Stellungsparameter $1, ... $9 und alle weiteren Aufrufargumente.

"$*" alle Aufrufargumente als ein Argument, also "$1 ... $9 ..."

$@ alle Aufrufargumente, also die Stellungsparameter $1, ... $9 und alle weiteren Aufrufargumente.

"$@" alle Aufrufargumente als eigenständige Argumente, also "$1" ... "$9" ...

$# die Anzahl der Aufrufargumente; dieser Wert kann größer sein als 9.

$- alle Optionen, die in der aktuellen Shell gesetzt sind

$? der Ende-Status des zuletzt ausgeführten Kommandos; gilt nicht für Hintergrund-Kommandos

$$ die Prozeß-Nummer (PID) der aktuellen Shell

$! die Prozeß-Nummer (PID) des zuletzt im Hintergrund gestarteten Kommandos

Diese Werte vergibt die Shell automatisch. Die Parameter $*, $@ und $# ändern Sie, wenn Sie mit *set* bzw. *sh -s* die Stellungsparameter mit neuen Werten versorgen oder wenn Sie *shift* aufrufen.

Shell-Parameter durch einen Wert ersetzen

Bei der Bearbeitung der Kommandozeile ersetzt die Shell die angegebenen Shell-Parameter durch ihren Wert (siehe Abschnitt *Wie bearbeitet die Shell die Kommandozeile?*, Schritt 4). Auf den Wert eines Shell-Parameters greifen Sie wie folgt zu:

```
$parameteroder${parameter}
```

Ist kein Wert definiert, ersetzt die Shell diesen Parameter durch die leere Zeichenkette.

Standard-Werte für Shell-Parameter vereinbaren

Wenn Sie sicher sein wollen, daß Shell-Parameter durch einen definierten Wert ersetzt werden, können Sie einen Standard-Wert vereinbaren. Zusammen mit dieser Vereinbarung geben Sie die Bedingung an, unter der die Shell diesen Standard-Wert einsetzen soll.

In diesem Abschnitt wird beschrieben, wie Sie diese Standard-Werte vereinbaren können. Für *parameter* können Sie den Namen einer Shell-Variablen oder eines der folgenden Zeichen angeben:

0 1 2 3 4 5 6 7 8 9 * @ # - ? $!

Die Bedeutung dieser Zeichen ist im Abschnitt *Shell-Parameter* erklärt.

Für *standard_wert* sind beliebige Zeichenketten erlaubt. Die Shell interpretiert *standard_wert* nur, wenn sie den entsprechenden Parameter durch diesen Wert ersetzt (siehe Beispiel am Ende dieses Abschnittes).

Die folgenden Vereinbarungen erkennt die Shell:

```
${parameter=standard_wert}
```

Die Shell prüft: ist *parameter* definiert, d.h. ist *parameter* bereits ein Wert zugewiesen? Der zugewiesene Wert kann auch die leere Zeichenkette sein!

Wenn ja, setzt die Shell den definierten Wert ein.

Wenn nein, setzt die Shell *standard_wert* ein.

Dem Parameter wird aber kein Wert zugewiesen, der Standard-Wert gilt nur für diesen Zugriff.

```
${parameter:=standard_wert}
```

Die Shell prüft: ist *parameter* definiert, d.h. ist *parameter* bereits ein Wert zugewiesen? Der zugewiesene Wert darf nicht die leere Zeichenkette sein, der Doppelpunkt schließt die leere Zeichenkette aus.

Wenn ja, setzt die Shell den definierten Wert ein.

Wenn nein, setzt die Shell *standard_wert* ein.

Dem Parameter wird aber kein Wert zugewiesen, der Standard-Wert gilt nur für diesen Zugriff.

```
${parameter=standard_wert}
```

Für *parameter* dürfen Sie nur den Namen einer Shell-Variablen angeben.
Die Shell prüft: ist *parameter* definiert, d.h. ist *parameter* bereits ein Wert zugewiesen? Der zugewiesene Wert kann auch die leere Zeichenkette sein!

Wenn ja, setzt die Shell den definierten Wert ein.

Wenn nein, weist die Shell der Variablen *parameter standard_wert* zu und setzt diesen Wert ein.

Nur Shell-Variablen können Sie so Werte zuweisen. Deshalb darf *$parameter* kein Stellungsparameter und kein besonderer Shell-Parameter sein.

```
${parameter:=standard_wert}
```

Für *parameter* dürfen Sie nur den Namen einer Shell-Variablen angeben.
Die Shell prüft: ist *parameter* definiert, d.h. ist *parameter* bereits ein Wert zugewiesen? Der zugewiesene Wert darf nicht die leere Zeichenkette sein, der Doppelpunkt schließt die leere Zeichenkette aus.

Wenn ja, setzt die Shell den definierten Wert ein.

Wenn nein, weist die Shell der Variablen *parameter standard_wert* zu und setzt diesen Wert ein.

Nur Shell-Variablen können Sie so Werte zuweisen. Deshalb darf *$parameter* kein Stellungsparameter und kein besonderer Shell-Parameter sein.

```
${parameter?[standard_wert]}
```

Die Shell prüft: ist *parameter* definiert, d.h. ist *parameter* bereits ein Wert zugewiesen? Der zugewiesene Wert kann auch die leere Zeichenkette sein!

Wenn ja, setzt die Shell den definierten Wert ein.

Wenn die Bedingung nicht erfüllt und *standard_wert* angegeben ist, bricht die Shell die Ausführung des entsprechenden Kommandos oder der Shell-Prozedur ab und schreibt die folgende Fehlermeldung auf die Standard-Fehlerausgabe:

```
parameter : standard_wert
```

Wenn die Bedingung nicht erfüllt und *standard_wert* nicht angegeben ist, bricht die Shell die Ausführung des entsprechenden Kommandos oder der Shell-Prozedur ab und schreibt die folgende Fehlermeldung auf die Standard-Fehlerausgabe:

parameter : parameter null or not set

Dem Parameter wird kein Wert zugewiesen.

```
${parameter:?standard_wert}
```

Die Shell prüft: ist *parameter* definiert, d.h. ist *parameter* bereits ein Wert zugewiesen? Der zugewiesene Wert darf nicht die leere Zeichenkette sein, der Doppelpunkt schließt die leere Zeichenkette aus.

Wenn ja, setzt die Shell den definierten Wert ein.

Wenn die Bedingung nicht erfüllt und *standard_wert* angegeben ist, bricht die Shell die Ausführung des entsprechenden Kommandos oder der Shell-Prozedur ab und schreibt die folgende Fehlermeldung auf die Standard-Fehlerausgabe:

parameter : *standard_wert*

Wenn die Bedingung nicht erfüllt und *standard_wert* nicht angegeben ist, bricht die Shell die Ausführung des entsprechenden Kommandos oder der Shell-Prozedur ab und schreibt die folgende Fehlermeldung auf die Standard-Fehlerausgabe:

parameter : parameter null or not set

Dem Parameter wird kein Wert zugewiesen.

```
${parameter?standard_wert}
```

Die Shell prüft: ist *parameter* definiert, d.h. ist *parameter* bereits ein Wert zugewiesen? Der zugewiesene Wert kann auch die leere Zeichenkette sein!

Wenn ja, setzt die Shell *standard_wert* ein.

Wenn nein, setzt die Shell die leere Zeichenkette ein.

```
${parameter:+standard_wert}
```

Die Shell prüft: ist *parameter* definiert, d.h. ist *parameter* bereits ein Wert zugewiesen? Der zugewiesene Wert darf nicht die leere Zeichenkette sein, der Doppelpunkt schließt die leere Zeichenkette aus.

Wenn ja, setzt die Shell *standard_wert* ein.

Wenn nein, setzt die Shell die leere Zeichenkette ein.

Beispiel

```
$ echo ${dvz:-`pwd`}
/usr1/karl/src
```

Die Shell führt *pwd* nur aus, wenn die Variable *dvz* nicht definiert oder ihr Wert gleich der leeren Zeichenkette ist.

KOMMANDOS DURCH IHRE AUSGABE ERSETZEN

Die Shell ersetzt in der Kommandozeile ein Kommando durch seine Ausgabe, wenn dieses Kommando in Gegenhochkommas eingeschlossen ist:

```
`kommando`
```

kommando
: Name des Kommandos, das die Shell ausführen und dessen Ausgabe sie statt *`kommando`* in die Kommandozeile einsetzen soll.

  Wenn in der Kommandozeile nur *`kommando`* angegeben ist, versucht die Shell, die Ausgabe von *kommando* auszuführen.

Dabei passiert der Reihe nach folgendes:

- Die aktuelle Shell erkennt in der Kommandozeile die Gegenhochkommas und ruft eine Subshell auf.
- Die Subshell bearbeitet *kommando* (siehe Abschnitt *Wie bearbeitet die Shell die Kommandozeile?*) und beendet sich.
- In der Kommandozeile wird *`kommando`* ersetzt durch die Ausgabe von *kommando*.
- Die aktuelle Shell zerlegt die durch die Ersetzung veränderte Kommandozeile nochmals in Argumente. Argument-Trennzeichen sind dabei die Zeichen, die der Variablen IFS zugewiesen sind.
- Die aktuelle Shell führt die Kommandozeile aus.

Wenn Sie innerhalb der Gegenhochkommas die Standard-Ausgabe umgelenkt haben, wird *`kommando`* ersetzt durch die leere Zeichenkette.

Innerhalb der Gegenhochkommas können auch mehrere Kommandos stehen, jeweils getrennt durch ein Kommando-Trennzeichen.

Beispiel

Das Bereitzeichen der Shell soll über den aktuellen Rechner informieren:

```
$ PS1="<`hostname`> "
<amadeus>
```

Wegen der Anführungszeichen ist das Leerzeichen nach > fester Bestandteil von *$PS1*.

EINGABE UND AUSGABE EINES KOMMANDOS UMLENKEN

Standardmäßig ist die Standard-Eingabe eines Kommandos der Tastatur zugeordnet, die Standard-Ausgabe und die Standard-Fehlerausgabe sind dem Bildschirm zugeordnet. Die Standard-Eingabe, die -Ausgabe und die -Fehlerausgabe lassen sich auch mit den folgenden Datei-Kennzahlen ansprechen:

0 für Standard-Eingabe
1 für Standard-Ausgabe
2 für Standard-Fehlerausgabe

Wenn die Shell ein Kommando im Hintergrund startet, lenkt sie die Standard-Eingabe für dieses Kommando um auf die leere Datei */dev/null*. Aber mit der Angabe *<datei* können Sie die Standard-Eingabe eines Hintergrund-Kommandos auf eine andere Datei umlenken.

Wenn Sie die Standard-Zuordnung für ein Kommando ändern wollen, geben Sie die Umlenkung an einer beliebigen Stelle im entsprechenden Kommando an. Nachfolgend ist beschrieben, wie Sie

- die Standard-Eingabe umlenken
- die Standard-Ausgabe umlenken
- die Standard-Fehlerausgabe umlenken.

Die Standard-Eingabe für ein Kommando umlenken

Das Kleinerzeichen < leitet die Umlenkung der Standard-Eingabe ein. Die Angabe <... ist gleichbedeutend mit 0<...

<datei
lenkt die Standard-Eingabe des Kommandos um auf *datei*. Das Kommando liest seine Eingabe aus dieser Datei.

datei
Name der Datei, auf die die Standard-Eingabe umgelenkt werden soll. Diese Datei muß bereits existieren und für Sie lesbar sein.

Beachten Sie, daß die Shell zu diesem Zeitpunkt die Sonderzeichen *, ? und [...] noch nicht interpretiert hat. Verwenden Sie deshalb bei der Angabe von *datei* diese Sonderzeichen nicht.

<&-
Für das Kommando ist die Standard-Eingabe geschlossen. Das Kommando erhält also nur Datei-Ende (EOF) als Eingabe.

Beispiel

```
$ wc -l <&-
0
```

<&zahl
lenkt die Standard-Eingabe des Kommandos um auf die Datei, die der Dateikennzahl *zahl* zugewiesen ist. Das Kommando liest seine Eingabe aus dieser Datei.

zahl
Dateikennzahl der Datei, auf die die Standard-Eingabe umgelenkt werden soll.

Das Zeichen & nach < bedeutet also für die Shell, daß die nachfolgende Angabe eine Dateikennzahl und nicht ein Dateiname ist.

<<[-]zeichenkette
leitet ein Here-Dokument ein. Here-Dokumente werden in Shell-Prozeduren verwendet für Kommandos, die von der Standard-Eingabe lesen.
Die Standard-Eingabe des Kommandos wird umgelenkt auf das Here-Dokument. Das sind die Zeilen der Shell-Prozedur, die zwischen der Einleitung des Here-Dokuments und der Zeile stehen, die als erste nur *zeichenkette* enthält. Fehlt eine solche Zeile, so endet das Here-Dokument mit dem Ende der Shell-Prozedur.
Das Kommando liest seine Eingabe aus dem Here-Dokument.

Die Shell bearbeitet die Zeilen des Here-Dokuments: sie ersetzt Variablen durch ihren Wert und Kommandos, die in Gegenhochkommas eingeschlossen sind, durch ihre Ausgabe. Das entsprechende Kommando erhält die Zeilen des Here-Dokuments mit diesen Ersetzungen (siehe Abschnitt *WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?, Schritt 5*).
Wenn die Shell die Zeichen \, $ und '...' nicht interpretieren soll, müssen Sie diese Zeichen innerhalb des Here-Dokuments mit einem vorangestellten \ entwerten. Für das ganze Here-Dokument können Sie die Zeichen \, $ und '...' entwerten, indem Sie nach << in *zeichenkette* ein beliebiges Zeichen entwerten, z.B. durch einen vorangestellten \.

-

Die Shell entfernt alle führenden Tabulatorzeichen aus *zeichenkette* und aus dem Here-Dokument.

- nicht angegeben:
Die Shell entfernt führende Tabulatorzeichen nicht.

zeichenkette

beliebige Zeichenkette, die aber nicht innerhalb des Here-Dokuments vorkommen darf. Die erste Zeile im Here-Dokument, die nur *zeichenkette* enthält, beendet das Here-Dokument. Fehlt eine solche Zeile, gilt das Ende der entsprechenden Shell-Prozedur als Ende des Here-Dokuments.

Wenn in *zeichenkette* ein beliebiges Zeichen entwertet ist, z.B. durch einen vorangestellten \, sind alle Sonderzeichen der Shell im Here-Dokument für die Shell entwertet.
Als Ende-Kriterium dürfen Sie *zeichenkette* nur ohne Entwertungszeichen angeben.

Beispiel für ein Here-Dokument

Die Shell-Prozedur *post* hat folgenden Inhalt, allerdings ohne Zeilennummern:

```
1  mail $* <<ENDE

2  An alle Systemverwalter!!
3
4     Die naechste Besprechung zum Thema Datensicherheit findet
5     am Freitag, den 13. Dezember um 10 Uhr im Besprechungsraum
6     "Marseille" statt. Bitte nicht vergessen.
7
8  ENDE
```

Zum Here-Dokument gehören die Zeilen 2 bis 7, die Zeichenkette ENDE beendet das Here-Dokument. Die Standard-Eingabe des Kommandos *mail* wird auf das Here-Dokument umgelenkt.

Mit der folgenden Eingabe wird die Post an alle Systemverwalter verschickt, deren Namen in der Datei *system* eingetragen sind:

```
$ sh post `cat system`
```

Die Standard-Ausgabe für ein Kommando umlenken

Das Größerzeichen > leitet die Umlenkung der Standard-Ausgabe ein. Die Angabe >... ist gleichbedeutend mit 1>...

>datei
lenkt die Standard-Ausgabe des Kommandos um auf *datei*. Das Kommando schreibt seine Ausgabe in *datei*.

datei
Name der Datei, auf die die Standard-Ausgabe umgelenkt werden soll.

Existiert diese Datei bereits, wird ihr bisheriger Inhalt gelöscht. Existiert diese Datei noch nicht, wird sie neu angelegt.

Beachten Sie, daß die Shell zu diesem Zeitpunkt die Sonderzeichen *, ? und [...] noch nicht interpretiert hat. Verwenden Sie deshalb bei der Angabe von *datei* diese Sonderzeichen nicht.

>&-
Für das Kommando ist die Standard-Ausgabe geschlossen. Das Kommando gibt also nichts aus.

Beispiel

```
$ ls >&-
$
```

>&zahl
lenkt die Standard-Ausgabe des Kommandos um auf die Datei, die der Dateikennzahl *zahl* zugewiesen ist. Das Kommando schreibt seine Ausgabe in diese Datei.

zahl
Dateikennzahl der Datei, auf die die Standard-Ausgabe umgelenkt werden soll.

Das Zeichen & nach > bedeutet also für die Shell, daß die nachfolgende Angabe eine Dateikennzahl und nicht ein Dateiname ist.

>>datei
lenkt die Standard-Ausgabe des Kommandos um auf *datei*. Das Kommando schreibt seine Ausgabe in *datei*.

datei
Name der Datei, auf die die Standard-Ausgabe umgelenkt werden soll.

Existiert diese Datei bereits, wird die Ausgabe des Kommandos an den bisherigen Inhalt der Datei angehängt. Existiert diese Datei noch nicht, wird sie neu angelegt.

Beachten Sie, daß die Shell zu diesem Zeitpunkt die Sonderzeichen *, ? und [...] noch nicht interpretiert hat. Verwenden Sie deshalb bei der Angabe von *datei* diese Sonderzeichen nicht.

Die Standard-Fehlerausgabe für ein Kommando umlenken

Die Angabe 2> leitet die Umlenkung der Standard-Fehlerausgabe ein. Die Dateikennzahl 2 müssen Sie angeben.

2>datei
lenkt die Standard-Fehlerausgabe des Kommandos um auf *datei*. Das Kommando schreibt seine Fehlermeldungen in *datei*.

datei
Siehe Beschreibung zu *>datei*.

2>&1
Die Standard-Fehlerausgabe wird umgelenkt auf die Datei, der die Dateikennzahl 1 zugewiesen ist (siehe Beschreibung zu *>&zahl*). Die Fehlermeldungen und die Ausgaben des Kommandos werden nacheinander in die entsprechende Datei eingetragen.

2>>datei
lenkt die Standard-Fehlerausgabe des Kommandos um auf *datei*. Das Kommando schreibt seine Fehlermeldungen in *datei*.

datei
Siehe Beschreibung zu *>>datei*.

Mehrere Umlenkungen zu einem Kommando

Umlenkungen können Sie an beliebiger Stelle innerhalb der Kommandozeile angeben. Auch mehrere Umlenkungen in einer Kommandozeile sind möglich:

- Sie können gleichzeitig die Standard-Eingabe und die Standard-Ausgabe eines Kommandos umlenken. Die beiden Dateien, in die umgelenkt wird, sollten aber verschiedene Namen tragen. Sonst löscht die Shell die Datei, bevor das Kommando die Eingabe lesen konnte.

  Beispiel

  ```
  $ <liste sort >liste
  ```

  Nach der Ausführung dieses Kommandos ist die Datei *liste* leer. Die Shell lenkt die Standard-Eingabe und die Standard-Ausgabe für *sort* um, bevor sie *sort* startet. Die Angabe *>liste* bedeutet, der alte Inhalt der Datei *liste* wird gelöscht. Deshalb erhält *sort* keine Daten zum Sortieren und gibt auch nichts aus.

- Sie können gleichzeitig die Standard-Ausgabe und die Standard-Fehlerausgabe eines Kommandos umlenken. Dabei ist die Reihenfolge dieser Umlenkungen zu beachten. Die Shell wertet die Umlenkungen von links nach rechts aus.

  Beispiel

  ```
  1>ausgabe 2>&1
  ```

  Die Shell lenkt zuerst die Standard-Ausgabe um auf die Datei *ausgabe*. Dann lenkt sie die Standard-Fehlerausgabe um auf die Datei, der die Dateikennzahl 1 zugewiesen ist. Das ist die Datei *ausgabe*.
  Die Datei *ausgabe* enthält also die Ausgaben und die Fehlermeldungen des entsprechenden Kommandos.

  In anderer Reihenfolge:

  ```
  2>&1 1>ausgabe
  ```

  Die Shell lenkt zuerst die Standard-Fehlerausgabe um auf die Datei, der die Dateikennzahl 1 zugewiesen ist. Das ist standardmäßig der Bildschirm, d.h. die Datei */dev/tty*. Dann lenkt sie die Standard-Ausgabe um auf die Datei *ausgabe*. Die Datei *ausgabe* enthält also nur die Ausgaben, aber nicht die Fehlermeldungen des entsprechenden Kommandos.

ARGUMENTE DURCH PASSENDE DATEINAMEN ERSETZEN

Bei der Bearbeitung der Kommandozeile sucht die Shell nach Argumenten, die eines der folgenden Sonderzeichen enthalten (siehe Abschnitt *Wie bearbeitet die Shell die Kommandozeile?*, Schritt 9):

* ? [

Diese Argumente vergleicht die Shell als Muster mit den Dateinamen im entsprechenden Dateiverzeichnis. Wenn die Shell zu einem Argument einen passenden Dateinamen findet, ersetzt sie das Argument durch diesen Dateinamen. Wenn mehrere Dateinamen passen, ersetzt die Shell das Argument durch die sortierte Liste aller passenden Dateinamen. Die Sortierreihenfolge ergibt sich aus den Zeichensatzwerten. Jeder Dateiname ist ein eigenständiges Argument.

Wenn kein Dateiname paßt, verwendet die Shell das unveränderte Muster als Dateinamen.

Wenn das Muster keinen Schrägstrich / enthält, sucht die Shell die passenden Dateinamen immer im aktuellen Dateiverzeichnis. Wenn das Muster zu Dateinamen passen soll, die mit einem Punkt . beginnen, muß dieser Punkt als erstes Zeichen des Musters angegeben sein. Punkte an beliebiger Stelle innerhalb eines Dateinamens dagegen müssen Sie im Muster nicht angeben.

Ein Muster für Dateinamen ist eine beliebige Zeichenkette, die eines der folgenden Sonderzeichen oder eine beliebige Kombination daraus enthält:

*

als eigenständiges Muster:
Die Shell ersetzt * durch die Namen aller Dateien im aktuellen Dateiverzeichnis, die nicht mit einem Punkt . beginnen.
Ist das aktuelle Dateiverzeichnis leer, verwendet die Shell das Zeichen * unverändert als Dateinamen.

als Bestandteil eines Musters:
Die Shell ersetzt * durch kein, ein oder mehrere Zeichen, entsprechend den Dateinamen im aktuellen Dateiverzeichnis, zu denen das Muster paßt. Unter den Zeichen, die * ersetzen, kann auch ein Punkt sein, aber nur, wenn * nicht das erste Zeichen des Musters ist.
Ist das aktuelle Dateiverzeichnis leer oder paßt kein Dateiname zum angegebenen Muster, verwendet die Shell das unveränderte Muster mit dem Zeichen * als Dateinamen.

Das Sonderzeichen * wird nicht ersetzt durch:

einen Schrägstrich /
einen Punkt ., wenn * das erste Zeichen des Musters ist.

?

Die Shell ersetzt ? durch genau ein Zeichen, entsprechend den Dateinamen im aktuellen Dateiverzeichnis, zu denen das Muster paßt. Das Sonderzeichen ? kann auch durch einen Punkt ersetzt werden, aber nur, wenn ? nicht das erste Zeichen des Musters ist.
Ist das aktuelle Dateiverzeichnis leer oder paßt kein Dateiname zum angegebenen Muster, verwendet die Shell das unveränderte Muster mit dem Zeichen ? als Dateinamen.

Das Sonderzeichen ? wird nicht ersetzt durch:

einen Schrägstrich /
einen Punkt ., wenn * das erste Zeichen des Musters ist.

[s]

ersetzt die Shell durch genau ein Zeichen, das in der Zeichenkette *s* enthalten ist, entsprechend den Dateinamen im aktuellen Dateiverzeichnis, zu denen das Muster paßt. Die eckigen Klammern müssen Sie angeben.
Ist das aktuelle Dateiverzeichnis leer oder paßt kein Dateiname zum angegebenen Muster, verwendet die Shell das unveränderte Muster *[s]* als Dateinamen.

s

Zeichenkette. *s* darf einfache Zeichen außer dem Schrägstrich / enthalten. Wenn das Muster zu Dateinamen passen soll, die mit einem Punkt . beginnen, darf dieser Punkt nicht innerhalb der eckigen Klammern stehen. Beachten Sie, daß das Ausrufezeichen ! als erstes Zeichen und der Bindestrich - innerhalb der eckigen Klammern eine Sonderbedeutung haben. Die Zeichen * und ? haben innerhalb der eckigen Klammern keine Sonderbedeutung.

[!s]

ersetzt die Shell durch genau ein Zeichen, das in der Zeichenkette *s* nicht enthalten ist, entsprechend den Dateinamen im aktuellen Dateiverzeichnis, zu denen das Muster paßt. Die eckigen Klammern müssen Sie angeben.
Ist das aktuelle Dateiverzeichnis leer oder paßt kein Dateiname zum angegebenen Muster, verwendet die Shell das unveränderte Muster *[!s]* als Dateinamen.

s

wie bei *[s]*.

[c1-c2]
ersetzt die Shell durch genau ein Zeichen aus dem innerhalb der eckigen Klammern angegebenen Bereich, entsprechend den Dateinamen im aktuellen Dateiverzeichnis, zu denen das Muster paßt. Die eckigen Klammern müssen Sie angeben. Ist das aktuelle Dateiverzeichnis leer oder paßt kein Dateiname zum angegebenen Muster, verwendet die Shell das unveränderte Muster mit den eckigen Klammern als Dateinamen.

c1-c2
legt einen Zeichenbereich fest. Zu diesem Bereich gehören alle Zeichen, die im ASCII-Zeichensatz zwischen *c1* und*c2* liegen. *c1* und *c2* gehören zu diesem Bereich. Sie können auch mehrere Zeichenbereiche ohne Trennzeichen angeben.

c1
c1 darf ein einfaches Zeichen außer Schrägstrich / sein.

Wenn das Muster zu Dateinamen passen soll, die mit einem Punkt . beginnen, muß dieser Punkt das erste Zeichen des Musters sein, also vor der öffnenden eckigen Klammer [stehen.

Beachten Sie, daß das Ausrufezeichen | als erstes Zeichen und der Bindestrich - innerhalb der eckigen Klammern eine Sonderbedeutung haben.
Die Zeichen * und ? haben innerhalb der eckigen Klammern keine Sonderbedeutung.

c2
c2 darf ein einfaches Zeichen außer Schrägstrich / sein.

[!c1-c2]
ersetzt die Shell durch genau ein Zeichen, das nicht zu dem innerhalb der eckigen Klammern angegebenen Bereich gehört, entsprechend den Dateinamen im aktuellen Dateiverzeichnis, zu denen das Muster paßt. Die eckigen Klammern müssen Sie angeben.
Ist das aktuelle Dateiverzeichnis leer oder paßt kein Dateiname zum angegebenen Muster, verwendet die Shell das unveränderte Muster mit den eckigen Klammern als Dateinamen.

c1-c2
legt einen Zeichenbereich fest (siehe Beschreibung zu *[c1-c2]*.

WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?

Im Dialog schließen Sie mit ⏎ Ihre Eingabe ab und übergeben die Kommandozeile an die Shell. In welchen Portionen die Shell die Eingabe erhält, ist abhängig von der Einstellung der jeweiligen Datensichtstation (siehe *stty*).
Eine Prozedur-Shell erhält als Eingabe den Inhalt der entsprechenden Shell-Prozedur.

Die Kommandozeile kann mehrere Kommandos enthalten, die durch Kommando-Trennzeichen voneinander getrennt sind. Innerhalb einer Shell-Prozedur sind auch Neue-Zeile-Zeichen als Kommando-Trennzeichen zugelassen. Die Shell bearbeitet der Reihe nach jedes einzelne Kommando und führt es aus. Deshalb bezeichnet in der nachfolgende Beschreibung der Begriff Kommandozeile die logische Einheit, die durch nicht entwertete Kommando-Trennzeichen begrenzt ist.
Erst wenn die Shell alle Kommandos der Eingabe ausgeführt hat, gibt sie wieder ihr Bereitzeichen aus.

Die Shell bearbeitet jede Kommandozeile in mehreren Schritten. Die einzelnen Bearbeitungsschritte führt die Shell für jede Kommandozeile einmal aus. Die Reihenfolge dieser Schritte liegt fest.
Diese Reihenfolge sollten Sie beachten, damit die Shell Ihre Eingaben wie gewünscht bearbeiten kann.

Wie die Shell eine Kommandozeile bearbeitet, ist auch abhängig von den Shell-Optionen, die Sie beim Aufruf dieser Shell (siehe *sh*) oder nachträglich mit *set* gesetzt haben.

Die einzelnen Schritte in chronologischer Reihenfolge:

1. Die Shell liest die Kommandozeile bis zum ersten Kommando-Trennzeichen, das nicht entwertet ist, und sucht nach Hochkommata, Anführungszeichen und Gegenhochkommas.
2. Die Shell stellt fest, ob die Kommandozeile Eingabe- bzw. Ausgabe-Umlenkungen oder Variablen-Zuweisungen enthält.
3. Die Shell zerlegt die Kommandozeile in einzelne Argumente.
4. Die Shell ersetzt angegebene Variablen durch ihren Wert.
5. Die Shell ersetzt Kommandos, die in Gegenhochkommas eingeschlossen sind, durch ihre Ausgabe.
6. Die Shell lenkt die Standard-Eingabe, die Standard-Ausgabe bzw. die Standard-Fehlerausgabe auf die angegebenen Dateien um.
7. Die Shell führt Variablen-Zuweisungen aus.
8. Die Shell zerlegt die so bearbeitete Zeile nochmals in Argumente, entsprechend den Argument-Trennzeichen, die der Variablen *IFS* zugewiesen sind.

9. Die Shell ersetzt Argumente durch passende Dateinamen, falls sie eines der folgenden Zeichen bzw. Zeichenketten enthalten: * ? [...] [!...]
10. Die Shell führt das Kommando aus.

Die Bearbeitungsschritte 5, 6 und 7 werden nur dann ausgeführt, wenn die Shell in Schritt 1 und Schritt 2 die entsprechenden Sonderzeichen in der Kommandozeile erkannt hat.

Schritt 1: bis zum ersten Kommando-Trennzeichen lesen

Die Shell liest von der Standard-Eingabe oder aus einer Datei bis zum ersten Kommando-Trennzeichen:

Neue-Zeile-Zeichen ; & | && ||

Wenn Sie nach dem Bereitzeichen der Shell kein Kommando eingeben, sondern sofort die Taste END drücken, beendet sich die aktuelle Shell. Wenn die Subshell, die eine Shell-Prozedur ausführt, das letzte Kommando der Prozedur ausgeführt hat, erhält sie Datei-Ende (EOF) und beendet sich. Das gilt jedoch nicht, wenn Sie eine Shell-Prozedur mit dem eingebauten *sh*-Kommando . (Punkt) starten. Die aktuelle Shell, die diese Prozedur ausführt, beendet sich nicht, wenn sie Datei-Ende erhält.

Dabei durchsucht die Shell die gelesenen Daten

- nach Entwertungszeichen:

  \ '...' "..."

  Die dadurch entwerteten Sonderzeichen sind ab jetzt vor der Interpretation durch die Shell geschützt (siehe *Sonderzeichen für die Shell entwerten*).

- nach Gegenhochkommas:

  `*kommando*`

  Dieser Ausdruck soll in Schritt 5 durch die Ausgabe von *kommando* ersetzt werden (siehe Abschnitt *KOMMANDOS DURCH IHRE AUSGABE ERSETZEN*). Die Shell führt diese Aktion in Schritt 5 nur aus, wenn sie die Gegenhochkommas in Schritt 1 erkennt.

  Die Shell schützt *'kommando'* vor der Bearbeitung in Schritt 3. Die Behandlung ist so, als ob der Ausdruck in Anführungszeichen eingeschlossen wäre: " *'kommando'* ".

Wenn in der aktuellen Shell die Option *-v* gesetzt ist, gibt die Shell die Kommandozeile nach dem 1. Schritt aus (siehe *sh* und *set*).

Schritt 2: die Sonderzeichen > < und = suchen

Die Shell sucht nach den folgenden Sonderzeichen, die in einem späteren Bearbeitungsschritt Aktionen auslösen:

> oder <
in Schritt 6 soll die Standard-Eingabe, die Standard-Ausgabe oder die Standard-Fehlerausgabe entsprechend den weiteren Angaben umgelenkt werden (siehe *EINGABE UND AUSGABE EINES KOMMANDOS UMLENKEN*).

=
in Schritt 7 soll einer Variablen ein Wert zugewiesen werden (siehe Abschnitt *SHELL-VARIABLEN*).

Die Shell führt diese Aktionen nur aus, wenn sie die angegebenen Sonderzeichen zu diesem Zeitpunkt erkennt.

Schritt 3: in Argumente zerlegen

Die Shell zerlegt die Kommandozeile in Argumente. Argument-Trennzeichen für diesen Bearbeitungsschritt sind:

Leerzeichen Tabulatorzeichen

Anschließend entfernt sie alle Argument-Trennzeichen aus der Kommandozeile.
Auf die Variable IFS greift die Shell erst im achten Schritt zu, wenn sie die Kommandozeile zum zweiten Mal in Argumente zerlegt.

Außerdem prüft die Shell, ob das erste Argument der Kommandozeile, also das Argument, das direkt auf ein Kommando-Trennzeichen folgt, eines der folgenden Schlüsselwörter oder Sonderzeichen ist (siehe auch *Die Shell als Programmiersprache*):

if elif case for while until { (

Wenn ja, entfernt sie das Schlüsselwort bzw. die öffnende runde Klammer aus der Kommandozeile. Dann wiederholt sie die Schritte 1 bis 3 für die nachfolgenden Kommandozeilen bis zu der Kommandozeile, die mit dem entsprechenden abschließenden Schlüsselwort oder der schließenden Klammer beginnt:

fi esac done })

Stößt die Shell dabei auf einen Syntax-Fehler, so gibt sie eine Fehlermeldung aus und bricht die weitere Bearbeitung ab.

Schritt 4: Variablen durch ihren Wert ersetzen

Die Shell sucht nach dem Zeichen $ gefolgt von einem zulässigen Variablen-Namen:

$0 $1 $2 ... $9 $* $@ $# $? $! $$ $- $*name* ${...}

Die entsprechenden Shell-Variablen ersetzt die Shell durch ihren Wert. Nicht definierte Variablen ersetzt die Shell durch die leere Zeichenkette
(siehe auch *SHELL-VARIABLEN* und *Shell-Parameter*).

Schritt 5: Kommandos durch ihre Ausgabe ersetzen

Die Shell bearbeitet die Zeichenketten, die in Gegenhochkommas `...` eingeschlossen sind. Allerdings gilt das nur für die Gegenhochkommas, die die Shell bereits in Schritt 1 erkannt hat:

`*kommando*`

Wenn Sie innerhalb der Gegenhochkommas die Standard-Ausgabe umgelenkt haben, wird `*kommando*` ersetzt durch die leere Zeichenkette.

Innerhalb der Gegenhochkommas können auch mehrere Kommandos stehen, jeweils getrennt durch ein Kommando-Trennzeichen (siehe auch *KOMMANDOS DURCH IHRE AUSGABE ERSETZEN*).

Schritt 6: Standard-Eingabe, -Ausgabe bzw. -Fehlerausgabe umlenken

Die Shell bearbeitet die Argumente der Kommandozeile, bei denen sie in Schritt 2 eines der folgenden Zeichen gefunden hat:

< <&- <& << <<- > >&- >& >>

Nach der Umlenkung entfernt die Shell die dafür verantwortlichen Argumente aus der Kommandozeile. Das Kommando selbst merkt nichts von der Umlenkung (siehe auch *EINGABE UND AUSGABE EINES KOMMANDOS UMLENKEN*).

Schritt 7: Werte an Variablen zuweisen

Die Shell bearbeitet die Stellen der Kommandozeile, an denen sie in Schritt 2 auf ein Gleichheitszeichen = gestoßen ist (siehe auch *SHELL-VARIABLEN* und *DIE UMGEBUNG*).

Die Kommandozeile kann nach den sechs vorausgegangenen Bearbeitungsschritten wie folgt aussehen:

name=wert␣...
: Wenn die Kommandozeile nur eine Zuweisung, aber keinen Kommandonamen enthält, legt die aktuelle Shell die Variable *name* an und weist ihr *wert* zu. Eine Kommandozeile kann auch mehrere Zuweisungen enthalten.

  Wenn die Shell allen angegebenen Variablen die entsprechenden Werte zugewiesen hat, ist die Kommandozeile fertig bearbeitet. Die Shell führt also die Schritte 8 bis 10 nicht mehr aus.

  Die aktuelle Umgebung enthält ab jetzt diese neuen Variablen. Sollen die Variablen auch in einer Subshell bekannt sein, so müssen sie exportiert werden (siehe eingebautes *sh*-Kommando *export*). Wenn die Shell-Option *-a* gesetzt ist, werden diese Variablen automatisch exportiert (siehe eingebautes *sh*-Kommando *set* und *sh*).

name=wert␣...␣kommando
: Wenn die Kommandozeile zusätzlich zu einer Zuweisung einen Kommandonamen enthält, trägt die aktuelle Shell die angegebene Variable nicht in die aktuelle Umgebung ein, sondern nur in die Umgebung für *kommando*. Vor dem Kommandonamen können auch mehrere Zuweisungen stehen.

  Anschließend entfernt sie die Zuweisungen aus der Kommandozeile und startet das angegebene *kommando* in der neuen Umgebung.

  Wenn Sie in der aktuellen Shell mit *set* die Option *-k* gesetzt haben, können Sie diese Zuweisungen auch hinter den Kommandonamen schreiben (siehe *set*).

Schritt 8: in Argumente zerlegen

Die Shell zerlegt die Kommandozeile nochmals in Argumente. Argument-Trennzeichen sind die Zeichen, die der Variablen IFS zugewiesen sind. Standardmäßig sind der Variablen IFS zugewiesen:

Leerzeichen Tabulatorzeichen Neue-Zeile-Zeichen

In Schritt 8 interpretiert die Shell ein Neue-Zeile-Zeichen nicht mehr als Kommando-Trennzeichen.

Diese zweite Zerlegung ist notwendig, weil sich die Kommandozeile seit Schritt 3 möglicherweise verändert hat:

- Die Variablen sind durch ihre Werte ersetzt worden (Schritt 4).
- Kommandos in Gegenhochkommas sind durch ihre Ausgaben ersetzt worden (Schritt 5).
- Schlüsselwörter und runde Klammern hat die Shell aus der Kommandozeile entfernt (Schritt 3).
- Umlenkungs- und Zuweisungsargumente hat die Shell aus der Kommandozeile entfernt (Schritt 6 und Schritt 7).

Anschließend entfernt die Shell alle Argument-Trennzeichen aus der Kommandozeile. Außerdem entfernt sie die leeren Zeichenketten, die nicht in Anführungszeichen oder Hochkommata eingeschlossen sind. Solche nicht geschützten leeren Zeichenketten entstehen, wenn die Shell nicht definierte Variablen durch die leere Zeichenkette ersetzt.
Leere Zeichenketten der Form "" oder '' bleiben als Argumente erhalten.

Nach dieser Zerlegung enthält das erste Argument den späteren Kommandonamen, die weiteren Argumente die späteren Aufrufargumente für das Kommando.

Schritt 9: Argumente durch passende Dateinamen ersetzen

Die Shell sucht nach Argumenten, die eines der folgenden Sonderzeichen enthalten:

* ? [

Diese Argumente vergleicht die Shell als Muster mit den Dateinamen im entsprechenden Dateiverzeichnis (siehe Abschnitt *ARGUMENTE DURCH PASSENDE DATEINAMEN ERSETZEN*).

Wenn die Shell zu einem Argument einen passenden Dateinamen findet, ersetzt sie das Argument durch diesen Dateinamen. Wenn mehrere Dateinamen passen, ersetzt die Shell das Argument durch die sortierte Liste aller passenden Dateinamen. Jeder Dateiname ist ein eigenständiges Argument.

Wenn kein Dateiname paßt, verwendet die Shell das unveränderte Muster als Dateinamen.

Wenn in der aktuellen Shell die Option *-f* gesetzt ist, entfällt dieser Bearbeitungsschritt. Wenn in der aktuellen Shell die Option *-n* gesetzt ist, gibt die Shell die Kommandozeile nach Schritt 9 aus. Der Schritt 10 entfällt (siehe *sh* und *set*).

Schritt 10: das Kommando ausführen

Die Kommandozeile enthält jetzt nur noch die Argumente, die für die Ausführung des entsprechenden Kommandos nötig sind. Die Shell interpretiert das erste Argument als den Namen des Kommandos, das sie ausführen soll. Die übrigen Argumente interpretiert die Shell als Aufrufargumente für dieses Kommando und übergibt sie beim Aufruf an das Kommando.

Den Kommandonamen prüft die Shell in der folgenden Reihenfolge:

- Ist das Kommando ein eingebautes *sh*-Kommando?

  Wenn ja, führt die Shell dieses Kommando selbst aus.
  Wenn nein, prüft die Shell weiter.

- Ist das Kommando eine Shell-Funktion?

  Wenn ja, führt die Shell diese Funktion selbst aus.

  Wenn nein, nimmt die Shell an, daß es sich um ein externes Kommando handelt. Deshalb erzeugt sie mit *fork()* einen neuen Prozeß. Dieser Prozeß ist eine Kopie der aufrufenden Shell, der neue Prozeß erbt also die Umgebung der aufrufenden Shell.
  Dieser neue Shell-Prozeß arbeitet jetzt weiter:

- Enthält der Kommandoname einen Schrägstrich / ?

  Wenn ja, versucht der Shell-Prozeß, das entsprechende Kommando unter dem angegebenen Namen mit *exec()* zu starten, ohne Zugriff auf die Variable PATH.

  Eine eingeschränkte Shell weist Kommandos sofort ab, wenn im Namen ein Schrägstrich enthalten ist (siehe *sh*, Option *-r*).

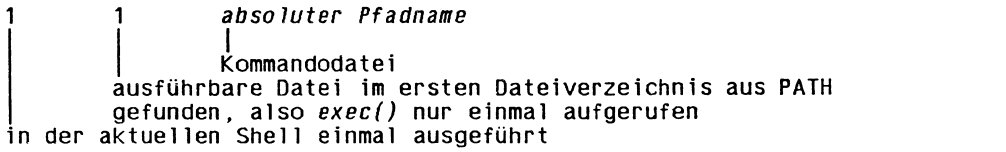
  Enthält der Kommandoname keinen /, wird der Name in der aktuellen Hash-Tabelle (siehe *hash*) gesucht:

  - Enthält die Hash-Tabelle diesen Namen, versucht der Shell-Prozeß, das Kommando unter dem in der Hash-Tabelle angegebenen absoluten Pfadnamen mit *exec()* zu starten, ohne Zugriff auf die Variable *PATH*.
  - *Steht der Kommandoname nicht in der Hash-Tabelle, wird die Variable PATH* ausgewertet. Der absolute Pfadname des Kommandos wird zusammengesetzt aus dem ersten Pfadnamen der Variablen *PATH*, einem / und dem Kommandonamen. Anschließend versucht der Shell-Prozeß, das Kommando unter diesem absoluten Pfadnamen mit *exec()* zu starten.

- Kann der Systemaufruf *exec()* die Datei laden?

  Wenn ja, wird das Kommando ausgeführt. Bei Kommandos, die im Hintergrund ausgeführt werden sollen, meldet sich die aufrufende Shell sofort mit ihrem Bereitzeichen zurück. Bei allen anderen Kommandos wartet sie auf das Ende der Ausführung.

  Bei Kommandos, die mit ihrem einfachen Dateinamen aufgerufen wurden, wird die Hash-Tabelle aktualisiert. Ein Kommando, das noch nicht in der Hash-Tabelle stand, wird wie folgt eingetragen:



  War das Kommando bereits in die Hash-Tabelle eingetragen, so wird nur in der entsprechenden Zeile der Wert in der Spalte *hits* um 1 erhöht.

  Wenn *exec()* die Datei nicht laden kann, hat das eine der folgenden Ursachen:

  - Die Datei dieses Namens ist keine Binär-Datei (z.B. a.out-Format).
    Dann muß die Datei eine Shell-Prozedur sein. Der aktuelle Shell-Prozeß liest die Datei und führt die Kommandos entsprechend den hier beschriebenen Schritten aus.

- Der Benutzer, der das Kommando aufgerufen hat, hat kein Ausführrecht für diese Datei.
  In diesem Fall meldet sich die aufrufende Shell zurück mit der Fehlermeldung:
  `kommando: execute permission denied.`

- *exec()* findet diese Datei nicht.
  Wenn der Aufruf-Name einen / enthielt oder das Kommando bereits in der Hash-Tabelle eingetragen war (also kein Zugriff auf die Variable PATH), meldet sich die aufrufende Shell zurück mit der Fehlermeldung: `kommando: not found.`

  Andernfalls wertet der Shell-Prozeß nochmals die Variable *PATH* aus. Er versucht, das Kommando mit *exec()* unter dem absoluten Pfadnamen zu starten, der sich aus dem zweiten Pfadnamen der Variablen *PATH* ergibt.

  Wenn *exec()* die Datei nicht findet, wiederholt sich der Vorgang mit dem nächsten Pfadnamen der Variablen *PATH*. Weitere Wiederholungen sind solange möglich, bis alle Pfadnamen aus *PATH* ausgewertet sind.

  Wenn *exec()* die Datei nicht findet, obwohl alle Pfadnamen der Variablen *PATH* ausgewertet sind, meldet sich die aufrufende Shell zurück mit der Fehlermeldung *kommando: not found.*

Die Shell als Programmiersprache

Die Shell bietet die wesentlichen Elemente einer Programmiersprache:

- Variablen
- Parameter
- eingebaute Funktionen; das sind die eingebauten *sh*-Kommandos
- Kommentare
- Makros; das sind die Shell-Funktionen
- Ablaufanweisungen

Jede Kommandozeile, die Sie nach dem Bereitzeichen der Shell eingeben, ist bereits ein kleines Shell-Programm. Die Shell interpretiert diese Zeile nach bestimmten Regeln und führt das entsprechende Kommando aus (siehe *WIE BEARBEITET DIE SHELL DIE KOMMANDOZEILE?*).

Größere Programme erstellen Sie mit einem Editor: Sie legen eine Datei an und tragen dort alle Kommandos ein, die die Shell der Reihe nach ausführen soll. Diese Shell-Prozeduren müssen nicht übersetzt werden. Wenn Sie eine Shell-Prozedur mit *sh dateiname* aufrufen, erzeugt die Shell eine Subshell, die jede Zeile der Datei liest, interpretiert und ausführt. Sind alle Kommandos aus dieser Datei ausgeführt, beendet sich die Subshell, und die übergeordnete Shell meldet sich zurück.

Dieser Abschnitt beschreibt,

- wie Sie Shell-Prozeduren kommentieren,
- wie Sie Shell-Funktionen erstellen und aufrufen,
- und die Ablaufanweisungen der Shell in alphabetischer Reihenfolge.

KOMMENTARE IN SHELL-PROZEDUREN

Das Nummernzeichen # leitet einen Kommentar ein. Die Shell ignoriert alle auf # folgenden Zeichen bis zum ersten Neue-Zeile-Zeichen, auch wenn es entwertetet ist.

Beispiel

In einer Shell-Prozedur können Kommentare wie folgt angegeben sein:

```
date # Kommentar
echo \# leitet keinen Kommentar ein
pwd   #Kommentar
date # ein Kommentar, der trotzdem hier endet \
echo \# ein Kommentar geht nicht ueber mehrere Zeilen!
```

Wenn Sie # entwerten, verliert es seine Sonderbedeutung als Kommentar-Zeichen.

Es gibt aber noch eine andere Möglichkeit, Kommentare einzuleiten, und zwar mit dem eingebauten *sh*-Kommando *:* (Doppelpunkt). Dieses Kommando gibt nur den Ende-Status 0 zurück und tut sonst nichts.

Der nachfolgende Kommentar muß vom Doppelpunkt durch ein Leer- oder ein Tabulatorzeichen getrennt sein. Bei der Kommentierung mit *:* müssen Sie folgendes beachten:

- Die Shell interpretiert die Aufrufargumente, also den Kommentar. Insbesondere ersetzt sie darin vorkommende Shell-Parameter durch deren Wert. Wenn Sie den Kommentar in Hochkommata einschließen, entwerten Sie alle Sonderzeichen für die Shell.
- Der Doppelpunkt muß immer als erstes Zeichen in einer Zeile stehen oder auf ein Kommando-Trennzeichen folgen, z.B. auf einen Strichpunkt ;. Andernfalls erkennt die Shell den Doppelpunkt nicht als Kommando.

Beispiel für Kommentare mit *:* (Doppelpunkt):

```
: Die Shell-Prozedur zeigt moegliche Kommentare \
  sie gibt die Prozess-Nummer der Shell und das Datum aus
echo $$;    : '$$ ist die Prozess-Nummer der aktuellen Shell'; date
```

Mit *:* (Doppelpunkt) können Sie Kommentare über mehrere Zeilen schreiben oder zwischen zwei Kommandos, die in derselben Zeile stehen.

SHELL-FUNKTIONEN

Eine Shell-Funktion faßt mehrere Kommandos unter einem Namen zusammen.

Dieser Abschnitt beschreibt,

- wie Sie eine Shell-Funktion definieren,
- wie Sie eine Shell-Funktion ausführen,
- wie sich Shell-Funktionen und Shell-Prozeduren unterscheiden,
- und das eingebaute *sh*-Kommando *return*.

Eine Shell-Funktion definieren

Die Definition einer Shell-Funktion hat folgendes Format:

```
name( ) {kommandofolge;}
```

name
: Name der Shell-Funktion. Ein Funktionsname muß mit einem Buchstaben oder mit einem Unterstrich _ beginnen. Dem ersten Zeichen dürfen Buchstaben, Ziffern und Unterstriche in beliebiger Reihenfolge folgen.

  Eine Shell-Funktion und eine Shell-Variable können nicht den gleichen Namen tragen. Wenn Sie eine Shell-Funktion mit einem Namen definieren, unter dem es bereits eine Shell-Variable gibt, so geht der Wert der Shell-Variablen verloren, und umgekehrt.

kommandofolge
: ein oder mehrere Kommandos, wie sie auch in Shell-Prozeduren erlaubt sind. Die einzelnen Kommandos werden durch ein Neue-Zeile-Zeichen oder ein anderes Kommando-Trennzeichen getrennt.
  Eine Shell-Funktion kann selbst wieder Shell-Funktionen aufrufen oder definieren.

  Die Kommandos aus *kommandofolge* werden ausgeführt, wenn Sie die Shell-Funktion aufrufen.

Das Leerzeichen nach der öffnenden geschweiften Klammer kann durch ein Neue-Zeile-Zeichen ersetzt werden, ebenso wie der Strichpunkt vor der schließenden geschweiften Klammer.

Beispiel

Im Dialog die Shell-Funktion *ll* definieren:

```
$ ll() {
> ls -l $*
> }
```

Mit der Shell-Funktion *ll* können Sie jetzt das Kommando *ls -l* simulieren.

Nach der Definition ist die Shell-Funktion der aktuellen Shell bekannt. Das eingebaute *sh*-Kommando *set* schreibt alle in der aktuellen Shell definierten Shell-Variablen mit ihren Werten und alle in der aktuellen Shell definierten Funktionen auf die Standard-Ausgabe. Das eingebaute *sh*-Kommando *type* mit einem Funktionsnamen als Argument gibt ebenfalls die Definition der Shell-Funktion aus.

Sie können Shell-Funktionen nicht exportieren, sie sind nur der Shell bekannt, in der sie definiert wurden.
Bereits definierte Shell-Funktionen können Sie mit *unset* wieder löschen.

Innerhalb einer Shell-Funktion können Sie auf alle Variablen zugreifen, die der aktuellen Shell bekannt sind. Wie auch bei Shell-Prozeduren können Sie beim Aufruf Stellungsparameter übergeben. Der Shell-Parameter $0 enthält jedoch nicht den Funktionsnamen, sondern den Aufruf-Namen der aktuellen Shell oder den Namen der Shell-Prozedur, je nachdem wo Sie die Funktion aufgerufen haben.

Shell-Funktionen in einer Datei definieren

Wenn Sie eine Shell-Funktion immer wieder verwenden wollen, sollten Sie die Definition in eine Datei schreiben. Diese Datei führen Sie mit dem eingebauten *sh*-Kommando . (Punkt) in der aktuellen Shell aus. Anschließend können Sie auf diese Funktion in der aktuellen Shell zugreifen. Sie können in dieser Datei auch mehrere Shell-Funktionen sammeln.

Eine Shell-Funktion ausführen

Eine Shell-Funktion rufen Sie auf wie ein Kommando. Die aktuelle Shell führt die Shell-Funktion aus, sie erzeugt also keine Subshell wie bei Shell-Prozeduren.

Beispiel

```
$ ll /dev
```

So wird die gerade definierte Shell-Funktion *ll* aufgerufen. Den Shell-Parameter $* versorgt die Shell mit dem Dateiverzeichnis */dev*.

Beim Aufruf können Sie wie bei Shell-Prozeduren weitere Argumente angeben. Mit diesen Argumenten versorgt die aktuelle Shell die Stellungsparameter. Sie ändern also beim Aufruf einer Shell-Funktion die Stellungsparameter der aktuellen Shell wie mit *set*.

Sie können jedoch eine Shell-Funktion nur in der Shell aufrufen, in der Sie sie definiert haben.

Variablen-Zuweisungen innerhalb einer Shell-Funktion verändern die Umgebung der aktuellen Shell.

Unterschiede zwischen Shell-Funktionen und Shell-Prozeduren

Shell-Funktionen sind effizienter als Shell-Prozeduren. Sie werden von der Shell wie die eingebauten Kommandos bevorzugt ausgeführt. Die Shell erzeugt keine Subshell, und sie muß die Daten nicht erst aus einer Datei laden.

Eine Shell-Funktion ist an die Shell gebunden, in der sie definiert wurde.

Jeder Aufruf einer Shell-Funktion setzt alle Stellungsparameter neu.

Mit dem eingebauten *sh*-Kommando *return* können Sie Shell-Funktionen mit einem vorgegebenen Ende-Status beenden. Das Kommando *return* kann nur innerhalb von Shell-Funktionen aufgerufen werden.
Standardmäßig beendet sich eine Shell-Funktion, wenn alle Kommandos der Funktion ausgeführt sind.

Der Ende-Status einer Shell-Funktion

Standardmäßig ist der Ende-Status einer Shell-Funktion der Ende-Status des innerhalb der Funktion zuletzt ausgeführten Kommandos. Diesen Ende-Status können Sie beispielsweise mit *echo $?* abfragen.

Wenn Sie eine Shell-Funktion mit einem definierten Ende-Status beenden wollen, sollten Sie das eingebaute *sh*-Kommando *return* verwenden. Dieses Kommando können Sie nur innerhalb einer Shell-Funktion aufrufen.

Wie in Shell-Prozeduren könnten Sie auch mit dem eingebauten *sh*-Kommando *exit* einen Ende-Status festlegen. Das Kommando *exit* beendet aber nicht nur die Shell-Prozedur bzw. die Shell-Funktion, sondern auch die ausführende Shell.

Eine Shell-Funktion beenden

Das eingebaute *sh*-Kommando *return* beendet Shell-Funktionen. Beim Aufruf können Sie mit einer Zahl den Ende-Status der entsprechenden Shell-Funktion festlegen.

Auch ohne Aufruf von *return* beenden sich Shell-Funktionen:

- wenn alle Kommandos der Funktion ausgeführt sind. Ende-Status ist der Ende-Status des zuletzt ausgeführten Kommandos.
- wenn die ausführende Shell ein Kommando nicht findet oder Syntax-Fehler erkennt. Ende-Status ist ein Wert ≠0.

Sie können *return* nur innerhalb einer Shell-Funktion aufrufen.

```
return[ ende_status]
```

ende_status
: Eine Zahl zwischen 0 und 256. Mit diesem Ende-Status wird die Shell-Funktion beendet.
Wenn Sie eine größere Zahl angeben, erhalten Sie als Ende-Status nicht die angegebene Zahl, sondern den entsprechenden Wert modulo 256: Die angegebene Zahl wird durch 256 geteilt, der ganzzahlige Rest ist dann der entsprechende Wert modulo 256.

  Mit dem Kommando *echo $?* können Sie den Ende-Status abfragen.

  ende_status nicht angegeben:
  Ende-Status der Shell-Funktion ist der Ende-Status des Kommandos, das vor dem Aufruf von *return* ausgeführt wurde.

Beispiel

Das Kommando *false* läßt sich durch folgende Shell-Funktion realisieren:

```
false() {
return 255
}
```

ABLAUFANWEISUNGEN DER SHELL

Zu einer Programmiersprache gehören Ablaufanweisungen. Die Shell bietet die folgenden:

break
Schleifen steuern

case ... esac
abfragen und verzweigen

continue
Schleifen steuern

for ... do ... done
Liste in Schleifen abarbeiten

if ... then ... elif ... then ... else ... fi
Bedingungen abfragen und Kommandos ausführen

until ... do ... done
Schleife mit Abbruch-Bedingung

while ... do ... done
Schleife mit Abbruch-Bedingung

Diese Ablaufanweisungen können Sie im Dialog eingeben oder in einer Shell-Prozedur verwenden. Die folgenden Schlüsselwörter erkennt die Shell nur, wenn sie am Beginn eines Kommandos stehen, also nach einem Kommando-Trennzeichen, und wenn sie nicht entwertet sind:

case	for
do	if
done	then
elif	until
else	while
esac	{
fi	}

Die geschweiften Klammern fassen die Ausgabe der eingeschlossenen Kommandos zusammen (siehe *Kommandofolgen klammern*).

Die Ablaufanweisungen *if*, *until* und *while* prüfen zuerst eine Bedingung. Bedingungen kann man in der Shell nur durch ein Kommando oder eine Kommandofolge (siehe *Kommandofolgen*) darstellen. Die Shell wertet den Ende-Status aus.

- Bedingung wahr
  Das als Bedingung angegebene Kommando oder die Kommandofolge hat einen Ende-Status gleich 0.
- Bedingung falsch
  Das als Bedingung angegebene Kommando oder die Kommandofolge hat einen Ende-Status ≠0.

Die Ablaufanweisungen können Sie beliebig kombinieren und schachteln.

break - Schleife abbrechen

Das eingebaute *sh*-Kommando *break* bricht die Abarbeitung der *for-*, *until-* oder *while*-Schleife ab, in der es angegeben ist. Eine Shell-Prozedur wird mit dem Kommando fortgesetzt, das hinter dem nächsten nicht erreichten *done* steht.

Schleifen können beliebig tief geschachtelt sein. Wenn Sie mit *break* mehrere verschachtelte Schleifen abbrechen wollen, können Sie dies durch eine entsprechende Zahl beim Aufruf angeben

```
break [␣n]
```

n
: die Anzahl der verschachtelten *for-*, *until-* oder *while*-Schleifen, die *break* abbrechen soll.

  n nicht angegeben:
  break bricht die *for-*, *until-* oder *while*-Schleife ab, in der es angegeben ist; für *n* wird also 1 eingesetzt.

case-Anweisung - abfragen und verzweigen

Mit der *case*-Anweisung können Sie eine Eingabe mit vorgegebenen Mustern vergleichen und entsprechend dem Ergebnis zu Kommandos verzweigen.

```
case␣name␣in
muster[|muster]...)␣[kommandofolge];;
.
.
.
muster[|muster]...)␣[kommandofolge]
esac
```

name
: beliebige Zeichenkette, normalerweise ein Shell-Parameter. Die Shell vergleicht *name* mit den angegebenen Mustern.

muster
: Zeichenkette, mit der *name* verglichen werden soll. Diese Zeichenkette kann so angegeben werden wie die Argumente, die durch passende Dateinamen ersetzt werden sollen (siehe *ARGUMENTE DURCH PASSENDE DATEINAMEN ERSETZEN*). Allerdings werden die Sonderzeichen *,? und [...] in *muster* durch beliebige Zeichen ersetzt:

Sonderzeichen	Ersetzung
*	wird ersetzt durch kein, ein oder mehrere beliebige Zeichen; auch durch Punkt . oder Schrägstrich /
?	wird ersetzt durch genau ein beliebiges Zeichen; auch durch Punkt . oder Schrägstrich /
[s]	wird ersetzt durch genau ein Zeichen, das in der Zeichenkette *s* enthalten ist. *s* darf beliebige einfache Zeichen (auch Punkt . und Schrägstrich /) enthalten.
[!s]	wird ersetzt durch genau ein Zeichen, das in der Zeichenkette *s* nicht enthalten ist.
[c1-c2]	wird ersetzt durch genau ein Zeichen aus dem innerhalb der eckigen Klammern angegebenen Zeichenbereich; *c1* und *c2* gehören zum Bereich. *c1* und *c2* können beliebige einfache Zeichen (auch Punkt . und Schrägstrich /) sein.
[!c1-c2]	wird ersetzt durch genau ein Zeichen, das nicht zu dem innerhalb der eckigen Klammern angegebenen Zeichenbereich gehört.

Wenn *name* zu diesem Muster paßt, führt die Shell die danach angegebene Kommandofolge aus. Nach der Ausführung dieser Kommandofolge ist die *case*-Anweisung beendet.

Wenn *name* nicht zu diesem Muster paßt, wird das nächste Muster geprüft.

Wenn bei mehreren Mustern die gleiche Kommandofolge ausgeführt werden soll, können Sie diese Muster vor die entsprechende Kommandofolge setzen, jeweils getrennt durch einen senkrechten Strich |.

Wenn Sie für *muster* das Zeichen * angeben, werden die nachfolgenden Muster nicht mehr überprüft. Es wird also nur die Kommandofolge ausgeführt, die zum Muster * gehört.

kommandofolge
Folge von Kommandos, die ausgeführt werden sollen, wenn *name* zum entsprechenden Muster paßt.

Jede Kommandofolge bis auf die letzte muß mit zwei Strichpunkten ;; abgeschlossen werden.

Wenn Sie kein Kommando angeben wollen, müssen Sie, abgesehen von der letzten Kommandofolge, zumindest die beiden Strichpunkte ;; angeben.

Das Schlüsselwort *esac* schließt die *case*-Anweisung ab.

continue - Schleife abbrechen

Die Ablaufanweisung *continue* beendet den aktuellen Durchlauf der *for-*, *until-* oder *while*-Schleife, in der es angegeben ist. Die Abarbeitung der entsprechenden Schleife wird mit dem nächsten Durchlauf fortgesetzt. Eine Shell-Prozedur wird ebenfalls mit dem nächsten Durchlauf dieser Schleife fortgesetzt.

Schleifen können beliebig tief geschachtelt sein. Wenn Sie mit *continue* den Durchlauf mehrerer Schleifen abbrechen wollen, die den Aufruf von *continue* umgeben, können Sie dies durch eine entsprechende Zahl beim Aufruf angeben.

```
continue [ n]
```

n

Nummer der *for-*, *until-* oder *while*-Schleife, die den Aufruf von *continue* umgibt. Mit dem Durchlauf dieser Schleife soll die Shell-Prozedur fortgesetzt werden.

n nicht angegeben:
continue setzt die Shell-Prozedur mit dem nächsten Durchlauf der *for-*, *until-* oder *while*-Schleife fort, in der es angegeben ist. Für *n* wird also 1 eingesetzt.

for-Anweisung - Liste in Schleifen abarbeiten

Mit der *for*-Anweisung können Sie Kommandos mehrmals wiederholen. Für jedes Argument in der angegebenen Liste wird die *for*-Schleife einmal durchlaufen.

```
for␣name[␣in␣liste]
do␣kommandofolge
done
```

name
: Name einer Shell-Variablen. Dieser Variablen weist die Shell als Wert nacheinander die Argumente aus *liste* zu. Nach jeder Zuweisung wird die Kommandofolge ausgeführt. Der Schlüsselwort-Parameter *$name* spricht den jeweils gültigen Wert der Variablen *name* an.

liste
: Liste von Argumenten; Argument-Trennzeichen sind die Zeichen, die der Variablen IFS zugewiesen sind. Jedes Argument wird der Reihe nach der Variablen *name* als Wert zugewiesen und *kommandofolge* einmal ausgeführt. Die *for*-Anweisung ist beendet, wenn *liste* kein weiteres Argument enthält.

  Enthält ein Argument eines der Sonderzeichen *, ? oder [...], so wird es durch passende Dateinamen ersetzt (siehe *ARGUMENTE DURCH PASSENDE DATEINAMEN ERSETZEN*).

  liste nicht angegeben:
  Die Angabe *in* entfällt ebenfalls. Die Shell setzt "$@" ein; die *for*-Schleife wird also für jedes Aufrufargument einmal durchlaufen, das beim Aufruf der entsprechenden Shell-Prozedur angegeben bzw. mit *set* in der aktuellen Shell gesetzt ist.

kommandofolge
: Folge der Kommandos, die entsprechend oft ausgeführt werden soll.

Das Schlüsselwort *done* schließt die *for*-Anweisung ab.

if-Anweisung - Bedingungen abfragen und Kommandos ausführen

Mit der *if*-Anweisung können Sie Bedingungen abfragen und je nach Ergebnis verschiedene Kommandos ausführen.

Dazu führt die Shell die Kommandofolge aus, die Sie direkt nach *if* angegeben haben:

- Die Bedingung ist wahr, wenn das letzte Kommando aus dieser Kommandofolge den Ende-Status 0 zurückgibt.
- Die Bedingung ist falsch, wenn das letzte Kommando aus dieser Kommandofolge einen Ende-Status ≠0 zurückgibt.

```
if␣kommandofolge1
then␣kommandofolge2
[elif␣kommandofolge3
then␣kommandofolge4] ...
[else␣kommandofolge5]
fi
```

kommandofolge1
: Folge von Kommandos, die die Shell ausführt, um die Bedingung abzufragen.

  Wenn die Bedingung wahr ist, verzweigt die Shell zu *then*.
  Wenn die Bedingung falsch ist, verzweigt die Shell zum nächsten *elif* bzw. *else*.

kommandofolge2
: Folge von Kommandos, die die Shell ausführen soll, wenn die Bedingung nach *if* wahr ist. Nach der Ausführung dieser Kommandos ist die *if*-Anweisung beendet.

kommandofolge3
: Folge von Kommandos, die die Shell ausführt, um die Bedingung abzufragen. Die Shell führt *kommandofolge3* nur aus, wenn die Bedingung nach *if* falsch ist.

  Wenn die Bedingung wahr ist, verzweigt die Shell zum nächsten *then*.
  Wenn die Bedingung falsch ist, verzweigt die Shell zum nächsten *elif* bzw. *else*.

  Der *elif*-Zweig ist optional. Wenn Sie *elif* nicht angeben, entfällt auch das auf *elif* folgende *then*.

kommandofolge4
: Folge von Kommandos, die die Shell ausführen soll, wenn die Bedingung nach *if* falsch und die Bedingung nach *elif* wahr ist. Nach der Ausführung dieser Kommandos ist die *if*-Anweisung beendet.

kommandofolge5
Folge von Kommandos, die die Shell ausführen soll, wenn die Bedingungen nach *if* und nach *elif* falsch waren. Nach der Ausführung dieser Kommandos ist die *if*-Anweisung beendet.

Wenn der *else*-Zweig fehlt, führt die Shell Kommandos nur aus, wenn eine Bedingung wahr ist.

Das Schlüsselwort *fi* schließt die *if*-Anweisung ab.

Wenn die Shell weder die Kommandos aus einem *then*-Zweig noch die Kommandos aus dem *else*-Zweig ausgeführt hat, gibt die *if*-Anweisung als Ende-Status den Wert 0 zurück. In diesem Fall war also keine Bedingung erfüllt.

until-Anweisung - Schleife mit Abbruch-Bedingung

In der Shell gibt es zwei Schleifen mit Abbruch-Bedingung, die *until*- und die *while*-Schleife.

Die *until*-Anweisung prüft, ob die angegebene Bedingung falsch ist. Sobald die Bedingung wahr wird, ist die *until*-Anweisung beendet.

```
until␣kommandofolge1
do␣kommandofolge2
done
```

kommandofolge1
Folge von Kommandos, die die Shell ausführt, um die Bedingung abzufragen.

Wenn die Bedingung wahr ist, ist die *until*-Anweisung beendet. Die Kommandos aus *kommandofolge2* werden nicht ausgeführt.

Wenn die Bedingung falsch ist, führt die Shell die Kommandos aus *kommandofolge2* aus. Anschließend führt die Shell erneut die Kommandos aus *kommandofolge1* aus, um die Bedingung abzufragen.

kommandofolge2
Folge von Kommandos, die die Shell ausführt, solange die Bedingung falsch ist.

Das Schlüsselwort *done* schließt die *until*-Anweisung ab.

Wenn die Shell die Kommandos nach dem Schlüsselwort *do* kein einziges Mal ausgeführt hat, gibt die *until*-Anweisung als Ende-Status den Wert 0 zurück. In diesem Fall war also die Bedingung bereits bei der ersten Prüfung wahr.

while-Anweisung - Schleife mit Abbruch-Bedingung

In der Shell gibt es zwei Schleifen mit Abbruch-Bedingung, die *while*- und die *until*-Schleife.

Die *while*-Anweisung prüft, ob die angegebene Bedingung wahr ist. Sobald die Bedingung falsch wird, ist die *while*-Anweisung beendet.

```
while␣kommandofolge1
do␣kommandofolge2
done
```

kommandofolge1
: Folge von Kommandos, die die Shell ausführt, um die Bedingung abzufragen.

  Wenn die Bedingung wahr ist, führt die Shell die Kommandos aus *kommandofolge2* aus. Anschließend führt die Shell erneut die Kommandos aus *kommandofolge1* aus, um die Bedingung abzufragen.

  Wenn die Bedingung falsch ist, ist die *while*-Anweisung beendet. Die Kommandos aus *kommandofolge2* werden nicht ausgeführt.

kommandofolge2
: Folge von Kommandos, die die Shell ausführt, solange die Bedingung wahr ist.

Das Schlüsselwort *done* schließt die *while*-Anweisung ab.

Wenn die Shell die Kommandos nach dem Schlüsselwort *do* kein einziges Mal ausgeführt hat, gibt die *while*-Anweisung als Ende-Status den Wert 0 zurück. In diesem Fall war also die Bedingung bereits bei der ersten Prüfung falsch.

shift
Die Werte der Stellungsparameter nach links verschieben

Das in die Bourne-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.

```
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.

FEHLERMELDUNG

```
prozedur: cannot shift
```

Diese Fehlermeldung erhalten Sie, wenn $# gleich 0 ist. Dem Stellungsparameter $1 konnte also kein Wert zugewiesen werden.

BEISPIELE

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.

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, *sh*

shl Schichtenverwaltung für Shells (shell layer manager)

shl ermöglicht es Ihnen, an einer Datensichtstation mit mehreren Shells gleichzeitig zu arbeiten.
shl ist ein Verwaltungsprogramm, mit dem Sie das Verhalten der einzelnen Schichten (layers) oder Shells kontrollieren und beeinflussen können.
Welche der Shells (*sh*, *jsh*, *ksh* oder *csh*) Sie zum Arbeiten in den einzelnen Schichten verwenden, ist Ihnen überlassen.
shl benötigt einen Eintrag in */dev/sxt*. Ist dieser nicht vorhanden, bricht *shl* mit Fehlermeldung ab.

Arbeitsweise

shl ist ein Verwaltungsprogramm, mit dem Sie die einzelnen Schichten mit ihren Shells aufbauen und kontrollieren können. *shl* schaltet ein sogenanntes virtuelles Terminal oder Gerät (*/dev/sxt*) zwischen die Shell einer Schicht und Ihre Datensichtstation. Das virtuelle Terminal können Sie, wie die Leitung einer Datensichtstation, mit *stty* oder *ioctl* beeinflussen.

Unmittelbar nach dem Aufruf erwartet *shl* Kommandos zum Aufbau wenigstens einer Schicht. Dazu meldet sich die Verwaltung von *shl* - und später zur besseren Unterscheidung - mit einem eigenen Bereitzeichen `>>>`. Wurde eine Schicht aufgebaut, dann zieht sich die Verwaltung zurück und übergibt die Kontrolle an die Shell der Schicht, die sich mit ihrem von *shl* modifiziertem Bereitzeichen *PS1* (siehe unten) meldet.
Diese Schicht wird die aktuelle Schicht genannt, die von *shl* aufgerufene Shell ist der Prozeßgruppenchef, ähnlich wie bei einer *login*-Shell. Jede Schicht hat somit ihre eigene Prozeßgruppennummer. Die in der Schicht aufgerufene Shell wird durch den Wert der Variablen *SHELL* bestimmt. Sie können mit *shl* bis zu sieben Schichten gleichzeitig benutzen.

Die Eingabe von Ihrer Datenstation wird entweder von der Verwaltung oder von der aktuellen Schicht gelesen. Will ein Prozeß von einer anderen Schicht von der Standard-Eingabe - also von der Datensichtstation - lesen, dann wird er blockiert, bis seine Schicht zur aktuellen gemacht wird. Mit dem *swtch*-Zeichen (siehe *stty*)
- auf CTRL z gesetzt, falls noch nicht definiert - können Sie von der aktuellen Schicht jederzeit in die Verwaltung umschalten.

Alle Schichten können ihre Standard-Ausgabe auf die Datensichtstation schreiben, falls Sie dies nicht durch ein Kommando für einzelne Schichten unterbinden.
Dies erreichen Sie durch ein Kommando an die Verwaltung oder durch Aufruf des *stty*-Kommandos mit der Option *loblk* (layer output blocking). Durch *-loblk* oder Aktivieren der Schicht können Sie diese Sperre wieder aufheben.

Vorsicht

Verwenden Sie eine Shell mit Auftragssteuerung, dann müssen Sie das *switch*-Zeichen (siehe *stty*) neu definieren, damit das *suspend*-Zeichen nicht deaktiviert wird.

Kommandos

Die folgenden Kommandos können Sie nach dem Bereitzeichen der Verwaltung > > > zur Steuerung und Information eingeben. Sie können jede eindeutige Abkürzung (Präfix) verwenden. Die Namen *(1)* bis *(7)* können durch die Ziffer abgekürzt werden.

Ein *name* besteht aus einer Folge von Zeichen, die durch ein Leer-, Tabulator- oder Neue-Zeile-Zeichen begrenzt werden. Nur die ersten acht Zeichen sind signifikant. Die Namen *(1)* bis *(7)* dürfen nicht zum Erzeugen einer Schicht (siehe unten *create*) verwendet werden. diese Namen werden als Standardnamen von *shl* eingesetzt, wenn bei *create* kein Argument angegeben wurde.

create[␣name]
Die Schicht mit der Referenz *name* wird erzeugt und zur aktuellen Schicht gemacht. Haben Sie das Argument *name* nicht angegeben, wird von *shl* eine Name vergeben.
Dieser wird aus der letzten Ziffer des Namens des virtuellen Geräts, an das die Schicht gebunden ist, gebildet. Die Ziffer wird zur Darstellung in runden Klammern eingeschlossen: z.B. *(3)* bei */dev/sxt003*.
Dem Bereitzeichen *PS1* der Shell wird als Wert der Name der Schicht, gefolgt von einem Leerzeichen, zugewiesen. Sie können maximal sieben Schichten erzeugen.

block␣name[␣name]...
Falls die Schicht *name* nicht die aktuelle Schicht ist, wird die Standard-Ausgabe der Prozesse der Schicht blockiert. Dieses Kommando entspricht der Ausführung von *stty loblk* (*l*ayer *o*utput *bl*ocking) in der Schicht.

delete␣name[␣name]...
Jede Schicht, deren *name* als Argument angegeben ist, wird beendet. Allen Prozessen der Prozeßgruppe der angegebenen Schichten wird das Signal *SIGHUP* zur Beendigung geschickt (siehe *signal()* [14]).

help oder **?**
Die Kommandosyntax von *shl* wird auf die Standard-Ausgabe geschrieben.

layers[␣**-l**][␣name]...
Von jeder durch das Argument *name* benannten Schicht wird der Name, gefolgt von der Prozeßgruppennummer, auf die Standard-Ausgabe geschrieben.
Durch die Option *-l* wird eine ausführlichere Ausgabe ähnlich der des *ps* Kommandos erzeugt. Ohne Argument *name* schreibt dieses Kommando die Information über alle Schichten auf die Standard-Ausgabe.

resume[␣name]
Die Schicht *name* wird zur aktuellen Schicht gemacht. Ohne Argument *name* wird die letzte (noch existierende) aktuelle Schicht aktiviert.

toggle
Aktiviert die Schicht, die vor der letzten aktuellen Schicht aktuell war.

unblock␣name[␣name]...
Falls die Schicht *name* nicht die aktuelle Schicht ist, wird die Standard-Ausgabe der Prozesse der Schicht nicht blockiert. Dieses Kommando entspricht der Ausführung von *stty -loblk* (*l*ayer *o*utput *bl*ocking) in der Schicht.

quit
Dieses Kommando beendet *shl*. Den Prozessen aller Schichten wird das Signal *SIGHUP* zur Beendigung geschickt.

name
Die Schicht *name* wird zur aktuellen Schicht gemacht.

DATEI

/dev/sxt/??[0-7]
Virtuelles Terminal oder Gerät.

UMGEBUNGSVARIABLE

SHELL
enthält den aufgerufenen Kommandointerpreter.

BEISPIELE

Typische Arbeitsfolge mit der *shl*:

```
$ shl
create eins
eins
...
CTRL z
create
(3)
...
CTRL z
layers -l
----
----
----
(3)
CTRL z
toggle
eins
...
CTRL z
quit
$
```

SIEHE AUCH

sh, stty
ioctl(), signal() [14]
sxt [5]

sinfilt
Dateien mit sprachabhängigen Sonderzeichen für SINIX lesbar machen

sinfilt wandelt MS-DOS-Dateien mit sprachabhängigen Sonderzeichen (wie z.B. ä, ö, ü, ß) bzw. mit Daten im IBM-Zeichensatz derart um, daß sie für SINIX lesbar und bearbeitbar werden (siehe *Tabellen und Verzeichnisse, Zeichensatz ISO 646*). *sinfilt* schreibt standardmäßig auf die Standard-Ausgabe.

```
sinfilt␣-sprache[␣-o␣ausdatei][␣datei]...
```

-sprache
Sie müssen eine der folgenden Sprachen wählen:

d –	deutsch	**g** –	britisch
a –	dänisch	**i** –	italienisch
b –	belgisch	**n** –	norwegisch
e –	spanisch	**s** –	schwedisch
f –	französisch	**z** –	schweizerisch

x – Mit dieser Option setzen Sie die IBM-Zeichen der umzuwandelnden Datei in die Zeichen des ISO-Zeichensatzes 8859-1 um.

-o ausdatei
Name der Ausgabedatei

datei
Name der umzuwandelnden Datei

datei nicht angegeben:
dosfilt liest von der Standard-Eingabe

Um SINIX-Dateien in MS-DOS-Dateien umzuwandeln, benutzen Sie das Kommando *dosfilt*.

Die Kommandos *sinfilt -x* und *dosfilt -x* setzen die angegebenen Dateien vom IBM-Zeichensatz in den Zeichensatz nach ISO 8859-1 um (*sinfilt*) und umgekehrt (*dosfilt*). Die beiden Zeichensätze besitzen allerdings nicht den gleichen Zeichenvorrat. Daher ist folgendes zu beachten:

- 52 Zeichen (inklusive aller ASCII-Zeichen und deutschen Umlaute) können direkt 1 : 1 übertragen werden.
- Sämtliche Semigraphikzeichen, viele mathematische und griechische Zeichen des IBM-Zeichensatzes (insgesamt 63 Zeichen mit Hexwerten größer als 7f) können nicht direkt in identische Zeichen des ISO-Zeichensatzes umgewandelt werden. Sie werden durch ähnliche Zeichen ersetzt.
- Einige nationale Sonderzeichen des ISO-Zeichensatzes 8859-1 (insgesamt 44 Zeichen) können nicht direkt in identische Zeichen des IBM-Zeichensatzes umgewandelt werden. Sie werden ebenfalls durch ähnliche Zeichen ersetzt.

BEISPIEL

Umwandeln der MS-DOS-Datei *TERMINE* im IBM-Zeichensatz in die für SINIX lesbare Datei *termine*.

```
$ sinfilt -x -o termine TERMINE
```

SIEHE AUCH

dosfilt

Tabellen und Verzeichnisse, Zeichensatz ISO 646

sleep Prozesse zeitweise stillegen

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.

```
sleep␣zeit
```

zeit
Zeit in Sekunden, um die die Ausführung des Prozesses verzögert werden soll. Sie müssen für *zeit* eine nicht-negative Dezimalzahl angeben.

FEHLERMELDUNG

```
sleep: bad character in argument
```

Sie haben für *zeit* eine negative Dezimalzahl oder einen nichtnumerischen Ausdruck angegeben.

BEISPIELE

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
   ```

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()* [14]

sort
Dateien sortieren und/oder mischen

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-Feldtennzeichen zum folgenden Feld. Führende Leerzeichen in einer Zeile gehören somit standardmäßig zum ersten Feld.

```
sort[␣option]...[␣+pos[␣-pos]]...[␣datei]...
```

Keine Option angegeben
: *sort* sortiert die Eingabezeilen lexikographisch, wobei als Einzelzeichen jeweils ein Byte verwendet wird. Für die Bytes gilt die Sortierreihenfolge der Maschine.

option

Optionen, die das Verhalten von sort ändern

Mehrere dieser Optionen müssen Sie 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 Performance 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.

-zrecsz
Mit der Option *-z* stellen Sie für die Mischphase Puffer von ausreichender Größe bereit. Dies ist nur nowendig, 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* hat jedoch nur eine Wirkung, 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, 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. Das 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.

-tx

sort behandelt das Zeichen, das Sie für *x* angeben, als Feldtrennzeichen. Im Unterschied zu den Standard-Feldtrennzeichen gehört *x* selbst nicht zu einem Feld. Es kann aber Teil eines Sortierfeldes sein, z.B. wenn das Sortierfeld vom ersten bis dritten jeweils durch *x* getrennten Feld reicht. Jedes Trennzeichen *x* ist signifikant, d.h. *xx* begrenzt ein leeres Feld.

-tx 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.

+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.

DATEI

/var/tmp/stm???
temporäre Dateien

BEISPIELE

1. Der Inhalt von *eingabe_datei* soll nach dem zweiten Feld sortiert werden.

```
$ sort +1 -2 eingabe_datei
```

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
```

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
```

4. Die Datei */etc/passwd* soll nach den Benutzernummern (3.Feld) sortiert ausgegeben werden.

```
$ sort -t: +2n -3 /etc/passwd
```

5. 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

spell
Rechtschreibfehler suchen

spell vergleicht die Wörter in einer Datei mit einer Wortliste und eignet sich so zur Suche nach Rechtschreibfehlern. Wörter, die weder in der standardmäßig englischen Wortliste stehen, noch nach Grammatikregeln der englischen Sprache ableitbar sind (z.B. durch Präfixe, Suffixe oder Deklination), werden auf die Standard-Ausgabe geschrieben. Sie können die englische Wortliste mit einer eigenen, z.B. deutschen Wortliste ergänzen, nach der die Wörter einer Datei überprüft werden sollen.

```
spell[␣option]...[␣+lokal_datei][␣datei]...
```

option

-b

(b - british) Es wird überprüft, ob die Wörter aus *datei* der korrekten englischen im Unterschied zur amerikanischen Schreibweise entsprechen. Als korrekt angesehen werden z.B. englische Schreibweisen wie *centre*, *colour*, *programme*, *speciality*, *travelled* usw. sowie die Endung *-ise* in Wörtern wie *standardise*.

-b nicht angegeben:
Es wird überprüft, ob die Wörter aus *datei* der korrekten amerikanischen Schreibweise entsprechen.

-l

Normalerweise folgt *spell* (wie *deroff*)
Ketten einzufügender Dateien (über die *troff* Kommandos zum Texteinzug *.so* und *.nx*), falls die Pfadnamen dieser Dateien nicht mit */usr/lib* beginnen. Durch die Option *-l* wird *spell* gezwungen, den Ketten aller einzufügenden Dateien zu folgen.

-v

Alle Wörter aus *datei* werden ausgegeben, die nicht buchstabengetreu mit Wörtern aus der Wortliste übereinstimmen. Die Wortliste steht in der Datei */usr/ucblib/dict/words*.

Wörter, die möglicherweise eine abgeleitete Form von Wörtern aus der Wortliste darstellen, werden wie folgt gekennzeichnet:
+präfix wort
+suffix wort

-x

Für jedes Wort aus *datei* wird mit = der mögliche Wortstamm ausgegeben.

+lokal_datei
spell gibt alle die Wörter von *datei* aus, die nicht in *lokal_datei* stehen. Die Wörter in *lokal_datei* sieht *spell* als korrekt geschrieben an. *lokal_datei* ist eine von Ihnen erstellte Datei, in der eine nach ASCII sortierte Wortliste enthalten ist, wobei in jeder Zeile ein Wort stehen muß. Die in dieser Datei enthaltenen Wörter sieht *spell* als korrekt geschrieben an. Für *lokal_datei* müssen Sie das Leserecht (siehe *chmod*) haben.

+lokaldatei nicht angegeben:
spell sieht die Wörter in der Datei */usr/.dict/words* als korrekt geschrieben an.

datei
Name der Datei, deren Inhalt nach Rechtschreibfehlern überprüft werden soll. Für *datei* müssen Sie das Leserecht (siehe *chmod*) haben.
Wenn Sie für *datei* das Zeichen - angeben, liest *spell* von der Standard-Eingabe.

datei nicht angegeben:
spell liest von der Standard-Eingabe.

troff, *tbl* und *eqn* Anweisungen in der Eingabe werden von *spell* meistens ignoriert.

DATEI

/usr/lib/spell/spellprog
Ausführbares Programm, das *spell* benutzt.

/usr/share/lib/spell/hlist[ab]
Amerikanische und britische Wortliste als Hash-Datei. Diese Rechtschreibeverzeichnisse basieren auf vielen Quellen. Obwohl sie zufälliger sind als ein gewöhnliches Wörterbuch, sind sie doch effektiver bezüglich spezieller Namen und weitverbreiteter technischer Wörter. Spezielles Vokabular aus Biologie, Medizin und Chemie ist nur in geringem Umfang vorhanden.

/usr/share/lib/spell/hstop
Stopliste als Hash-Datei. Sie filtert falsch geschriebene Wörter (z.B. thier=thy-y+ier) aus, die sonst ungemeldet passieren würden.

/var/adm/spellhist
enthält eine Protokolldatei (history file), in der eine Kopie aller Ausgaben gesammelt wird.

Sie können diese Dateien erweitern und pflegen. Dafür stehen die Kommandos *spellin*, *hashmake* und *hashcheck* zur Verfügung (siehe dort).

UMGEBUNGSVARIABLEN

Hilfsdateien wie Wortlisten, Stoplisten und Protokolldateien können über Umgebungsvariablen angegeben werden. Die Voreinstellungen sind wie folgt:

D_SPELL=/usr/share/lib/spell/hlist[ab]
enthält die Wortliste als Hash-Datei.

S_SPELL=/usr/share/lib/spell/hstop
enthält die Stop Liste als Hash-Datei.

H_SPELL=/var/adm/spellhist
enthält die Protokolldatei (history file).

BEISPIEL

Die Datei *farben* soll anhand der in *liste* enthaltenen Wörter nach Rechtschreibfehlern überprüft werden:

```
$ cat farben
bleu gelb roth grun lila schwarz weis orrange

$ cat liste
blau
gelb
grün
lila
orange
rot
schwarz
weiß

$ spell -v +liste farben
bleu
grun
orrange
roth
weis
```

Die Ausgabe zeigt alle Wörter aus *farben*, die nicht in *liste* enthalten sind.

SIEHE AUCH

deroff, *hashmake*, *hashcheck*, *spellin*, *sed*, *sort*, *tee*

eqn, *tbl*, *troff* [1-2]

spellin
Komprimierte Wortliste erzeugen

spellin gehört, zusammen mit den Programmen *hashmake* und *hashcheck*, zu einer Gruppe ergänzender Kommandos für das Rechtschreibeprüfprogramm *spell*. Das Kommando *spell* benutzt Vergleichslisten für die Rechtschreibeüberprüfung. Mit den aufgeführten Kommandos können neue Rechtschreibelisten erstellt werden, bereits bestehende können gepflegt und erweitert werden.

spellin erzeugt aus den angegebenen codierten Worten komprimierte Wortlisten, die von dem Kommando *spell* verwendet werden können. Bereits existierende komprimierte Wortlisten können erweitert werden.

```
/usr/lib/spell/spellin anzahl
```

spellin liest *anzahl* Hash-Codes, wie sie von *hashmake* erzeugt wurden, von der Standardeingabe und erstellt daraus eine komprimierte Liste. Diese wird auf die Standardausgabe geschrieben.

spellin wird verwendet, um neue Rechtschreibelisten zu erzeugen, oder um bestehende Rechtschreibelisten zu erweitern.

FEHLERMELDUNGEN

```
spellin: arg count
```

Die Angabe von *anzahl* fehlt.

SIEHE AUCH

hashmake, *hashcheck*, *spell*

split
Datei auf mehrere Dateien verteilen

split teilt Dateien in kleinere Abschnitte auf. Die Abschnitte schreibt *split* in einzelne Ausgabedateien. Die ursprüngliche Datei bleibt erhalten.

```
split[ -n][ datei[ name]]
```

-n
split teilt die Eingabedatei in Abschnitte zu je *n* Zeilen auf.

-n nicht angegeben:
split teilt die Eingabedatei in Abschnitte zu je 1000 Zeilen auf.

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 in lexikographischer Reihenfolge.

name darf maximal 12 Zeichen lang sein.
split legt höchstens 676 Ausgabedateien an.

Wenn Sie einen Wert für *name* angeben, dann müssen Sie auch einen Wert für *datei* angeben.

name nicht angegeben:
Die Ausgabedateien heißen *xaa*, *xab* usw.

BEISPIELE

1. Der Inhalt der Datei *beispiel* soll zu je 20 Zeilen auf verschiedene Dateien verteilt werden:

```
$ split -20 beispiel
$ ls
beispiel
xaa
xab
xac
xad
```

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 -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

bfs, *csplit*
statvfs() [14]

srchtxt
Inhalt von Meldungsdateien anzeigen, nach Zeichenketten suchen

srchtxt zeigt den Inhalt von Meldungsdateien an, oder führt eine Suche nach speziellen Zeichenketten in den Meldungsdateien durch.

Das *srchtxt* Dienstprogramm zeigt alle Meldungstexte einer Meldungsdatenbank an, oder sucht nach einer bestimmten Zeichenkette in den Meldungsdatenbanken (siehe auch *mkmsgs*). Die Datenbanken sind im Dateiverzeichnis /usr/lib/locale/*sprachraum*/*LC_MESSAGES* abgelegt.
Über die Option *-m* können aber auch andere Suchpfade spezifiziert werden.
Das Dateiverzeichnis *sprachraum* kann als der Name der Sprache angesehen werden, in der die Meldungstexte verfaßt sind.

```
srchtxt[␣option]...[␣text]
```

Keine Option angegeben
Der Sprachraum, auf den zugegriffen wird, wird über die Umgebungsvariable *LC_MESSAGES* bestimmt. Ist diese nicht gesetzt, ist der Wert der Umgebungsvariablen *LANG* ausschlaggebend.

option

-s
(s - suppress) Die Angabe der Folgenummer jeder einzelnen Meldung unterbleibt.

-l␣sprachraum
(l - locale) Zugriff auf Dateien im Dateiverzeichnis /usr/lib/locale/*sprachraum*/*LC_MESSAGES*.

-l nicht angegeben:
Der zu durchsuchende Sprachraum wird über die Umgebungsvariable *LC_MESSAGES* bestimmt. Ist diese nicht gesetzt, ist der Wert der Umgebungsvariablen *LANG* ausschlaggebend.

In Kombination mit der *-m messagedatei* Option wird *sprachraum* ignoriert, falls *messagedatei* einen Schrägstrich / enthält, also einen Pfadnamen darstellt.

-m␣messagedatei
Die in *messagedatei* spezifizierten Dateien werden angezeigt oder durchsucht. Es können mehrere Dateien, durch Kommata voneinander getrennt, angegeben werden.

Enthält *messagedatei* einen Schrägstrich /, wird angenommen, daß es sich um einen Pfadnamen handelt. Ansonsten wird *messagedatei* im oben beschriebenen Dateiverzeichnis gesucht.

text
Die Zeichenkette *text* wird gesucht. Jedes Auftreten von *text* wird angezeigt.

text kann auch als regulärer Ausdruck (siehe *ed*) angegeben werden.

text nicht angegeben:
Alle Zeichenketten der durchsuchten Dateien werden angezeigt.

Ausgabeformat

Den von *srchtxt* angezeigten Meldungstexten ist die Folgenummer innerhalb der Datei vorangestellt. Sie ist in spitze Klammern eingeschlossen:
<meldungsdatei:meldungsnummer>. *meldungsdatei* ist die Datei, die den nachfolgenden Meldungstext enthält, *meldungsnummer* ist die Folgenummer des angezeigten Meldungstextes innnerhalb der *meldungsdatei*.

Die übrige Anzeige erfolgt im Format, wie es bei *gettxt* beschrieben ist.

FEHLERMELDUNGEN

Die von *srchtxt* ausgegebenen Fehlermeldungen sind selbsterklärend. Sie weisen auf Fehler beim Aufruf des Kommandos hin, oder melden Fehler, die beim Suchen nach bestimmten Meldungsdateien aufgetreten sind.

UMGEBUNGSVARIABLEN

LC_MESSAGES
Enthält den zu durchsuchenden Sprachraum.

LANG
Enthält den zu durchsuchenden Sprachraum (alternativ zu *LC_MESSAGES*).

SIEHE AUCH

ed, *exstr*, *gettxt*, *mkmsgs*
gettxt(), *setlocale()* [14]

strchg
Konfiguration eines Datenstroms ändern (change stream configuration)

strchg ändert die Konfiguration des Datenstroms (stream), der mit Ihrer Standard-Eingabe verbunden ist. *strchg* aktiviert oder deaktiviert Module für diesen Datenstrom. Nur der Systemverwalter oder Eigentümer eines Geräts darf die Konfiguration des zugehörigen Datenstroms ändern. Wenn Sie weder Eigentümer des Datenstroms sind noch Systemverwalter-Berechtigung haben, schlägt das Kommando *strchg* fehl. Dies gilt auch für den Fall, daß Sie keine Lese-Berechtigung für den Datenstrom und keine Systemverwalter-Berechtigung haben.
Wenn Sie Module für einen Datenstrom in der falschen Reihenfolge aktivieren, kann das zur Folge haben, daß der Datenstrom nicht wie gewünscht funktioniert. Wenn Sie bei einer Datensichtstation das Modul für die Übertragungsprozedur (line discipline) nicht an der richtigen Stelle aktivieren, kann es passieren, daß die Datensichtstation überhaupt nicht mehr auf Eingaben reagiert.

```
strchg␣-h␣modul1[,␣modul2...]                    Format 1
strchg␣-p[␣option]                               Format 2
strchg␣-f␣datei                                  Format 3
```

Format 1: Module für einen Datenstrom aktivieren

strchg␣-h␣modul1[,␣modul2...]

Als Argumente geben Sie den bzw. die Namen eines oder mehrerer aktivierbarer Module für Datenströme an. Die Module werden in der angegebenen Reihenfolge aktiviert. *modul1* wird also als erstes aktiviert, dann *modul2* usw.

Format 2: Module für einen Datenstrom deaktivieren

strchg␣-p[␣option]

Wenn Sie außer *-p* keine Option angeben, deaktiviert *strchg* das oberste Modul des Datenstroms.

option

-a
Wenn Sie *-a* zusammen mit *-p* angeben, werden für den Datenstrom alle Module über dem obersten Treiber deaktiviert. Sie können *-a* nur zusammen mit *-p* angeben und nicht zusammen mit *-u*.

-u␣modul
Für den Datenstrom werden alle Module über *modul* deaktiviert. *modul* selbst wird

nicht deaktiviert. Sie können *-u* nur zusammen mit *-p* angeben und nicht zusammen mit *-a*.

Format 3: Konfiguration eines Datenstroms festlegen

strchg␣-f␣datei

datei enthält eine Liste von Modulen, die die gewünschte Konfiguration des Datenstroms darstellt. Dabei muß jedes Modul auf einer eigenen Zeile stehen. Die erste Zeile enthält den Namen des obersten Moduls und die letzte Zeile den Namen des untersten Moduls, also des Moduls, das dem Treiber am nächsten sein soll. *strchg* stellt die augenblickliche Konfiguration des Datenstroms fest und aktiviert bzw. deaktiviert dann die notwendigen Module in der richtigen Reihenfolge, bis die gewünschte Konfiguration erreicht ist.

BEISPIELE

1. Mit dem folgenden Kommando aktivieren Sie das Modul *ldterm* für den Datenstrom, der mit Ihrer Standard-Eingabe verbunden ist:

```
strchg -h ldterm
```

2. Mit dem folgenden Kommando deaktivieren Sie das oberste Modul für den Datenstrom, der mit */dev/term/24* verbunden ist. Dazu müssen Sie Eigentümer dieser Gerätedatei sein oder Systemverwalter-Berechtigung haben:

```
strchg -p </dev/term/24
```

3. Die Datei *conf* enthalte die folgenden drei Zeilen:

```
compat
ldterm
ptem
```

Dann können Sie mit dem Kommando

```
strchg -f conf
```

den Datenstrom, der mit Ihrer Standard-Eingabe verbunden ist, so konfigurieren, daß das Modul *ptem* direkt über dem Treiber aktiviert wird. Darüber wird das Modul *ldterm* aktiviert und das Modul *compat* wird als oberstes Modul aktiviert, also am nächsten zum Kopfmodul des Datenstroms.

SIEHE AUCH

strconf
streamio [13]

strconf
Konfiguration eines Datenstroms abfragen (query stream configuration)

Mit *strconf* können Sie die Konfiguration des Datenstroms (stream) abfragen, der mit der Standard-Eingabe verbunden ist.

```
strconf [␣-t|␣-m␣modul]
```

Keine Option angegeben
: *strconf* gibt eine Liste aller Module aus, die für den Datenstrom aktiviert sind, sowie den obersten Treiber. In jeder Zeile wird ein Name ausgegeben. Der erste Name gibt dabei das oberste Modul für den Datenstrom an (sofern es eines gibt) und der letzte Name ist der Name des Treibers.

-t
: (t - top) Es wird nur das oberste Modul ausgegeben, sofern es eines gibt.

-m␣modul
: *strconf* überprüft, ob das angegebene Modul für den Datenstrom aktiviert ist. Falls dies der Fall ist, gibt *strconf* die Meldung *yes* aus und liefert den Ende-Status 0. Falls das angegebene Modul nicht aktiviert ist, gibt *strconf* die Meldung *no* aus und liefert einen Ende-Status ungleich Null.

Sie dürfen nur eine der Optionen *-t* und *-m* angeben.

ENDE-STATUS

0 Der Datenstrom wurde erfolgreich überprüft. Bei den Optionen *-m* und *-t* besagt der Ende-Status 0, daß angegebene bzw. oberste Modul aktiviert ist.

≠0 Eine der Optionen *-m* oder *-t* wurde angegeben und
das gesuchte Modul ist nicht aktiviert. Bei verschiedenen Fehlerbedingungen wird ebenfalls ein Ende-Status ungleich Null zurückgegeben, z.B. bei falschen Optionen oder falls ein *ioctl*-Aufruf fehlschlägt.

BEISPIELE

1. Das Kommando *strconf* ohne Argumente erzeugt bei einem Datenstrom, für den nur das Modul *ldterm* über dem Treiber *ports* aktiviert ist, die folgende Ausgabe:

   ```
   ldterm
   ports
   ```

2. Mit dem folgenden Kommando können Sie überprüfen, ob das Modul *ldterm* für den Datenstrom aktiviert ist:

   ```
   strconf -m ldterm
   ```

   Falls dieses Modul tatsächlich aktiviert ist, erhalten Sie die Ausgabe

   ```
   yes
   ```

   und *strconf* beendet sich mit einem Ende-Status ungleich Null.

SIEHE AUCH

strchg
streamio [13]

strings
Druckbare Zeichenketten in Objekt- oder Binärdateien suchen

string 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, Mögliche Zeichensätze*). *strings* können Sie z.B. benutzen, um unbekannte Objektdateien zu identifizieren.

```
strings[␣[-]a][␣-o][␣[-]anzahl]␣datei...
```

-a
strings sucht in der ganzen Datei nach druckbaren Zeichenketten.

-a nicht angegeben:
strings sucht nur im initialisierten Datenbereich von Objektdateien nach druckbaren Zeichenketten.

-o
Vor jeder Zeichenkette wird ihre Position in der Datei ausgegeben.

[-]anzahl
Als Zeichenkette gilt jede Folge von *anzahl* oder mehr druckbaren Zeichen, die mit einem Neue-Zeile-Zeichen oder einem Null-Byte enden.

-anzahl nicht angegeben:
Als Zeichenkette gilt jede Folge von vier oder mehr druckbaren Zeichen, die mit einem Neue-Zeile-Zeichen oder einem Null-Byte enden.

datei
Name der Binärdatei, die *strings* nach druckbaren Zeichenketten durchsuchen soll.

BEISPIEL

Suchen aller druckbaren Zeichenketten in der ausführbaren Binärdatei *a.out*:

```
$ strings a.out
halli hallo
```

Dies kann die Ausgabe für eine Datei sein, deren Quellcode wie folgt aussah:

```
#include <stdio.h>

main()
{
printf("halli hallo\n");
}
```

SIEHE AUCH

od

stty
Eigenschaften einer Datensichtstation ausgeben oder ändern (set the options for terminal)

Mit *stty* können Sie Eigenschaften einer Datensichtstation ändern oder abfragen.

```
stty[option]...
```

Keine Option angegeben
stty gibt einige Eigenschaften der Datensichtstation aus, die die Standard-Eingabe von *stty* ist.

option

Optionen, die den Umfang der Ausgabe festlegen

-a
(a - all) *stty* gibt alle Einstellungen der Datensichtstation aus.

-g
stty gibt die Einstellungen in einem Format aus, daß Sie sie 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 acht Gruppen zusammengefaßt; zusätzlich gibt es eine Gruppe von Optionen, die durch Kombinationen von Optionen dieser Gruppen entstehen.

Erklärungen für die in Klammern angegebenen Optionen sind ebenfalls in Klammern gesetzt.

Das Kommando *stty* überprüft Kombinationen von Optionen nicht auf ihre Plausibilität, auch wenn viele Kombinationen nicht sinnvoll sind.

Datenübertragung steuern

parenb (-parenb)
(parenb - parity enable) Erzeugung und Prüfung des Paritätsbits wird unterstützt (nicht unterstützt).

parext (-parext)
(parext - extended parity) Erweiterte Erzeugung und Überprüfung des Paritätsbits für "mark and space parity" wird unterstützt (nicht unterstützt).

parodd (-parodd)
(parodd - odd parity) Zeichenparität ist ungerade (gerade).
Falls *parext* wirksam ist: mark/space parity ist ungerade (gerade). Diese Optionen sind nur zusammen mit der Option *parenb* bzw. *parext* wirksam.

cs5 cs6 cs7 cs8
(cs - character size) Zeichengröße festlegen. Die Zahl gibt jeweils an, wieviele Bits zur Darstellung eines Zeichens bei der Ein- und Ausgabe verwendet werden (ohne das Paritätsbit).

0
"Telefon"-Leitungen sofort abbauen. Das gilt für alle Datensichtstationsleitungen - nicht nur für Modem-Leitungen. Das Signal SIGHUP wird an alle Prozesse gesendet, die mit dieser Datensichtstation verbunden sind.

ispeed zahl
(ispeed - input speed) Die Datenübertragunsrate für den Eingang der Datensichtstation wird auf *zahl* Baud gesetzt. Nicht alle Geräte unterstützen jedoch verschiedene Übertragungsraten für Eingang und Ausgang. Wird der Wert Null angegeben, so wird als Übertragungsrate für den Eingang die gleiche Übertragungsrate wie für den Ausgang gesetzt.

ospeed zahl
(ospeed - output speed) Die Datenübertragunsrate für den Ausgang der Datensichtstation wird auf *zahl* Baud gesetzt. Nicht alle Geräte unterstützen jedoch verschiedene Übertragungsraten für Eingang und Ausgang. Wird der Wert Null angegeben, so wird die Leitung sofort abgebaut.

zahl
Die Datenübertragungsrate wird, falls möglich, auf *zahl* Baud gesetzt. Nicht jede Hardware-Schnittstelle unterstützt jedoch jede Datenübertragungsrate.

hupcl (-hupcl)
(hupcl - hangup on close) Nach dem letzten Aufruf von *close()* wird die Leitung abgebaut (nicht abgebaut).

hup (-hup)
Wie *hupcl* (*-hupcl*)

cstopb (-cstopb)
(cstpb - character stop bit) Pro Zeichen werden zwei Stop-Bits (wird ein Stop-Bit) verwendet.

cread (-cread)
Der Daten-Empfang ist freigegeben (ist nicht freigegeben).

clocal (-clocal)
Es wird ein lokaler Direktanschluß ohne Modem-Steuersignale angenommen und deshalb werden die Modem-Steuersignale nicht unterstützt. (Die Modem-Steuersignale werden unterstützt.)

loblk (-loblk)
(loblk - block output from layer) Die Ausgabe einer nicht aktuellen Schicht (layer) wird blockiert (nicht blockiert).

Dateneingabe steuern

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 ermöglicht (nicht ermöglicht).

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 in der Eingabe werden in Wagenrücklauf-Zeichen (carriage return) verwandelt (nicht in Wagenrücklauf-Zeichen verwandelt).

igncr (-igncr)
(igncr - ignore carriage return) Wagenrücklauf-Zeichen 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 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 (arbeitet nicht) mit einer 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 fortgesetzt).

ixoff (-ixoff)
Das System sendet (sendet nicht) ein START- bzw. STOP-Zeichen, wenn die Eingabe-Warteschlange fast leer bzw. voll ist.

imaxbel (-imaxbel)
Wenn die Eingabezeile zu lang ist, wird das ASCII-Zeichen BEL (das Klingelzeichen) ausgegeben (nicht ausgeben).

Datenausgabe steuern

opost (-opost)
(opost - postprocess output) Ausgabe wird nachbearbeitet (nicht nachbearbeitet; alle anderen Optionen zur Steuerung der Daten-Ausgabe werden ignoriert).

olcuc (-olcuc)
(olcuc - output lower case to upper case) Kleinbuchstaben in der Ausgabe werden in die entsprechenden Großbuchstaben umgewandelt (nicht umgewandelt).

onlcr (-onlcr)
(onlcr - output newline to carriage return) Bei der Ausgabe werden Neue-Zeile-Zeichen in Wagenrücklauf-Zeichen (carriage return) umgewandelt (nicht umgewandelt).

ocrnl (-ocrnl)
(ocrnl - output carriage return to newline) Bei der Ausgabe werden Wagenrücklauf-Zeichen in Neue-Zeile-Zeichen umgewandelt (nicht umgewandelt).

onocr (-onocr)
Wagenrücklauf-Zeichen werden nicht auf Spalte 0 ausgegeben (werden auf Spalte 0 ausgegeben).

onlret (-onlret)
Ein Neue-Zeile-Zeichen in der Ausgabe führt auch die Funktion eines Wagenrücklauf-Zeichens aus (führt nur die Funktion eines Neue-Zeile-Zeichens aus).

ofill (-ofill)
Ausgabe von Füllzeichen für Verzögerungen verwenden (Zeitverzögerung verwenden).

ofdel (-ofdel)
Als Füllzeichen werden Abbruch-Zeichen DEL 0x7f verwendet
(werden Null-Zeichen NUL 0x0 verwendet).

cr0 cr1 cr2 cr3
(cr - carriage return) Art der Verzögerung bei der Ausgabe von Wagenrücklauf-Zeichen wählen.

nl0 nl1
(nl - newline) Art der Verzögerung bei der Ausgabe von Neue-Zeile-Zeichen wählen.

tab0 tab1 tab2 tab3
(tab - tabulator) Art der Verzögerung bei der Ausgabe von Tabulatorzeichen wählen.

bs0 bs1
(bs - backspace) Art der Verzögerung bei der Ausgabe von Rücksetz-Zeichen (backspace) wählen.

ff0 ff1
(ff - form feed) Art der Verzögerung bei der Ausgabe von Seitenvorschub-Zeichen (form feed) wählen.

vt0 vt1
(vt - vertical tabulator) Art der Verzögerung bei der Ausgabe von Vertikaltabulator-Zeichen wählen.

Lokale Merkmale

isig (-isig)
Die Zeichen werden auf die Steuerzeichen intr, quit und swtch abgeprüft (nicht abgeprüft).

icanon (-icanon)
(icanin - 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 Zeichenkette Rücksetz-Zeichen-Leerzeichen-Rücksetz-Zeichen (backspace-space-backspace) angezeigt (werden nicht als eine solche Zeichenkette angezeigt).

Vorsicht

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 ständig gemerkt wird.

echok (-echok)
(echok - echo kill) Nach einem kill-Zeichen wird zusätzlich ein Neue-Zeile-Zeichen ausgegeben (nicht ausgegeben).

lfkc (-lfkc)
Wie *echok* (*-echok*), veraltet.

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.

stwrap (-stwrap)
Zeilen mit mehr als 79 Zeichen werden bei einer synchronen Leitung nicht abgeschnitten (werden abgeschnitten).

tostop (-tostop)
Wenn ein Hintergrundprozeß auf die Datensichtstation schreibt, wird das Signal SIGTTOU geschickt (nicht geschickt).

echoctl (-echoctl)
(echoctl - echo control characters) Steuerzeichen werden als *^Zeichen* (Delete als ausgegeben (nicht ausgegeben).

echoprt (-echoprt)
Wenn ein Zeichen gelöscht wird, wird das Erase-Zeichen ausgeben (nicht ausgegeben).

echoke (-echoke)
Wenn eine ganze Zeile gelöscht wird, wird eine Folge von Rücksetz-Zeichen-Leerzeichen-Rücksetzzeichen ausgegeben (nicht ausgegeben).

flusho (-flusho)
(flusho - flush output) Die Ausgabe-Warteschlange wird geleert (wird nicht geleert).

pendin (-pendin)
(pendin - retype pending input) Nicht erledigte Eingabe wird angezeigt, wenn das nächste Zeichen gelesen oder eingegeben wird.

iexten (-iexten)
Erweiterte Funktionen für die Eingabedaten sind eingeschaltet (sind ausgeschaltet).

stflush (-stflush)
Bei einer synchronen Leitung wird nach jedem Aufruf von *write* die Ausgabewarteschlange geleert (nicht geleert).

stappl (-stappl)
(stappl - usr application mode) Bei synchroner Datenübertragung wird der Anwendungsmodus benutzt (nicht benutzt, d.h. in diesem Fall wird der Zeilenmodus benutzt).

Hardwareprotokoll steuern

rtsxoff (-rtsxoff)
Für die Eingabe wird das RTS-Hardwareprotokoll verwendet (nicht verwendet).

ctsxon (-ctsxon)
Für die Ausgabe wird das CTS-Hardwareprotokoll verwendet (nicht verwendet).

dtrxoff (-dtrxoff)
Für die Eingabe wird das DTR-Hardwareprotokoll verwendet (nicht verwendet).

cdxon (-cdxon)
Für die Ausgabe wird das CD-Hardwareprotokoll verwendet (nicht verwendet).

isxoff (-isxoff)
Für die Eingabe wird das isochrone Hardwareprotokoll verwendet (nicht verwendet).

Taktrate einstellen

xcibrg
Die Sendetaktrate wird vom internen Taktgenerator erzeugt.

xctset
Die Sendetaktrate wird der Leitung "transmitter signal element timing (DCE source)" entnommen (Schnittstelle CCITT V.24, Schaltung 114, EIA-232-D Anschluß 15).

xcrset
Die Sendetaktrate wird der Leitung "receiver signal element timing (DCE-source)" entnommen (Schnittstelle CCITT V.24, Schaltung 115, EIA-232-D Anschluß 17).

rcibrg
Die Empfangstaktrate wird vom internen Taktgenerator erzeugt.

rctset
Die Empfangstaktrate wird der Leitung "transmitter signal element timing (DCE source)" entnommen (Schnittstelle CCITT V.24, Schaltung 114, EIA-232-D Anschluß 15).

rcrset
Die Empfangstaktrate wird der Leitung "receiver signal element timing (DCE-source)" entnommen (Schnittstelle CCITT V.24, Schaltung 115, EIA-232-D Anschluß 17).

tsetcoff
Keine Taktrate an der Leitung "transmitter signal element timing" verfügbar.

tsetcrbrg
Der Takt vom Empfangstaktgenerator wird an der Leitung "transmitter signal element timing (DTE Source)" ausgegeben (Schnittstelle CCITT V.24, Schaltung 113, EIA-232-D Anschluß 24).

tsetctbrg
Der Takt vom Sendetaktgenerator wird an der Leitung "transmitter signal element timing (DTE Source)" ausgegeben (Schnittstelle CCITT V.24, Schaltung 113, EIA-232-D Anschluß 24).

tsetctset
"transmitter signal element timing (DCE source)" wird an der Leitung "transmitter signal element timing (DTE Source)" ausgegeben (Schnittstelle CCITT V.24, Schaltung 113, EIA-232-D Anschluß 24).

tsetcrset
"receiver signal element timing (DCE source)" wird an der Leitung "transmitter signal element timing (DTE Source)" ausgegeben (Schnittstelle CCITT V.24, Schaltung 113, EIA-232-D Anschluß 24).

rsetcoff
Keine Taktrate an der Leitung "receiver signal element timing" verfügbar.

rsetcrbrg
Der Takt vom Empfangstaktgenerator wird an der Leitung "receiver signal element timing (DTE Source)" ausgegeben (Schnittstelle CCITT V.24, Schaltung 128, kein entsprechender EIA-232-D Anschluß).

rsetctbrg
Der Takt vom Sendetaktgenerator wird an der Leitung "receiver signal element timing (DTE Source)" ausgegeben (Schnittstelle CCITT V.24, Schaltung 128, kein entsprechender EIA-232-D Anschluß).

rsetctset
"transmitter signal element timing (DCE source)" wird an der Leitung "receiver signal element timing (DTE Source)" ausgegeben (Schnittstelle CCITT V.24, Schaltung 128, kein entsprechender EIA-232-D Anschluß).

rsetcrset
"receiver signal element timing (DCE source)" wird an der Leitung "receiver signal element timing (DTE Source)" ausgegeben (Schnittstelle CCITT V.24, Schaltung 128, kein entsprechender EIA-232-D Anschluß).

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

evenp oder **parity**
entspricht *parenb* und *cs7*.

oddp
entspricht *parenb*, *cs7* und *parodd*.

spacep
entspricht *parenb*, *cs7* und *parext*.

markp
entspricht *parenb*, *cs7*, *parodd* und *parext*.

-parity oder **-evenp**
entspricht *-parenb* und *cs8*.

-oddp
enspricht *-parenb*, *-parodd* und *cs8*

-spacep
entspricht *-parenb*, *-parext* und *cs8*

-markp
entspricht *-parenb*, *-parodd*, *-parext* und *cs8*

raw (**-raw** oder **cooked**)
Raw-Modus einschalten (ausschalten). Raw-Modus heißt: *cs8* ist gesetzt, die Steuerzeichen erase, kill, intr, quit, swtch und eot haben keine Bedeutung, keine Nachbearbeitung der Ausgabe und keine Parität.

nl (**-nl**)
entspricht *-icrnl* und *-onlcr* (*icrnl* und *onlcr* und zusätzlich *-inlcr*, *-igncr*, *-ocrnl* und *-onlret*).

lcase (**-lcase**)
entspricht *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.

term
Alle Werte werden passend für ein Terminal vom Typ *term* eingestellt. *term* kann dabei sein: tty33, tty37, vt05, tn300, ti700 oder tek.

Fenstergröße einstellen

rows n
Die Fenstergröße wird auf *n* Zeilen festgesetzt.

columns n
Die Fenstergröße wird auf *n* Spalten festgesetzt.

ypixels n
Die Höhe des Fensters wird auf *n* Bildpunkte festgesetzt.

xpixels n
Die Breite des Fensters wird auf *n* Bildpunkte festgesetzt.

BEISPIELE

1. Aktuelle Einstellung abfragen:

```
$ stty -a
speed 38400 baud;
intr = DEL; quit = ^|; erase = ^h;
kill = @; 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 ff1
```

2. Einstellung so ändern, daß weitere Eingaben an der Tastatur nicht am Bildschirm angezeigt werden (Echo ausschalten):

```
$ stty -echo
```

   Die Einstellung kann man dann nur blind abfragen:

```
$ stty -a
speed 38400 baud; line = 2; intr = DEL; quit = ^|; erase = ^h;
kill = ^x; eof = ^d; eol <undef>; swtch <undef>
parenb parodd cs7 -cstopb hupcl cread clocal -loblk
ixon -ixany -ixoff
isig icanon -xcase -echo echoe echok -echonl -noflsh
opost -olcuc onlcr -ocrnl -onocr -onlret -ofill -ofdel
```

3. Einstellung ändern, so daß Eingaben am Bildschirm wieder angezeigt werden (Echo wieder einschalten):

```
$ stty echo
```

SIEHE AUCH

tabs
ioctl() [14]
termio, *termiox* [5]

su
Benutzerkennung vorübergehend wechseln (become superuser or another user)

Mit *su* können Sie vorübergehend unter einer anderen Benutzerkennung arbeiten.

Alle Benutzer außer dem Systemverwalter müssen nach dem Aufruf von *su* das Kennwort der Benutzerkennung angeben, unter der sie arbeiten wollen. Der Systemverwalter (Benutzername *root*) kann ohne Kennwort unter allen Benutzerkennungen arbeiten. Durch den Aufruf von *su* wird eine Sub-Shell gestartet, wenn für die angegebene Benutzerkennung in der Datei */etc/passwd* als Startprogramm eine Shell eingetragen ist. Ohne einen solchen Eintrag wird standardmäßig */usr/bin/sh* gestartet. Die Benutzer-Nummer der angegebenen Benutzerkennung wird zur effektiven und realen Benutzer-Nummer des aktuell laufenden Prozesses. Die Variable *PATH* (siehe *sh*) erhält den in der Arbeitsumgebung der angegebenen Benutzerkennung definierten Wert. Die Benutzerkennung *root* erhält für *PATH* den Wert */sbin:/usr/sbin:/usr/bin:/etc*. Die übrige Arbeitsumgebung des Benutzers, der *su* aufruft, z.B. *$USER* und *$HOME*, bleibt erhalten
(siehe aber Option -).

Alle systemweit (auch über ferne Rechner) unternommenen Versuche, mit *su* unter einer anderen Benutzerkennung zu arbeiten, werden in der Datei */var/adm/sulog* mitprotokolliert, wenn die Umgebungsvariable *SULOG* definiert ist. Alle systemweit (auch über ferne Rechner) unternommenen Versuche, mit *su* als Systemverwalter unter der Benutzerkennung *root* zu arbeiten, werden auf der Konsole ausgegeben, wenn die Umgebungsvariable *CONSOLE* definiert ist.

Mit der Taste END beenden Sie die aufgerufene Subshell und arbeiten wieder unter der vorher gültigen Benutzerkennung. Es gilt dann wieder vollständig die vorherige Arbeitsumgebung.

Aktuelle Benutzerkennung abfragen

Um zu überprüfen, unter welcher Benutzerkennung Sie gerade arbeiten, geben Sie das Kommando *id* ein.

Login - Benutzerkennung abfragen

Um zu überprüfen, unter welcher Benutzerkennung Sie sich mit dem letzten *login* am System angemeldet haben, geben Sie das Kommando *who am i* oder *logname* ein.

```
su[ -][ benutzerkennung[ arg] ...]
```

-

Wenn Sie - angeben und in der Datei */etc/passwd* für *benutzerkennung* als Startprogramm *sh* eingetragen ist, dann wird die Arbeits-Umgebung so abgeändert, als würden Sie sich unter *benutzerkennung* mit *login* am System anmelden. Es wird eine Login-Shell aufgerufen (siehe *login*) mit Präfix -, *-sh*, und zuerst */etc/profile* und dann das benutzerspezifische *$HOME/.profile* ausgeführt. In *.profile* können Sie *arg0* abfragen, ob es den Wert *-sh* oder den Wert *-su* hat. Im ersten Fall erfolgte der Aufruf vom Kommando *login*, im zweiten Fall vom Kommando *su* aus.

Ist als Startprogramm nicht */usr/bin/sh* sondern ein anderes *programm* eingetragen, dann wird *-programm* als *arg0* an *.profile* übergeben. Dabei ist es egal, ob der Aufruf von *login* oder *su* aus erfolgte.

benutzerkennung

Benutzerkennung, unter der Sie vorübergehend arbeiten möchten. Sie benötigen hierfür das Kennwort von *benutzerkennung*, das Sie nach der Eingabe von *su benutzerkennung* eingeben müssen.

Nur der Systemverwalter kann ohne Kennwort unter fremden Benutzerkennungen arbeiten.

benutzerkennung nicht angegeben:
Sie arbeiten als Systemverwalter unter der Benutzerkennung *root* weiter, vorausgesetzt, Sie wissen das Kennwort.

arg

Ein oder mehrere Argumente, die Sie an das in */etc/passwd* eingetragene Startprogramm, standardmäßig */bin/sh*, übergeben. Sie können dieselben Argumente angeben wie beim direkten Aufruf dieses Programms.

Ist das Startprogramm eine Shell, dann ist als einziges Argument auch *-c kommando* oder *-r* möglich.

-c kommando

kommando wird in der Startshell mit der Arbeitsumgebung der angegebenen Benutzerkennung ausgeführt.

-r

Eine eingeschränkte Subshell (restricted shell) wird aufgerufen.

FEHLERMELDUNGEN

`su: Unknown id: benutzerkennung`

Sie haben eine ungültige Benutzerkennung angegeben.

`su: sorry`

Sie haben ein ungültiges Kennwort eingegeben.

`arg: arg: cannot open`

Sie haben ein ungültiges Argument angegeben.

`arg: bad option(s)`

Sie haben eine ungültige Option angegeben.

DATEIEN

/etc/passwd
Systemweite Benutzer- und Kennwortdatei

/etc/profile
Systemweite Shell-Prozedur, die für jede Benutzerkennung von der Login-Shell ausgeführt wird.

$HOME/.profile
Benutzerspezifische Shell-Prozedur, die jeder Benutzer in seinem Home-Dateiverzeichnis einrichten kann. Sie wird von der Login-Shell nach */etc/profile* ausgeführt.

/var/adm/sulog
Datei, in der systemweit alle Versuche, mit *su* unter einem anderen Benutzernamen zu arbeiten, mitprotokolliert werden.

/etc/default/su
Diese Datei enthält, falls sie definiert sind, Standardeinstellungen für die Umgebungsvariablen *SULOG*, *CONSOLE*, *PATH* und *SUPATH*. z.B. ist *SULOG* auf */var/adm/sulog* gesetzt.

UMGEBUNGSVARIABLEN

SULOG
Falls dieser Variablen ein Wert zugewiesen wird, werden alle Versuche, mit *su* eine andere Benutzerkennung zu erlangen, in der Datei */var/adm/sulog* mitprotokolliert.

CONSOLE
Wenn diese Variable definiert ist, werden alle unternommenen Versuche, mit *su root* zu arbeiten, auf der Console mitprotokolliert

PATH
Variable, in der die Dateiverzeichnisse festgelegt sind, in denen die Shell aufgerufene Kommandos sucht. Die Dateiverzeichnisse sind durch Doppelpunkt : voneinander getrennt und werden in der angegebenen Reihenfolge durchsucht.

SUPATH
Variable für *root*, in der die Dateiverzeichnisse festgelegt sind, in denen die Shell aufgerufene Kommandos sucht. Die Dateiverzeichnisse sind durch Doppelpunkt : voneinander getrennt und werden in der angegebenen Reihenfolge durchsucht.

BEISPIELE

Die folgenden Beispiele zeigen den Unterschied zwischen den Kommandos *login* und *su*.

1. Anmelden am System (*login*) unter der Benutzerkennung *gast* und Abfragen der aktuellen Benutzerkennung (*id*), der Login-Benutzerkennung (*who am i*) und der Umgebungsvariablen *LOGNAME* (*echo $LOGNAME*).

```
$ login gast
Password: blinde Eingabe des Kennworts von gast
$ id
uid=100(gast)  gid=10(other)
$ who am i
gast      grtty   MAR 17 15:13
$ echo $LOGNAME
gast
```

2. Mit *su* ohne die Option Bindestrich - in die Benutzerkennung *fritz* wechseln, in deren Arbeitsumgebung die Variable PS1 (Bereitzeichen der Shell) auf den Wert FR gesetzt ist, aber nicht ausgewertet wird. Abfragen der aktuellen Benutzerkennung (*id*), der Login-Benutzerkennung (*who am i*) und der Umgebungsvariablen *LOGNAME* (*echo $LOGNAME*) und Zurückkehren in die Benutzerkennung *gast* (Taste END).

```
$ su fritz
Password: blinde Eingabe des Kennworts von fritz
$ id
uid=103(fritz)  gid=10(other)
$ who am i
gast      grtty   MAR 17 15:13
$ echo $LOGNAME
gast
$ END
$ id
uid=100(gast)  gid=10(other)
```

3. Mit *su* - in die Benutzerkennung *fritz* wechseln, in dessen Arbeitsumgebung die Variable PS1 (Bereitzeichen der Shell) auf den Wert FR gesetzt ist und jetzt wegen Option - auch ausgewertet wird. Abfragen der aktuellen Benutzerkennung (*id*), der Login-Benutzerkennung (*who am i*) und der Umgebungsvariablen *LOGNAME* (*echo $LOGNAME*) und Zurückkehren in die Benutzerkennung *gast* (Taste END).

```
$ su - fritz
Password: blinde Eingabe des Kennworts von fritz
FR id
uid=103(fritz)  gid=10(other)
FR who am i
gast      grtty   MAR 17 15:13
FR echo $LOGNAME
fritz
FR END
$
```

4. Mit *su - -c* unter der Benutzerkennung *fritz*, in dessen Arbeitsumgebung und mit dessen Erlaubnis Abfragen der aktuellen Benutzerkennung (*id*).

```
$ su - fritz -c "id"
Password: blinde Eingabe des Kennworts von fritz
uid=103(fritz)  gid=10(other)
$
```

SIEHE AUCH

env, login, sh
passwd, profile, environ [5]

sum
Prüfsumme einer Datei berechnen

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.

```
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.

Vorsicht

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.

BEISPIELE

1. Prüfsumme der Datei *monate* nach dem ersten Algorithmus:

```
$ sum monate
6658 1 monate
```

2. Prüfsumme der Datei *monate* nach dem zweiten Algorithmus:

```
$ sum -r monate
62412      1 monate
```

SIEHE AUCH

wc

sync
Systempuffer zurückschreiben

sync bewirkt, daß alle bisher noch nicht auf Festplatte oder Diskette 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.

sync

Anwendung

Wenn das System heruntergefahren werden soll, muß vorher *sync* aufgerufen werden, um sicherzustellen, daß das Dateisystem konsistent bleibt. Die Systemverwalter-Kommandos */etc/shutdown* und */etc/halt* rufen automatisch *sync* auf.

Durch *sync* werden nur lokale Puffer auf lokale Festplatten geschrieben. Die Pufferung auf fernen Rechnern kann durch *sync* nicht beeinflußt werden.

SIEHE AUCH

sync [14]

sysname
Informationen zum Betriebssystem ausgeben

sysname gibt folgende Informationen über das aktuelle Betriebssystem aus:

- den Systemnamen
- den Rechnernamen
- die Betriebssystem-Version
- das Datum der Freigabe
- die Systemkennung
- die eingestellte Sprache

sysname befindet sich im Dateiverzeichnis */etc*.

```
/etc/sysname[ option]...
```

Keine Option angegeben
Das Kommando gibt alle Informationen aus.

option

-s
(s - system) *sysname* gibt nur den Systemnamen aus. Der Systemname bezeichnet den Namen des Betriebssystems für einen bestimmten Rechnertyp (z.B. SINIX-H für den Rechner MX 300 oder SINIX-F für MX 500).

-n
(n - name) *sysname* gibt nur den Rechnernamen aus. Der Rechnername ist ein logischer Name des Rechners. Er kann mit dem Kommando *hostname* durch den Systemverwalter festgelegt werden und wird im Begrüßungsbildschirm angezeigt.

-v
(v - version) *sysname* gibt nur die Version des Betriebssystems aus.

-r
(r - release) *sysname* gibt nur das Datum der Freigabe aus.

-i
(i - identification) *sysname* gibt nur die Systemkennung aus. Die Systemkennung ist die Lizenznummer des installierten Systems.

-l
(l - language) *sysname* gibt nur die eingestellte Sprache aus.

SIEHE AUCH

hostname

tabs
Tabulatorstops setzen

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 ganz linke Spalte auf dem Bildschirm. Dies gilt auch für Geräte, deren Spaltenmarkierer mit 0 beginnen wie z.B. bei den Geräten DASI 300, DASI 300s und DASI 450.

```
tabs[␣tabspec][␣-Ttyp][␣+mn]
```

tabspec
Für die Spezifikation von Tabulatoren sind vier verschiedene Tabulatorangaben zulässig: -code, -n, n1,n2,..., --datei.

-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 Kartenspalte 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*[5]):
<: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 Kartenspalte 7. Durch Eingeben eines Leerzeichens wird auf Spalte 8 positioniert, mit einem Tabulatorzeichen auf Spalte 12. Dieses Format wird für COBOL empfohlen. Dateien mit dieser Tabulatoreinstellung sollten folgende Formatangabe enthalten (siehe *fspec*[5]):
<:t␣-c3␣m6␣s66␣d:>

-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 $1+n$,$1+2n$,$1+3n$... gesetzt werden. Bei Eingabe einer 8 erhalten Sie die "Standard"-Tabulatoreinstellung des UNIX-Systems. Bei Eingabe einer Null werden alle Tabulatorstops gelöscht.

n1,n2,...
Mit diesem Format können Sie die Spalten für die Tabulatorstops beliebig wählen. Sie können bis zu 64 Zahlen in aufsteigender Reihenfolge, durch Kommata getrennt, 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.

--datei
tabs liest die erste Zeile der Datei und sucht nach einer Formatangabe (siehe *fspecf* [5]). 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 sichergestellt 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*).

tabspec nicht angegeben:
tabs nimmt die Standardeinstellung *-8* vor, die den Standard-Tabulatoren des UNIX-Systems entspricht.

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.

Die folgenden Optionen können ebenfalls verwendet werden. Kommt eine Option mehr als einmal vor, gilt der zuletzt angegebene Wert.

-Ttyp
Datensichtstations-Typ, der zur Randeinstellung bekannt sein muß. *typ* ist ein in *term* [5] angegebener Name.

-T*typ* nicht angegeben:
tabs verwendet den Wert der Umgebungsvariablen *TERM*. Ist *TERM* in der Umgebung (siehe *environ* [5]) nicht definiert, verwendet *tabs* einen Typ, der für viele Datensichtstationen zutrifft.

+mn
Alle Tabulatoren werden um *n* Stellen nach rechts verschoben, die Spalte $n+1$ wird der linke 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 ganz linke Rand mit $+m0$ gesetzt. Der Rand wird bei den meisten Datensichtstationen nur nach rechts verschoben, wenn $+m$ explizit angegeben ist.

FEHLERMELDUNGEN

`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.

UMGEBUNGSVARIABLE

TERM
Typ der Datensichtstation

BEISPIELE

1. Sie wollen Tabulatorstops gemäß dem üblichen COBOL-Format setzten.

   ```
   $ tabs -c
   ```

2. Sie wollen in jeder 6.Spalte einen Tabulator-Stop setzen, d.h. in den Spalten 1,7,13,19,...

   ```
   $ tabs -6
   ```

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
   ```

4. Sie wollen Tabulatorstops gemäß der ersten Zeile (Formatangabe) Ihrer Datei *$HOME/tabspec.list/dat1* setzen.

   ```
   $ tabs --$HOME/tabspec.list/dat1
   ```

SIEHE AUCH

newform, *pr*, *tput*
fspec, *terminfo*, *environ*, *term* [5]

tail
Den letzten Teil einer Datei ausgeben

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.

```
tail[ stelle[bc][rf]][ datei]
```

stelle
Mit *stelle* legen Sie die Stelle in der Eingabeddatei fest, an der die Ausgabe beginnt. Sie können angeben:

+[zahl]
−[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.

Vorsicht

Abschnitte, die vom Dateiende aus gezählt werden, werden von *tail* in einem Puffer zwischengespeichert. Die Größe des Puffers ist auf 4 Kbyte beschränkt.

zahl nicht angegeben:
Standard-Wert ist 10.

stelle nicht angegeben:
Standard-Wert ist -10, d.h. 10 Einheiten vom Dateiende aus gezählt.

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.

l
(l - line) Die Ausgabe von *tail* beginnt an der durch *stelle* festgelegten Zeile.

Sie dürfen nur eine der Optionen *b*, *c* oder *l* angeben. Wenn Sie keine dieser drei

Optionen angeben, ist standardmäßig *l* gesetzt.

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.

r
(r - reverse) Die Zeilen der Datei werden in umgekehrter Reihenfolge ausgegeben, beginnend am angegebenen Startpunkt. Als Voreinstellung bei *r* werden alle Zeilen der Datei in umgekehrter Reihenfolge ausgegeben.

Sie dürfen nur eine der Optionen *f* und *r* angeben.

datei
Name der Eingabedatei, die *tail* ausgeben soll.

datei nicht angegeben:
tail liest von der Standard-Eingabe.

BEISPIELE

1. In der Datei *monate* steht jeweils in einer Zeile ein Monatsname. Die beiden *tail*-Aufrufe

```
$ tail -5 monate
```

```
$ tail +8 monate
```

haben die gleiche Ausgabe zur Folge:

```
August
September
Oktober
November
Dezember
```

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 -5f anna
Diese fuenf
Zeilen
standen vorher
schon in
der Datei
TUE MAR 21 11:40:35 MET 1989
TUE MAR 21 11:40:40 MET 1989
.
.
.
```

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 mit der Taste DEL abbrechen.

SIEHE AUCH

cat, head, more, pg

talk Dialog mit anderem Benutzer führen

Mit *talk* können Sie mit einem anderen Benutzer kommunizieren, der an einer Datensichtstation am gleichen oder an einem anderen Rechner arbeitet. *talk* sendet Eingabezeilen von Ihrer Datensichtstation zu der Ihres Kommunikationspartners. *talk* ist abhängig von der Rechnerarchitektur: es funktioniert nur zwischen Rechnern, die dieselbe Architektur haben.

```
talk␣benutzerkennung[␣tty-name]
```

benutzerkennung
: Benutzerkennung des Benutzers, mit dem Sie kommunizieren möchten. Wenn der Benutzer *benutzerkennung* 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, müssen Sie angeben: 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. Wer zuerst *talk* aufruft, ist Sender, der Benutzer mit der dabei angegebenen Benutzerkennung ist 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, können Sie mit CTRL l den Bildschirm neu erstellen, mit der Taste DELETE CHAR. ein Zeichen, mit DELETE WORD ein Wort und mit DELETE LINE eine Zeile löschen.

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.

FEHLERMELDUNGEN

Die wichtigsten Fehlermeldungen sind:

`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*).

SIEHE AUCH

mail, mesg, pr, who, write

tar
Archivieren von Dateien auf Magnetbandkassette, Band oder Diskette und Archive bearbeiten (tape file archiver)

Mit *tar* können Sie:

- Dateien oder Dateiverzeichnisse auf Magnetband, Magnetbandkassette oder Diskette archivieren (Format 1).
- den Inhalt eines Band- oder Diskettenarchivs lesen (Format 2).
- ein Archiv vollständig oder einzelne Dateien und Dateiverzeichnisse von Magnetband, Magnetbandkassette oder Diskette in ein Dateiverzeichnis einlesen (Format 3).

Als Systemverwalter können Sie mit *tar* beliebige Dateien und Dateiverzeichnisse sichern und wiederherstellen.

Arbeitsweise des Kommandos

tar arbeitet mit Archiven auf einem Datenträger. *tar* legt ein Archiv auf einem Datenträger an. Dabei setzt es eine Archiv-Anfangsmarke und eine Archiv-Endemarke. Mit einem *tar*-Kommando kann immer nur ein Archiv bearbeitet werden.

Auf einer Diskette kann jeweils immer nur ein Archiv angelegt werden.

Auf einem Magnetband bzw. einer Magnetbandkassette können mehrere Archive angelegt werden. Mit dem Kommando *mt* kann an den Anfang oder das Ende eines Archivs positioniert werden. *tar* liest immer nur aus dem aktuell positionierten Archiv. Befinden sich auf einem Magnetband mehrere Archive und sollen alle eingelesen werden, muß für jedes Archiv ein *tar*-Kommando gegeben werden.

Wenn Sie beim Einlesen die entsprechende Gerätedatei ohne automatisches Zurückspulen verwenden, ist die Bandposition nach dem Einlesen eines Archives automatisch der Beginn des nächsten Archivs. Sie brauchen dann nicht mit *mt* neu zu positionieren.

Vor dem Aufruf beachten

Bevor Sie *tar* ausführen können, müssen Sie das Magnetband, die Magnetbandkassette bzw. die Diskette in das entsprechende Laufwerk einlegen.

Wenn Sie ein Archiv anlegen wollen, muß der Schreibschutz des Datenträgers entfernt sein. Weitere Informationen hierzu finden Sie in der Betriebsanleitung Ihres Rechners. Wenn Sie ein Archiv auf Diskette anlegen wollen, muß diese Diskette formatiert sein. Dazu benutzen Sie das Kommando */etc/format* (siehe *format*)..

Nach dem Aufruf beachten

Wenn Sie *tar* ausgeführt haben, sollten Sie sofort den Datenträger aus dem entsprechenden Laufwerk nehmen. Ein anderer Benutzer könnte versehentlich mit einem weiteren *tar*-Kommando Ihr Archiv beschädigen.

```
tar␣funktion[attribut1...[␣argument]...]...␣datei␣...          Format 1
tar␣t[attribut2...][f␣gerätedatei][␣datei]...                  Format 2
tar␣x[attribut3...][f␣gerätedatei][␣datei]...                  Format 3
```

Format 1: Datei oder Dateiverzeichnis auf Datenträger archivieren

tar␣funktion[attribut1...[␣argument]...]...␣datei␣...

Als Benutzer ohne Systemverwalter-Rechte können Sie nur die Dateien und Dateiverzeichnisse archivieren, auf die Sie zugreifen dürfen. Für Dateien brauchen Sie das Leserecht, für Dateiverzeichnisse das Ausführrecht.

funktion

Für *funktion* geben Sie genau einen der folgenden Buchstaben mit oder ohne Bindestrich - an:

c

Diskette:

(c - create) *tar* legt auf dem Datenträger ein neues Archiv an und schreibt die angegebene Datei an den Beginn des neuen Archivs. Ein eventuell bereits vorhandenes Archiv auf dem Datenträger wird gelöscht. Werden mehrere Dateien angegeben, werden sie fortlaufend eingetragen. Wird für *datei* ein Dateiverzeichnis angegeben, werden alle Unterbäume des Dateiverzeichnisses archiviert.

Magnetband und Magnetbandkassette:

tar schreibt die angegebene Datei oder das angegebene Dateiverzeichnis ab der aktuellen Bandposition des eingelegten Magnetbandes bzw. Magnetbandkassette. Auf einem Magnetband kann mit dem Kommando *mt* an das Ende eines Archivs positioniert werden. So können mit *tar c* auf einem Magnetband mehrere Archive aneinandergehängt werden. Sie können verhindern, daß das Band nach dem Zugriff des *tar*-Kommandos zurückgespult wird, indem Sie die entsprechende Gerätedatei ansprechen (siehe *Tabellen und Verzeichnisse, Gerätedateien für Datenträger*).

Befinden sich mehrere Archive auf einem Magnetband und Sie überschreiben eines, so sind alle folgenden Archive nicht mehr lesbar.

Beispiel

Sie haben vier Archive auf einem Band. Sie positionieren auf den Beginn des zweiten und beschreiben ab dieser Position das Band mit einem neuen Archiv. Die alten Archive 2, 3 und 4 sind damit überschrieben.

Vorsicht

Paßt eine Datei nicht mehr auf den Datenträger, bricht *tar* mit folgender Fehlermeldung ab:

```
tar: write error
```

Die letzte Datei ist dann nicht ordnungsgemäß gesichert.

r

(r - replace) Diese Funktion gilt nur für Disketten oder Plattenarchive.
Die angegebene Datei bzw. die Dateien werden an das Ende eines bereits bestehenden Archivs angehängt.

u

(u - update) Diese Funktion gilt nur für Disketten oder Plattenarchive.
Die angegebene Datei bzw. Dateien werden dann an das Ende eines bereits bestehenden Archivs angehängt, wenn

- sie noch nicht im Archiv vorhanden sind, oder
- die Zeit der letzten Änderung der Datei im Archiv früher liegt, als die Zeit der letzten Änderung der zu archivierenden Datei.

attribut1

Die ausgewählte Funktion kann durch Angabe eines oder mehrerer Funktionsattribute gesteuert werden. Diese Attribute werden ohne Leerzeichen an die Funktion angefügt. Einige Attribute erwarten außerdem die Eingabe von Argumenten. Dabei ist folgendes zu beachten:

1. Geben Sie zuerst alle Attribute ohne Leerzeichen ein.
2. Geben Sie dann die Argumente getrennt durch Leerzeichen an. Die Reihenfolge der Argumente wird bestimmt durch die Reihenfolge, in der die zugehörigen Attribute eingegeben wurden.

attribut1 kann sein:

v

(v - verbose) *tar* gibt zu jeder Datei, die gerade bearbeitet wird, folgende Informationen aus:

```
a    pfadname  n blocks
```

Dabei bedeutet:

a (a - append) die angegebene Datei wurde in das Archiv eingetragen.

pfadname der Pfadname, unter dem die Datei eingetragen wurde.
n blocks die Anzahl der Blöcke, die die Datei belegt. Die Blockgröße ist abhängig vom verwendeten Medium.

v nicht angegeben:
tar gibt keine Meldungen aus.

w

(w - what) *tar* erwartet für jede Datei eine Bestätigung, bevor die Aktion ausgeführt wird. *tar* mit dem Attribut *w* gibt aus:

```
r dateiname:
```

Dabei bedeutet:
dateiname der Name der Datei oder des Dateiverzeichnisses, das archiviert werden soll

Nach dem Doppelpunkt : erwartet *tar* Ihre Eingabe. Nur wenn Sie mit y bejahen, führt *tar* die Aktion aus, sonst nicht.

w nicht angegeben:
tar führt die Aktion ohne Bestätigung aus.

f␣gerätedatei

(f - file) *tar* erwartet die Eingabe einer Gerätedatei, in die geschrieben werden soll, wenn Sie nicht die Standard-Gerätedatei benutzen wollen. *tar* interpretiert das zugehörige Argument als Gerätedatei.

gerätedatei
Name der Gerätedatei, in die geschrieben werden soll. Normalerweise verwenden Sie hier die Gerätedateien für Disketten-, Magnetband- oder Magnetbandkassettenlaufwerke, soweit Sie solche Geräte an Ihrem Rechner angeschlossen haben. Die Gerätedateien für alle Geräte befinden sich im Dateiverzeichnis */dev*.

Geben Sie für *gerätedatei* einen Bindestrich - an, dann schreibt *tar* auf die Standard-Ausgabe. Auf diese Weise können Sie *tar* in einer Pipe verwenden. So kann *tar* benutzt werden, um Dateiverzeichnisse oder Dateisysteme zu kopieren oder zu verschieben (siehe *Beispiel*).

Eine Liste der gebräuchlichen Gerätedateien befindet sich im Anhang dieses Handbuches.

Geben Sie hier eine normale Datei an, wird ein *tar*-Archiv auf der Festplatte angelegt.

gerätedatei nicht angegeben:
tar unternimmt folgende Schritte, um eine Gerätedatei zu bestimmen: Haben Sie in der Argumentliste zu *attribut1* eine Zahl zwischen 0 und 9 angegeben, wählt *tar* den entsprechenden Eintrag aus der Datei */etc/default/tar*

als Gerätedatei aus. Ist keine Zahl angegeben und die Shell-Variable *TAPE* gesetzt, nimmt *tar* den Inhalt von *TAPE*. Sonst wählt *tar* den Eintrag 0 aus */etc/default/tar*.

b␣blockungsfaktor
(b - blocking factor) Dieses Attribut können Sie verwenden, wenn Sie ein Magnetband bzw. Magnetbandkassette mit *tar* beschreiben.
tar interpretiert das dazugehörige Argument als Blockungsfaktor.

blockungsfaktor
Als Blockungsfaktor können Sie eine Zahl zwischen 1 und 20 angeben.

Der Blockungsfaktor gibt an, wieviele Band-Blöcke *tar* pro Bandzugriff schreiben soll. Je höher der Blockungsfaktor, desto mehr Daten passen auf das Band.

b blockungsfaktor nicht angegeben:
Blockungsfaktor = 1

L
tar verfolgt symbolische Verweise (symbolic links). Ohne dieses Attribut schreibt *tar* nur den symbolischen Verweis, aber nicht die zugehörigen Daten.

zahl
tar schreibt auf das Gerät, das im *zahl*-ten Eintrag in der Datei */etc/default/tar* angegeben ist. *zahl* muß im Bereich zwischen 0 und 9 liegen.

argument
Zu den Attributen *f* und *b* gehören Argumente (*gerätedatei* bzw. *blockungsfaktor*). Diese Argumente geben Sie anschließend an die Funktion und **alle** Attribute an, getrennt durch Leerzeichen. Das Kommando *tar* ordnet die Argumente anhand ihrer Reihenfolge den Attributen zu. Das heißt: geben Sie die Attribute *f* und *b* in der Reihenfolge *fb* ein, so wird das erste folgende Argument als *gerätedatei* und das zweite als *blockungsfaktor* interpretiert.

Beispiele

```
$ tar cfvb /dev/dsk/f0t 10 *.c
           |            |  |
           |            |  Dateien, die archiviert werden sollen
           |            |
           |            Blockungsfaktor zu b
           |
           Gerätedatei zu f
```

```
$ tar cv0 *.c
```

tar schreibt alle Dateien mit der Endung *.c* auf das Gerät 0 aus */etc/default/tar* (z.B. 3,5 Zoll Floppy).

datei
Name der Datei bzw. des Dateiverzeichnisses, die bzw. das archiviert werden soll. Mindestens ein Name muß angegeben sein. Sie können auch mehrere Dateien oder Dateiverzeichnisse angeben.

Ist die angegebene Datei ein Dateiverzeichnis, dann archiviert *tar* dieses Dateiverzeichnis mit allen darin enthaltenen Dateien und Unterdateiverzeichnissen.

Den Dateinamen können Sie auch mit Sonderzeichen der Shell angeben.

Ist eine Datei zu groß für einen Datenträger, dann muß sie vor dem Archivieren mit dem Kommando *split* geteilt werden. Sie kann dann später mit dem Kommando *cat* wieder zusammengefügt werden.

Format 2: Inhalt eines Band- oder Diskettenarchivs lesen

tar␣t[attribut2...][**f**␣gerätedatei][␣datei]...

t
(t - table) *tar* gibt den Namen der angegebenen Datei aus, falls sie im Archiv steht. Ist keine Datei angegeben, gibt *tar* das Inhaltsverzeichnis des gesamten Archivs aus.

attribut2
Das Kommando *tar t* kann durch Angabe eines oder mehrerer Funktionsattribute gesteuert werden. Diese Attribute werden ohne Leerzeichen an die Funktion angefügt. Dabei ist folgendes zu beachten:

1. Geben Sie zuerst alle Attribute (auch *f*) ohne Leerzeichen ein.
2. Geben Sie dann *gerätedatei* (Option *f*) getrennt durch Leerzeichen an.

attribut2 kann sein:

v
(v - verbose) *tar* gibt zu jeder Datei, die im Archiv gelesen wird, folgende Informationen aus:

- die gesetzten Zugriffsrechte
- Benutzernummer/Gruppennummer (UID/GID)
- Größe in byte
- Datum und Uhrzeit der Erstellung der Datei
- den Namen der Datei

Beispiel

```
rw------- 33/1   1045 Mar 16 15:01 1988 liste
```

v muß ohne Leerzeichen an *t* angefügt werden.

v nicht angegeben:
Das Kommando *tar* gibt nur den Namen der Dateien aus.

L

tar verfolgt symbolische Verweise (symbolic links). Ohne dieses Attribut schreibt *tar* nur den symbolischen Verweis, aber nicht die zugehörigen Daten.

zahl

tar schreibt auf das Gerät, das im *zahl*-ten Eintrag in der Datei */etc/default/tar* angegeben ist. *zahl* muß im Bereich zwischen 0 und 9 liegen.

f␣gerätedatei

(f - file) *tar* erwartet die Eingabe einer Gerätedatei, von der gelesen werden soll, wenn Sie nicht die Standard-Gerätedatei benutzen wollen (siehe *Format 1*).

datei

Name der Datei, deren Name ausgegeben werden soll, wenn sie im Archiv steht. Sie können mehrere Dateien angeben.

Ist die angegebene Datei ein Dateiverzeichnis, dann gibt *tar* dieses Dateiverzeichnis mit allen darin enthaltenen Dateien und Unterdateiverzeichnissen aus.

datei nicht angegeben:
tar gibt den Inhalt des gesamten Archivs aus.

Format 3: Dateien von Datenträger einlesen

tar␣x[attribut3]...[f␣gerätedatei][␣datei]...

x

(x - extract) *tar* liest die angegebene Datei
vom Archiv ein. Steht die Datei
mit relativem Pfadnamen im Archiv, so wird sie in das aktuelle Dateiverzeichnis kopiert. Steht die Datei mit absolutem Pfadnamen im Archiv, so wird sie in das entsprechende Dateiverzeichnis kopiert.

Als Benutzer ohne Systemverwalter-Rechte brauchen Sie das Schreibrecht für das Dateiverzeichnis, in das eingelesen wird.

Ist die angegebene Datei ein Dateiverzeichnis, so werden alle darin enthaltenen Dateien und Unterdateiverzeichnisse kopiert.

Existiert die angegebene Datei noch nicht in dem Dateiverzeichnis, in das sie kopiert werden soll, so wird diese Datei angelegt.

Benutzernummer des Eigentümers, Gruppennummer und Zeit der letzten Änderung werden von der archivierten Datei übernommen. Den archivierten Benutzer- und Gruppennummern werden die Namen zugewiesen, die in den Dateien */etc/passwd* bzw. */etc/group* eingetragen sind. Gibt es in diesen Dateien keinen Eintrag zu den archivierten Nummern, so gehört die kopierte Datei der entsprechenden Nummer als Eigentümer bzw. als Gruppe. Der Systemverwalter kann mit *chown* bzw. *chgrp* dieser Datei den gewünschten Eigentümer und die gewünschte Gruppe zuordnen. Das gleiche gilt, falls die angegebene Datei ein Dateiverzeichnis ist.

Die Zugriffsrechte werden so gesetzt, wie sie mit *umask* definiert sind.

Existiert die angegebene Datei bereits in dem Dateiverzeichnis, in das sie kopiert werden soll, so werden die Zugriffsrechte nicht geändert. Das s-Bit wird nur berücksichtigt, wenn *tar* vom Systemverwalter aufgerufen wird. Modifikationszeit, Eigentümer und Gruppe bleiben unverändert.

Befinden sich Dateien gleichen Namens mehrfach im Archiv (siehe *Format 1*) so überschreibt die nachfolgend vom Archiv eingelesene Datei eine bereits vorher kopierte Datei gleichen Namens. In diesem Fall wird die Modifikationszeit nicht berücksichtigt.

attribut3

Das Kommando *tar x* kann durch Angabe eines oder mehrerer Funktionsattribute gesteuert werden. Diese Attribute werden ohne Leerzeichen an die Funktion angefügt. Dabei ist folgendes zu beachten:

1. Geben Sie zuerst alle Attribute (auch *f*) ohne Leerzeichen ein.
2. Geben Sie dann *geräte_datei* (Option *f*) getrennt durch Leerzeichen an.

attribut3 kann sein:

l

(l - link) *tar* meldet, wenn ein Verweis auf andere Dateien nicht aufgelöst werden kann. Dies gilt nicht für symbolische Verweise.

l nicht angegeben:
tar gibt keine Fehlermeldung aus, wenn ein Verweis nicht aufgelöst werden kann.

m

(m - modify) *tar* setzt beim Kopieren der angegebenen Datei aus dem Archiv die Zeit der letzten Änderung für die Kopie auf das aktuelle Datum mit Unrzeit.

m nicht angegeben:
Die im Archiv gespeicherte Zeit der letzten Änderung bleibt unverändert.

o

(o - ownership) Die aus dem Archiv kopierte Datei erhält als Eigentümer den Benutzer, der *tar* aufgerufen hat. Als Gruppe wird der Datei ebenfalls die Gruppe des Benutzers zugewiesen, der *tar* aufgerufen hat.

o nicht angegeben:
Die im Archiv gespeicherten Nummern für Eigentümer und Gruppe werden übernommen.

v

(v - verbose) *tar* gibt zu jeder Datei bzw. Dateiverzeichnis, das eingelesen wird, aus:

```
x  pfadname,  k bytes,  n tape blocks
```

Dabei bedeutet:

x	(x - extract) die angegebene Datei wurde in das entsprechende Dateiverzeichnis eingetragen.
k byte	Die Datei ist *k byte* groß.
n tape blocks	Die Datei belegt *n* Blöcke. Die Blockgröße ist abhängig vom verwendeten Medium.

v nicht angegeben:
Das Kommando *tar* gibt keine Meldungen aus.

w

(w - what) *tar* erwartet für jede Datei eine Bestätigung, bevor die Aktion ausgeführt wird. *tar xw* gibt aus:

```
x    dateiname:
```

Dabei bedeutet:

x	extract
dateiname	Name der Datei, die eingelesen werden soll.

tar erwartet nach dem Doppelpunkt : Ihre Eingabe. Nur wenn Sie mit y bejahen, führt *tar* die Aktion aus, sonst nicht.

w nicht angegeben:
tar führt die Aktion ohne Bestätigung aus.

L

tar verfolgt symbolische Verweise (symbolic links). Ohne dieses Attribut schreibt *tar* nur den symbolischen Verweis, aber nicht die zugehörigen Daten.

zahl

tar schreibt auf das Gerät, das im *zahl*-ten Eintrag in der Datei */etc/default/tar* angegeben ist. *zahl* muß im Bereich zwischen 0 und 9 liegen.

f␣gerätedatei
(f - file) *tar* erwartet die Eingabe einer Gerätedatei, von der gelesen werden soll, wenn sie nicht die Standard-Gerätedatei benutzen wollen (siehe *Format 1*).

datei
Name der Datei, die eingelesen werden soll, wenn sie im Archiv steht. Sie können auch mehrere Dateien angeben, getrennt durch Leerzeichen.

Ist die angegebene Datei ein Dateiverzeichnis, dann liest *tar* dieses Dateiverzeichnis mit allen darin enthaltenen Dateien und Unterdateiverzeichnissen ein.

Wenn der Dateiname Sonderzeichen der Shell enthält, müssen Sie diesen in Hochkommata '...' einschließen. Geben Sie die Hochkommata nicht an, so versucht die Shell, die angegebenen Sonderzeichen entsprechend der Dateinamen im aktuellen Dateiverzeichnis und nicht bezüglich der im Archiv vorhandenen Dateien zu ersetzen.

datei nicht angegeben:
tar liest das gesamte Archiv ein.

FEHLERMELDUNGEN

`cannot open /dev/mt0`

Sie haben *tar cf /dev/mt0 ...* angegeben und die Standard-Gerätedatei */dev/mt0* steht nicht zur Verfügung.

`tape write error`

Sie haben den Datenträger nicht im richtigen Laufwerk oder die falsche Gerätedatei angegeben.
Tritt diese Fehlermeldung erst auf, wenn *tar* bereits angefangen hat zu arbeiten, ist es möglich, daß die angegebenen Dateien nicht auf den Datenträger passen. Die letzte Datei ist dann nicht ordnungsgemäß gesichert.

`directory checksum error`

Der Datenträger ist leer oder eine Datei ist nicht ordnungsgemäß gesichert.

DATEIEN

/etc/default/tar
Datei mit einer Liste aus bis zu 10 Gerätedateien, die über das Attribut *zahl* ausgewählt werden können.

/dev
Verzeichnis aller verfügbaren Gerätedateien

UMGEBUNGSVARIABLEN

TAPE
hier können Sie eine Gerätedatei eintragen, die verwendet wird, falls andere Angaben fehlen

BEISPIELE

Archivieren auf Diskette

Alle im aktuellen Dateiverzeichnis und seinen Unterverzeichnissen enthaltenen Dateien sollen auf eine Diskette geschrieben werden. Es soll ausgegeben werden, welche Dateien archiviert werden. Ein eventuell vorhandenes Archiv soll überschrieben werden.

```
$ tar cvf /dev/dsk/f0t *
a  hans/briefe/privat/tante  9 blocks
a  hans/briefe/privat/onkel  6 blocks
a  hans/briefe/dienst/chef1  3 blocks
a  hans/buch/manuscript  158 blocks
a  hans/buch/fehler  16 blocks
a  hans/tools/programm  20 blocks
    ...
```

Archivieren auf Magnetbandkassette

Das Dateiverzeichnis *dokumente* soll auf Magnetbandkassette als zweites Archiv mit Blockungsfaktor 10 archiviert werden. Das Band wurde bereits positioniert. Nach dem Zugriff soll das Band nicht zurückgespult werden. Deshalb wird die Gerätedatei */dev/rmt/c0s0r* verwendet.

```
$ tar cbf 20 /dev/rmt/c0s0r dokumente
```

Archivieren auf Magnetband

Die Datei *brief* soll auf das Magnetband mit Blockungsfaktor 20, Schreibdichte 1600 cpi archiviert werden. Es soll danach zurückgespult werden.

```
$ tar cfb /dev/rmt0 20 brief
```

Inhalt aller Archive eines Bandes lesen

Es soll der Inhalt eines ganzen Bandes gelesen und in ausführlicher Form ausgegeben werden. Auf dem Band befinden sich drei Archive. Deshalb wird die Gerätedatei ohne Zurückspulen benutzt und das *tar*-Kommando mehrfach eingegeben.

```
$ tar tvf /dev/c0s0n; tar tvf /dev/c0s0n; tar tvf /dev/nrt4
tar: blocksize = 20
rw------- 33/1   1586   Sep 19  13:40 1988 hans/briefe/privat/tante
rw------- 33/1   2024   Sep 19  15:23 1988 hans/briefe/privat/onkel
rw-rw---- 33/1   1365   Oct 20  08:12 1988 hans/briefe/dienst/chef1
   ...
tar: blocksize = 20
rw-r--r-- 45/3   2345   Jan 18  13:20 1989 otto/test
rwxr-xr-x 45/3    800   Jan 27  12:50 1989 otto/programm
  ...
tar: blocksize = 20
rw------- 40/3   4567   Apr 10  07:58 1988 fritz/texte/kap1
rw------- 40/3   2367   Apr 10  08:50 1988 fritz/texte/kap2
  ...
```

Dateien von Archiv in aktuelles Dateiverzeichnis einlesen

Der Benutzer *hans* mit UID 108 und GID 10 liest das Dateiverzeichnis *fritz/texte* von einem Archiv auf 3,5 Zoll Diskette in sein aktuelles Dateiverzeichnis ein. Die Dateien im Archiv haben als Eigentümer die UID 40 und als Gruppe die GID 3. Der Benutzer *hans* möchte, daß er bei den eingelesenen Dateien als Eigentümer gesetzt wird.

```
$ tar xfo /dev/dsk/f0t fritz/texte
```

Mit dem Kommando *ls -l* können die Einträge für Eigentümer und Gruppe überprüft werden.

```
$ ls -l
...
./fritz:
drwx-------2 hans    other     236   Apr  2   8:15 texte

./fritz/texte:
-rw------- 1 hans    other    4567   Apr 10  07:58 kap1
-rw------- 1 hans    other    2367   Apr 10  08:50 kap2
```

Verwendung von tar in einer Pipe

Das Dateiverzeichnis /usr1/hans soll nach /usr/hans verschoben werden.

```
$ (cd /home1/hans; tar cf - .) | (cd /home/hans; tar xf -)
```

SIEHE AUCH

ar, *cpio*, *ls*, *format*, *mt*

tee Pipes zusammenfügen und Eingabe kopieren

tee überträgt Daten von der Standard-Eingabe auf die Standard-Ausgabe und kopiert die Daten gleichzeitig in die angegebene Datei.

```
tee[ -a][ -i][ 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) Unterbrechungssignale (siehe *kill*) werden 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 *datei*en angeben, schreibt *tee* seine vollständige Ausgabe in jede dieser Dateien.

datei nicht angegeben:
tee schreibt seine Ausgabe nur auf die Standard-Ausgabe.

BEISPIEL

Das Kommando *tee* in einer Pipe

```
$ (date; who) | tee sicher | wc -l
       5
$ cat sicher
Fri Aug 04 14:16:10 MET 1989
felix    tty12   Aug  4 10:37
jane     tty13   Aug  4 08:37
tina     tty03   Aug  4 11:54
jens     tty04   Aug  4 11:13
```

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: 5 (4 angemeldete Benutzer und 1 Zeile für das Datum).

telnet Benutzerschnittstelle zum TELNET-Protokoll

telnet eröffnet wie *rlogin* eine Sitzung an einem fernen Rechner. Es wird jedoch im Gegensatz zu *rlogin* ein von DARPA genormtes Protokoll verwendet. Dadurch können auch Sitzungen an fernen Rechnern eröffnet werden, an denen kein SINIX- oder UNIX-Betriebssystem läuft.

telnet kommuniziert mit einem anderen Rechner unter Verwendung des TELNET-Protokolls. Auf dem fernen Rechner muß der *telnet*-Serverprozeß gestartet sein.

Um eine *telnet*-Sitzung zu eröffen, brauchen Sie eine Benutzerkennung und das zugehörige Kennwort.

TELNET ist ein zeilenorientiertes Protokoll für virtuellen Terminal-Betrieb. Bei einigen Fremdrechnern ist daher zusätzlich eine Full-Screen-Emulation nötig, um bildschirmweise arbeiten zu können.

telnet kennt zwei Bearbeitungs-Modi:

- den Kommando-Modus und
- den Eingabe-Modus.

Der Kommando-Modus

Wenn *telnet* ohne Argumente aufgerufen wird, geht es in den Kommando-Modus. Die Eingabeaufforderung ist *telnet*>. In diesem Modus nimmt es die Kommandos, die weiter unten aufgelistet sind, entgegen und führt sie aus.

Wird *telnet* mit Argumenten aufgerufen, führt es ein *open*-Kommando gemäß den Argumenten durch und eröffnet eine Verbindung. Sie befinden sich dann im Eingabe-Modus.

Welche Kommandos an einem Rechner vorhanden sind, hängt von der jeweiligen Implementierung des TELNET-Protokolls ab. Mit dem Kommando *?* können Sie die implementierten Kommandos abfragen.

Der Eingabe-Modus

Ist eine Verbindung einmal eröffnet, geht *telnet* in den Eingabe-Modus. Das heißt, es wird eine Sitzung am fernen Rechner eröffnet und Ihr Bildschirm verhält sich so, als wäre er jetzt direkt am fernen Rechner angeschlossen.

Wenn Sie am fernen Rechner arbeiten wollen, müssen Sie natürlich Kenntnisse über das System des fernen Rechners haben.

Im Eingabemodus wird entweder zeichenweise oder zeilenweise übertragen. Welcher Übertragungsmodus verwendet wird, hängt davon ab, was der ferne Rechner unterstützt.

Bei zeichenweiser Übertragung wird jedes eingegebene Zeichen sofort an den fernen Rechner zur Weiterverarbeitung übertragen.

Bei zeilenweiser Übertragung wird für den eingegeben Text lokal das Echo ausgegeben und normalerweise werden nur vollständige Zeilen zum fernen Rechner geschickt. Mit dem lokalen Echo-Zeichen (standardmäßig CTRL e) kann das lokale Echo ein- und ausgeschaltet werden (z.B. um ein Kennwort einzugeben, das nicht am Bildschirm angezeigt wird).

Falls *localchars* eingeschaltet ist, werden die Zeichen *quit*, *intr* und *flush* lokal abgefangen und als Sequenzen des TELNET-Protokolls zum fernen Rechner gesendet. Dies ist bei zeilenweiser Übertragung standardmäßig der Fall. Mit den Optionen *toggle autoflush* und *toggle autosynch* können Sie einstellen, daß nach den entsprechenden Aktionen die Ein- bzw. Ausgabepuffer für die Datensichtstation geleert werden.

Um im Eingabe-Modus *telnet*-Kommandos abzusetzen, müssen Sie sie mit dem *telnet*-Fluchtsymbol CTRL] beginnen lassen. Wenn *telnet* im Kommando-Modus ist, können Sie die normalen Eingabekonventionen verwenden.

Vorsicht

Während Sie *telnet* benutzen, sollten Sie nicht mit *layers* arbeiten.
Bei *telnet* können Sie keine Flußsteuerung verwenden.

Aufruf von telnet

```
telnet[ rechner[ port]]
```

rechner
: ist der Name des Rechners, zu dem die Verbindung aufgebaut werden soll. Die Verbindung zu diesem Rechner wird aufgebaut und *telnet* geht in den Eingabe-Modus. Damit ist eine Sitzung am angegebenen Rechner eröffnet.

  Wird kein Rechner angegeben, geht *telnet* in den Kommando-Modus.

port
: ist die Portnummer des zu verwendenden Ports.

  Keine Portnummer angegeben:
  Es wird das voreingestellte Port (TCP-23) verwendet.

Handelt es sich beim fernen System um ein UNIX-System, so erwartet das ferne Login-Programm die Eingabe der gewünschten Benutzerkennung und das zugehörige Kennwort. Wird beides richtig eingegeben, geht *telnet* in den Eingabe-Modus.

Eine *telnet*-Sitzung wird mit dem Kommando *close* oder *quit* beendet. TELNET geht dann in den Kommando-Modus.

TELNET-Kommandos

Die folgenden Kommandos stehen im Kommando-Modus zur Verfügung. Sie müssen jeweils nur soviele Zeichen eingeben, wie nötig sind, um das Kommando eindeutig zu identifizieren.

Welche Kommandos an einem Rechner vorhanden sind, hängt von der jeweiligen Implementierung des TELNET-Protokolls ab. Mit dem Kommando *?* können Sie die implementierten Kommandos abfragen.

Funktionale Kommando-Übersicht

Verbindungsauf- und -abbau

open	Eröffnen einer Verbindung zu einem fernen Rechner
close	Schließen aller offenen Verbindungen und Beenden von TELNET.
quit	wie close

TELNET steuern

mode	zwischen zeichenweiser und zeilenweiser Übertragung umschalten
send	Sonderzeichen zum fernen Rechner übertragen
set	*telnet*-Variablen ändern oder die zugehörige Funktion ganz abschalten
toggle	verschiedene Funktionen ein- oder ausschalten
z	*telnet* suspendieren

Informationen über TELNET

?	Über TELNET-Kommandos informieren
status	TELNET Status-Informationen ausgeben
display	einige oder alle Einstellungen anzeigen

TELNET-Kommandos alphabetisch

close

schließt eine *telnet*-Sitzung und kehrt in den Kommando-Modus zurück. Das Zeichen EOF im Kommandomodus bewirkt das selbe wie *close*. Das Zeichen EOF wird jedoch nur erkannt und zum fernen Rechner gesendet, wenn es das erste Zeichen der Zeile ist.

display[␣argument...]
Die Werte der als Argumente angegebenen Einstellungen werden angezeigt.

Kein Argument angegeben:
Es werden alle Einstellungen angezeigt.

mode␣typ
schaltet zwischen zeilenweiser und zeichenweiser Übertragung um. *typ* hat für zeilenweise Übertragung den Wert *line* und für zeichenweise Übertragung den Wert *character*. Es wird beim fernen Rechner angefragt, ob der gewünschte Modus erlaubt ist. Falls der ferne Rechner diesen Modus unterstützt, wird der Modus eingestellt.

open␣rechner␣[port]
Eröffnet eine Verbindung zu dem angegebenen Rechner und geht in den Eingabe-Modus.

rechner
ist der Name oder die Internet-Adresse des Rechners, mit dem eine Verbindung aufgebaut werden soll.
TELNET überprüft nicht, ob der Rechner in der Datei */etc/hosts* oder der Netzverwaltungs-Datei *hosts* eingetragen ist.
Falls der Rechner dort nicht eingetragen ist, muß er über seine Internet-Adresse angesprochen werden.

port
ist die Portnummer, über die die Verbindung aufgebaut werden soll.

Portnummer nicht angegeben:
TELNET versucht, mit einem *telnet*-Server über das voreingestellte Port (TCP-23) in Kontakt zu kommen.

quit
wie close

send␣argument...
Ein oder mehrere Sonderzeichen werden zum fernen Rechner gesendet. Die folgenden Argumente können angegeben werden, wobei auch mehrere Argumente zusammen zuläßig sind:

escape
Das eingestellte TELNET-Fluchtzeichen wird gesendet. (standardmäßig CTRL]).

synch
Die TELNET-Sequenz SYNCH wird gesendet. Diese Sequenz löscht auf dem fernen Rechner die bereits eingetippte, aber noch nicht gelesene Eingabe.

Die Sequenz wird als "urgent data" im TCP-Protokoll gesendet. Falls der ferne Rechner mit dem System 4.2 BSD arbeitet, kann es sein, daß die Sequenz nicht korrekt arbeitet. In diesem Fall wird manchmal das Zeichen *r* auf dem Bildschirm ausgegeben.

brk
Die TELNET-Sequenz BRK (Break) wird gesendet. Sie hat unter Umständen Bedeutung für den fernen Rechner.

ip
Die TELNET-Sequenz IP (Interupt Process) wird gesendet. Dadurch wird der augenblicklich laufende Prozeß auf dem fernen Rechner abgebrochen.

ao
Die TELNET-Sequenz AO (Abort Output) wird gesendet. Dadurch werden die Ausgabepuffer des fernen Rechners geleert und der Inhalt auf Ihrem Bildschirm angezeigt.

ayt
Die TELNET-Sequenz AYT (Are You There) wird gesendet. Es liegt am fernen Rechner, ob darauf eine Antwort erfolgt oder nicht.

ec
Die TELNET-Sequenz EC (Erase Character) wird gesendet. Diese Sequenz löscht das letzte eingegebene Zeichen.

el
Die TELNET-Sequenz EL (Erase Line) wird gesendet. Dadurch wird der ferne Rechner angewiesen, die zuletzt eingegebene Zeile zu löschen.

ga
Die TELNET-Sequenz GA (Go Ahead) wird gesendet. Diese Sequenz hat normalerweise keine Bedeutung für den fernen Rechner.

nop
Die TELNET-Sequenz NOP (No Operation) wird gesendet.

?
gibt Hilfe-Information für das Kommando *send* aus.

set␣*argument*␣*wert*

weist einer *telnet*-Variablen einen Wert zu. Der besondere Wert *off* schaltet die zugehörige Funktion aus. Die Einstellungen der Variablen können mit dem Kommando *display* abgefragt werden. Die möglichen Argumente (Variablen) sind:

echo

schaltet das Echo auf dem lokalen System bei zeilenweiser Übertragung ein bzw. aus. (z.B. um ein Kennwort einzugeben, das nicht am Bildschirm angezeigt wird). Bei manchen fernen Systemen muß das Echo bei zeilenweiser Übertragung manuell ausgeschaltet werden.

Voreinstellung:
Das Zeichen *echo* ist CTRL e.

escape

ist das *telnet*-Fluchtsymbol. Mit der Eingabe von CTRL und dem definierten Fluchtsymbol wechselt TELNET vom Eingabe-Modus einer eröffneten Sitzung in den TELNET-Kommandomodus.
Für *Fluchtsymbol* kann jedes abdruckbare Zeichen verwendet werden.

Voreinstellung:
Das Zeichen *escape* ist CTRL].

interrupt

Wenn *localchars* eingeschaltet ist (siehe *toggle localchars*) und Sie geben das *interrupt*-Zeichen ein, so wird die TELNET-Sequenz IP (siehe *send ip*) zum fernen Rechner gesendet.

Voreinstellung:
Das Zeichen *interrupt* ist das Zeichen *intr* Ihrer Datensichtstation (siehe *stty*).

quit

Wenn *localchars* eingeschaltet ist (siehe *toggle localchars*) und Sie geben das *quit*-Zeichen ein, so wird die TELNET-Sequenz BRK (siehe *send brk*) zum fernen Rechner gesendet.

Voreinstellung:
Das Zeichen *quit* ist das Zeichen *quit* Ihrer Datensichtstation (siehe *stty*)

flushoutput

Wenn *localchars* eingeschaltet ist (siehe *toggle localchars*) und Sie geben das *flushoutput*-Zeichen ein, so wird die TELNET-Sequenz AO (siehe *send ao*) zum fernen Rechner gesendet.

Voreinstellung:
Das Zeichen *flushoutput* ist das Zeichen *flush* Ihrer Datensichtstation (siehe *stty*)

erase
Wenn *localchars* eingeschaltet ist (siehe *toggle localchars*) und zeichenweise übertragen wird, so wird bei Eingabe des *erase*-Zeichens die TELNET-Sequenz EC (siehe *send ec*) zum fernen Rechner gesendet.

Voreinstellung:
Das Zeichen *erase* ist das Zeichen *erase* Ihrer Datensichtstation (siehe *stty*)

kill
Wenn *localchars* eingeschaltet ist (siehe *toggle localchars*) und zeichenweise übertragen wird, so wird bei Eingabe des *kill*-Zeichens die TELNET-Sequenz EL (siehe *send el*) zum fernen Rechner gesendet.

Voreinstellung:
Das Zeichen *kill* ist das Zeichen *kill* Ihrer Datensichtstation (siehe *stty*)

eof
Bei zeilenweiser Übertragung wird dieses Zeichen sofort zum fernen Rechner übertragen, wenn es als erstes Zeichen einer Zeile eingegeben wird.

Voreinstellung:
Das Zeichen *eof* ist das Zeichen *eof* Ihrer Datensichtstation (siehe *stty*)

status
Das Kommando zeigt den aktuellen Status von TELNET an.

Es wird ausgegeben, ob und zu welchem Rechner eine Verbindung besteht, und was das aktuelle Fluchtsymbol ist.

toggle␣argument...

Damit können Sie einstellen, wie *telnet* auf verschiedene Ereignisse reagiert. Jedes Argument steht für eine Funktion, die entweder ein- oder ausgeschaltet wird. Es wird jedesmal in den jeweils anderen Zustand umgeschaltet. Die augenblicklichen Einstellungen können Sie mit dem Kommando *display* abfragen. Folgende Argumente sind zuläßig und können auch zusammen angegeben werden:

autoflush
Falls sowohl *autoflush* als auch *localchars* eingeschaltet sind und die Zeichen *ao*, *intr* oder *quit* erkannt werden (die in TELNET-Sequenzen umgewandelt werden, siehe *set*), so werden so lange keine Daten mehr auf dem Bildschirm ausgegeben, bis der ferne Rechner bestätigt, daß diese Sequenzen verarbeitet wurden.

Voreinstellung:
autoflush ist eingeschaltet, falls nicht *stty noflsh* eingestellt wurde, andernfalls ausgeschaltet.

autosynch
Falls sowohl *autosynch* als auch *localchars* eingeschaltet sind und eines der Zeichen *intr* oder *quit* eingegeben wird (siehe *set*), so wird nach der erzeugten TELNET-Sequenz noch die TELNET-Sequenz SYNCH gesendet. Damit soll der ferne Rechner veranlaßt werden, jede vorherige Eingabe zu löschen, bis die beiden TELNET-Sequenzen gelesen und verarbeitet wurden.

Voreinstellung:
autosynch ist ausgeschaltet.

crmod
veranlaßt, daß der ferne Rechner beim Übertragen von Daten am Zeilenende eine Carriage-Return-Zeichenfolge einfügt.

Standardmäßig werden keine Carriage-Return-Zeichenfolgen eingefügt.

Die Funktion wird durch einmalige Eingabe des Kommandos eingeschaltet. Bei einer erneuten Eingabe des Kommandos wird sie wieder ausgeschaltet.

Vorsicht

Sendet das ferne System standardmäßig Carriage-Return-Zeichenfolgen (wie z.B. SINIX) und die Funktion *crmod* ist eingeschaltet, führt das zu einer Verdoppelung des Zeilenvorschubs.

Voreinstellung:
crmod ist ausgeschaltet.

debug
Die Unterstützung für die Fehlersuche auf Ebene der Socket-Schnittstelle wird ein- bzw. ausgeschaltet. Dies ist nur für den Systemverwalter hilfreich.

Voreinstellung:
debug ist ausgeschaltet.

localchars
Wenn *localchars* eingeschaltet ist, werden die Zeichen *flush*, *interrupt*, *quit*, *erase* und *kill* (siehe *set*) lokal erkannt und in die entsprechenden Steuersequenzen des TELNET-Protokolls umgewandelt. Diese sind *ao*, *ip*, *brk*, *ec* und *el* (siehe *send*).

Voreinstellung:

Bei zeilenweiser Übertragung ist *localchars* eingeschaltet, bei zeichenweiser Übertragung ausgeschaltet.

netdata
Die Anzeige aller Netzwerk-Daten wird ein- bzw. ausgeschaltet. Die Anzeige erfolgt in hexadezimalem Format.

Voreinstellung:
netdata ist ausgeschaltet.

options
schaltet das Sichtbarmachen der TELNET-Optionen, die gerade verarbeitet werden, ein bzw. aus. TELNET-Optionen werden normalerweise während des Verbindungsaufbaus mit dem Server ausgetauscht.

Wenn der Modus eingeschaltet ist, werden alle vereinbarten Optionen angezeigt. Optionen, die von TELNET gesendet werden, werden als SENT angezeigt, während Optionen, die von TELNET empfangen werden, als RCVD angezeigt werden.

Voreinstellung:
options ist ausgeschaltet.

?
Alle zulässigen Argumente für das Kommando *toggle* werden angezeigt.

z
telnet suspendieren. Dieses Kommando arbeitet nur, wenn Sie einen Kommandointerpreter mit Ablaufsteuerung (job control) verwenden (z.B. *sh*).

?[␣kommando]
Mit *?* können Sie sich Hilfeinformationen ausgeben lassen. Ohne Angabe von Argumenten gibt TELNET eine Übersichtsinformation über die implementierten TELNET-Kommandos als Hilfe aus. Wenn *kommando* angegeben ist, gibt es die zu diesem Kommando verfügbaren Informationen aus.

SIEHE AUCH

rlogin, *sh*, *stty*, *inet*

test Bedingungen prüfen

Das in die Bourne-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.

Als Ergebnis liefert *test* zurück:

- Ende-Status 0 (wahr), falls die Bedingung erfüllt ist.
- Ende-Status 1 (falsch), falls die Bedingung nicht erfüllt ist oder falls Sie die Bedingung nicht vollständig angegeben haben. Den gleichen Ende-Status erhalten Sie, wenn Sie keine Bedingung angeben.

Abhängig vom Ende-Status können Sie unterschiedliche Kommandos ausführen, Schleifen abbrechen usw.

Für das eingebaute *sh*-Kommando *test* gibt es zwei Schreibweisen (siehe Syntax). Die Wirkung ist dieselbe.

```
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 *Bedingungen verknüpfen*).
Folgende Bedingungen prüft *test*:

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.

-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 - sticky bit) wahr, wenn *datei* existiert und das sticky- oder t-Bit gesetzt ist.

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

-t[␣dateikennzahl]
(t - terminal) wahr, wenn die angegebene *dateikennzahl* einer Datensichtstation zugeordnet ist.

dateikennzahl
für *dateikennzahl* können Sie angeben:

0 für Standard-Eingabe
1 für Standard-Ausgabe
2 für Standard-Fehlerausgabe

dateikennzahl nicht angegeben:
Standardmäßig wird 1 eingesetzt, also Standard-Ausgabe.

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 Hochkommata '', setzt *test* den Namen Ihres aktuellen Dateiverzeichnisses ein.

Wenn Sie *datei* nicht angeben, gibt *test* eine Fehlermeldung aus und bricht mit Ende-Status 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 Hochkommata ein.

Die leere Zeichenkette geben Sie an durch zwei aufeinanderfolgende Anführungszeichen oder Hochkommata. Wenn Sie keine Zeichenkette angeben, gibt *test* eine Fehlermeldung aus und bricht mit Ende-Status 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 sollten Sie in Anführungszeichen einschließen.

-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, daß einer Shell-Variablen kein Wert zugewiesen ist. Den entsprechenden Shell-Parameter sollten Sie in Anführungszeichen einschließen.

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, sollten Sie diese in Anführungszeichen einschließen.

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 Ende-Status 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 Ende-Status 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.

ENDE-STATUS

0 der angegebene Ausdruck ist wahr

1 der angegebene Ausdruck ist falsch oder Sie haben eine Bedingung nicht vollständig angegeben oder Sie haben überhaupt keine Bedingung angegeben.

FEHLERMELDUNG

```
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.
Deshalb sollten Sie Shell-Parameter immer in Anführungszeichen einschließen. Andernfalls fehlt ein Argument, wenn die entsprechende Shell-Variable nicht definiert ist.

BEISPIELE

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.

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.

3. Es soll festgestellt werden, ob der Shell-Variablen TAPE ein Wert zugewiesen ist. Dazu gibt es mehrere Möglichkeiten:

   (a)
```
if [ ! -z "$TAPE" ]
   then echo Der Variablen TAPE ist ein Wert zugewiesen
   else ....
fi
```

   (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, sh

tftp
Einfaches Dateiübertragungs-Programm

tftp gestattet Ihnen das Übertragen von Dateien vom und zum fernen Rechner.

```
tftp[␣ferner␣rechner]
```

tftp ist die Schnittstelle zum TFTP-Server am fernen Rechner. Den Rechner rufen Sie bereits in der Kommandozeile auf, wobei *ferner-rechner* bei weiteren Übertragungen als Standardwert aufgefaßt wird (siehe *tftp*-Kommando *connect*). Wenn Sie *tftp* eingeben, läuft das Programm und gibt das Aufforderungszeichen *tftp*>. Sie können dann die unten beschriebenen *tftp*-Kommandos eingeben.

tftp-Kommandos

ascii
Setzt den Dateiübertragungs-Typ auf ASCII-Code.

binary
Setzt den Dateiübertragungs-Typ auf Binär-Code.

connect␣ferner-rechner[␣anschlußnummer]
Verbindet den lokalen Rechner mit dem angegebenen fernen TFTP-Server *ferner-rechner*. Wenn Sie den Zugang zum fernen Rechner mit *anschlußnummer* angeben, wird über diese Nummer die Verbindung aufgebaut.
Anders als bei FTP, stellt TFTP keine wirkliche Verbindung her, sondern merkt sich den von Ihnen angegebenen Rechner für Dateiübertragungen.
Der ferne Rechner kann daher auch erst mit dem Sende- oder Empfangsauftrag (siehe *put* und *get*) angegeben werden.

get␣datei
get␣ferne-datei␣lokale-datei
get␣datei1␣datei2␣datei3...dateiN
Holt eine oder mehrere (mindestens drei) Dateien vom angebenen fernen Rechner. Die Adresse des fernen Rechners können Sie auf zwei verschiedene Arten angeben:

- nur den oder die Dateinamen, wenn Sie den fernen Rechner bereits angegeben haben.
- als Zeichenkette *ferner-rechner:dateiname*, wenn Sie den fernen Rechner vorher noch nicht angegeben haben. In diesem Fall setzen Sie zugleich den Standardwert *ferner-rechner* für weitere Dateiübertragungen.

mode übertragungs-modus
Setzt den Übertragungs-Modus auf *ascii* oder *binary*. Der Voreinstellwert ist *ascii*.

put datei
put lokale-datei ferne-datei
put datei1 datei2...dateiN fernes-dvz
Sendet eine oder mehrere Dateien an den angegebenen fernen Rechner. Die Adresse des fernen Rechners können Sie auf zwei verschiedene Arten angeben:

- nur den oder die Dateinamen, wenn Sie den fernen Rechner bereits angegeben haben.
- als Zeichenkette *ferner-rechner:dateiname*, wenn Sie den fernen Rechner vorher noch nicht angegeben haben. In diesem Fall setzen Sie zugleich den Standardwert *ferner-rechner* für weitere Dateiübertragungen.

Wenn Sie nur das ferne Dateiverzeichnis als Adresse angeben, muß der angesprochene Rechner unter UNIX-System laufen.

quit
Beendet *tftp*. Die gleiche Wirkung hat END.

rexmt zeitwahl
Setzt den Zeitpunkt *zeitwahl* in Sekunden fest, an dem eine Dateiübertragung blockweise gestartet werden soll.

status
Zeigt den aktuellen *tftp*-Status an.

timeout absolute-zeitwahl
Setzt den absoluten Zeitpunkt in Sekunden fest, an dem eine Dateiübertragung stattfinden soll.

trace
Schaltet die Ablaufunterbrechung ein bzw. aus (Voreinstellwert).

verbose
Schaltet den Protokoll-Modus ein bzw. aus. Voreinstellwert: aus.

?[kommando]
Gibt Hilfestellungen zu den *tftp*-Kommandos.

Hinweis

Da Sie durch *tftp* nicht wirklich mit dem fernen Rechner verbunden sind, kann es vorkommen, daß die Dateiübertragung scheitert.

BEISPIEL

```
$ tftp ↵
tftp > connect fernweh ↵
Passwort eingeben: holiday ↵
tftp > status ↵
Mode: netascii Verbose: off Tracing: off
Rexmt-interval: 5 seconds Max-timeout: 25 seconds
tftp > timeout 14:30 ↵
tftp > put datei1 ↵
tftp > quit ↵
$
```

SIEHE AUCH

ftp

time Laufzeit eines Kommandos messen

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 Zehntel Sekunden steht.

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.

```
time␣prog[␣arg]...
```

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*.

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        2.0
user        0.9
sys         0.8
```

SIEHE AUCH

times, *timex*
times() [14]

times
Gesamt-Laufzeit der bisher gestarteten Prozesse ausgeben

Das in die Bourne-Shell *sh* eingebaute Kommando *times* gibt aus, wieviel Zeit die Prozesse, die die aktuelle Shell bisher gestartet hat, insgesamt verbraucht haben. Die Ausgabe ist aufgeschlüsselt nach Benutzer-Zeit und System-Zeit, angegeben in Minuten (m) und Sekunden (s).

Die Benutzer-Zeit ist die Zeit, die die Prozesse in der Benutzer-Phase verbraucht haben. Die System-Zeit 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*.

times

BEISPIEL

Die Gesamt-Laufzeit aller Prozesse abfragen, die die aktuelle Shell bisher gestartet hat:

```
$ times
0m15s 0m24s
```

Die Prozesse haben in der Benutzer-Phase 15 Sekunden und in der System-Phase 24 Sekunden verbraucht.

SIEHE AUCH

time, timex
times() [14]

timex
Laufzeit eines Kommandos messen, Prozessdaten und Systemaktivitäten anzeigen (time execution)

timex veranlaßt die Ausführung eines Programms oder einer Shell-Prozedur und mißt die Zeiten, die für die Ausführung verbraucht werden. Die gemessenen Rechenzeiten werden als Sekundenwerte auf die Standard-Fehlerausgabe geschrieben.

timex gibt die folgenden Werte aus: 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.

Über Optionen können zusätzliche Prozeßinformationen über das ausgeführte Kommando und seine Sohnprozesse angefordert werden, sowie Daten über die Gesamtauslastung des Systems zur Laufzeit des Kommandos.

```
timex[␣option] . . prog[␣arg] . . .
```

Keine Option angegeben
timex führt das angegebene Kommando *prog* aus und zeigt die Laufzeiten an, die zur Ausführung des Kommandos benötigt wurden.

option

-p
(p - process) Anzeige von Prozeßinformationen über *prog* und seine Sohnprozesse. Diese Option wird nur ausgeführt, falls das Process Accounting aktiviert ist.
Die Optionen *f*, *h*, *k*, *m*, *r* und *t* sind Unteroptionen zu *-p* und werden nur in Kombination mit *-p* wirksam.

-o
Anzeige der Gesamtanzahl der von *prog* und seinen Sohnprozessen gelesenen oder geschriebenen Blöcken. Die Anzahl der übertragenen Zeichen wird ebenfalls ausgegeben.
Diese Option wird nur wirksam, wenn das Process Accounting aktiviert ist.

-s
Anzeige der Gesamtauslastung des Systems während der Laufzeit von *prog*. Die Ausgabe entspricht dem *sar* Kommando.

Die folgenden Optionen werden nur in Kombination mit der *-p* Option wirksam:

-f
Die Ausgabe enthält zusätzliche Spalten für die fork- und exec-Flags, sowie für den Endestatus des Systems.

-h
(h - hog factor) Anstatt des gemittelten Speicherbedarfs des Prozesses wird der Anteil der gesamten zur Verfügung stehenden CPU Zeit angegeben, der vom Prozeß zur Laufzeit verbraucht wurde. Dieser Wert wird als Monopolisierungsfaktor (hog factor) bezeichnet und wird folgendermaßen berechnet: Gesamt-CPU-Zeit/verbrauchte Zeit.

-k
Zeigt kumulierte Speichergröße in Kilobytes an, den ein Prozeß insgesamt pro Laufzeit-Minute beansprucht hat.

-m
(mean core size) Der Mittelwert des Hauptspeicherbedarfs wird angezeigt.
Diese Option ist standardmäßig eingestellt.

-r
Ausgabe des CPU Faktors (user-time/(system-time + user-time)).

-t
Die CPU-Zeit, die das System verbraucht, wird getrennt von der Zeit, die der Benutzer verbraucht, aufgelistet. Die Anzahl der gelesenen und geschriebenen Blöcke, und die Anzahl der übertragenen Zeichen werden immer mit angegeben.

prog
Programm oder Shell-Prozedur, die von *timex* ausgeführt wird.

arg
Argument für das Programm oder die Shell-Prozedur *prog*. Ob und wieviele Argumente angegeben werden, ist von *prog* abhängig.

SIEHE AUCH

time, *sar*
times() [14]

touch Änderungs- und Zugriffszeiten aktualisieren

touch setzt den Zeitpunkt der letzten Änderung bzw. des letzten Zugriffs für Dateien auf das aktuelle oder ein gewünschtes Datum.

```
touch[␣option] ...[␣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)

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.

ENDE-STATUS

Wenn *touch* im richtigen Format aufgerufen wurde, dann ist der Ende-Status gleich der Anzahl der Dateien, deren Zeiten nicht geändert werden konnten; hierbei werden auch die Dateien, die nicht vorhanden waren und nicht angelegt wurden, mitgezählt.

FEHLERMELDUNG

```
date: bad conversion
```

Sie haben ein unzulässiges Datum angegeben, z.B. 13010000.

BEISPIELE

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.

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() [14]

tput
Datensichtstation initialisieren oder Datenbasis terminfo abfragen

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).

```
tput[␣-Ttyp]␣capname[␣parameter]...                    Format 1
tput␣-S␣<<!
>> ␣capname[␣parameter]...
>> !                                                   Format 2
tput[␣-Ttyp]␣init                                      Format 3
tput[␣-Ttyp]␣reset                                     Format 4
tput[␣-Ttyp]␣longname                                  Format 5
```

Format 1: Eine Eigenschaft einer Datensichtstation ausgeben

tput gibt Eigenschaften einer Datensichtstation aus, die in der Datenbasis *terminfo* festgelegt sind.

tput[␣-Ttyp]␣capname[␣parameter]...

-Ttyp
: Für *typ* geben Sie den Typ der Datensichtstation an, dessen Eigenschaften Sie abfragen wollen. Die Shell-Variablen *LINES* und *COLUMNS* bleiben unverändert, sowie die Größe des Fensters, das mit dem Kommando *layers* erzeugt wurde.

  -Ttyp nicht angegeben:
  Für *typ* wird der Wert der Umgebungsvariablen TERM eingesetzt.

capname
: Für *capname* geben Sie den in der Datenbasis *terminfo* verwendeten Kurznamen der Datensichtstationseigenschaft (capability name) an, deren Wert Sie ausgeben lassen wollen. Wenn die zu diesem Kurznamen gehörende Eigenschaft vom Typ Zeichenkette ist, gibt *tput* eine Zeichenkette aus. Wenn die zu diesem Kurznamen gehörende Eigenschaft vom Typ ganze Zahl ist, gibt *tput* eine ganze Zahl aus. Wenn die zu diesem Kurznamen gehörende Eigenschaft vom Typ Boolescher Wert ist, liefert *tput* nur einen Ende-Status zurück und zwar 0 für *wahr*, wenn die Datensichtstation die entsprechende Eigenschaft hat, und 1 für *falsch*, wenn die Datensichtstation die Eigenschaft nicht hat.

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␣<<!
>> ␣capname[␣parameter]...
>> !

Mit einem Aufruf von *tput* können mehrere Eigenschaften der aktuellen Datensichtstation ausgegeben werden. Sie übergeben die Eigenschaften nicht von der Kommandozeile, sondern von der Standardeingabe (siehe *Beispiel*). Es ist nur jeweils eine Datensichtstationseigenschaft pro Zeile zulässig. Die Bedeutung der Werte 0 und 1 als Ende-Status hat sich geändert. *capname* und *parameter* sind - ähnlich wie bei Format 1 - Eigenschaften und Parameter einer Datensichtstation.

Format 3: Datensichtstation initialisieren

tput[␣-Ttyp]␣init

Wenn die Datenbasis *terminfo* existiert und einen Eintrag für die aktuell benutzte Datensichtstation enthält, initialisiert *tput* diese entsprechend dem Terminaltyp, der bei *-Ttyp* angegeben ist. Im einzelnen führt *tput* folgende Aktionen durch:

- Die Zeichenketten für die Initialisierung der Datensichtstation (*is1*, *is2*, *is3*, *if*, *ifprog*) 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 die Aktion ohne Fehlermeldung.

-Ttyp
Für *typ* geben Sie den Typ der Datensichtstation an, mit dessen Eigenschaften Sie Ihre Datensichtstation initialisieren wollen.

-Ttyp nicht angegeben:
Für *typ* wird der Wert der Umgebungsvariablen TERM eingesetzt.

Format 4: Datensichtstation zurücksetzen

tput[␣-Ttyp]␣reset

Statt der Zeichenketten für die Initialisierung werden die Zeichenketten für das Zurücksetzen der Datensichtstation ausgegeben (*rs1*, *rs2*, *rs2*, *rf*). Falls dafür keine Einträge vorhanden sind, jedoch für die Initialisierung, werden die Zeichenketten für die Initialisierung ausgegeben. Ansonsten verhält sich *reset* wie *init*.

-Ttyp
Für *typ* geben Sie den Typ der Datensichtstation an, mit dessen Eigenschaften Sie Ihre Datensichtstation zurücksetzen wollen.

-Ttyp nicht angegeben:
Für *typ* wird der Wert der Umgebungsvariablen TERM eingesetzt.

Format 5: Ausführlichen Namen einer Datensichtstation ausgeben

tput[␣-Ttyp]␣longname

Wenn die Datenbasis *terminfo* existiert und einen Eintrag für die benutzte Datensichtstation (vgl. Format 1, Option *-Ttyp*) 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 Datenbasis *terminfo*.

ENDE-STATUS

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 der Standardausgabe wird nichts ausgegeben).

capname vom Typ Boolescher Wert oder Zeichenkette und Option *-S*:

0 wenn alle Zeilen erfolgreich abgearbeitet werden konnten.
1 Ende-Status 1 kann niemals vorkommen.

capname vom Typ ganze Zahl:

0 Ende-Status immer 0. Anhand der Standardausgabe kann entschieden werden, ob *capname* für den angegebenen Datensichtstationstyp definiert ist oder nicht. Der Wert -1 bedeutet, daß *capname* nicht definiert ist.

Fehlersituationen werden durch einen Ende-Status 2, 3 oder 4 angezeigt.

FEHLERMELDUNGEN

Je nach Ende-Status gibt *tput* folgende Fehlermeldungen aus:

`usage error`

Falscher Aufruf, Ende-Status 2.

`tput: unknown terminal type`

Unbekannter Datensichtstationstyp *type* oder keine *terminfo* Datenbasis vorhanden, Ende-Status 3

`tput: unknown terminfo capability capname`

Unbekannte Datensichtstationseigenschaft *capname*, Ende-Status 4.

UMGEBUNGSVARIABLEN

TERM
Standard-Wert für den Typ der Datensichtstation, der eingesetzt wird, wenn *-Ttyp* nicht angegeben ist.

DATEIEN

*/usr/share/lib/terminfo/?/**
Dateiverzeichnis mit der übersetzten Datenbasis zur Beschreibung der Eigenschaften der 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

BEISPIELE

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 Datensichttsationstyp gilt.

2. Anzahl der Spalten der aktuellen Datensichtstation ausgeben

```
$ tput cols
80
```

3. Ist die aktuelle Datensichtstation ein Hardcopy-Terminal?

```
$ tput hc
$ echo $?
1
```

hc ist vom Typ Boolescher Wert, *tput* liefert daher nur einen Ende-Status zurück. Diesen Ende-Status fragen Sie mit *echo $?* ab. Der Ende-Status ist 1, die aktuelle Datensichtstation ist also kein Hardcopy-Terminal.

4. Mehrere Eigenschaften der Datensichtstation mit einem *tput*-Aufruf ausgeben

```
$ tput -S <<!
> clear
> cup 10 10
> bold
> !
```

Der Bildschirm wird gelöscht, die Schreibmarke auf Position 10, 10 gebracht und verstärkte Darstellung eingeschaltet. Ein einzelnes Ausrufezeichen in einer eigenen Zeile beendet die Liste.

SIEHE AUCH

stty, clear, tabs
profile, terminfo [5]

Leitfaden für Programmierer [8]
Referenzhandbuch für Programmierer [14]

tr Zeichen ersetzen oder löschen (transliterate)

tr liest einen Eingabetext von der Standard-Eingabe, ersetzt (Format 1) oder löscht (Format 2) einzelne Zeichen und schreibt das Ergebnis auf die Standard-Ausgabe.

```
tr␣[-c][␣-s][␣zeichenkette1[␣zeichenkette2]]                    Format 1
tr␣-d[␣-c][␣-s][␣zeichenkette1[␣zeichenkette2]]                 Format 2
```

Wenn Sie bei einem Aufruf mehrere Optionen angeben, dann müssen Sie diese in beiden Formaten mit einem führenden Bindestrich und ohne Leerzeichen aneinanderreihen, z.B. -cs oder -dc.

Format 1: Zeichen ersetzen

tr[␣**-c**][␣**-s**][␣zeichenkette1[␣zeichenkette2]]

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*).

-c

(c - complement) *zeichenkette1* wird bezüglich des ASCII-Zeichensatzes (mit oktalen Werten 001 bis 377) komplementiert. Die komplementierte *zeichenkette1* enthält dann alle Zeichen des ASCII-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* jede Folge von gleichen Zeichen, die in *zeichenkette2* vorkommen, zu einem Zeichen (siehe *Beispiel 4*).

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 Hochkommata '...' einschließen oder die Zeichen mit einem vorangestellten Gegenschrägstrich \ entwerten.

Sie können jedes Zeichen folgendermaßen oktal angeben: \\oktalzahl oder '\oktalzahl' oder "\oktalzahl", wobei *oktalzahl* eine ein-, zwei- oder dreistellige Oktalzahl ist. Den Gegenschrägstrich müssen Sie, wie angegeben, entwerten, damit die Ziffern als Oktalzahl erkannt werden.

tr bearbeitet das Zeichen NUL (oktal 000) in den angegebenen Zeichenketten nicht; kommt das Zeichen NUL als Eingabezeichen vor, wird es stets gelöscht.

Zeichenbereiche oder Folgen von gleichen Zeichen können Sie in beiden Zeichenketten wie folgt abkürzen; die eckigen Klammern [und] müssen Sie hier explizit angeben!

[a-z]
steht für die Folge der ASCII-Zeichen von *a* bis *z* einschließlich.

[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.

Format 2: Zeichen löschen

tr␣-d[␣-c][␣-s][␣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 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.

BEISPIELE

Zeichen ersetzen (Format 1)

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 Hochkommata sind notwendig, da die Zeichenketten Shell-Sonderzeichen enthalten.

Zeichen löschen (Format 2)

2. 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

Mit dem in die Bourne-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 (Format 1):
  - 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. Jede Shell ignoriert immer das Signal SIGTERM (15). Dieses Verhalten können Sie mit *trap* nicht ändern.
  - 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.

```
trap[␣kommandoliste]␣signalnummer␣...          Format 1
trap                                           Format 2
```

Format 1: Signalbehandlung ändern

trap[␣kommandoliste]␣signalnummer␣...

kommandoliste
: legt fest, wie die Shell auf die nachfolgend angegebenen Signale reagieren soll, ob sie
  - Kommandos ausführen soll, wenn das angegebene Signal eintrifft,
  - das angegebene Signal ignorieren soll,
  - 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 Hochkommata '...' 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 *Signalbehandlung in der Shell*).

Beachten Sie dabei, daß die Shell die angegebenen Kommandos zweimal interpretiert:

- Das erste Mal, wenn die Shell *trap* ausführt, und
- das zweite Mal, wenn das entsprechende Signal eintrifft und die Shell die vereinbarten Kommandos ausführt.

Deshalb haben Hochkommata 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.
Das Signal SIGTERM (15) wird von jeder Shell ignoriert. Dieses Verhalten können Sie mit *trap* nicht verändern.

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 *Signalbehandlung in der Shell*).

kommandoliste nicht angegeben:
Die Shell reagiert auf die angegebenen Signale wieder standardmäßig (siehe *Signalbehandlung in der Shell*).

signalnummer
Nummer des Signals, auf das die Shell wie angegeben reagieren soll (siehe *signal()* [14]). Sie können mehrere Signalnummern angeben, jeweils getrennt durch Leerzeichen. Sobald eines dieser Signale eintrifft, wird *kommandoliste* ausgeführt.

Sinnvolle Angaben für die Shell sind:

0 Beendigung der Shell (EOF)
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.

Folgende Angaben haben keine Wirkung:

1 SIGHUP wird immer ignoriert

9 SIGKILL führt immer zum Abbruch

15 SIGTERM wird immer ignoriert

Die Angabe 11 bzw. SIGSEGV ist bei *trap* nicht erlaubt.

Format 2: Geänderte Signalbehandlung ausgeben

trap

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 *Signalbehandlung in der Shell*).

Signalbehandlung in der Shell

Ein Prozeß kann jederzeit ein Signal erhalten, das er entweder selbst, das 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 Shell sind folgende Signale von Bedeutung:

Signal-Nummer	Signal	Ursache
1	SIGHUP	Verbindung zur Datensichstation ist unterbrochen
2	SIGINT	Taste DEL
3	SIGQUIT	Tasten CTRL \
9	SIGKILL	Kommando *kill -9 PID*; PID ist die Prozeß-Nummer der entsprechenden Shell
15	SIGTERM	Kommando *kill -15 PID*; PID ist die Prozeß-Nummer der entsprechenden Shell

Abhängig davon, ob Sie mit *trap* eine Signalbehandlung vereinbart haben oder nicht, reagiert die Shell auf diese Signale wie folgt:

Signal-Nummer	Dialog-Shell	Prozedur-Shell	Prozedur-Shell im Hintergrund
1	ignorieren	ignorieren	ignorieren
2	vereinbarte Signalbehandlung ausführen sonst: ignorieren	vereinbarte Signalbehandlung vor dem nächsten Kommando ausführen sonst: abbrechen	ignorieren
3	vereinbarte Signalbehandlung nach dem nächsten Kommando ausführen sonst: ignorieren	vereinbarte Signalbehandlung vor dem nächsten Kommando ausführen sonst: abbrechen und gegebenenfalls *core* schreiben	ignorieren
9	abbrechen	abbrechen	abbrechen
15	ignorieren	ignorieren	ignorieren

In C-Programmen vereinbaren Sie mit der Funktion *signal()*, wie das Programm auf eintreffende Signale reagieren soll (siehe *signal()* [14]).

BEISPIELE

1. In einer Shell-Prozedur soll das Signal 2 ignoriert werden. Deshalb enthält diese Prozedur die folgende Zeile:

```
trap '' 2
```

   Die zwei Hochkommata, 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.

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 20 22:17:23 MEZ 1989
```

   Hier wird vereinbart, daß bei Beendigung der aktuellen Shell eine Meldung in die Datei *$HOME/logdatei* geschrieben werden soll. Die Kommandoliste muß in Hochkommata eingeschlossen sein, damit das Kommando *date* erst bei Beendigung der Shell ausgeführt wird.

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 Prozeß-Nummer 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 Ende-Status 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 Ende-Status 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 Ende-Status 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*, *sh*
signal() [14]

true
Ende-Status 0 zurückgeben

true gibt den Ende-Status 0 zurück und tut sonst nichts.
true verwendet man in Shell-Prozeduren, um die Bedingung *wahr* zu erzeugen.
Die Bedingung *falsch* (Endestatus ungleich 0) erzeugen Sie mit dem Kommando *false*.

```
true
```

ENDE-STATUS

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, sh, : (Shell-Kommando)

truss Systemaufrufe und Signale protokollieren

truss führt das angegebene Kommando aus und protokolliert mit, welche Systemaufrufe es aufruft, welche Signale es empfängt und welche Maschinenfehler es verursacht. Jede Zeile des ausgegebenen Protokolls enthält entweder einen Maschinenfehler oder den Namen eines Signals oder einen Systemaufruf mit seinen Argumenten und Rückkehrwerten. Die Argumente von Systemaufrufen werden symbolisch dargestellt, wobei nach Möglichkeit die Define-Werte aus den entsprechenden Header-Dateien verwendet werden. Fehler-Rückkehrwerte werden mit den verschlüsselten Fehlernamen ausgegeben, die im Einleitungskapitel des *Referenzhandbuchs für Programmierer* [14] beschrieben sind.

```
truss [option] ... kommando
```

option

Bei den folgenden Optionen ist es möglich, für ein Argument mehrmals dieselbe Option mit verschiedenen Optionsargumenten anzugeben. In diesen Fällen schränken immer die weiter rechts stehenden Optionen die weiter links stehenden ein.

-a

truss gibt bei jedem Aufruf des Systemaufrufs *exec()* die Zeichenketten aus, die als Argumente übergeben werden.

-c

truss gibt die protokollierten Systemaufrufe, Fehler und Signale nicht aus, sondern zählt sie mit und gibt das Ergebnis aus, sobald das angegebene Kommando beendet oder *truss* unterbrochen wird. Das Ergebnis enthält je eine eigene Summe für Systemaufrufe, Fehler und Signale.

Ist zusätzlich Option *-f* angegeben, zählt *truss* die Systemaufrufe, Fehler und Signale auch für alle Kindprozesse mit.

-e

truss gibt bei jedem Aufruf des Systemaufrufs *exec()* die Zeichenketten aus, die als Umgebung übergeben werden.

-f

truss protokolliert auch alle durch *fork()* erzeugte Kindprozesse und die zugehörigen Signale, Fehler und Systemaufrufe. Jede Ausgabezeile enthält zusätzlich eine PID, die anzeigt, welcher Prozeß den Systemaufruf aufgerufen oder das Signal empfangen hat.

-f nicht angegeben:
Nur das Kommando bzw. der Prozeß, der auf der ersten Stufe abläuft, wird protokolliert.

-i

truss protokolliert Systemaufrufe, die aktuell schlafen und unterbrechbar sind, wie z.B. *open()* und *read()* auf Terminal-Gerätedateien oder Pipes, erst, wenn sie abgeschlossen werden.

-i nicht angegeben:
truss protokolliert schlafende und unterbrechbare Prozesse, sobald sie länger als eine Sekunde schlafen, einmal und ein zweites Mal, wenn sie abgeschlossen werden.

-m[␣][!]fehler,...

Sie geben einen oder mehrere Maschinenfehler an, die *truss* protokollieren soll. Sollen alle Maschinenfehler protokolliert werden, können Sie für *fehler* die Zeichenkette *all* angeben. Wenn Sie vor *fehler* ein Ausrufezeichen *!* angeben, protokolliert *truss* die angegebenen Fehler nicht. Maschinenfehler können Sie entweder mit ihrem Namen oder mit ihrer Nummer (siehe *<sys/fault.h>*) angeben.

-m[␣][!]fehler nicht angegeben:
entspricht der Angabe *-m all -m !fltpage*, d.h. alle Maschinenfehler werden ausgegeben, bis auf FLTPAGE (recoverable page fault).

-o[␣]ausdatei

truss schreibt seine Ausgabe in die Datei *ausdatei*.

-o[␣]ausdatei nicht angegeben:
truss schreibt seine Ausgabe auf die Standard-Fehlerausgabe.

-p

truss interpretiert die übergebenen Argumente als Liste von PIDs für ablaufende Prozesse (siehe *ps*) und nicht als Kommando das ausgeführt werden soll. Mehrere Argumente können Sie nur in Form von PIDs angeben, in Form eines Kommandos können Sie nur ein Argument angeben. Geben Sie mehrere Kommandos an, werden alle außer dem ersten ignoriert. *truss* kontrolliert jeden angegebenen Prozeß und protokolliert ihn, vorausgesetzt die UID und GID (siehe *ps*) des aufrufenden Benutzers entsprechen der UID und GID des Prozesses oder der aufrufende Benutzer ist Systemverwalter. Prozesse können Sie auch über ihre im Dateiverzeichnis */proc* stehenden Namen angeben, z.B. */proc/1234*. Dies ist auch für fern eingehängte *proc*-Dateiverzeichnisse möglich.

-r[␣][!]fd,...

truss gibt für jeden Systemaufruf *read()* auf einen der für *fd* angegebenen Dateideskriptoren den Inhalt des Ein-/Ausgabepuffers vollständig aus. Alle Dateideskriptoren können Sie mit der Zeichenkette *all* für *fd* angeben. Wenn Sie vor *fd* ein Ausrufezeichen *!* angeben, gibt *truss* für die angegebenen Dateideskriptoren nichts aus. Die Ausgabe enthält 32 byte je Zeile und stellt jedes Byte als ASCII-Zeichen mit einem Leerzeichen davor dar oder Steuerzeichen als C-Escapefolge aus zwei Zeichen wie z.B. `\t` für Tabulatorzeichen und `\n` für Neue-Zeile-Zeichen. Ist die ASCII-Darstellung nicht möglich, wird der Hexadezimal-Code des Bytes ausgegeben.

-r[␣][!]fd nicht angegeben:
entspricht der Angabe *-r !all*, d.h. für keinen Dateideskriptor gibt *truss* etwas aus. Für jeden protokollierten Systemaufruf *read()* oder *write()* werden die ersten 12 byte des Ein-/Ausgabepuffers ausgegeben.

-s[␣][!]signal,...
Sie geben ein oder mehrere durch Komma getrennte Signale an, die *truss* protokollieren soll. Sollen alle Signale protokolliert werden, können Sie für *signal* die Zeichenkette *all* angeben. Wenn Sie vor *signal* ein Ausrufezeichen *!* angeben, protokolliert *truss* die angegebenen Signale nicht. *truss* protokolliert in seiner Ausgabe jedes angegebene empfangene Signal, auch wenn der Prozeß es ignoriert (nicht blockiert). (Blockierte Signale werden erst empfangen, wenn der Prozeß sie freigibt). Signale können Sie entweder mit ihrem Namen oder mit ihrer Nummer (siehe *<sys/signal.h>*) angeben.

-s[␣][!]signal nicht angegeben:
entspricht der Angabe *-s all*, d.h. alle Signale werden mitprotokolliert.

-t[␣][!]systemaufruf,...
Sie geben einen oder mehrere durch Komma getrennte Systemaufrufe an, die *truss* protokollieren soll. Sollen alle Systemaufrufe protokolliert werden, können Sie für *systemaufruf* die Zeichenkette *all* angeben. Wenn Sie vor *systemaufruf* ein Ausrufezeichen *!* angeben, protokolliert *truss* die angegebenen Systemaufrufe nicht. Systemaufrufe können Sie entweder über ihre Namen oder Nummern angeben (siehe *Referenzhandbuch für Programmierer, Kapitel 2* [14]).

-t[␣][!]systemaufruf nicht angegeben:
entspricht der Angabe *-t all*, d.h. alle Systemaufrufe werden mitprotokolliert.

-v[␣][!]systemaufruf,...
truss gibt den Inhalt jeder Struktur aus, die über ihre Adresse an den/die angegebenen Systemaufruf(e) übergeben wird. Übergabewerte und die Werte, die das Betriebssystem zurückgibt, werden ausgegeben. Für jedes Feld, das für Ein- und Ausgabe benützt wird, wird nur der Ausgabewert ausgegeben. Alle Systemaufrufe können Sie mit der Zeichenkette *all* für *systemaufruf* angeben. Wenn Sie vor *systemaufruf* ein Ausrufezeichen *!* angeben, gibt *truss* für die angegebenen Systemaufrufe nichts aus. Systemaufrufe können Sie entweder über ihre Namen oder Nummern angeben (siehe *Referenzhandbuch für Programmierer, Kapitel 2* [14]).

-v[␣][!]systemaufruf nicht angegeben:
entspricht der Angabe *-v !all*, d.h. kein Systemaufruf wird angegeben.

-w[␣][!]fd,...
wie Option *-r*, nur für den Systemaufruf *write()*.

-x[␣][!]systemaufruf,...
truss gibt die Argumente, die an den/die angegebenen Systemaufruf(e) übergeben werden, nicht symbolisch, sondern in hexadezimaler Form aus. Alle Systemaufrufe

können Sie mit der Zeichenkette *all* für *systemaufruf* angeben. Wenn Sie vor *systemaufruf* ein Ausrufezeichen *!* angeben, gibt *truss* für die angegebenen Systemaufrufe nichts aus. Systemaufrufe können Sie entweder über ihre Namen oder Nummern angeben (siehe *Referenzhandbuch für Programmierer, Kapitel 2* [14]).

-x[␣][!]systemaufruf nicht angegeben:
entspricht der Angabe *-x !all*, d.h. kein Systemaufruf wird angegeben.

ARBEITSWEISE

Wenn Sie *truss* mit Option *-o* aufrufen oder die Standard-Fehlerausgabe in eine Datei umlenken, die keine Terminal-Gerätedatei ist, ignoriert *truss* die Signale SIGHUP, SIGINT und SIGQUIT. Dadurch ist es leichter, interaktive Programme zu protokollieren, die solche Signale vom Terminal erhalten.

Wenn *truss* seine Ausgabe auf ein Terminal ausgibt oder wenn mit Option *-p* bereits existierende Prozesse protokolliert werden, reagiert *truss* auf die Signale SIGHUP, SIGINT und SIGQUIT, indem es alle protokollierten Prozesse freigibt und sich beendet. Dadurch ist es möglich, eine allzu ausführliche Ausgabe von *truss* abzubrechen und Prozesse wieder freizugeben, die vor dem Start von *truss* bereits existierten. Solche Prozesse laufen dann normal weiter als ob nichts gewesen wäre.

Einige der im *Referenzhandbuch für Programmierer, Abschnitt 2* [14] beschriebenen Systemaufrufe weichen von den aktuell im Betriebssystem vorhandenen Schnittstellen ab. Es ist möglich, daß die Ausgabe von *truss* von den Beschreibungen im *Referenzhandbuch für Programmierer* [14] ein wenig abweicht.

Jeder Maschinenfehler (ausgenommen Paging-Fehler)
führt dazu, daß der den Fehler verursachende Prozeß ein Signal erhält. Der Empfang des Signals wird jeweils sofort nach der Meldung des Maschinenfehlers (außer Paging-Fehler) gemeldet, es sei denn, der Prozeß blockiert das Signal.

Das Betriebssystem schränkt die Protokollierung von Prozessen aus Sicherheitsgründen in einigen Fällen ein: so kann ein Benutzer, der den Objektcode eines Kommandos (a.out-Datei) nicht lesen kann, die Prozesse dieses Kommandos nicht protokollieren. Kommandos, bei denen für den Eigentümer oder für die Gruppe das s-Bit (siehe *chmod*) gesetzt ist, kann nur der Systemverwalter protokollieren. *truss* verliert die Kontrolle über jeden Prozeß, der einen *exec()*-Systemaufruf für eine Objektdatei durchführt, für die das s-Bit gesetzt ist oder die nicht lesbar ist, es sei denn, *truss* wird vom Systemverwalter aufgerufen. Solche Prozesse werden ab dem Aufruf von *exec()* unbeeinflußt von *truss* normal fortgesetzt.

Um Konflikte mit anderen kontrollierenden Prozessen zu vermeiden, protokolliert *truss* keinen Prozeß, von dem es bemerkt, daß er bereits von einem anderen Prozeß über die Schnittstelle */proc* kontrolliert wird. Im Dateiverzeichnis */proc* wird für jeden laufenden Pprozeß eine Datei angelegt, auf die Aufrufe wie *open()*, *close()* und *iocntl* zugreifen können. Diesen Mechanismus verwenden zum Beispiel Debugger wie *sdb* alternativ zu *ptrace()*. Auch *truss* verwendet diesen Mechanismus. Wenn ein Prozeß bereits von einem anderen Prozeß durch *proc()* kontrolliert wird, kann dieser Prozeß von *truss* nicht mehr bearbeitet werden, wie folgendes Beispiel zeigt:

```
$ truss -o /dev/null sh&
```

Angenommen, der Prozeß *sh* hat die Prozeß-Id 18027:

```
$ truss -p 18027
truss: someone else is tracing process 18027
```

Jedoch ist es möglich, truss auf einen Debugger anzuwenden, der seinerseits einen Prozeß per *proc()* kontrolliert. Dabei kann *truss* dann zwar den Debugger protokollieren, nicht aber den von dem Debugger gestarteten Prozeß. Ebenso kann *truss* auf sich selbst angewendet werden, z.B.:

```
$ truss truss ls
          |     |
          |     kann vom ersten truss nicht mitprotokolliert werden,
          |     da bereits vom zweiten truss per proc()
          |     kontrolliert.
          |
          kann protokolliert werden.
```

Die Ausgabe von *truss* enthält Tabulatorzeichen, wobei von Standard-Tabulatorstopps in jeder 8.Spalte ausgegangen wird.

Die Ausgabe für mehrere Prozesse erfolgt in keiner festgelegten zeitlichen Reihenfolge. Zum Beispiels kann es sein, daß der Systemaufruf *read()* auf eine Pipe vor dem entsprechenden Systemaufruf *write()* protokolliert wird. Für einen einzigen Prozeß erfolgt die Ausgabe exakt in der richtigen zeitlichen Reihenfolge.

Beim Protokollieren von Kindprozessen kann es passieren, daß das System an die Grenze der pro Benutzer gleichzeitig möglichen Prozesse stößt. Werden mehr als ein Prozeß protokolliert, läuft *truss* für jeden Prozeß, der protokolliert werden soll, als eigener kontrollierender Prozeß. Das Kommando *spell* benötigt zum Beispiel 9 Prozesse und *truss* fügt weitere 9 hinzu, so daß insgesamt 18 Prozesse laufen. Damit ist man bereits sehr nah an der im System festgelegten Grenze von 25 möglichen Prozessen pro Benutzer.

truss benutzt bei der Protokollierung von mehr als einem Prozeß die Mechanismen *shared memory* und *Semaphoren* (Optionen *-f* und *-p* bei mehr als einer PID). Wenn diese Mechanismen auf einem System nicht zur Verfügung stehen, aber gebraucht werden, gibt *truss* eine Warnung aus und arbeitet weiter. In diesem Fall kann aber die Ausgabe von *truss* durcheinandergeraten und die Ausgabe bei Option *-c* protokolliert nur das Kommando der höchsten Ebene oder die erste PID und keine Kindprozesse.

FEHLERMELDUNGEN

`truss: someone else is tracing process <PID>`

truss wurde auf einen bereits anderweitig kontrollierten Prozeß angewandt.

`truss: invalid process id: <PID>`

Nicht vorhandene PID.

`truss: cannot control process <PID>`

Zugriffsberechtigung für PID fehlt.

`truss: Cannot find program: name`

truss findet das angegebene Programm *name* nicht.

DATEIEN

/proc
Dateiverzeichnis mit den für jeden laufenden Prozeß angelegten Dateien.

BEISPIELE

1. Protokollieren des Kommandos *find* und Ausgabe des Protokolls auf dem Bildschirm (Standard-Fehlerausgabe), wobei die Ausgabe von *find* (Standard-Ausgabe) in die Datei *find.aus* umgelenkt wird:

```
$ truss find . -print > find.aus
```

2. Protokollieren der Systemaufrufe *open()*, *close()*, *read()* und *write()*, die das Kommando *find* aufruft:

```
$ truss -t open,close,read,write find . -print > find.aus
```

3. Protokollieren des Kommandos *spell*, dem die Datei *document* als Argument übergeben wird, wobei die Ausgabe von *truss* mit der Option *-o* in die Datei *truss.aus* umgelenkt wird. Da *spell* ein Shell-Skript ist, muß *truss* mit der Option *-f* aufgerufen werden, damit auch die Kindprozesse protokolliert werden.

```
$ truss -f -o truss.aus spell document
```

4. Protokollieren der Aktivitäten des *init*-Prozesses. Dazu ist nur der Systemverwalter berechtigt.

```
$ truss -p -v all 1
```

tty
Pfadnamen der aktuellen Datensichtstation ausgeben (terminal type)

tty gibt den Pfadnamen der Datensichtstation aus, mit der der Prozeß verbunden ist. Der Ende-Status 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.

```
tty[␣-l][␣-s]
```

-l
Falls die aktuelle Datensichtstation über eine synchrone Leitung angeschlossen ist, wird deren Nummer ausgegeben.

-s
tty gibt nichts aus, sondern liefert nur den Ende-Status.

-s nicht angegeben:
Ist die Standard-Eingabe keine Datensichtstation, meldet *tty* dies.

ENDE-STATUS

0 Standard-Eingabe ist eine Datensichtstation.
1 Standard-Eingabe ist keine Datensichtstation.
2 Eine ungültige Option wurde angegeben.

FEHLERMELDUNGEN

`not on an active synchronous line`

Die Option *-l* wurde angegeben und die Standardeingabe ist keine Datensichtstation, die über eine synchrone Leitung angeschlossen ist.

`not a tty`

Die Standardeingabe ist keine Datensichtstation und die Option *-s* wurde nicht angegeben.

BEISPIELE

1. Den Namen der aktuellen Datensichtstation ausgeben:

```
$ tty
/dev/tty003
```

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'
  .
  .
```

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

Das in die Bourne-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 Korn-Shell *ksh* ist **type** eine Alias-Variable für *whence -v*, wobei *whence* ein eingebautes Korn-Shell-Kommando ist.

```
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, oder den absoluten Pfadnamen bzw. einen Pfadnamen der Form *./name*, falls *name* eine ausführbare Datei ist.
  Ist *name* eine Shell-Funktion, gibt *type* zusätzlich die Funktionsdefinition aus. In der Korn-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.

FEHLERMELDUNG

```
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.

UMGEBUNGSVARIABLE

PATH
Suchpfad der Shell

BEISPIEL

Der folgende Bildschirm-Dialog zeigt, welche Informationen das eingebaute *sh*-Kommando *type* liefert:

```
$ type tar
tar is /bin/tar
$ type type
type is a shell builtin
$ type ll
ll is a function
ll(){
ls -al $* ≠ pg
}
$ type typetest
typetest is ./typetest
```

SIEHE AUCH

sh, ksh

ulimit
Datei-Größe für das Schreiben begrenzen oder aktuellen Grenzwert abfragen (user limit)

Mit dem in die Bourne-Shell *sh* eingebauten Kommando *ulimit* können Sie

- abfragen, welche Grenzwerte für die aktuelle Shell oder ihre Sohn-Prozesse festgelegt sind.
- die Grenzwerte einzeln für die aktuelle Shell und alle ihre Sohn-Prozesse ändern. Als Benutzer ohne Systemverwalter-Privilegien können Sie diese Werte nur herabsetzen. Die neuen Werte gelten für die aktuelle Shell und ihre Sohn-Prozesse.

  Einen herabgesetzten Wert können Sie nicht mehr erhöhen. Sie müssen die Shell beenden, in der Sie den Grenzwert herabgesetzt haben.

  Nur der Systemverwalter kann diesen Grenzwert erhöhen.

Die folgenden Grenzwerte, die in *getrlimit()* genauer beschrieben sind, stehen Ihnen für die aktuelle Shell und alle ihre Sohn-Prozesse zur Verfügung:

c - core size
: Maximale Größe eines Speicherabzugs in der Datei *core* (in 512 Byte-Blöcken), wenn ein Prozeß fehlerhaft abgebrochen wurde (siehe *signal()* [14]). Ist *core size* gleich 0, wird keine *core*-Datei angelegt.

d - data segment
: Maximale Größe des Datensegments oder *heap* (in Kbyte) eines Prozesses.

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 diesen Standardwert überschreiten erhalten Sie entweder eine Fehlermeldung vom entsprechenden Kommando oder die neue Datei enthält nur die Daten bis zum Erreichen des Grenzwertes.

n - number of filedescriptors
: Maximale Anzahl (geöffneter) Dateikennzahlen eines Prozesses plus 1.

s - stack size
: Maximale Größe des Stacksegments (in Kbyte) eines Prozesses.

t - time
: Maximal verbrauchbare CPU-Zeit (in Sekunden) für einen Prozeß.

v - virtuell memory size
: Maximale Größe des virtuellen Speichers (in Kbyte) eines Prozesses.

Beispiel für die Dateigröße:

Nach *ls -lR >datei* enthält *datei* nur so viele Bytes, wie der aktuelle Grenzwert erlaubt.
Bei *cp* erhalten Sie die Fehlermeldung *bad copy to datei*, wenn die Datei, die kopiert werden soll, größer ist als der aktuelle Grenzwert.

```
ulimit[␣-H][␣-S][␣option] ...                                    Format 1
ulimit[␣-H][␣-S][␣option]␣grenzwert                              Format 2
```

Format 1: Grenzwerte abfragen

ulimit[␣-H][␣-S][␣option]...

ulimit schreibt die durch *option* abgefragten Grenzwerte auf die Standard-Ausgabe. Sie können die Optionen beliebig kombinieren. Wollen Sie alle Grenzwerte gleichzeitig sehen, verwenden Sie einfach die Option *-a*.

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.

grenzwert
: legt den Grenzwert für die aktuelle Shell und jeden ihrer Sohn-Prozesse entsprechend der angegebenen Option fest. Für *grenzwert* können Sie als Benutzer ohne Systemverwalter-Privilegien nur Werte angeben, die kleiner sind als der aktuelle Grenzwert. Als Systemverwalter 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.

-H
: Abfragen (Format 1) oder Setzen (Format 2) eines *harten* Grenzwerts (hard limit). Als Benutzer ohne Systemverwalter-Privilegien können Sie jeden *harten* Grenzwert herabsetzen. Aber nur der Systemverwalter darf einen *harten* Grenzwert erhöhen.

-S
: Abfragen (Format 1) oder Setzen (Format 2) eines *weichen* Grenzwerts (soft limit). Jeder Benutzer kann einen *weichen* Grenzwert auf einen Wert kleiner dem *harten* Grenzwert setzen.

Weder *-H* noch *-S* angegeben
Beim Format 1 schreibt *ulimit* die *weichen* Grenzwerte auf die Standard-Ausgabe. Beim Format 2 setzt *ulimit harte* und *weiche* Grenzwerte auf den angegebenen Wert.

option

Durch Optionen können Sie die abzufragenden (Format 1) oder zu setzenden (Format 2) Grenzwerte angeben.
Alle Optionen sind für Format 1 und Format 2 zugelassen, mit Ausnahme der Option *-a*, die nur für Format 1 zugelassen ist.

Keine Option angegeben
ulimit verwendet die Option *-f*.

-a
Nur bei Format 1: Abfrage aller Grenzwerte.

-c
Maximale Größe eines Speicherabzugs in der Datei *core* (in 512 Byte-Blöcken).

-d
Maximale Größe des Datensegments oder *heap* (in Kbyte).

-f
Maximale Dateigröße (in 512 Byte-Blöcken).

-n
Maximale Anzahl von Dateikennzahlen plus 1.

-s
Maximale Größe des Stackelements (in Kbyte).

-t
Maximal verbrauchbare CPU-Zeit (in Sekunden).

-v
Maximale Größe des virtuellen Speichers (in Kbyte).

FEHLERMELDUNG

`Bad ulimit`

Sie haben versucht, den aktuellen Grenzwert zu erhöhen. Dies darf nur der Systemverwalter.

BEISPIEL

Den aktuellen Grenzwert abfragen und anschließend herabsetzen:

```
$ ulimit
4194303
$ ulimit 20000
$ sh
$ ulimit
20000
```

Dieser neue Grenzwert ist auch in der Subshell gültig. Ab jetzt können nur noch Dateien angelegt werden, die kleiner sind als 20.000 * 512 byte.

SIEHE AUCH

sh
getrlimit(), signal(), ulimit() [14]

umask
Standard-Vergabe der Zugriffsrechte ändern (user mask)

Das in die Bourne-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.

In der Datei */etc/profile* kann der Systemadministrator z.B. mit *umask* die Schutzbit-Maske auf den Wert 066 setzen. Da */etc/profile* von jeder Login-Shell ausgeführt wird, gelten für jeden Benutzer nach der Anmeldung am System folgende Zugriffsrechte (siehe *Beispiel 1*):

- für neu angelegte Dateien: `rw-------`
- für neu angelegte Dateiverzeichnisse: `rwx--x--x`

Aus der Schutzbit-Maske die Zugriffsrechte bestimmen

Beim Anlegen von Dateien und Dateiverzeichnissen werden in der Regel folgende Zugriffsrechte als Grundeinstellung vergeben (siehe *open()* [14]):

- `rw-rw-rw-` an Dateien; das ist binär 110110110
- `rwxrwxrwx` an Dateiverzeichnisse; das ist binär 111111111

Mit *umask* können Sie von dieser Grund-Einstellung nur Rechte wegnehmen. Das bedeutet, Sie können mit *umask* nie erreichen, daß an Dateien, die Sie z.B. mit *cat* oder einem Editor neu anlegen, automatisch das Ausführrecht vergeben wird. Das entsprechende x-Bit können Sie aber mit dem Kommando *chmod* setzen.

Die Schutzbit-Maske ist die 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 Grund-Einstellung mit UND; d.h das Ergebnis hat nur an den Stellen eine Eins, an der beide Summanden eine Eins haben, sonst 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`.

umask [*maske*]

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*).

  Unabhängig von der Angabe für *maske* erreichen Sie mit *umask* nie, daß an Dateien das Ausführrecht vergeben wird. Verwenden Sie dazu *chmod*.

  maske nicht angegeben:
  umask gibt die derzeit gültige Schutzbit-Maske aus. Die Ausgabe hat folgendes Format:

  0nnn

  0
  : steht für Oktaldarstellung

  nnn
  : die aktuelle Schutzbit-Maske in Oktaldarstellung

DATEI

/etc/profile
Der Systemadministrator legt hier normalerweise einen Schutzmasken-Wert für normale Benutzer und einen für *root* fest.

BEISPIELE

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`

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 anschließend an neue Dateien und Dateiverzeichnisse vergeben werden.

SIEHE AUCH

chmod, sh
chmod(), creat(), open(), umask() [14]
profile [5], [10], [11]

uname
Namen des aktuellen Systems ausgeben

uname schreibt Informationen über das aktuelle Betriebssystem auf die Standard-Ausgabe.

```
uname [␣option] ...                                    Format 1
uname [␣-S␣systemname]                                 Format 2
```

Format 1: Den Systemnamen ausgeben

uname[␣option]...

Keine Option angegeben
Der Name des Betriebssystems wird ausgegeben, z.B. *SINIX-L*. Unter diesem Namen kann das System innerhalb eines Netzes angesprochen werden.

-a
(a - all) Alle im folgenden genannten Informationen werden ausgegeben, z.B. *SINIX-L bolte V5.40 N5 i386 MX300I*.

-m
(m - machine type) Der Name des Maschinentyps wird ausgegeben, z.B. *MX300I*.

-n
(n - node) Der Knotenname des Betriebssystems wird ausgegeben. Unter diesem Namen kann das System innerhalb eines Netzes angesprochen werden, z.B. *bolte*.

-p
(p - processor type) Die Bezeichnung des Prozessors von dem Rechner, mit dem der Benutzer aktuell arbeitet, wird ausgegeben.

-r
(r - release) Die Versionsnummer des Betriebssystems wird ausgegeben, z.B. *V5.40*.

-s
(s - system) Der Name des Betriebssystems wird ausgegeben, z.B. *SINIX-L*. Unter diesem Namen ist das Betriebssystem auf der lokalen Installation bekannt.

-v
(v - version) Der Nachtragsstand der Version des Betriebssystems wird ausgegeben.

Format 2: Den Systemnamen festlegen bzw. ändern

uname␣-S␣systemname

Format 2 ist nur für den Systemverwalter bestimmt.

systemname
Knotennname, den das System erhalten soll.

SIEHE AUCH

sysname(), *uname()* [14]

uncompress
Komprimierte Dateien expandieren

uncompress dekomprimiert Dateien, die mit *compress* komprimiert wurden. Die Dateien werden wieder in ihren Originalzustand zurückgebracht.

uncompress gehört, zusammen mit dem Programmen *compress* und *zcat* zu einer Gruppe von Kommandos, mit der Sie Dateien komprimieren, dekomprimieren und komprimierte Dateien ausgeben können. Eine ähnliche Funktionalität bieten Ihnen die Kommandogruppe *pack*, *unpack* und *pcat*.

```
uncompress[ option] ... [ datei] ...
```

Keine Option angegeben
: Die angegebenen Dateien oder die Daten der Standardeingabe werden dekomprimiert.

option

-c
: *uncompress* schreibt die dekomprimierten Daten nur auf die Standardausgabe. Es werden keine Dateien verändert oder angelegt. *uncompress -c* ist in seiner Funktion identisch mit *zcat*.

-v
: (v - verbose) Die prozentuale Platzersparnis für jede komprimierte Datei wird angezeigt.

datei
: Name einer mit *compress* komprimierten Datei. Sie können den Namen mit dem oder ohne das Suffix *.Z* angeben, der Name der Datei muß aber mit *.Z* enden. Ist diese Datei eine mit *compress* komprimierte Datei, so wird sie durch die dekomprimierte Version ersetzt. Der Name der neuen Datei hat nicht mehr die Endung *.Z*. Die dekomprimierte Datei erhält dieselben Zugriffsrechte, dasselbe Zugriffs- und Änderungsdatum und denselben Eigentümer wie die komprimierte Datei.
Wenn auf die komprimierte Datei Verweise eingetragen sind, gibt *uncompress* die Warnung
`uncompress: datei.Z: Warning: file has links`
aus, die Datei wird aber trotzdem dekomprimiert. Der Verweis auf die komprimierte Datei bleibt erhalten.

ENDE-STATUS

0 bei Erfolg
1 bei Fehler

FEHLERMELDUNGEN

Bei allen im folgenden beschriebenen Fehlern wird das Kommando *uncompress* nicht ausgeführt, d.h., die Datei bleibt in komprimiertem Zustand und der Original-Zustand wird nicht wiederhergestellt.

`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 unerlaubtem Segmentzugriff) wurde empfangen, was normalerweise bedeutet, daß die Eingabedatei beschädigt ist.

SIEHE AUCH

compress, *zcat*, *pack*, *unpack*, *pcat*

uniq
Mehrfache Zeilen suchen (unique lines)

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.

```
uniq[ option[ +n][ -m]][ eingabe_datei[ ausgabe_datei]]
```

Es ist nicht sinnvoll, mehrere Optionen gleichzeitig anzugeben.

Keine Option angegeben
eingabe_datei wird ausgegeben und Wiederholungen werden weggelassen. Die Ausgabe von *uniq* ohne Option und *uniq -ud* ist identisch.

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. Die Zahl steht rechtsbündig bis Spalte 4, der Zeileninhalt beginnt in Spalte 6. *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 bzw. ab Feld $m+1$ werden beim Vergleichen der Zeilen nicht berücksichtigt. Ein Feld ist eine nichtleere Zeichenfolge, die durch ein Tabulator- oder Leerzeichen vom Nachbarfeld getrennt ist.

$+n$ nicht angegeben:
Die Zeilen werden ab Zeilenanfang bzw. ab Anfang von Feld $m+1$ verglichen.

-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.

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 die die Ausgabe geschrieben werden soll.

ausgabe_datei nicht angegeben:
uniq schreibt auf die Standard-Ausgabe.

BEISPIELE

1. Durchsuchen einer Datei nach gleichen Zeilen unabhängig davon, wo sie in der Datei stehen. Für jede dieser Zeilen ist auszugeben, wie oft sie vorkommt.

```
$ sort datei | uniq -c
```

2. Ausgeben der 10 häufigsten Wörter in der Datei *text*.

```
$ cat text | sed 's/␣␣*/\ ↵
> /g' | sort | uniq -c | sort -r | awk '{ if(NR<11) print $0 }'
```

Erläuterung:

- *sed* erzeugt aus *text* eine Liste aller Wörter, indem ein oder mehrere Leerzeichen durch ein Neue-Zeile-Zeichen ersetzt werden.
- *sort* sortiert diese Liste nach ASCII.
- *uniq -c* entfernt aus der sortierten Liste mehrfache Zeilen und schreibt vor alle übriggebliebenen Zeilen die Häufigkeit ihres Auftretens.
- *sort -r* sortiert diese Häufigkeitsliste rückwärts, d.h. die häufigste Zeile steht in der ersten, die seltenste in der letzten Zeile.
- *awk* gibt diese Liste zeilenweise ($0) aus, solange die Variable NR kleiner als 11 ist. In NR wird die Anzahl der abgearbeiteten Zeilen mitgezählt (siehe auch *awk*).

SIEHE AUCH

comm, sort

units
Einheiten umrechnen

units berechnet Umrechungsfaktoren zwischen verschiedenen Einheiten.

```
units
```

ARBEITSWEISE

Sie können die Umrechnungsfaktoren je zweier Einheiten berechnen. *units* arbeitet interaktiv. Nach der Eingabe des Kommandos werden Sie aufgefordert, die erste Einheit einzugeben:

```
$ units ↵
you have:
```

Haben Sie die erste Einheit eingegeben, werden Sie aufgefordert, die zweite Einheit anzugeben:

```
you want:
```

Wenn Sie auch die zweite Einheit angegeben haben, gibt *units* die Umrechnungsfaktoren aus. Der erste Umrechnungsfaktor gibt an, mit welcher Zahl Angaben in der ersten Einheit multipliziert werden müssen, um Angaben in der zweiten Einheit zu erhalten. Dem ersten Umrechnungsfaktor ist ein Stern * vorangestellt. Der zweite Umrechnungsfaktor gibt an, durch welche Zahl Angaben in der zweiten Einheit dividiert werden müssen, um Angaben in der ersten Einheit zu erhalten. Dem zweiten Umrechnungsfaktor ist ein Schrägstrich / vorangestellt. Beide Umrechnungsfaktoren sind in wissenschaftlicher Notation und mit sechs Nachkommastellen angegeben, z.B. steht 9.797299e-01 für $9{,}797299 * 10^{-1}$, d.h. für 0,9797299.

Beispiel

```
$ units
you have: inch
you want: cm
        * 2.540000e+00
        / 3.937008e-01
```

Die umzurechnenden Einheiten können auf folgende Weisen aus anderen Einheiten kombiniert werden:

- durch ein oder mehrere Leerzeichen getrennte Einheiten werden miteinander multipliziert
- durch einen Schrägstrich / getrennte Einheiten werden dividiert
- nachgestellte ganze Zahlen geben Potenzen an
- vorangestellte ganze Zahlen werden als multiplikative Konstanten interpretiert.

Beispiel

```
$ units
$ you have: 15 lbs force/in2
  you want: atm
          * 1.020689e+00
          / 9.797299e-01
```

Sie können mit *units* nur Umrechnungen berechnen, die sich multiplikativ ausdrücken lassen, eine Umrechnung von Grad Celsius in Grad Fahrenheit ist also z.B. nicht möglich.

Neben den üblichen Einheiten und ihren Abkürzungen, kennt *units* auch einige Konstanten aus Chemie, Physik und Mathematik
:

pi	Verhältnis von Kreisumfang zu Kreisdurchmesser
c	Lichtgeschwindigkeit
e	Ladung eines Elektrons
g	Erdbeschleunigung
force	wie *g*
mole	Avogadro'sche Zahl

Die Datei */usr/share/lib/unittab* enthält eine vollständige Tabelle der Einheiten und Konstanten, die *units* bekannt sind, und die verwendeten Umrechnungsfaktoren.

Nur für den Systemverwalter

Der Systemverwalter kann die Datei */usr/share/lib/unittab* verändern, z.B. die Umrechnungsfaktoren für die verschiedenen Währungen aktualisieren oder neue Einheiten mit den entsprechenden Umrechnungsfaktoren hinzufügen.

Sie beenden die Ausführung von *units*, indem Sie die Taste DEL oder END drücken.

FEHLERMELDUNGEN

`cannot recognize xyz`

Sie haben eine Einheit angegeben, die *units* nicht kennt. In der Datei */usr/share/lib/unittab* können Sie nachsehen, welche Einheiten *units* bekannt sind.

`conformability`

Sie haben Einheiten angegeben, die *units* nicht umrechnen kann. Zusätzlich gibt *units* noch die Definitionen der angegebenen Einheiten aus.

DATEI

/usr/share/lib/unittab
enthält eine vollständige Liste der *units* bekannten Einheiten und Konstanten

BEISPIELE

1. Meilen in Kilometer umrechnen:

```
$ units
you have: mile
you want: km
        * 1.609344e+00
        / 6.213712e-01
END
$
```

2. Bar in Atmosphären umrechnen:

```
$ units
you have: bar
you want: atm
        * 9.869233e-01
        / 1.013250e+00
END
$
```

3. Gallonen in Barrel umrechnen:

```
$ units
you have: gallon
you want: barrel
        * 2.380952e-02
        / 4.200000e+01
END
$
```

unpack
Komprimierte Dateien expandieren

unpack dekomprimiert Dateien, die mit *pack* komprimiert wurden, bringt sie also wieder in ihren Original-Zustand.

```
unpack datei ...
```

datei
: Name einer mit *pack* komprimierten Datei. Sie können den Namen mit dem oder ohne das Suffix *.z* angeben, der Name der Datei muß aber mit *.z* enden.
Ist diese Datei eine mit *pack* komprimierte Datei, so wird sie durch die dekomprimierte Version ersetzt. Der Name der neuen Datei hat nicht mehr die Endung *.z*.
Die dekomprimierte Datei erhält dieselben Zugriffsrechte, dasselbe Zugriffs- und Änderungsdatum und denselben Eigentümer wie die komprimierte Datei.
Wenn auf die komprimierte Datei Verweise eingetragen sind, gibt *unpack* die Warnung `unpack: datei.z: Warning: file has links` aus, die Datei wird aber trotzdem dekomprimiert.

ENDE-STATUS

Der Ende-Status von *unpack* ist die Anzahl der Dateien, die nicht dekomprimiert werden konnten. In welchen Fällen eine Datei nicht dekomprimiert werden kann, erfahren Sie im Abschnitt Fehlermeldungen.

FEHLERMELDUNGEN

Bei allen im folgenden beschriebenen Fehlern wird das Kommando *unpack* nicht ausgeführt, d.h., die Datei bleibt in komprimiertem Zustand und der Original-Zustand wird nicht wiederhergestellt.

`unpack: datei: already exists`

Es existiert bereits eine nicht komprimierte Datei *datei*.

`unpack: datei: cannot create`

Die nicht komprimierte Datei kann nicht erstellt werden, weil Sie kein Schreibrecht für das Dateiverzeichnis haben, in dem diese Datei angelegt würde.

```
unpack: datei.z: cannot open
```

Sie haben kein Leserecht für die komprimierte Datei *datei.z* oder der Datei fehlt die Endung *.z*.

```
unpack: ganz_langer_dateiname: file name too long
```

Der Dateiname ohne *.z* darf aus höchstens 12 Zeichen bestehen.

```
unpack: datei.z: not in packed format
```

datei.z ist keine mit *pack* komprimierte Datei.

BEISPIEL

Die Datei *katharina* wird mit *pack* komprimiert und dann mit *unpack* dekomprimiert, also wieder in ihren Originalzustand gebracht:

```
$ ls -l
total 532
-rw-------   1 anna     other     268716   Jan 05 10:03 katharina

$ pack katharina
pack: katharina: 41.7% Compression

$ ls -l
total 312
-rw-------   1 anna     other     156692   Jan 05 10:03 katharina.z

$ unpack katharina
unpack: katharina: unpacked

$ ls -l
total 532
-rw-------   1 anna     other     268716   Jan 05 10:03 katharina
```

SIEHE AUCH

cat, *compress*, *pcat*, *uncompress*, *pack*, *zcat*

unset
Shell-Variablen oder Shell-Funktionen aus der Umgebung löschen

Das in die Bourne-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.

```
unset name...
```

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.

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, *sh*, *ksh*

uucp
Dateien zwischen Unix-Systemen kopieren (Unix to Unix copy)

uucp kopiert Dateien innerhalb eines Unix-Systems oder zwischen verschiedenen Unix-Systemen.

Vorsicht

Die Menge der Dateien, auf die von einem fernen Rechner aus zugegriffen werden darf, kann (und sollte normalerweise aus Sicherheitsgründen!) stark eingeschränkt sein. Sie werden höchstwahrscheinlich nicht berechtigt sein, Dateien über den Pfadnamen zu holen; bitten Sie stattdessen den betreffenden Benutzer auf dem fernen System, Ihnen die Dateien zu senden. Aus den gleichen Gründen werden Sie wahrscheinlich nicht berechtigt sein, Dateien zu beliebigen Pfadnamen zu senden. Die Zugriffsberechtigungen werden ueber die Datei */usr/lib/uucp/USERFILE* geregelt.

Alle Dateien, die über *uucp* empfangen werden, haben als Eigentümer den UUCP-Verwalter.

uucp kopiert keine Dateien, bei denen für "Andere" kein Leserecht besteht; d.h. die Dateien müssen mindestens die Zugriffsrechte 0444 besitzen.

```
uucp[␣option]...␣ausgangsdatei␣...␣zieldatei
```

option

-a
Es soll kein *getwd()* ausgeführt werden, um das aktuelle Dateiverzeichnis zu finden. (Diese Option wird manchmal benutzt, um die Effizienz zu steigern.)

-c
Die Ausgangsdatei wird direkt nach *zieldatei* kopiert und nicht zuerst in das Spool-Dateiverzeichnis. (Diese Option ist standardmäßig gesetzt.)

-C
Die Ausgangsdatei wird zuerst in das Spool-Dateiverzeichnis kopiert und von dort aus nach *zieldatei*.

-d
Es werden alle nötigen Dateiverzeichnisse für das Kopieren angelegt. (Diese Option ist standardmäßig gesetzt.)

-f
Es werden keine Zwischendateiverzeichnisse für das Kopieren angelegt.

-ggrad
Mit dieser Option können Sie eine Priorität festlegen, mit der *uucp* die Dateien überträgt. *grad* ist ein Buchstabe oder eine Ziffer: 0,...,9,A,...,Z,a,...,z. 0 bezeichnet die höchste Priorität, *z* die niedrigste. Standardmäßig hat *uucp* die Priorität *n*. Bei umfangreichen Aufträgen sollten Sie eine niedrigere Priorität setzen.
Zum Vergleich: *uux* hat standardmäßig die Priorität *A*, Nachrichten werden per *mail* normalerweise mit Priorität *C* gesendet.

-m
Ist der Kopiervorgang beendet, wird eine Nachricht an den Benutzer geschickt, der *uucp* aufgerufen hat.

Die Option *-m* wirkt nur, wenn Sie Dateien senden oder eine einzige Datei empfangen. Wenn Sie mehrere Dateien empfangen, die über Shell-Sonderzeichen ?, *, [...] angesprochen wurden, wirkt die Option *-m* nicht.

-nbenutzerkennung
Der Benutzer am fernen System erhält eine Nachricht, daß eine Datei gesendet wurde.

-r
Die Dateiübertragung wird nicht gestartet, sondern nur der Auftrag in die Warteschlange gesetzt.

-sspooldvz
Das Dateiverzeichnis *spooldvz* wird anstelle des Standard-Spool-Dateiverzeichnisses als Spool-Dateiverzeichnis verwendet.

-xdebug
Die Fehlersuche (debugging) soll mit Level *debug* erfolgen. Das bedeutet: Wenn Fehler auftreten, werden diese auf der Standard-Fehlerausgabe kommentiert. Für *debug* können Sie eine Zahl von 0 bis 99 angeben. Je größer die Zahl, desto ausführlicher sind die Informationen, die Sie erhalten. Mit Level 9 erhalten Sie bereits sehr ausführliche Informationen, mit Level 99 erhalten Sie extrem viel Text.

ausgangsdatei
Datei, die kopiert werden soll. In welcher Form Sie die Ausgangsdateien angeben können, ist im Abschnitt *Angabe der Dateien* beschrieben.

zieldatei
Für *zieldatei* können Sie entweder eine Datei oder ein Dateiverzeichnis angeben. Geben Sie ein Dateiverzeichnis an, werden die Ausgangsdateien unter dem gleichen einfachen Dateinamen in das Dateiverzeichnis kopiert.
Die Zieldatei erhält die gleichen Zugriffsrechte wie die Ausgangsdatei, mindestens aber die Zugriffsrechte 0644.

In welcher Form Sie *zieldatei* angeben können, ist im Abschnitt *Angabe der Dateien* beschrieben.

Angabe der Dateien

Für *ausgangsdatei* und *zieldatei* können Sie angeben:

- entweder einen Pfadnamen; dieser Pfadname bezieht sich dann auf Ihren (d.h. den lokalen) Rechner
- oder eine Zeichenkette der Form

  ```
  systemname!pfadname
  ```

  wobei *systemname* ein Systemname sein muß, den *uucp* kennt; der Pfadname bezieht sich dann auf das angegegebene (ferne) System.

Die Pfadnamen können Sie auf eine der folgenden vier Arten angeben:

1. als absoluten Pfadnamen.
2. als relativen Pfadnamen; diesem wird dann das aktuelle Dateiverzeichnis vorangestellt.
3. als Pfadnamen, dem *~benutzerkennung* vorangestellt ist; dabei muß *benutzerkennung* eine Benutzerkennung sein, die auf dem betreffenden System existiert. *~benutzerkennung* wird dann durch das Login-Dateiverzeichnis des angegebenen Benutzers ersetzt.
4. in der Form *~/datei*; *datei* wird dann an das Public-Dateiverzeichnis des betreffenden Systems angehängt (normalerweise */var/spool/uucppublic*).

Wenn Sie innerhalb der Pfadnamen Shell-Sonderzeichen zur Dateinamen-Generierung ?, * und [...] angeben, so werden diese auf dem System expandiert, auf das sich der Pfadname bezieht.

Ist das Ergebnis der Pfadnamen-Auswertung ein fehlerhafter Pfadname für das lokale bzw. ferne System, kann nicht kopiert werden.

Wenn Sie auf ein System kopieren wollen, das mehrere "Stationen" entfernt ist, können Sie *mail* verwenden. Ein *uucp*-Aufruf der Form

uucp meinedatei system1!system2!system3!deinedatei

ist nicht erlaubt.

DATEIEN

*/usr/lib/uucp/**
UUCP-interne Dateien und nicht für den Anwender bestimmte Kommandos

/usr/lib/uucp/USERFILE
Liste der Dateibäume, auf die über UUCP zugegriffen werden darf

/var/spool/uucp
Spool-Dateiverzeichnis

BEISPIELE

Das Kommando

```
$ uucp /usr1/lit/datei brummbox!~fritz/kopie
```

sendet die Datei */usr1/lit/datei* des lokalen Systems zum System *brummbox* und schreibt den Inhalt dort in die Datei *kopie* im Login-Dateiverzeichnis des Benutzers *fritz*.

SIEHE AUCH

mail, *uux*

D.A.Nowitz and M.E.Lesk:
Unix Programmer's Manual, A Dial-Up Network of UNIX Systems [32]

D.A.Nowitz:
Unix Programmer's Manual, Uucp Implementation Description [33]

uudecode
Datei nach der Übertragung per mail decodieren (UUCP decode)

uuencode und *uudecode* werden zusammen benutzt, um eine Text- oder Binärdatei über UUCP-*mail* (oder anderes *mail*) zu senden. Sie können die Datei nicht nur direkt, sondern auch über eine *mail*-Kette von jeweils unmittelbar benachbarten Systemen senden.

uudecode liest eine codierte Datei, entfernt alle Zeilen am Anfang oder Ende, die das *mail*-System für die Übertragung hinzugefügt hat, 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.

Weitere Informationen siehe *uuencode*.

```
uudecode[ datei]
```

datei
Datei, die decodiert werden soll.

datei nicht angegeben:
uudecode liest von der Standard-Eingabe.

SIEHE AUCH

mail, *uucp*, *uuencode*, *uux*

uuencode
Datei für die Übertragung per mail codieren (UUCP encode)

uuencode und *uudecode* werden zusammen benutzt, um eine Text- oder Binärdatei über UUCP-*mail* (oder anderes *mail*) zu senden. Sie können die Datei nicht nur direkt, sondern auch über eine *mail*-Kette von jeweils unmittelbar benachbarten Systemen senden.

uuencode erhält als Eingabe eine Datei 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.

```
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.

Gebrauch

uuencode und *uudecode* sollten folgendermaßen eingesetzt werden:

Sie rufen *uuencode* wie folgt auf:

```
uuencode [ausgangsdatei] zieldatei | mail 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.

SIEHE AUCH

mail, *uucp*, *uudecode*, *uux*

uuglist
Service-Liste angeschlossener UNIX-Rechner

uuglist gibt eine Liste der verfügbaren Dienstleistungen aller am System angeschlossenen UNIX-Rechner aus. Diese Dienstleistungen benötigen Sie, um die Kommandos *uucp* und *uux* mit der Option *-g* zu verwenden.

```
uuglist[-u]
```

-u
Gibt die Liste der Dienstleistungen aus, die dem Benutzer für die Option *-g* der Kommandos *uucp* und *uux* zur Verfügung stehen.

DATEIEN

/usr/lib/uucp/Grades
Beinhaltet die Liste der graduell abgestuften Dienstleistungen

SIEHE AUCH

uucp, *uux*,

uulog
UUCP-Protokolldateien ausgeben (UUCP log files)

uulog fragt Aufzeichnungen über *uucp*- und *uux*-Transaktionen ab, die in entsprechenden Protokoll-Dateien im Dateibaum */var/spool/uucp/LOG* eingetragen sind.

Vorsicht

Die UUCP-Aufzeichnungen darüber, welcher Benutzer die jeweilige Transaktion gestartet hat, sind unter Umständen nicht richtig.

```
uulog[␣-s␣system][␣-f␣system][␣-x][␣-zahl]
```

-s␣system
uulog gibt Informationen über Transaktionen aus, an denen das angegebene System beteiligt war.

-f␣system
uulog wendet auf die Protokolldatei von *system* das Kommando *tail -f* an. *tail* können Sie dann nur durch Drücken von END verlassen.

-x
uulog fragt die Protokoll-Datei */var/spool/uucp/LOG/uuxqt* für das angegebene System ab. Diese Option kann nur in Kombination mit *-f* oder *-s* verwendet werden.

-zahl
Diese Option kann nur in Kombination mit *-f* verwendet werden. Dem *tail*-Kommando wird dann *zahl* als Zeilenanzahl übergeben.

DATEIEN

/var/spool/uucp/LOG/uucico
/var/spool/uucp/LOG/uucp
/var/spool/uucp/LOG/uux
/var/spool/uucp/LOG/uuxqt
Logdateien für UUCP-Aktivitäten

SIEHE AUCH

uucp, *uux*

uuname
Namen von UUCP-Systemen auflisten (UUCP names)

uuname listet die UUCP-Namen von allen bekannten Systemen auf.

```
uuname [-l]
```

-l
gibt den Namen des lokalen Systems aus; dieser Name kann sich von dem Systemnamen, den das Kommando *hostname* ausgibt, unterscheiden, falls der *hostname*-Name sehr lang ist.

SIEHE AUCH

uucp, *uux*

uupick
Dateiübertragung zwischen UNIX-Rechnern

uupick überprüft Dateien, die an den Benutzer gesendet werden, und nimmt sie an oder weist diese zurück.

```
uupick [option] senddateien bestimmungsort
```

option

-s
Durchsucht nur das Dateiverzeichnis *PUBDIR* nach Dateien, die vom fernen Rechner gesendet werden.

senddateien
Die Dateien (oder Teilbäume, wenn Sie ein Dateiverzeichnis angegeben haben) werden in ein allgemein zugängliches Dateiverzeichnis des Zielrechners geschrieben, das beim *uucp*-Kommando definiert sein muß. Standardmäßig heißt dieses Dateiverzeichnis /var/spool/uucppublic. Bestimmte Dateien werden in das Dateiverzeichnis *PUBDIR/receive/user/mysystem/files* geschrieben

bestimmungsort
Den Bestimmungsort der Datei geben Sie in der Form an:

rechner[!rechner]... !benutzer

Der angegebene Rechner muß bei *uucp* eingetragen sein (siehe *uuname*). *benutzer* ist die Benutzerkennung eines Benutzers am Zielrechner.

Wenn Sie Dateien angeben, durchsucht *uupick* das Dateiverzeichnis *PUBDIR* nach Dateien an den Benutzer. Für jeden gefundenen Eintrag (Datei oder Dateiverzeichnis) schreibt *uupick* folgende Meldung auf die Standardausgabe:

from *rechner name*: [datei] [dateiverzeichnis]

anschließend liest *uupick* von der Standardeingabe, um Daten zu dieser Datei einzugrenzen. Dabei können Sie folgende Angaben machen:

↵	nächsten Eintrag bearbeiten
d	lösche die eingetragene Datei
m [dir]	kopiere die eingetragene Datei in das Dateiverzeichnis *dir*. Wenn Sie das Dateiverzeichnis nicht mit kompletten Pfadnamen angeben, hängt *uupick* die Datei in ein dem aktuellen Arbeitsdateiverzeichnis nächstgelegenen Dateiverzeichnis an. Geben Sie kein Dateiverzeichnis an, wird die Datei standardmäßig an das aktuelle Ziel-Dateiverzeichnis angehängt.

a [dir] Wie *m [dir]*; es gilt jedoch für alle vom fernen Rechner gesendeten Dateien.

p schreibt den Inhalt einer Datei auf die Standard-Ausgabe.

q beendet *uupick* (wie END).

END beendet *uupick* (wie *q*).

!*kommando*
verläßt *uupick* vorübergehend, um ein Shell-Kommando auszuführen.

Vorsicht

Wenn Sie Dateien, die mit einem Punkt beginnen, verschicken wollen, müssen Sie den Punkt explizit angeben.

Beispiel:

Sie wollen die Datei *.profile* verschicken, aber aus Gründen der Zeitersparnis den Dateinamen mit Shell-Sonderzeichen generieren.

Richtig:	.profile	.prof*	.profil?
Falsch:	*prof*	?profile	

DATEIEN

var/spool/uucppublic
An dieses Dateiverzeichnis im Zielrechner werden die gesendeten Dateien und Dateiverzeichnisse standardmäßig angehängt.

PUBDIR/empfänger/benutzer/senderechner/dateien
Dieses Dateiverzeichnis erstellen Sie im Zielrechner, wenn das Standard-Dateiverzeichnis nicht definiert ist.

SIEHE AUCH

mail, *uucp*, *uustat*, *uux*, *uupick*
uucleanup [5]

uustat
Kontroll-Funktion zur Dateiübertragung in öffentlichen UNIX-Systemen

Mit *uustat* können Sie *uucp*-Kommandos steuern oder generelle Information zu *uucp*-Verbindungen abfragen. Der Verarbeitungsstand von bereits abgesendeten *uucp*-Kommandos kann abgefragt werden. Sie können die Kommandos auch abbrechen. Generell können aktuelle Leistungsinformationen (wie durchschnittliche Übertragungsleistung, Zeiten in der Warteschlange) oder die allgemeinen system- und benutzerspezifischen Daten der *uucp*-Verbindungen des fernen Systems ausgegeben werden.

```
uustat[␣option]                                    Format1
uustat[␣option[␣option]]                           Format2
uustat␣option[␣option]                             Format3
```

Keine Option angegeben
uustat gibt den Zustand aller *uucp*-Anfragen aus, die Sie als aktueller Benutzer erteilt haben.

Format 1: Informationen zu Auftragssteuerung und Prozeßdaten

uustat[-q][-m][-p][-ubenutzer][Sqric]

Diese Optionen können Sie jeweils nur einzeln angeben.

-m
Zeigt die Zugriffsmöglichkeit auf alle Rechner an.

-p
Das Kommando *ps-flp* (ausführliche Prozeßdaten auflisten) wird für alle Prozeßnummern, die sich in Sperrdateien befinden, ausgeführt.

-q
Listet die Aufträge in den Warteschlangen der einzelnen Rechner auf. Falls ein Status-Feld für einen Rechner existiert, werden Datum, Uhrzeit und Informationen zum Status ausgegeben. Wenn eine Zahl die Angaben zu *C*- oder *X*-Dateien ergänzt, zeigt diese das Alter der ältesten *C*- bzw. *X*--Datei dieses Systems in Tagen an.

Aus dem Retry-Feld ist zu ersehen, in wievielen Stunden der nächste Aufruf möglich ist.

Das Zähl-Feld zeigt die Anzahl der Fehlversuche beim Aufruf.
Beachten Sie: in Systemen mit einer entsprechenden Anzahl ausstehender Aufträge kann die Ausführung um 30 Sekunden oder mehr (Echtzeit) verzögert sein.

-Sqric
Gibt den Zustand eines Auftrags aus; dabei bedeutet:

q: Auftrag in die Warteschlange eingereiht,
die Übertragung findet noch statt.

r: Auftrag läuft gerade
Von Beginn der Übertragung ab läuft der Auftrag

i: Auftrag unterbrochen
Die Übertragung wurde abgebrochen, bevor der ganze Auftrag beendet war.

c: Auftrag ausgeführt.
Der Auftrag ist komplett übertragen.

Die vollständigen Zustandsinformationen sind im Abrechnungs-Logbuch enthalten, das aber nicht automatisch angelegt wird und daher nicht vorausgesetzt werden kann. Die Parameter können in beliebiger Kombination verwendet werden, nötig ist immer mindestens einer davon. Die Option *-S* kann mit den Optionen -s*ystem* und -u*benutzer* kombiniert werden. Die Ausgabe entspricht in diesem Fall der von *-s* und *-u*, nur der Zustand der Aufträge ist an die letzte Stelle der Ausgabe gesetzt.

-ubenutzer
Mit dieser Option können Sie sich über den Stand aller *uucp*-Anfragen, die Ihnen zugeordnet sind, informieren.

Format 2: Systemeingriffe und -meldungen

uustat[-kauftrags-nr[␣-n]][-a][-rauftrags-nr[␣-n]][-ssystem[␣-j]]

Die Optionen -k*auftragsnr*, -r*auftragsnr*, -s*ystem* dürfen nur einzeln angegeben werden. Die Option *-n* funktioniert nur in Verbindung mit -k*auftragsnr* oder -r*auftragsnr*, die Option *-j* nur mit -s*ystem*.

-a
Alle Aufträge, die sich in der Warteschlange befinden, werden genannt.

-j
Die Anzahl der angezeigten Aufträge wird ausgegeben. Diese Option kann nur in Verbindung mit den Optionen -a oder -s*ystem* verwendet werden.

-kauftragsnr
uucp bricht die Übertragung des unter *auftragsnr* laufenden Prozesses ab, sofern Sie der Eigentümer der Übertragung sind. Der Systemverwalter oder *uucp*-Administrator kann von anderen Benutzern gestartete Funktionen abbrechen; in diesem Fall wird an den Eigentümer eine Meldung über den Abbruch geschickt.

-n
Unterdrückt die Ausgabe auf die Standardausgabe, die Standardfehlerausgabe bleibt davon unberührt. Diese Option funktioniert nur in Verbindung mit den Optionen -k*auftragsnr* und -r*auftragsnr*.

-rauftragsnr
r*auftragsnr* setzt für die Dateien, die mit *auftragsnr* bezeichnet werden, die Zeitangabe der letzten Bearbeitung auf die aktuelle Zeit. Dies verhindert, daß der Reinigungsdämon den unveränderten Auftrag als veraltet ausmustert.

-ssytem
Der Stand aller *uucp-Anfragen* an ferne Syteme wird ausgegeben. In Verbindung mit der Option *-j* wird die Gesamtanzahl der Aufträge ausgegeben.

Format 3: Leistungsdaten des fernen Systems

uustat␣tsystem[␣-dnr][␣-c]

Beachten Sie bei der Fehlersuche, daß *-t*-Optionen keine Meldung ausgeben, wenn die für die Berechnung nötigen Daten nicht gespeichert sind.

-c
Die Option kann nur in Verbindung mit der Option -t*system* verwendet werden. Wenn *-c* gesetzt ist, wird die durchschnittliche Zeit in der Warteschlange berechnet, wenn nicht, wird die durchschnittliche Übertragungsleistung berechnet.

-dnummer
Die Option kann nur in Verbindung mit der Option -*t*system verwendet werden. Für *nummer* setzen Sie die Anzahl der Minuten ein. Damit überschreiben Sie den Default-Wert von 60 Minuten, der bei Berechnungen verwendet wird. Diese Berechnungen basieren auf Daten aus dem Optionalen Performance Logbuch, die nicht immer zur Verfügung stehen; sie können nur von der letzten Bereinigung des Performance-Logbuchs an durchgeführt werden.

-tsystem
Gibt die durchschnittliche Übertragungsrate oder die in der Warteschlange verbrachte Zeit des fernen Systems *system* aus.

Beachten Sie:

Nachdem der Benutzer eine *uucp*-Anfrage erteilt hat, gibt *uustat* eine Dateigröße von -99999 byte aus, wenn die angegebene Datei übertragen oder gelöscht wurde, oder wenn sie bei der *uucu*-Anfrage nicht mit der Option *-c* ins Spoolverzeichnis übertragen wurde. Ein solcher Auftrag scheitert, wenn die zu übertragenden Dateien nicht gefunden wurden.

DATEIEN

*var/spool/uucp/**
Spoolverzeichnisse

var/uucp/.Admin/account
Datei mit den Abrechnungsdaten

var/uucp/.Admin/perflog
Datei mit den Leistungsdaten

BEISPIEL

Das Kommando

```
$uustat -teagle -d50 -c
```

erzeugt eine Ausgabe der folgenden Form:

```
Last change: Basic Network Utilities average queue time to eagle for last minu-
tes: 5 seconds

Letzte Änderung: Basic Network Utilities Durchschnittliche Wartezeit für Rech-
ner eagle in den letzten Minuten: 5 Sekunden
```

Mit der Verwendung der Option *-c* als Ergänzung zur Option *-t*, ist in der Ausgabe die durchschnittliche Wartezeit für den Rechner *eagle* enthalten (5 Sekunden).

Dasselbe Kommando ohne die Option *-c* erzeugt die Ausgabe:

```
average transfer rate with eagle for last 50 minutes:  2000.88 bytes/sec

Durchschnittliche Übertragungsleistung zum Rechner eagle in den  letzten 50
Minuten:  2000.88 bytes/sec
```

Die Verwendung der Optionen -s*system* und -u*benutzer* erzeugt Ausgaben in der folgenden Form:

```
eagleN1bd7    4/07-11:07  S  eagle   dan  522   /home/dan/A
eagleN1bd7    4/07-11:07  S  eagle   dan   59   D.3b2a12ce4924
              4/07-11:07  S  eagle   dan  rmail mike

AdlerN1bd7    4/07-11:07  S  Adler1  dan  522   /home/dan/A
AdlerN1bd7    4/07-11:07  S  Adler1  dan   59   D.3b2a12ce4924
              4/07-11:07  S  Adler1  dan  rmail josef
```

In den einzelnen Spalten stehen folgende Informationen:

Spalte 1: Auftragsnummer
Spalte 2: Datum und Uhrzeit
Spalte 3: **S** = Datei wurde gesendet
R = auf Datei wurde mit einer Abfrage zugegriffen
Spalte 4: Zielrechner
Spalte 5: Benutzernummer bzw. Benutzer, der den Auftrag erteilt hat
Spalte 6: Dateigröße (in byte) bzw. ferne Übertragung vorgesehen (rmail)
Spalte 7: Dateiname

Ist die der Dateigröße angegeben, enthält diese Spalte den vom Benutzer vergegebenen (z.B. josef) oder einen systeminternen Namen (wie z.B. D.3b2a12ce4924). Systeminterne Namen werden im Falle einer fernen Ausführung (rmail) erzeugt.

SIEHE AUCH

uucp

uuto
Dateiübertragung zwischen UNIX-Rechnern

uuto sendet die angegebenen Dateien an den Bestimmungsort. Dabei verwendet es die Möglichkeiten, die *uucp* zum Versenden von Dateien benutzt, erlaubt jedoch darüberhinaus dem lokalen System, den Zugriff auf die Datei zu kontrollieren.

```
uuto[option]␣senddateien␣bestimmungsort
```

option

-p
Kopiert die Sendedatei vor der Übertragung ins Spool-Dateiverzeichnis.

-m
Sobald der Sendeauftrag erfolgreich ausgeführt wurde, schickt *uuto* eine Nachricht davon in den Standard-Briefkasten des Senders.

senddateien
Die Dateien (oder Teilbäume, wenn Sie ein Dateiverzeichnis angegeben haben) werden in ein öffentliches Dateiverzeichnis des Zielrechners geschrieben, das beim *uucp*-Kommando definiert sein muß. Standardmäßig heißt dieses Dateiverzeichnis /var/spool/uucppublic. Wenn Sie ein Dateiverzeichnis in der Form *PUBDIR/receive/user/mysystem/files* angeben, werden die kopierten Dateien dort angehängt.

bestimmungsort
Den Bestimmungsort der Datei geben Sie in der Form an:

rechner[!rechner]... !benutzer

Der angegebene Rechner muß bei *uucp* eingetragen sein (siehe *uuname*). *benutzer* ist die Benutzerkennung eines Benutzers am Zielrechner.

Vorsicht

Wenn Sie Dateien, die mit einem Punkt beginnen, verschicken wollen, müssen Sie den Punkt explizit angeben.

Richtig:	.profile	.prof*	.profil?
Falsch:	*prof*	?profile	

DATEIEN

var/spool/uucppublic
An dieses Dateiverzeichnis im Zielrechner werden die gesendeten Dateien und Dateiverzeichnisse standardmäßig angehängt.

PUBDIR/empfänger/benutzer/senderechner/dateien
Dieses Dateiverzeichnis erstellen Sie im Zielrechner, wenn das Standard-Dateiverzeichnis nicht definiert ist.

SIEHE AUCH

mail, *uucp*, *uustat*, *uux*, *uupick*
uucleanup [5]

uux
Kommando auf fernem System ausführen (Unix to Unix command execution)

uux holt null, eine oder mehrere Dateien von verschiedenen Systemen, führt ein Kommando auf einem bestimmten System aus und sendet die Standard-Ausgabe dieses Kommandos zu einer Datei auf einem bestimmten System.

uux benachrichtigt Sie über *mail*, wenn das Kommando auf dem fernen System ausgeführt worden ist. Durch entsprechende Einträge in der Konfigurationsdatei */usr/lib/uucp/L.cmds* können Sie diese Benachrichtigung unterbinden oder einschränken auf die Fälle, in denen das Kommando auf dem fernen System nicht ausgeführt werden kann. Ebenso können Sie diese Benachrichtigung mit der Option *-n* für den Einzelfall verhindern.

Vorsicht

Aus Sicherheitsgründen ist die Menge der Kommandos, die über *uux* ausgeführt werden dürfen, meist stark eingeschränkt. Oft ist über *uux* nicht viel mehr erlaubt, als Nachrichten zu empfangen (siehe *mail*). Welche Kommandos ausgeführt werden dürfen, wird in der Konfigurationsdatei */usr/lib/uucp/L.cmds* festgelegt.

```
uux [option] ... kommando
```

option

-
Die Standard-Eingabe von *uux* wird zur Standard-Eingabe des angegebenen Kommandos.

-aname
Eine Benachrichtigung über die Ausführung des Kommandos auf dem fernen System wird statt an die Benutzerkennung des *uux*-Aufrufers an die Benutzerkennung *name* geschickt.

-c
Lokale Dateien werden vor der Dateiübertragung auf das ferne System nicht in das Spool-Dateiverzeichnis kopiert.
Diese Option ist standardmäßig gesetzt.

-C
Lokale Dateien werden vor der Dateiübertragung auf das ferne System in das Spool-Dateiverzeichnis kopiert.

-ggrad
Mit dieser Option können Sie eine Priorität festlegen, mit der *uux* die Dateien über-

trägt. *grad* ist ein Buchstabe oder eine Ziffer: 0,...,9,A,...,Z,a,...,z. 0 bezeichnet die höchste Priorität, *z* die niedrigste. Standardmäßig hat *uux* die Priorität *A*. Bei umfangreichen Aufträgen sollten Sie eine niedrigere Priorität setzen.
Zum Vergleich: *uucp* hat standardmäßig die Priorität *n*, Nachrichten werden per *mail* normalerweise mit Priorität *C* gesendet.

-l

Es wird versucht, einen Verweis von der Originaldatei, die geholt werden soll, in das Spool-Dateiverzeichnis zu erzeugen. Ist das nicht möglich, wird die Datei in das Spool-Dateiverzeichnis kopiert.

-L

uucico wird mit Option *-L* aufgerufen. Dies bewirkt, daß Kommandos nur in Systemen eines *lokalen* Verbundes aufgerufen werden (siehe *uucico*).

-n

Sie erhalten auf keinen Fall eine Nachricht, auch nicht, wenn das Kommando auf dem fernen System nicht erfolgreich ausgeführt werden konnte.

-p

Gleiche Wirkung wie Option -:
Die Standard-Eingabe von *uux* wird zur Standard-Eingabe des angegebenen Kommandos.

-r

Die Dateiübertragung wird nicht gestartet, sondern nur der Auftrag in die Warteschlange gesetzt.

-xdebug

Die Fehlersuche (debugging) soll mit Level *debug* erfolgen. Das bedeutet: Wenn Fehler auftreten, werden diese auf der Standard-Fehlerausgabe kommentiert. Für *debug* können Sie eine Zahl von 0 bis 99 angeben. Je größer die Zahl, desto ausführlicher sind die Informationen, die Sie erhalten. Mit Level 9 erhalten Sie bereits sehr ausführliche Informationen, mit Level 99 erhalten Sie extrem viel Text.

-z

Sie werden nur dann benachrichtigt, wenn das angegebene Kommando nicht erfolgreich ausgeführt wurde.

kommando

Kommando, das ausgeführt werden soll.
kommando besteht aus einem oder mehreren Argumenten, die wie eine Shell-Kommandozeile aussehen, nur muß den Kommando- und Dateinamen ! oder *systemname*! vorangestellt werden. Fehlt *systemname*, so bezieht sich das betreffende Argument auf das lokale System, sonst auf das angegebene System.

Wenn Sie für *kommando* eine Shell-Pipeline angeben, können Sie nur für das erste Kommando der Pipeline einen Systemnamen angeben; alle anderen Kommandos werden auf dem System des ersten Kommandos ausgeführt.

Dateinamen können Sie auf eine der folgenden vier Arten angeben:

1. als absoluten Pfadnamen.
2. als relativen Pfadnamen; diesem wird dann das aktuelle Dateiverzeichnis vorangestellt.
3. als Pfadnamen, dem *~benutzerkennung* vorangestellt ist; dabei muß *benutzerkennung* eine Benutzerkennung sein, die auf dem betreffenden System existiert. *~benutzerkennung* wird dann durch das Login-Dateiverzeichnis des angegebenen Benutzers ersetzt.
4. als Pfadnamen der Form *~/datei*; *datei* wird dann an das Public-Dateiverzeichnis des betreffenden Systems angehängt (normalerweise */var/spool/uucppublic*).

Wenn Sie innerhalb der Pfadnamen Shell-Sonderzeichen zur Dateinamen-Generierung ?, * und [...] angeben, so werden diese auf dem System expandiert, auf das sich der Pfadname bezieht.

Shell-Sonderzeichen wie z.B. <>;|, müssen entwertet werden, indem Sie entweder das ganze *kommando* oder nur die Sonderzeichen in Anführungszeichen "..." einschließen.
Das Shell-Sonderzeichen * sollten Sie nicht verwenden, da es wahrscheinlich nicht das gewünschte Ergebnis liefern wird. Die Shell-Ausdrücke << und >> werden nicht unterstützt.

uux versucht, alle angegebenen Dateien auf das System zu holen, wo das Kommando ausgeführt wird. Ausgabedateien auf einem anderen System müssen Sie deshalb kennzeichnen, indem Sie das betreffende Argument in \(...\) einschließen, z.B. \(system!/usr/datei\).

Siehe auch *Beispiel 1* und *Beispiel 2*.

DATEIEN

*/usr/lib/uucp/**
UUCP-interne Dateien und nicht für den Anwender bestimmte Kommandos (siehe *uucico, DATEIEN*)

/usr/lib/uucp/L.cmds
Liste der Kommandos, die von fernen Systemen aus über *uux* ausgeführt werden dürfen

/var/spool/uucp
Spool-Dateiverzeichnis

BEISPIELE

1. Das Kommando

```
$ uux "!diff usg!/usr/dan/datei1 pwba!/a4/dan/datei2 >
> !~/dan/datei.diff"
```

holt die Dateien */usr/dan/datei1* und */a4/dan/datei2* von den Rechnern *usg* bzw. *pwba* in das lokale System, führt das Kommando *diff* auf dem lokalen System aus und schreibt die Standard-Ausgabe von *diff* in die lokale Datei */var/spool/uucppublic/dan/datei.diff*.

2. Das Kommando

```
$ uux a!wc b!/usr/datei1 \(c!/usr/datei2\)
```

holt die Datei */usr/datei1* vom System *b* in das System *a*, führt auf dem System *a* das Kommando *wc* aus und schreibt die Standard-Ausgabe von *wc* in die Datei */usr/datei2* auf dem System *c*.

SIEHE AUCH

uucp

vacation
Post automatisch speichern und beantworten

Erreicht neue Post den Briefkasten des Empfängers, überprüft das Kommando *mail* zuerst, ob dort ein Vermerk existiert, der anzeigt, daß die Post an einen anderen Empfänger oder an ein Kommando übergeben werden soll (siehe *mail*). *vacation* vermerkt in diesem Briefkasten, daß die Post in einen Urlaubsbriefkasten weitergeschickt werden soll und schickt dem Sender eine automatische Antwort zu. Die Post wird im Urlaubsbriefkasten gespeichert.

Um die Funktion von *vacation* wieder aufzuheben, müssen Sie folgendes Kommando eingeben: `mail n-F ""`

```
vacation[␣-l␣logdat][␣-m␣postdat][␣-M␣antwdat][␣-F␣stellvertreter][␣-d]
```

-l␣logdat
Datei, in die *vacation* die Namen der Sender schreibt, die bereits eine automatische Antwort erhalten haben.

-l␣logdat nicht angegeben:
vacation benutzt die Datei *$HOME/.maillog.*

-m␣postdat
Urlaubsbriefkastendatei, in der die Post gespeichert wird. Die Datei soll nicht den Namen der Standard-Briefkastendatei */var/mail/*benutzerkennung haben, da diese nur entweder Post oder Vermerke zum Weiterleiten der Post enthalten kann.

-m␣postdat nicht angegeben:
vacation benutzt die Datei *$HOME/.mailfile.*

-M␣antwdat
Datei, in der die Antwort steht, die automatisch an den Sender geschickt wird.

-M␣antwdat nicht angegeben:
vacation benutzt die Datei */usr/share/lib/mail/std␣vac␣msg.* Diese Datei hat folgenden Inhalt:

```
Subject: AUTOANSWERED!!!

I am on vacation. I will read (and answer if necessary)
your e-mail message when I return.

This message was generated automatically and you will
receive it only once, although all messages you send
me while I am away WILL be saved.
```

Das bedeutet:

Betreff: Automatische Antwort !!!

Ich bin im Urlaub. Ich werde Ihre elektronische Post lesen (und wenn notwendig beantworten), wenn ich zurückkomme.

Diese Nachricht wurde automatisch erzeugt und Sie werden sie nur einmal bekommen. Es wird jedoch die gesamte Post, die Sie mir während meiner Abwesenheit schicken, gespeichert.

-F␣stellvertreter
Für *stellvertreter* geben Sie eine Benutzerkennung an, an die Ihre Post geschickt wird, wenn es bei der Übertragung in die Urlaubsbriefkastendatei Probleme gibt. Die Post wird dann nicht an den Sender zurückgeschickt.

-F␣fehlsicher nicht angegeben:
Bei Problemen wird die Post zum Sender zurückgeschickt.

-d
logdat wird mit dem aktuellen Datum versehen.

DATEIEN

$HOME/.maillog
enthält standardmäßig die Namen der Sender, die bereits eine automatische Antwort erhalten haben.

$HOME/.mailfile
standardmäßiger Urlaubsbriefkasten.

/var/mail/benutzerkennung
standardmäßiger Briefkasten des Benutzers.

/usr/share/lib/mail/std_vac_msg
enthält den standardmäßigen Antworttext.

*/tmp/notif**
temporäre Zwischenspeicherungsdatei.

/usr/lib/mail/vacation2
Shell-Script, das vom Kommando *vacation* aufgerufen wird.

BEISPIEL

Sie wollen während Ihrer Abwesenheit Ihre Post in der Datei */usr/mail/urlaub* speichern. Die Antwort, die Sie dem Sender automatisch zuschicken wollen, soll in der Datei */usr/mail/antwort* stehen. Jeder Sender, der diese Antwort erhalten hat, soll in die Datei */usr/mail/sender/antwort* eingetragen werden.

```
$ vacation -l /usr/mail/sender/antwort -m /usr/mail/urlaub -M /usr/mail/antwort
```

SIEHE AUCH

mail

Leitfaden für Benutzer [2]

vi Bildschirmorientierter Editor (visual)

Der *vi* (visual) ist ein bildschirmorientierter Texteditor

Aufbau dieser Beschreibung

Nach der Beschreibung des Aufrufs von *vi* auf SINIX-Ebene finden Sie folgende Abschnitte:

Arbeitsweise des *vi*

- Bildschirmaufbau
- Akustisches Signal
- Modi des *vi*
- *ex*-Kommandos eingeben
- Editor-Puffer sichern und *vi* verlassen
- Positionieranweisungen
- Textpuffer des *vi*
- Voreinstellung des *vi*
- Anpassung des *vi* an die Datensichtstation
- Signalbehandlung

vi-Kommandos

- Definitionen
- Control-Kommandos
- Kommandos des *vi*-Kommandomodus

Übersicht

Der *vi* zeigt auf dem Bildschirm ein Fenster von maximal 23 Zeilen in die editierte Datei. Er bietet Ihnen folgende Möglichkeiten:

- Text erstellen, ändern, kopieren, löschen
- Viele Kommandos zur Positionierung der Schreibmarke (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
- Subshell aufrufen
- Mit einem SINIX-Kommando einen Bereich der aktuellen Datei bearbeiten
- Mehrere Dateien während einer Sitzung bearbeiten und Text von einer Datei zu einer anderen 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.

```
vi[␣option]...[␣datei]...
view[␣option]...[␣datei]...
vedit[␣option]...[␣datei]...
```

view verhält sich wie *vi*, außer daß *datei* nur zum Lesen geöffnet ist und nicht verändert werden darf.

vedit ist eine spezielle Version von *vi* für Anfänger. *vedit* verhält sich wie *vi*, wenn die Optionen *report=1*, *showmode* und *nomagic* (siehe *ex*) und *novice* gesetzt sind.

option

-t␣markierung
(t - tag) Die Datei mit *markierung* wird zum Editieren aufgerufen. Der Editor positioniert die Zeile mit der Definition der Markierung in der 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
Der Lisp-Modus wird eingeschaltet. Der Text wird so eingerückt, daß er ein für Lisp-Programme typisches Aussehen bekommt. Die folgenden *vi*-Kommandos erhalten eine Bedeutung für Lisp: (,), {, }, [[und]].

-r␣datei
(r - recover) Mit der Option *-r* können Sie Ihre Sitzung wiederherstellen, falls das System oder der *vi* während der Sitzung abgestürzt ist.
Bei einem Absturz des Systems oder des *vi*-Editors werden die Änderungen, die nur im Editor-Puffer stehen, nicht in die Datei geschrieben. SINIX versucht jedoch den Inhalt des Puffers zu retten, indem eine Kopie des Pufferinhalts angelegt wird - falls möglich. *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
Nach einem Absturz des Systems oder des Editors wird eine Liste aller geretteten Dateien ausgegeben.

-r ohne Option liefert dasselbe Ergebnis wie *-L*.

-R
(R - Read-only) *datei* wird nur zum Lesen geöffnet. Damit können Sie ein versehentliches Überschreiben von *datei* verhindern.

Vorsicht

Die Datei kann mit dem *ex*-Kommando *w!* (siehe *Arbeitsweise des vi, ex-Kommandos eingeben*) dennoch überschrieben werden.

-wn

Der Bildschirm wird aus *n* Textzeilen aufgebaut.

-wn nicht angegeben:
Die Bildschirmlänge wird durch die Umgebungsvariable *LINES* festgelegt. Ist diese nicht definiert, wird der Bildschirm aus 23 Textzeilen aufgebaut.

-c␣kommando

Mit der Option *-c* können Sie beim Aufruf des *vi* auf eine bestimmte Zeile der zu editierenden Datei positionieren oder ein *ex*-Kommando ausführen lassen. 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 nichts anderes bewirkt (z.B. Suchen).

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 *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 Hochkommata '*muster*' einschließen, damit die Shell die Sonderzeichen nicht interpretiert.

'ex-kommando'
"ex-kommando"

ex-kommando kann irgendein *ex*-Kommando sein. Das *ex*-Kommando muß in Hochkommata 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 und der *vi*-Modus angezeigt (vgl. das Beispiel bei *ex*).

-x

Verschlüsselungsoption: *vi* führt beim Aufruf das *ex*-Kommando *X* aus und fragt Sie nach einem Schlüssel. Dieser Schlüssel wird dann zum Ent- und Verschlüsseln von Text durch den Algorithmus des Kommandos *crypt* verwendet. *vi* klärt ab, ob der einzulesende Text verschlüsselt ist oder nicht. Die temporäre Pufferdatei wird ebenfalls mit einer Transformation des eingegebenen Schlüssels verschlüsselt. (Siehe auch Kommando *crypt*.)

-C

Verschlüsselungsoption: wie *-x*, nur daß *vi* das *ex*-Kommando *C* ausführt. Das Kommando *C* entspricht dem Kommando *X* mit dem Unterschied: es wird angenommen, daß jeder zu lesende Text verschlüsselt ist. (Siehe auch Kommando *crypt*.)

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*). Wenn Sie keine Datei angeben, arbeiten Sie zunächst mit einem leeren Editor-Puffer. Den Inhalt dieses Puffers können Sie dann mit dem *ex*-Kommando *w datei* in die Datei *datei* schreiben.

ARBEITSWEISE DES VI

Der *vi* arbeitet immer mit einem Editor-Puffer. Bei Beginn einer *vi*-Sitzung wird eine Kopie der Datei, die Sie bearbeiten, im Editor-Puffer angelegt. Falls Sie keinen Dateinamen angeben oder die Datei noch nicht existiert, beginnen Sie mit einem leeren Editor-Puffer. Während einer Sitzung erfolgen alle Änderungen am Editor-Puffer. Die Orginaldatei wird erst verändert, wenn Sie den Inhalt des Editor-Puffers 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 Editor-Puffer in die Datei geschrieben wird.

Bildschirmaufbau

Der Bildschirm hat 24 Zeilen und 80 Spalten.

Davon dienen standardmäßig 23 Bildschirm-Zeilen der Darstellung des Textes der editierten Datei; die unterste Zeile auf dem Bildschirm ist eine Statuszeile. 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.

Als Markierung der ersten 23 Bildschirmzeilen, die zur Darstellung des Textes dienen, verwendet der *vi* in Spalte 1 folgende Zeichen:

~

Die Tilde ~ kennzeichnet Zeilen, die auf dem Bildschirm, aber nicht im Editor-Puffer 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 korrekt darstellen kann:

Erstens können zwar 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. (Denn eine Textzeile kann ja mehrere Zeilen auf dem Bildschirm bedeuten.)

Zweitens kann es sein, daß der *vi* nach dem Löschen von einzelnen Zeilen den Bildschirm nicht wieder neu aufbaut. In diesem Fall können Sie den Bildschirm mit dem *vi*-Kommando CTRL l neu aufbauen.

Die letzte Zeile auf dem Bildschirm (Zeile 24) 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).

Akustisches Signal

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.

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 können Sie Text im Editor-Puffer ergänzen oder ändern. 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 *vi-Kommandos, Kommandos des vi-Kommandomodus*). Alle folgenden Eingabezeichen, auch verschiedene nicht-druckbare Zeichen, werden auf den Bildschirm und in den Editor-Puffer geschrieben. Im Eingabemodus können Sie die meisten *vi*-Kommandos nicht verwenden. Es gibt jedoch einige wenige *vi*-Kommandos, die im Eingabemodus eine spezielle Bedeutung haben. Dies 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 müssen Sie eingeben, 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
Bei automatischer Einrückung um einen "Tabulatorschritt" nach links positionieren. Die Schrittbreite können Sie mit der Option *shiftwidth* festlegen. Dieses Kommando ist nur am Anfang einer neuen Eingabezeile möglich, also z.B. sinnvoll mit gesetzter Option *autoindent*.

CTRL h
1 Zeichen zurück.

CTRL t
1 Tabulatorschritt nach rechts, entsprechend *shiftwidth*. Dies ist nur am Anfang einer neuen Eingabezeile möglich.

CTRL v
Nicht druckbare Zeichen einfügen.

CTRL w
1 Wort zurück.

\
Escape für Korrekturzeichen ⌫.

Allerdings können Sie die Schreibmarke mit den meisten dieser Kommandos nur im Bereich des gerade eingegebenen Textes bewegen und diesen nicht verlassen.

vi-Zeilen-Kommandomodus

Im *vi*-Zeilen-Kommandomodus können Sie *vi*-, *ex*- und SINIX-Kommandos in der Statuszeile eingeben. Den Zeilen-Kommandomodus schalten Sie ein durch Eingabe eines Doppelpunktes :, eines Schrägstriches / oder eines Fragezeichens ?. 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 kann die Eingabe abgebrochen werden.

:

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 Statuszeile, und der Doppelpunkt und das eingegebene Kommando werden in der untersten Statuszeile angezeigt. Fast alle *ex*-Kommandos können verwendet werden; ausgenommen sind davon z.B. die *ex*-Kommandos, die den *ex* in den *ex*-Eingabemodus schalten würden.
Für den Namen der im Editor-Puffer 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.

/, ?

Nach Eingabe eines Schrägstriches / bzw. Fragezeichens ? springt die Schreibmarke in die Statuszeile, und Sie können einen regulären Ausdruck angeben, nach dem der *vi* vorwärts (bei /) oder rückwärts (bei ?) 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 noch nicht in die Statuszeile. Erst nachdem Sie eine *vi*-Positionieranweisung auf eine Zeile gegeben haben, springt die Schreibmarke in die Statuszeile, und 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 SINIX-Kommando eingeben, das mit ↵ oder ESC abgeschickt werden muß. Der Bereich von der aktuellen Zeile bis zur angegebenen Position wird, falls das SINIX-Kommando von der Standardeingabe liest, zur Eingabe des SINIX-Kommandos und nach Ende des Kommandos durch die Ausgabe (Standard-Ausgabe und Standard-Fehlerausgabe) des SINIX-Kommandos ersetzt. Falls das Kommando nichts ausgibt (z.B. *true*), wird der angegebene Bereich gelöscht, also durch "nichts" 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*. Zurück in den *vi* gelangen Sie, wenn 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 Editor-Puffer geschrieben. Sie beenden diesen Modus, indem Sie eine einzelne Zeile mit einem Punkt in der ersten Spalte eingeben.

ex-Kommandos 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.

Editor-Puffer sichern und vi verlassen

Um den Editor-Puffer 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 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 *vi* zu verlassen, haben Sie folgende Möglichkeiten:

q
(q - quit, *ex*-Kommando) *vi* 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*-Kommando) *vi* verlassen; am Editor-Puffer vorgenommene Änderungen gehen verloren.

x
ZZ
vi verlassen und den geänderten Editor-Puffer in die editierte Datei schreiben. *x* ist ein *ex*-Kommando; *ZZ* ist ein *vi*-Kommando.

wq[␣datei]
(wq - write and quit, *ex*-Kommando) *vi* verlassen und den Editor-Puffer in die angegebene Datei *datei* schreiben.

datei nicht angegeben:
Der Inhalt des Editor-Puffers wird in die editierte Datei geschrieben.

Positionieranweisungen

Der *vi* bietet eine große Menge von Kommandos und Suchkommandos zur Positionierung der Schreibmarke. Eine Positionieranweisung wird im *vi*-Kommandomodus gegeben.
In der folgenden Übersicht sind die wichtigsten Positionieranweisungen zusammengestellt. Die Begriffe Absatz, Abschnitt, Satz, Wort und Langwort sind im Abschnitt *vi-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]fzeichen
[zahl]Fzeichen
auf *zahl*-tes Zeichen *zeichen* positionieren, rückwärts positionieren

[zahl]tzeichen
[zahl]Tzeichen
vor (hinter) *zahl*-tes Zeichen *zeichen* (rückwärts) positionieren

Wortweise positionieren

[zahl]**b**
[zahl]B
auf den Anfang des *zahl*-ten vorhergehenden Wortes, Langwortes

[zahl]**e**
[zahl]E
auf das Ende des *zahl*-ten Wortes, Langwortes

[zahl]**w**
[zahl]W
auf den Anfang des *zahl*-ten Wortes, 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 aktuellen Absatzes

[zahl]**}**
auf den Anfang des nächsten Absatzes

[zahl]**[[**
zurück auf *zahl*-te vorangehende Abschnittsgrenze

[zahl]**]]**
auf die *zahl*-te Abschnittsgrenze

%
zur korrespondierenden Klammer

Bildschirmbezogen positionieren

[zahl]**H**
auf das erste Zeichen in der *zahl*-ten Textzeile auf dem Bildschirm

M
auf Bildschirmmitte

[zahl]**L**
auf das erste Zeichen in der *zahlt*-letzten Textzeile auf dem Bildschirm

[zeile]**z+**
[zeile]**z.**
[zeile]**z-**
aktuelle Zeile oder Zeile mit Nummer *zeile* an den oberen Bildschirmrand, die Bildschirmmitte, den unteren Bildschirmrand positionieren

Muster und Marken suchen

/Muster
?Muster
auf *Muster* positionieren, rückwärts positionieren

'marke
auf markierte Zeile positionieren (Zeilenanfang)

'marke
auf Marke positionieren

Text-Puffer des vi

Der *vi* verfügt über

- einen temporären Puffer
- 9 numerierte Puffer und
- 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 Editor-Puffer 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 deshalb Ihr letztes Kommando, das den Editor-Puffer 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 sonstige 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.

Vorsicht

Bei jedem Kommando, das den Editor-Puffer 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.

Numerierte Puffer

Wenn Sie eine oder mehrere Zeilen löschen (*vi*-Kommando *d*), wird der gelöschte Text in den ersten der neun numerierten 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 numerierten Puffer können Sie mit den *vi*-Kommandos *P* und *p* zurückholen. Diesen Kommandos müssen Sie ein Anführungszeichen " und die Nummer des Puffers voranstellen, 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 numerierten Puffer zurückgeholt. Nach obigem Beispiel holen Sie also den Inhalt des Puffers 5 zurück. Der Inhalt der numerierten 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 werden mit den Kleinbuchstaben *a-z* oder mit den Großbuchstaben *A-Z* bezeichnet. 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*).

Voreinstellung des vi

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.

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 Editor-Puffer
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. Ein Beispiel für den nicht-Boole'schen Typ ist *scroll*. *set all* gibt den Wert aus, auf den eine Option vom nicht-Boole'schen Typ gesetzt ist.

Nehmen wir an, Sie wollen die Voreinstellungen des *vi* ändern. Dies können Sie auf drei Arten tun:

- während einer *vi*-Sitzung
- mit der Umgebungsvariablen EXINIT
- mit der Datei *.exrc*.

Nehmen wir weiter an, Sie wollen sich zum Beispiel die Zeilennummern und den *vi*-Modus anzeigen lassen.

- Während einer *vi*-Sitzung:

  Um sich die Zeilennummern anzeigen zu lassen, benutzen Sie während einer *vi*-Sitzung das 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, 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*.

Wie verhält es sich nun, wenn Sie die Möglichkeiten kombinieren? Wenn sich die definierten Voreinstellungen nicht widersprechen, sind sie alle gültig. Widersprechen sie sich, dann gilt für die Priorität folgende Reihenfolge (die Priorität nimmt von oben nach unten ab):

- vi-Sitzung
- EXINIT
- *$HOME/.exrc*

Daraus folgt:

- Während einer *vi*-Sitzung definierte Voreinstellungen haben also höchste Priorität. Sie können daher während einer Sitzung jederzeit die aktuellen Voreinstellungen mit dem *ex*-Kommando *set* ändern.
- Voreinstellungen, die mit EXINIT definiert wurden, haben Vorrang vor widersprechenden Voreinstellungen in der Datei *$HOME/.exrc*.

Anpassung des vi an die Datensichtstation

Die Bildschirmausgabe des *vi* hängt vom Typ der verwendeten Datensichtstation ab. In SINIX brauchen Sie den *vi* nicht anzupassen.

vi benutzt den Inhalt der Umgebungsvariablen TERM, um in *terminfo* die notwendigen Informationen über die verwendete Datensichtstation zu finden. Falls TERM nicht definiert ist oder Sie an einer anderen Datensichtstation arbeiten, müssen Sie TERM neu setzen. Dazu gibt es verschiedene Möglichkeiten:

Vorsicht

Geben Sie nur den tatsächlichen Namen Ihrer Datensichtstation für TERM an. Falls andere Namen verwendet werden, kann dies zur Störung Ihrer Datensichtstation führen.

- Shell-Variable TERM setzen und exportieren
- TERM in der Datei *.profile* definieren und exportieren
- Mit dem *ex*-Kommando *set term* (nur im *ex*-Kommandomodus möglich)
- In der Datei *.exrc* mit dem *ex*-Kommando *set term.*
- Durch Setzen mittels EXINIT

Signalbehandlung

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.

VI-KOMMANDOS

Nach dem Aufruf befindet sich der *vi* im *vi*-Kommandomodus. Falls Sie in den *vi*-Eingabemodus gewechselt sind, kehren Sie durch Eingabe von ESC (bzw. DEL) wieder in den *vi*-Kommandomodus zurück.

ex-Kommandos können Sie, wie bereits oben beschrieben (siehe *Arbeitsweise des vi, ex-Kommandos eingeben*), entweder vom *vi*-Kommandomodus aus (siehe das *vi*-Kommando Doppelpunkt :) 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 ein akustisches Signal aus.

Vorsicht

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
(Paragraph) Ein Absatz ist festgelegt durch den Wert der *ex*-Option *paragraphs*. Leere Zeilen und Zeilen, wo ein Abschnitt beginnt, sind ebenfalls Absatzgrenzen.

Abschnitt
(Section) 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, zählt jede öffnende runde Klammer (am Anfang einer Zeile ebenfalls als Abschnittsgrenze.

Langwort
Ein Langwort ist eine Folge von Zeichen, die durch Leer-, Tabulator- oder Neue-Zeile-Zeichen getrennt sind. Vergleiche *Wort*.

Satz
Ein Satz wird beendet durch ein 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 Ge-

genhochkommata ' 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 folgendem 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 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 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, können Sie die Eingabe durch ESC ungeschehen machen. Durch *zahl* geben Sie an:

- eine Größe bei einem Positionierkommando, z.B.
  4CTRL u rollt den Bildschrirm fünf Zeilen zurück.
- einen Wiederholfaktor, z.B.
  4CTRL b blättert den Schirm um vier Seiten zurück.

Wenn Sie *zahl* nicht angeben, ersetzt *vi zahl* durch den Wert eins; außer es ist ausdrücklich anders angegeben.

CTRL@
Im *vi*-Eingabemodus:
Muß als erstes Zeichen angegeben werden! Danach auutomatischens Zurückwechseln in den Kommandomodus. Der zuletzt eingegebene Text wird an der Stelle eingefügt, an der Sie CTRL@ eingeben. Sie müssen zwischendurch in den Kommandomodus gewechselt haben.

CTRL@ funktioniert nur bis maximal 127 Zeichen.

[zahl]CTRL b

(B - backward) Die Bildschirmausgabe wird um 21 Bildschirm-Zeilen zurückgeblättert; es wird also die vorhergehende Bildschirmseite gezeigt. 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. zuletzt bei CTRL d oder CTRL u verwendeter Wert *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 *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.

0CTRL d
eendet 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; es wird also die folgende Bildschirmseite gezeigt. Mit *zahl* kann angegeben werden, um wieviele Bildschirmseiten vorwärts geblättert werden soll.

CTRL g

In der Statuszeile wird 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 Editor-Puffer

- der Prozent-Anteil der aktuellen Zeile an der Gesamtlänge der Datei in Zeilen.

Entspricht dem *ex*-Kommando *f*.

[*zahl*]CTRL h = ⌫
Im *vi*-Kommandomodus:

zahl nicht angegeben:
zahl =1

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 von *zahl* möglich.

Die Schreibmarke wird um ein Zeichen nach links bewegt, höchstens bis zum Anfang des gerade eingegebenen Textes.

[zahl]CTRL j = 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 = ↵
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
Wirkt ähnlich wie CTRL l. Es werden nur @-Zeilen gelöscht, deren Zeilenanfang fehlt, d.h. in den ersten Zeilen des Bildschirms nicht aber in den letzten.

CTRL t

Im *vi*-Eingabemodus:

Tabulatorzeichen einfügen. Die Schrittweite eines Tabulators entspricht *shiftwidth* (siehe *ex*, *ex-Optionen*). Die Tabulatoren werden allerdings erst nach Beendigung des *vi*-Einfüge-Modus "sichtbar". 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. zuletzt bei CTRL d oder CTRL u verwendeter Wert *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 ugespeichert.

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) Im Eingabemodus wird die Schreibmarke 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 Editor-Puffer 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 [= 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:

Beendet den *vi*-Eingabemodus und wechselt in den *vi*-Kommandomodus.

ESC
Siehe CTRL[.

Kommandos des vi-Kommandomodus

Den meisten *vi*-Kommandos können Sie eine ganze Zahl voranstellen. Die eckigen Klammern 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, 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]**G**
zeile ist eine positive ganze Zahl, die die Nummer der Zeile angibt, auf die *G* positioniert. Die erste Zeile der Datei hat die Nummer 1.

zeile nicht angegeben:
vi positioniert die Schreibmarke auf den Anfang der letzten Zeile der Datei.

[zeile]**z**
vi positioniert die Zeile mit dieser 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 | positioniert.

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 bearbeitete Bereich von der aktuellen Position der Schreibmarke (einschließlich) bis zur Position, die die Schreibmarke nach dem Positionierkommando hätte. 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
[zahl]**cc**
(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, werden alle ganz oder teilweise betroffenen Zeilen gelöscht. Haben Sie nicht zeilenweise, z.B. wortweise, positioniert, 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 numerierten Puf-

fern 1-9 gesichert, d.h. er kann z.B. mit *"1p* wiederhergestellt werden.

zahl
Zahl wirkt als Multiplikator auf den angegebenen Bereich.

c
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]dposition
[zahl]**dd** } Format 1

"Puffer[zahl]dposition
"Puffer[zahl]**dd** } Format 2

Format 1: Text löschen

(d - delete) Textbereich löschen. Auf dieses Kommando muß ein Positionierungskommando 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 numerierten 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.

Format 2: Text löschen und in alphabetischen Puffer sichern

Der Text wird genau wie bei Format 1 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 nächste (*zahl*-te) Ende eines Wortes positioniert.

[zahl]E
(E - end) Die Schreibmarke wird auf das nächste Ende eines Langwortes positioniert.

[zahl]fzeichen
(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 können dieses Suchkommando mit dem Strichpunkt ; in Suchrichtung und mit dem Komma , entgegen der Suchrichtung wiederholen.

[zahl]Fzeichen
(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 können dieses Suchkommando mit dem Strichpunkt in Suchrichtung und mit dem Komma entgegen der Suchrichtung wiederholen.

[zeile]G
(G - go to) *vi* positioniert die Schreibmarke auf den Anfand der Zeile mit der Nummer *zeile*.

zeile nicht angegeben:
vi positioniert die Schreibmarke auf die letzte Zeile der Datei.

[zahl]h
Die Schreibmarke wird um *zahl* Zeichen nach links bewegt, jedoch nur in der aktuellen Zeile.

[zahl]H
Die Schreibmarke wird auf den Anfang der ersten Bildschirmzeile positioniert. Wird eine Zahl *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 Text Zeile sein.

[zahl]**j**
Die Schreibmarke wird um *zahl* Textzeilen in der gleichen Spalte nach unten bewegt.
Entspricht CTRL j und CTRL n.

J
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 eine Zahl kann gegebenenfalls 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**
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 eine Zahl *zahl* angegeben, so wird die Schreibmarke zu der Textzeile bewegt, die sich *zahl* Textzeilen oberhalb der untersten Textzeile befindet. Wird ein Operator verwendet, so sind ganze Zeilen betroffen.

Beispiel

d3L löscht alle Zeilen von der aktuellen Zeile (einschließlich) bis zur dritten Textzeile (einschließlich) oberhalb des unteren Bildschirmrandes.

mmarke
(m - mark) Marke an der aktuellen Position der Schreibmarke setzen. Auf *m* muß ein Kleinbuchstabe *marke* folgen; mit diesem Buchstaben wird die aktuellen 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) Unter der aktuellen Zeile wird eine Zeile eingefügt, und *vi* wechselt in den *vi*-Eingabemodus.

O
(O - open) Oberhalb der aktuellen Zeile wird eine neue Zeile eingefügt, und *vi* wechselt in den *vi*-Eingabemodus.

[zahl]**p** Format 1
"puffer**p** Format 2

Format 1: Text aus temporärem Puffer holen

(p - put) Schreibt den Text aus dem temporären Puffer in einfacher oder *zahl*-facher Ausführung hinter die aktuelle Position der Schreibmarke. Wenn der Text im Puffer mit *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.

Format 2: Text aus numeriertem oder alphabetischem Puffer holen

puffer kann sein:

- eine Ziffer von 1-9: der Text wird aus dem entsprechenden numerierten Puffer geholt
- ein Buchstabe: der Text wird aus dem entsprechenden alphabetischen Puffer geholt.

[zahl]P
"pufferP
(P - put) Entspricht *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 müsssen Sie *vi*↵ eingeben.

[zahl]rzeichen
(r - replace) Das Zeichen, auf dem die Schreibmarke steht, wird durch das nächste eingegebene Zeicnen *zeichen* ersetzt. Wird eine Zahl angegeben, so werden *zahl* Zeichen durch das nächste Zeichen ersetzt.

R
(R - replace) Mehrere Zeichen ersetzen. *vi* schaltet auf der aktuellen Position der Schreibmarke in den *vi*-Eingabemodus (Replace Mode). 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 blaESC
```

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 blaESC
```

[zahl]S
(S - substitute) Ganze Zeilen werden ersetzt (entspricht *cc*). Mit *zahl* kann angegeben werden, wieviele Zeilen ersetzt werden sollen.

[zahl]t
(t - to) Auf *t* muß ein einzelnes Zeichen folgen. *vi* durchsucht die aktuelle Zeile in Vorwärtsrichtung (nach rechts) nach diesem Zeichen und positioniert die Schreibmarke gegebenenfalls unmittelbar vor dieses Zeichen. Wenn Sie einen Wiederholungsfaktor angeben, positioniert *vi* vor das *zahl*-te Zeichen.

Sie können dieses Suchkommando mit dem Strichpunkt in Suchrichtung und mit dem Komma entgegen der Suchrichtung wiederholen.

[zahl]T
(T - to) Auf *T* muß ein einzelnes Zeichen folgen. *vi* durchsucht die aktuelle Zeile in Rückwärtsrichtung nach diesem Zeichen und positioniert die Schreibmarke gegebenenfalls hinter dieses Zeichen. Wenn Sie einen Wiederholungsfaktor angeben, positioniert *vi* hinter das *zahl*-te Zeichen.

Sie können dieses Suchkommando mit dem Strichpunkt ; in Suchrichtung und mit dem Komma , entgegen der Suchrichtung wiederholen.

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 numerierten 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 unterhalb der Schreibmarke 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]yposition
[zahl]yy } Format 1

"puffer[zahl]yposition
"puffer[zahl]yy } Format 2

Format 1: 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`	aktuelle und drei folgende Textzeilen in den temporären Puffer sichern.

Format 2: 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	aktuelle und drei folgende Textzeilen in den alphabetischen Puffer *a* sichern.
"ap	Inhalt von *a* im Anschluß an aktuelle Position ausgeben.

[zahl]Y Format 1
"puffer[zahl]Y Format 2

Format 1: 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*).

Format 2: 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

Das Kommando *z* baut den Bildschirm neu auf. Dabei wird die durch *zeile* bzw. *Muster* bestimmte Textzeile 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 mit dem ersten Vorkommen von Muster.

zahl ist eine ganze Zahl, die auf *z* folgt. *zahl* gibt die Anzahl der Bildschirmzeilen an, mit denen der Bildschirm neu aufgebaut wird.

zahl nicht angegeben:
zahl = 23.

ZZ

vi beenden. Wenn am Text-Puffer seit der letzten Sicherung (mit dem *ex*-Kommando *w*) Änderungen vorgenommen wurden, dann wird der Inhalt des Text-Puffers in die Datei mit dem aktuellen Dateinamen (siehe *ex*-Kommando *f*) gesichert. Entspricht *ex*-Kommando *x*.

[zahl]**Leerzeichen**

zahl nicht angegeben:
zahl = 1

Die Schreibmarke wird um *zahl* Zeichen nach rechts bewegt (höchstens bis zum Zeilenende).

Entspricht *l*.

[zahl]!position

Mit Ausrufezeichen ! können Sie in den *vi*-Zeilen-Kommandomodus wechseln, um ein SINIX-Kommando einzugeben. Auf ! muß einen Positionieranweisung auf eine andere Zeile folgen. Erst dann wechseln Sie in den *vi*-Zeilen-Kommandomodus. Der Bereich von der aktuellen Zeile bis zur angegebenen Zeile wird zur Standard-Eingabe des SINIX-Kommandos und durch die Ausgabe (Standard-Ausgabe und Standard-Fehlerausgabe) des SINIX-Kommandos ersetzt. Die Zeilen werden immer ganz, nie als Teilzeilen übergeben.
Die Schreibmarke springt erst nach der Eingabe der Positionieranweisung in die Statuszeile. Die Eingabe des SINIX-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 SINIX-Kommando übergeben. Dem zweiten Ausführungszeichen kann eine ganze Zahl vorangestellt werden, damit wird angegeben, wieviele Zeilen einschließlich der aktuelllen Zeile an das SINIX-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 und den beiden folgenden Zeilen.

"

Das Anführungszeichen " wird dem Namen eines alphabetischen oder numerierten Puffers vorangestellt. Siehe *vi*-Kommandos *d*, *p* und *y*.

[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).

%

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 kommerziellem Und & wird das letzte *ex*-Kommando *s* wiederholt. Entspricht dem *ex*-Kommando *&*.

'marke — Format 1
" — Format 2

(Hochkomma)

Format 1: Schreibmarke auf markierte Zeile bewegen

marke ist ein Kleinbuchstabe, der eine Marke bezeichnet, die mit dem *vi*-Kommando *m* gesetzt wurde. Mit Hochkomma Marke *'marke* wird die Schreibmarke zum ersten Zeichen der markierten Zeile bewegt, das kein Leer- oder Tabulatorzeichen ist.

Format 2: 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*↵.

Gilt für Format 1 und 2:

Wird das '-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.

Beispiele

d'a
Die Zeilen von der aktuellen Zeile bis zur markierten Zeile einschließlich werden gelöscht.

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 wiederholen.

[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. Auch der Beginn eines Abschnittes oder Absatzes wird als Beginn eines Satzes interpretiert (siehe { und [[). Siehe *ex*, *ex-Optionen*.

[zahl])
Mit der schließenden runden Klammer) wird die Schreibmarke zum Anfang des nächsten (*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 im 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 Bindstrich - 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 *zahl*-mal wiederholt, das den Editor-Puffer veränder hat.

Wenn Ihr letztes Kommando den Inhalt eines numerierten Puffers ausgegeben hat, dann gibt ein darauf folgendes Punkt-Kommando den Inhalt des numerierten Puffers mit der um eins erhöhten Nummer aus u.s.w..

/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 Datei-Ende 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, 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.
```

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 ↵ abschickt.

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* 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 wären bei ihrer Eingabe *lisp* und *autoindent* gesetzt. Mit *zahl* kann angegeben werden, wieviele Zeilen bearbeitet werden sollen. Position kann auch ein zweites Gleichheitszeichen sein, dann werden die aktuelle Zeile bzw. *zahl* Zeilen eingerückt.

[zahl]>position

Das *vi*-Kommando Größerzeichen > verschiebt Zeilen um *shiftwidth* nach rechts (siehe <).

?

Das Fragezeichen bewirkt eine Suche in Rückwärtsrichtung; ist also das Gegenstück zu / (siehe /).

[zahl][[

Mit 2 öffnenden eckigen Klammern wird zurück auf die vorangehende (*zahl*-te) Abschnittsgrenze positioniert.

[zahl]]]

Mit 2 schließenden eckigen Klammern wird vorwärts auf die nächste (*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.

'marke — Format 1
" — Format 2

(Gegenhochkomma)

Format 1: 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.

Format 2: Schreibmarke auf vorherige aktuelle Position bewegen

Durch das Kommando zweifaches 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↵.

Gilt für Format 1 und Format 2:

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.

[zahl]{
Mit der öffnenden geschweiften Klammer { wird die Schreibmarke zum Anfang des aktuellen 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

[zahl]}
Mit der schließenden geschweiften Klammer } wird die Schreibmarke zum Anfang des nächsten Absatzes bewegt. Durch *zahl* kann ein Wiederholungsfaktor angegeben werden.

~
Die Tilde ~ bewirkt eine Zeichenkonvertierung. Steht unter der Schreibmarke ein Kleinbuchstabe, wird er in einen Großbuchstaben umgewandelt - und umgekehrt. Die Schreibmarke bewegt sich anschließend um ein Zeichen nach rechts weiter.

DATEIEN

$HOME/.exrc
Datei mit Voreinstellungen für *ex* und *vi*. Diese Voreinstellungen sind nicht wirksam, wenn durch *EXINIT* etwas widersprüchliches definiert wird.

UMGEBUNGSVARIABLEN

LINES
Anzahl der Zeilen pro Bildschirm

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.

SIEHE AUCH

edit, *ex*, *vedit*

wait
Auf die Beendigung von Hintergrund-Prozessen warten

Das in die Bourne-Shell *sh* eingebaute Kommando *wait* wartet,

- bis der vorher gestartete Hintergrund-Prozeß mit der angegebenen Prozeß-Nummer (PID) beendet ist, oder
- bis alle vorher gestarteten Hintergrund-Prozesse beendet sind, wenn Sie beim Aufruf keine Prozeß-Nummer angeben.

Im Dialog wartet die Shell standardmäßig nicht auf die Beendigung eines vorher gestarteten Hintergrund-Kommandos, 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 Hintergrund-Kommando rechtzeitig beendet ist.

```
wait[ prozess_nummer]
```

prozess_nummer
: Nummer des Hintergrund-Prozesses, auf dessen Ende *wait* warten soll. Sie können nur eine Prozeß-Nummer angeben. Das Kommando *wait* gibt als Ende-Status den Ende-Status des Hintergrund-Prozesses zurück. Wenn Sie mehrere Prozeß-Nummern angeben, gibt *wait* keine Fehlermeldung aus, sondern wartet nur auf die Beendigung des Hintergrund-Prozesses mit der Prozeß-Nummer, die Sie beim Aufruf als erste angegeben haben. Bei der Prozeß-Nummer eines bereits beendeten Hintergrundprozesses verhält sich *wait* wie bei nicht angegebener Prozeß-Nummer.

  Der Variablen *!* (Ausrufezeichen) weist die Shell immer die Prozeß-Nummer des zuletzt im Hintergrund gestarteten Kommandos zu. Mit *$!* können Sie auf den Inhalt dieser Variablen zugreifen.

  prozess_nummer nicht angegeben:
  wait wartet auf die Beendigung aller Hintergrund-Prozesse, die vor ihm gestartet wurden. In diesem Fall gibt *wait* den Ende-Status 0 zurück.

ENDE-STATUS

Wenn Sie *wait* ohne Argument aufgerufen haben, ist der Ende-Status immer gleich 0.

Wenn Sie *wait* mit einer Prozeß-Nummer aufrufen, gibt *wait* als Ende-Status den Ende-Status des so ausgewählten Hintergrund-Prozesses zurück.

SIEHE AUCH

sh

wc
Wörter, Zeichen und Zeilen zählen (word count)

wc gibt die Anzahl der Zeilen, Wörter und Zeichen von Dateien auf die Standard-Ausgabe aus.

```
wc[␣option]...[␣datei]...
```

Keine Option angegeben
wc gibt drei Zahlenwerte aus für die Anzahl der Zeilen, Wörter, Zeichen.

option

-c
wc gibt die Anzahl der Zeichen aus. Leer-, Tabulator- und Neue-Zeile-Zeichen werden mitgezählt.

-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.

BEISPIELE

1. Ausgeben der Anzahl der Zeilen, Wörter und Zeichen für die Dateien *logik*, *plan* und *rest*.

```
$ wc logik plan rest
     27     139    1077 logik
      5      15     140 plan
      3       6      51 rest
     35     160    1268 total
```

2. Feststellen, wieviele Dateien im aktuellen Dateiverzeichnis eingetragen sind.

```
$ ls | wc -l
      31
```

3. Feststellen, wieviele Benutzer gerade am Rechner arbeiten.

```
$ who | wc -l
       6
```

4. Zählen, 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.

who
Aktive Benutzerkennungen anzeigen

who informiert Sie darüber,
- unter welcher Benutzerkennung und an welcher Datensichtstation Sie gerade arbeiten
- welche Benutzer seit wann mit *login* 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 und wann Systemabstürze stattgefunden haben, seit 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.

```
who[␣option]...[␣datei]                    Format 1
who␣-q␣n␣zahl[␣datei]                      Format 2
who␣am␣i                                   Format 3
who␣am␣I                                   Format 4
```

Format 1: Ausführliche Informationen ausgeben

who[␣option]...[␣datei]

Die allgemeine Form der Ausgabe sieht folgendermaßen aus:

Name [Zustand] Leitung Zeit [inaktiv] [PID] [Kommentar] [Ende-Status]

Die Informationen zu *Name* (Benutzerkennung), *Leitung* (Datensichtstation) und *Zeit* werden bei allen Optionen außer *-q* ausgegeben. *Zustand* wird nur bei *-T* ausgegeben, *inaktiv* und *PID* (Prozeßnummer) nur bei *-u* und *-l*, *Kommentar* und *Ende-Status* nur bei *-a*. Die Ausgabe bei *-p*, *-d* und *-r* wird bei den jeweiligen Optionen erläutert.

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.

option

-a
(a - all) Alle Optionen außer *-q* sind wirksam.

-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 *Ende-Status* 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 nur die Datensichtstationen aufgelistet, bei denen das System auf eine Anmeldung wartet. *Name* enthält in diesem Fall den Eintrag *LOGIN*. Die anderen Felder haben die übliche Bedeutung - außer, daß das Feld *Zustand* nicht existiert.

-p
(p - process) *who* listet alle Prozesse auf, die vom *init*-Prozeß gestartet wurden. *Name* gibt den Namen des von *init* gestarteten Programms an, wie er in der Datei */sbin/inittab* steht. *Zustand*, *Leitung* und *inaktiv* haben in diesem Fall keine Bedeutung. *Kommentar* enthält das Identifikationsfeld der entsprechenden Zeile in */sbin/inittab*.

-r
(r - run level) *who* gibt den aktuellen Betriebszustand des Prozesses *init* aus. *inaktiv*, *PID* und *Kommentar* enthalten die Signalnummer, die den Prozeß terminiert hat, die Prozeßnummer bzw. den Ende-Status.

-s
(s - standard) *who* gibt nur *Name*, *Leitung* und *Zeit* aus. Diese Option ist die Standardeinstellung.

-T
(T - terminal) Wie *-s*, nur wird zusätzlich der Zustand der Datensichtstation ausgegeben. *Zustand* gibt an, ob ein anderer Benutzer auf diese Datensichtstation schreiben darf: Ein Pluszeichen + gibt an, daß jeder Benutzer auf diese Datensichtstation schreiben darf, bei einem Minuszeichen - ist dies nicht möglich. Der Systemverwalter kann auf alle Datensichtstationen schreiben. Bei einer defekten Leitung 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 nur die Benutzer auf, die momentan angemeldet sind. *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. *inaktiv* gibt an, wie lange keine Ein- oder Ausgabe mehr auf der Datensichtstation erfolgte. Ein Punkt . bedeutet dabei, 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. Mit diesem Eintrag stellen Sie also feststellt, ob an einer Datensichtstation gearbeitet wird. *PID* gibt die Prozeßnummer des Kommandointerpreters (der Shell) an, mit dem ein Benutzer arbeitet. *Kommentar* enthält zur jeweiligen Datensichtstation einen Kommentar, wie er in der Datei */sbin/inittab/* steht. Dieser Kommentar kann z. B. Informationen über den Aufstellungsort der Datensichtstation o.ä. enthalten.

-q

(q - quick) *who* gibt nur die Namen und die Zahl aller aktuell angemeldeten Benutzer aus. Alle anderen Optionen außer *-n* werden ignoriert.

datei

Name der Datei, aus der *who* seine Informationen bezieht. Wenn Sie für *datei*, wie es üblich ist, die Datei */var/adm/wtmp* angeben, erzeugt *who* Informationen über alle An- und Abmeldungen am System und Systemabstürze, seit der Systemverwalter die Datei */var/adm/wtmp* zum letzten Mal auf die Größe 0 reduziert hat.

datei nicht angegeben:
who holt seine Informationen aus der Datei */var/adm/utmp*

Format 2: Kurzinformation ausgeben

who ␣ **-qn** ␣ zahl [␣ datei]

-q

wie bei Format 1

-n zahl

who gibt pro Zeile *zahl* Benutzer aus. *zahl* muß mindestens 1 sein.

datei

wie bei Format 1

Format 3, Format 4: Eigeninformation ausgeben

```
who␣am␣i
who␣am␣I
```

who gibt aus:
- Benutzerkennung, unter der Sie sich angemeldet haben
- Name der Datensichtstation (ohne */dev/*), an der Sie sich angemeldet haben
- Anmeldezeitpunkt.

DATEIEN

/var/adm/utmp
Datei, aus der *who* standardmäßig seine Informationen holt.

/var/adm/wtmp
Datei, aus der *who* seine Informationen holt, wenn sie beim Aufruf angegeben wird. Der Systemverwalter reduziert die Größe dieser Datei von Zeit zu Zeit auf 0.

/sbin/inittab
Datei, die alle von *init* gestarteten Prozesse enthält.

BEISPIEL

```
$ who am i
moritz   tty10    Jan  19 13:00
```

moritz ist die Benutzerkennung, tty10 der Name der Datensichtstation..

SIEHE AUCH

date, login, mesg,
su, init
inittab, utmp [5]
wait() [14]

whois
Internet-Service zum Auffinden von Benutzerkennungs-Dateiverzeichnissen

whois sucht für einen bestimmten Identifikator einen Eintrag in einem Internet-Dateiverzeichnis. *whois* greift auf Datenbasen zu, die Informatione über das Netz wie NIC handles (NIC = Network Information Center) Namen des Netzverwalters enthalten.

```
whois[␣-h␣rechner]␣identifikator
```

-h␣rechner
Name eines Rechners, der in der Datei */etc/hosts* eingetragen ist.

identifikator
Sie können entweder einen Namen wie z.B. "Schmidt"
angeben oder einen Verweis (NIC handle) wie z.B. "SRI-NIC". Das NIC (Network Information Center) ist eine Einrichtung von SRI international, die IP Network-Nummern und Domain-Namen verwaltet. Ein NIC handle ist ein eindeutiger, einem Netzwerk zugeordneten Identifikator, der von SRI international verwaltet wird. Um eine Suche nur nach dem Namen zu erzwingen, geben Sie vor dem Namen einen Punkt . ein. Um eine Suche nur nach dem Verweis zu erzwingen, geben Sie vor dem Verweis ein Ausrufezeichen ! ein.

Um nach einem Eintrag für eine Gruppe oder Organisation zu suchen, geben Sie vor dem Argument einen Stern * ein. Es wird dann die Liste aller Gruppenmitglieder mitausgegeben.
Sie können natürlich Ausrufezeichen und Stern oder Punkt und Stern zusammen angeben.
Wenn Sie direkt an *identifikator* drei Punkte ... anhängen, sucht *whois* nach allen Namen oder Verweisen, die bis zu den Punkten mit der Angabe übereinstimmen, z.B. wird bei der Angabe "Fra..." gefunden: Franz, Francesca, Fraunberg etc.

BEISPIELE

1. Nach dem Namen oder Verweis "Schmidt" suchen:

```
$ whois Schmidt
```

2. Nach dem Verweis "SRI-NIC" suchen:

```
$ whois !SRI-NIC
```

3. Nach dem Namen "Hans Otto" suchen:

```
$ whois Otto,Hans
```

write
Nachricht an einen Benutzer senden

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, können Sie keine Nachrichten mit *write* schicken. Ein Benutzer mit Systemverwalterberechtigung kann Nachrichten an alle Datensichtstationen schicken.

```
write␣empfänger[␣station]
text
...
... END
```

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.

station
: Nummer der Datensichtstation, an der der Empfänger angemeldet ist.

station nicht angegeben:
write sucht die Datensichtstation aus der Datei */var/adm/utmp*. Wenn ein Benutzer mehrfach angemeldet ist und es daher mehrere Einträge gibt, nimmt *write* den ersten Eintrag.

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 Dateiende (EOT) aus und beendet sich.
- Nichtdruckbare Zeichen werden vor dem Senden umgewandelt. Steuerzeichen werden als Folge Dach ^, ASCII-Zeichen dargestellt; Zeichen, bei denen das achte Bit gesetzt ist, erscheinen in "Meta-Notation". So wird zum Beispiel '\003' als ^C dargestellt und '\372' als 'M-z'.

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. Der erste Benutzer erkennt an einem zweifachen Klingelzeichen, daß die Verbindung geklappt hat und der zweite Benutzer Nachrichten entgegennehmen kann.
2. Der zweite Benutzer ruft deswegen *write* mit der Kennung des ersten Benutzers auf. 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 die END- oder die DEL-Taste drücken. Wenn Sie zusätzlich verhindern möchten, daß der andere Benutzer weiterhin Nachrichten sendet, dann rufen Sie *mesg -n* auf.

FEHLERMELDUNGEN

`user is not logged on.`

Der Empfänger ist nicht angemeldet.

`Permission denied.`

Die Datensichtstation des Empfängers ist schreibgeschützt (siehe *mesg*).

`Warning: You have your terminal set to "mesg -n". 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 !.

BEISPIEL

An die Benutzerin *karin* eine Meldung schicken:

```
$ write karin
!date
Mon Oct 15 19:00:13 MET 1990
heute am 15.10.
bin ich gespannt,
was passiert END
```

SIEHE AUCH

mail, *mesg*, *pr*, *sh*, *who*

xargs Argumentliste(n) aufbauen und Kommando ausführen

xargs verbindet beim Aufruf angegebene Argumente und Argumente, die es von der Standard-Eingabe liest, miteinander und führt 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 Hochkommata '...' 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 Hochkommata '...' eingeschlossen werden oder ihnen ein Gegenschrägstrich \ vorangestellt wird.

```
xargs[␣option]...[␣kommando[␣anfangs_argument]...]
```

option

Mit den Optionen *-i*, *-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*, *-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*, *-l* oder *-n*:
Wenn sich Optionen gegenseitig aufheben, etwa *-l* und *-n*, so gilt die zuletzt angegebene Option.

-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.

ersetzungszeichenkette nicht angegeben:
Für *ersetzungszeichenkette* wird ein Paar geschweifter Klammern {} angenommen.

-l[zeilenanzahl]
(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.

zeilenanzahl
positive ganze Zahl

zeilenanzahl nicht angegeben:
Für *zeilenanzahl* wird 1 angenommen.

-narganzahl
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.

-t
(t - trace) Das *kommando* und jede erstellte Argumentliste werden unmittelbar bevor sie abgearbeitet werden auf die Standard-Fehlerausgabe ausgegeben.

-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. Wenn Ihre Antwort mit *y* (yes) beginnt, so wird der Aufruf von *kommando* ausgeführt; die übrigen Buchstaben Ihrer Antwort spielen keine Rolle. Bei jeder anderen Antwort, die nicht mit einem *y* beginnt, wird der Kommandoaufruf nicht ausgeführt.

-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* oder *-l* gesetzt sind. Wenn keine der Optionen *-i*, *-l* oder *-n* gesetzt ist, dann wird die Ausführung von *xargs* beendet, wenn die Gesamtlänge aller Argumente *maxgröße* (siehe Option *-s*) überschreitet.

-smaxgröße
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*, *-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.

maxgröße
positive ganze Zahl <= 470.

-smaxgröße nicht angegeben:
wirkt wie die die Angabe *-s470*.

-e[dateiende]
xargs liest die Standard-Eingabe entweder bis zum Erreichen des tatsächlichen Dateiendes oder bis es das angegebene logische Dateiende erkennt.

dateiende
Für *dateiende* geben Sie eine Zeichenkette an. Diese Zeichenkette wird bei der Ausführung von *xargs* als logisches Dateiende interpretiert.

dateiende nicht angegeben:
Es ist kein logisches Dateiende definiert. Der Unterstrich _ hat keine Sonderbedeutung und wird als normales Zeichen verarbeitet.

-edateiende nicht angegeben:
Der Unterstrich _ wird als logisches Dateiende interpretiert.

kommando
Für *kommando* können Sie ein beliebiges Kommando angeben. Wenn die Ausführung von *kommando* den Ende-Status -1 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 -1 zu vermeiden.

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* ist gesetzt.

anfangs_argument nicht angegeben:
Die Argumentlisten werden nur aus den von der Standard-Eingabe eingelesenen Argumenten aufgebaut.

BEISPIELE

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*, {} ist Ersetzungszeichenfolge, da nichts anderes angegeben ist). Vor jedem Aufruf des Kommandos *mv* wird das Kommando und die Argumentliste ausgegeben (Option *-t*).

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 Wed Mar 29 14:21:06 MET 1989 wer_und_wann
```

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 -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

zcat Komprimierte Dateien ausgeben

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. Eine ähnliche Funktionalität bieten Ihnen die Kommandogruppe *pack*, *unpack* und *pcat*.

```
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.

ENDE-STATUS

0 bei Erfolg
1 bei Fehler

FEHLERMELDUNGEN

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 unerlaubtem Segmentzugriff) wurde empfangen, was normalerweise bedeutet, daß die Eingabedatei beschädigt ist.

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

compress, *uncompress*, *pack*, *unpack*, *pcat*, *cat*

:
Ende-Status 0 zurückgeben

Das in die Bourne-Shell *sh* eingebaute Kommando *:* (Doppelpunkt) gibt den Ende-Status 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 (siehe *sh*, *Standard-Werte für Shell-Parameter vereinbaren*).
- 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.

```
: [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 (siehe *sh*). Das eingebaute *sh*-Kommando *:* gibt nur den Ende-Status 0 zurück.

  argument nicht angegeben:
  Das eingebaute *sh*-Kommando *:* gibt nur den Ende-Status 0 zurück und tut sonst nichts.

ENDE-STATUS

0 in jedem Fall

BEISPIELE

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!".

2. In der folgenden Shell-Prozedur wird der Shell-Variablen *name* die Prozeß-Nummer des aktuellen Shell-Prozesses zugewiesen, falls diese Variable noch nicht definiert ist oder die leere Zeichenkette enthält:

```
: ${name:=$$}
echo $name
```

SIEHE AUCH

sh, *true*

. Shell-Prozeduren in der aktuellen Shell ausführen

Das in die Bourne-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.

```
. 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 Leserecht.

UMGEBUNGSVARIABLE

PATH
Suchpfad der Shell

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

Das in die Bourne-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:

- Ende-Status 0 (wahr), falls die Bedingung erfüllt ist.
- Ende-Status 1 (falsch), falls die Bedingung nicht erfüllt ist oder falls Sie die Bedingung nicht vollständig angegeben haben. Den gleichen Ende-Status erhalten Sie, wenn Sie keine Bedingung angeben.

Abhängig vom Ende-Status können Sie unterschiedliche Kommandos ausführen, Schleifen abbrechen usw.

Für das eingebaute *sh*-Kommando *[␣␣]* gibt es zwei Schreibweisen (siehe Syntax). Die Wirkung ist dieselbe.

```
[␣ausdruck␣]  oder  test␣ausdruck
```

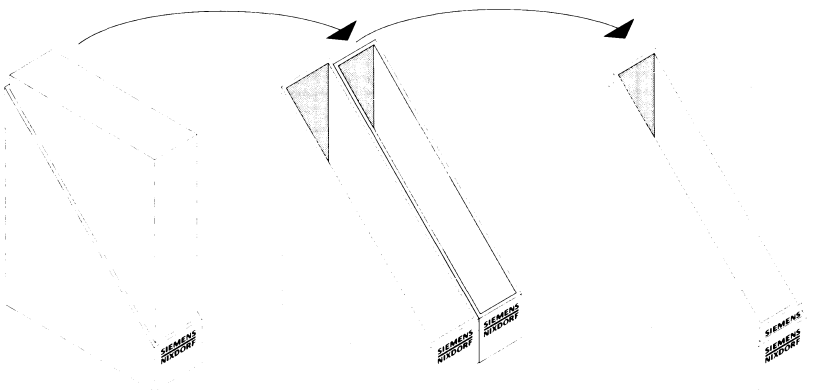
Die Beschreibung der Syntax: siehe eingebautes *sh*-Kommando *test*.

SIEHE AUCH

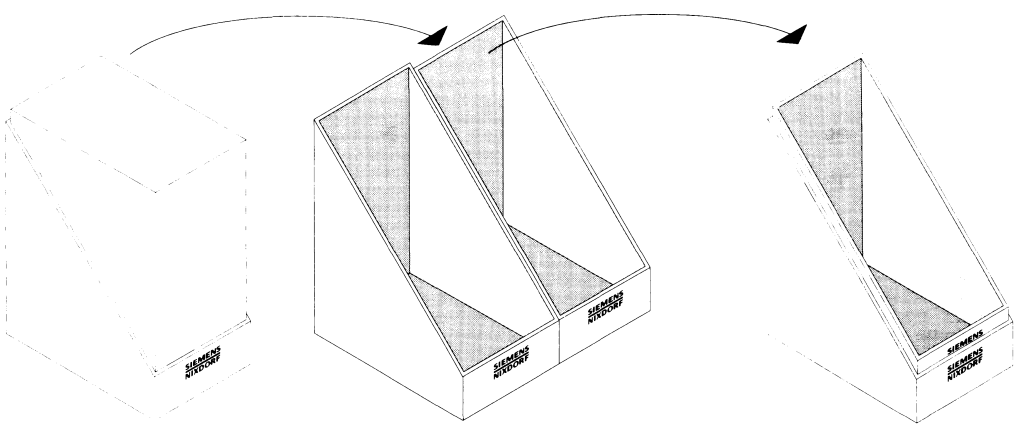
sh, *test*

Sammelboxen

Für Handbücher des vorliegenden Formates bieten wir zweiteilige Sammelboxen in zweierlei Größen an. Der Bestellvorgang entspricht dem für Handbücher.



Breite: ca. 5 cm
Bestellnummer: U3775-J-Z18-1



Breite: ca. 10 cm
Bestellnummer: U3776-J-Z18-1

9164 12

Herausgegeben von/Published by
Siemens Nixdorf Informationssysteme AG
Postfach 2160, W-4790 Paderborn
Postfach 830951, W-8000 München 83

Bestell-Nr./Order No. ***U6417-J-Z95-1***
Printed in the Federal Republic of Germany
10710 AG 12905.2 (13390)