Targon® /31

User's Reference Manual

NIXDORF
COMPUTER

Printed in W.-Germany 05.1989

Systemliteratur
Targon® /31

Systemliteratur
Targon® /31

Systemliteratur
Targon® /31

Bitte abtrennen und in die Tasche im Handbuchrücken einstecken.

Systemliteratur
Targon® /31

Systemliteratur
Targon® /31

Organisationsblatt

Dieses Blatt gibt eine Übersicht über alle Änderungen, die seit der ersten Auflage an diesem Handbuch durchgeführt wurden. Es wird bei jeder Änderungsmitteilung mitgeliefert und ist jeweils auszutauschen.

Erstauflage:	1.5.89	Rel. 4

Dieses Handbuch wurde mit troff erstellt.

Inhaltsverzeichnis

EINFÜHRUNG

Dieses *User's Reference Manual* beschreibt die Kommandos, die die Basis-Software des Targon /31 Systems darstellen.

Verschiedene andere Handbücher enthalten weitere wichtige Informationen:

- Der *Administrator's Guide* ist eine Übersicht über das UNIX-System und enthält Übungen für die Verwendung der Texteditoren, die Automatisierung von Aufgaben, die sich wiederholen, und das Senden von Informationen an andere.
- Der *Programmer's Guide* gibt einen Überblick über die Programmierungumgebung des UNIX-Systems sowie Übungen zu verschiedenen Programmierwerkzeugen.
- Das *Programmer's Reference Manual* beschreibt Kommandos, Systemaufrufe, Bibliotheksfunktionen und Dateiformate, die der Programmierer braucht.
- Der *Administrator's Guide* enthält die Beschreibung von Verwaltungsabläufen und die Erklärung der Verwaltungsaufgaben.
- Das *Administrator's Reference Manual* beschreibt die Kommandos für den Systemverwalter.

Die Programme gehören zwar jeweils zu einem Paket von speziellen Dienstprogrammen, sie sind hier jedoch in alphabetischer Reihenfolge in einem Kapitel mit dem Titel "Kommandos" zusammengefaßt. Die folgenden verschiedenen Pakete von Dienstprogrammen kommen in diesem Kapitel vor:

1. Basis-Dienstprogramme für Netzwerke
2. Dienstprogramme für Magnetbänder
3. Dienstprogramme für Verzeichnis- und Dateiverwaltung
4. Editoren
5. Basis-Dienstprogramme
6. Grafik-Dienstprogramme
7. Dienstprogramme für Interprozeß-Kommunikation
8. Dienstprogramme für Drucker
9. Dienstprogramme für Leistungsmessung
10. Dienstprogramme für Datenschutz
11. Spell-Dienstprogramme
12. Dienstprogramme zur Unterstützung von Terminals
13. Dienstprogramme zur Verwaltung von Terminal-Parametern

14. Dienstprogramme für Benutzerkonfiguration

Die Dienstprogramme für Datenschutz (security) werden nur an USA-Kunden geliefert.

Kapitel 1 Kommandos

Die Einträge in Kapitel 1 behandeln Programme, die vom Benutzer direkt oder in Kommando-Prozeduren aufgerufen werden, im Gegensatz zu Funktionen, die innerhalb von Benutzer-Programmen aufgerufen werden. Die Kommandos stehen im allgemeinen im Verzeichnis **/bin** (für **Bin**ärprogramme). Außerdem stehen einige Programme in **/usr/bin**. Diese Verzeichnisse werden automatisch vom Kommando-Interpreter, genannt *Shell*, durchsucht. UNIX-Systeme auf dem Targon-/31-System haben zusätzlich ein Verzeichnis **/usr/lbin**, das lokale Kommandos enthält.

Die Zahlen hinter dem Kommando sollen Querverweise vereinfachen. Eine **(1)**, **(1C)** oder **(1G)** hinter einem Kommando bedeutet meistens, daß das Kommando in diesem Handbuch enthalten ist. (Kommandos in Kapitel 1, die für Programmierer geeignet sind, sind im *Programmer's Reference Manual* zu finden.) Eine **(1M)**, **(C)** oder **(7)** hinter einem Kommando bedeutet, daß dieses Kommando im entsprechenden Kapitel des *Administrator's Reference Manual* steht. Eine **(2)**, **(3)**, **(4)** oder **(5)** hinter einem Kommando bedeutet, daß das Kommando im entsprechenden Kapitel des *Programmer's Reference Manual* steht.

Jeder Eintrag im Kapitel Kommandos erscheint unter einem einzigen Namen, der in der oberen Ecke der entsprechenden Seite(n) angegeben ist. Die Einträge außer der Einführung *intro*(1) sind alphabetisch geordnet. Einige Einträge können auch mehrere Kommandos beschreiben. In solchen Fällen erscheint der Eintrag nur einmal und ist unter seinem "Hauptnamen" alphabetisch eingeordnet, d. h. dem Namen der in der oberen Ecke einer Seite erscheint. Die "Sekundärkommandos" werden direkt unter dem zugehörigen Hauptkommando aufgelistet.

Die Einträge sind in folgendem Format beschrieben (jedoch kommen nicht alle Abschnitte in jedem Eintrag vor):

- **BEZEICHNUNG** enthält den Hauptnamen (und, falls vorhanden, den (die) Sekundärnamen,) und eine Kurzbeschreibung.
- **ÜBERSICHT** zeigt die Definition des beschriebenen Programms. Einige zu erläuternde Konventionen werden benutzt, insbesondere in der **ÜBERSICHT:**

- Zeichenfolgen in **Fettdruck** sind Literale und müssen genauso eingegeben werden, wie sie angegeben sind.
- Zeichenfolgen in *Kursivschrift* bezeichen meistens ersetzbare Argument-Parameter oder Kommandonamen, die an anderer Stelle im Handbuch stehen.
- Eckige Klammern [ ], die einen Argument-Paramter einschließen, bedeuten, daß das Argument wahlweise ist. Der Parameter *name* oder *file* steht immer für einen Dateinamen.
- Punkte ... bedeuten, daß das vorherige Argument wiederholt werden darf.
- Die letzte Konvention betrifft die Kommandos: Ein Argument, das mit einem Minuszeichen (-), einem Pluszeichen (+) oder einem Gleichheitszeichen (=) beginnt, wird häufig als eine Art von Schalter angesehen, selbst wenn es an einer Stelle steht, an der ein Dateiname stehen könnte. Sie sollten daher keine Dateinamen verwenden, die die mit -, + oder = anfangen.

- **BESCHREIBUNG** beschreibt die Verwendung dieser Kommandos.
- **BEISPIEL(E)** gibt gegebenenfalls ein oder mehrere Beispiele für die Anwendung.
- **DATEIEN** enthält die Dateinamen, auf die sich das Programm bezieht.
- **ENDECODES** beschreibt die Werte, die bei Beendigung des Kommandos gesetzt werden können. Der gesetzte Wert ist über die Shell-Umgebungsvariable '?' erhältlich (vgl. sh(1)).
- **BEMERKUNGEN** liefert Information, die hilfreich ist in bestimmten Fällen, die jeweils beschrieben sind.
- **SIEHE AUCH** gibt Hinweise auf verwandte Einträge.
- **DIAGNOSE** erläutert die Fehlermeldungen, die hervorgerufen werden können. Meldungen, die unmittelbar verständlich sind, werden nicht aufgeführt.
- **ACHTUNG!** weist auf die Grenzfälle der jeweiligen Kommandos hin.
- **FEHLER** beschreibt bekannte Fehler in der Software, die nicht berichtigt worden sind. Gelegentlich wird ein Vorschlag für einen Ausweg angegeben.

Vor dem Kapitel 1 stehen ein "Inhaltsverzeichnis" mit einer Auflistung der Haupt- und Sekundärkommandonamen.

Kennenlernen des Systems

Dieses Kapitel enthält die wichtigsten Informationen, die zum Kennenlernen des UNIX-Systems benötigt werden: Wie man sich anmeldet und abmeldet, wie man über das Terminal im Dialog arbeitet, und wie ein Programm ausgeführt wird. (Vgl. *User's Guide*, der eine vollständige Einführung in das System enthält.)

Anmelden

Sie müssen sich über ein Vollduplex-ASCII- Terminal im UNIX-System anmelden. Außerdem müssen Sie eine gültige Login-Kennung haben, die Sie eventuell (zusammen mit Hinweisen für den Einstieg in das UNIX-System) vom Systemverwalter erhalten. Normale Geschwindigkeiten für Terminals sind 120, 240, 480 und 960 Zeichen pro Sekunde (1200, 2400, 4800 und 9600 Baud). Einige UNIX-Systeme sehen für jede mögliche Geschwindigkeit eine andere Zugriffsart vor, während andere Systeme mehrere Geschwindigkeiten über ein gemeinsames Zugriffsverfahren anbieten. Im letzteren Fall gibt es eine "bevorzugte" Geschwindigkeit; greift man von einem Bildchirmgerät, das auf eine andere Geschwindigkeit gesetzt ist, auf diese Geschwindigkeit zu, erscheint eine Folge sinnloser Zeichen auf dem Bildschirm (die **login:**-Meldung bei falscher Geschwindigkeit). Man betätigt dann die "Break-", "Unterbrechungs-" oder "Achtung-" Taste solange, bis die **login:**-Meldung erscheint.

Die meisten Terminals besitzen einen Geschwindigkeitsschalter, der auf die geeignete Geschwindigkeit gesetzt werden sollte, sowie einen Halb- /Vollduplex-Schalter, der auf Vollduplex gesetzt werden sollte. Nach der Herstellung einer Verbindung gibt das System **login:** aus. Man antwortet durch Eingabe der eigenen Login-Kennung mit anschließendem "Return". Hat der Benutzer ein Paßwort, wird er vom System danach gefragt, das dieses Wort jedoch weder am Bildschirm wiederholt noch ausgibt. Nach erfolgter Anmeldung haben die Tasten "Return", New-Line und "Zeilenvorschub" jeweils dieselbe Bedeutung.

Der Login-Name muß in Kleinbuchstaben eingegeben werden. Gibt man hier Großbuchstaben ein, nimmt das UNIX-System, daß das entsprechende Terminal nur Großbuchstaben erzeugen kann, und behandelt in der folgenden Sitzung alle eingegebenen Buchstaben als Großbuchstaben. Bei erfolgreicher Anmeldung gibt die Shell ein **$** auf dem Bildschirm aus.

Nach dem Anmelden kann eine Tagesmeldung auf dem Bildschirm erscheinen, bevor das Gerät das Bereit-Zeichen ausgibt. Weitere Informationen sind bei *login*(1) zu finden, wo das Anmelden im Einzelnen erläutert wird, sowie bei *stty*(1), wo Sie erfahren, wie Sie Ihr Terminal für das System beschreiben können. In *profile*(4) (im *Programmer's Reference Manual*) wird erklärt, wie man erreicht, daß diese Aufgabe bei jedem Anmelden automatisch ausgeführt wird.

Abmelden

Zum Abmelden hat man zwei Möglichkeiten:

- Hat man sich mit dem Telefon angemeldet, braucht man zum Abmelden lediglich den Hörer aufzulegen.
- Man kann sich abmelden, indem man der Shell Dateiende eingibt (ASCII EOT-Zeichen, normalerweise "CONTROL-D"). Die Shell beendet sich daraufhin und die **login:**-Meldung erscheint erneut.

Kommunikation über das Terminal

Wenn man Eingabe für das UNIX-System über die Tastatur eingibt, werden die einzelnen Buchstaben gesammelt und vorübergehend gespeichert. Obwohl die Zeichen an Ihrem Bildschirm wiederholt werden, erfolgt die Übergabe dieser Zeichen an ein Programm erst nach der Eingabe eines "Return" (oder "New-Line-Zeichens"), wie oben im Abschnitt "Anmelden" beschrieben.

Die Ein-/Ausgabe des UNIX-Systems für Terminals erfolgt im Vollduplexverfahren. Es wird immer "vorausgelesen", das heißt die Eingabe kann jederzeit erfolgen, auch wenn ein Programm gerade eine Ausgabe durchführt. Selbstverständlich werden dabei Eingabezeichen mit den Ausgabezeichen vermischt. Die eingegebenen Zeichen werden immer aufgehoben und auch in der richtigen Reihenfolge interpretiert. Es gibt eine Grenze für das Vorauslesen, jedoch ist die Grenze großzügig ausgelegt und wird wahrscheinlich nicht überschritten.

Das Zeichen @ löscht alle Zeichen, die vor diesem Zeichen in einer Zeile stehen, d. h. die Zeile wird praktisch gelöscht. (@ wird als Zeichen für Zeilenlöschen bezeichnet.) Das Zeichen # löscht das letzte eingegebene Zeichen. Wiederholtes Betätigen von # löscht die Zeichen jeweils von rechts nach links, wobei nicht über den Anfang der Zeile hinaus gelöscht wird; @ und # können als das jeweilige Zeichen selbst angegeben werden, wenn man ihnen ein \ voranstellt (um ein \ zu löschen, braucht man daher zwei #). Diese standardmäßigen Lösch- und Zeilenlöschzeichen können auch geändert werden; vgl. *stty*(1).

CONTROL-S (auch als das ASCII -**DC3**-Zeichen bekannt) wird durch gleichzeitiges Betätigen der Steuertaste (Control- Taste) und der Taste mit dem Buchstaben **s** eingegeben und dient zum vorübergehenden Anhalten der Ausgabe. Dies ist bei Terminals nützlich, weil man damit verhindern kann, daß die Ausgabe vom Bildschirm verschwindet, bevor man sie gelesen hat. Die Ausgabe geht weiter, wenn ein **CONTROL-Q** (auch als **DC1** bekannt) eingegeben wird. Wenn man also beispielsweise **cat yourfile** eingegeben hat, und der Inhalt von **yourfile** so schnell auf dem Bildschirm abläuft, daß man ihn nicht lesen kann, würde man **CONTROL-S** eingeben, um die Ausgabe kurzfristig anzuhalten. Nach der Eingabe von **CONTROL-Q** läuft die Ausgabe mit der Geschwindigkeit wie zuvor weiter. Die Zeichen **CONTROL-S** und **CONTROL-Q** werden an kein anderes Programm weitergegeben, wenn sie auf diese Art benutzt werden.

Das ASCII-Zeichen **DEL** ("Löschen")wird nicht an Programme weitergegeben, sondern erzeugt ein *Unterbrechungssignal* genau wie das Signal "break" (Abbrechen), "interrupt" (Unterbrechung) oder "attention" (Achtung). Dieses Signal bewirkt im allgemeinen, daß das jeweils laufende Programm abgebrochen wird. Eine typische Anwendung ist zum Beispiel, der Abbruch einer langen Ausgabe, die man nicht wünscht. Programme können allerdings (statt abgebrochen zu werden) dieses Signal entweder ignorieren oder eine spezielle Behandlung vorsehen, wenn das Signal eintritt. Der Editor *ed*(1) fängt beispielsweise Unterbrechungen ab und stoppt, was *er* gerade ausführt, anstatt sich zu beenden. Daher kann eine Unterbrechung dazu benutzt werden, eine Editor-Ausgabe zu beenden ohne die Datei, die gerade editiert wird, zu verlieren.

Abgesehen von der Anpassung an die Geschwindigkeit des Terminals, versucht das UNIX-System, intelligent zu sein und zu ermitteln, ob der Benutzer ein Terminal mit "New-Line"-Funktion hat oder ob diese Funktion durch "Carriage-Return" und "Zeilenvorschub" simuliert werden muß. Im letzteren Fall werden alle *Eingabe*-"Carriage-Return"-Zeichen in "Zeilenvorschub"-Zeichen (Standardzeilenbegrenzer) umgewandelt, und ein "Carriage-Return"- und "Zeilenvorschub"-Paar wird am Bildschirm wiedergegeben. Gerät man aus Versehen in den falschen Modus, kann das Kommando *stty*(1) helfen.

In den Quellprogrammen des UNIX-Systems können Tabulatorzeichen frei benutzt werden. Wenn Ihr Terminal keine Tab-Funktion hat, können Sie dafür sorgen, daß Tabulatorzeichen bei der Ausgabe in Leerzeichen umgewandelt und bei der Eingabe als Leerzeichen wiedergegeben werden. Auch hier wird dieser Modus mit dem Kommando *stty*(1) gesetzt bzw. zurückgesetzt. Das System geht davon aus, daß Tabulatoren an jeder achten Zeichenstelle gesetzt sind. Mit dem Kommando *tabs*(1) werden Tabulatoren am Bildschirm gesetzt, sofern dies möglich ist.

Ein Programm ausführen

Nach erfolgreicher Anmeldung im UNIX-System steht ein Programm, das Shell heißt, mit Ihrem Terminal in Verbindung. Die Shell liest jede von Ihnen eingegebene Zeile, teilt die Zeile in einen Kommandonamen und seine Argumente auf und führt das Kommando aus. Ein Kommando ist ganz einfach ein ablauffähiges Programm. Normalerweise durchsucht die Shell zunächst das aktuelle Verzeichnis (vgl. "Das aktuelle Verzeichnis") nach dem angegebenen Programm, und wenn es dort nicht zu finden ist, die Systemdateiverzeichnisse, wie zum Beispiel **/bin** und **/usr/bin**. Kommandos, die das System bereitstellt, sind nichts besonderes, außer daß sie in Verzeichnissen stehen, in denen sich die Shell findet. Man kann die Kommandos auch in eigenen Verzeichnissen aufbewahren und die Shell veranlassen, die Kommandos in diesen Verzeichnissen zu suchen. Vgl. die Beschreibung für *sh*(1) im Abschnitt "Austauschen von Parametern", in dem die Shell-Umgebungsvariable **$PATH** erläutert wird.

Der Kommandoname ist das erste Wort einer Eingabezeile für die Shell; das Kommando und seine Argumente werden durch Leerzeichen oder durch Tabulatoren voneinander getrennt.

Wenn ein Programm fertig ist, übernimmt die Shell normalerweise die Steuerung wieder und gibt dem Benutzer das Bereitzeichen aus, um ihm anzuzeigen, daß sie für ein weiteres Kommando bereit ist. Die Shell hat zahlreiche andere Leistungsmerkmale, die im einzelnen in *sh*(1) beschrieben sind.

Das aktuelle Verzeichnis

Das UNIX-System hat ein Dateisystem, das als Hierarchie von Verzeichnissen angeordnet ist. Wenn Ihre Login-Kennung eingerichtet wird, erstellt der Systemverwalter gleichzeitig ein Verzeichnis für Sie (normalerweise mit demselben Namen wie Ihre Login-Kennung, das als Ihr *login*- oder *home*-Verzeichnis bezeichnet wird). Beim Anmelden wird dieses Verzeichnis dann das *aktuelle* Verzeichnis, und standardmäßig wird angenommen, daß sich jeder Dateiname, den Sie eingeben, in diesem Verzeichnis befindet. Da Sie der Eigentümer dieses Verzeichnisses sind, haben Sie Erlaubnis zum Lesen, Schreiben, Ändern oder Löschen des Verzeichnisinhalts. Die Erlaubnis, in andere Verzeichnisse und Dateien zu wechseln bzw. diese zu ändern, wird von den jeweiligen Eigentümern oder vom Systemverwalter erteilt oder verweigert. Um das aktuelle Verzeichnis zu wechseln, verwendet man *cd*(1).

Pfadnamen

Um Dateien oder Verzeichnisse anzusprechen, die nicht im aktuellen Verzeichnis sind, muß man einen Pfadnamen verwenden. Vollständige Pfadnamen beginnen mit einem /. Das ist der Name des *root*-Verzeichnisses des gesamten Dateisystems. Hinter dem Schrägstrich folgt dann der Reihe nach der Name (abgeschlossen mit /) von jedem Verzeichnis, das jeweils das nächste Unterverzeichnis enthält, bis schließlich der gewünschte Datei- oder Verzeichnisname erreicht ist (z.B. bezieht sich **/usr/ae/filex** auf die Datei **filex** im Verzeichnis **ae**, **ae** ist selbst wieder ein Unterverzeichnis von **usr** und **usr** ein Unterverzeichnis des Root-Verzeichnisses). Den vollständigen Pfadnamen des Verzeichnisses, in dem Sie gerade arbeiten, können Sie mit *pwd*(1) abfragen. Vgl. *intro*(2) im *Programmer's Reference Manual*, wo eine formale Definition von *Pfadname* angegeben wird.

Wenn das aktuelle Verzeichnis Unterverzeichnisse enthält, beginnen die Pfadnamen der Dateien in einem Unterverzeichnis mit dem Namen des entsprechenden Unterverzeichnisses (*ohne* einen vorangestellten /). Ein Pfadname kann überall dort verwendet werden, wo ein Dateiname benötigt wird.

Wichtige Kommandos, die Dateien betreffen, sind *cp*(1), *mv* (vgl. *cp*(1)) und *rm*(1), die Dateien kopieren, umbenennen bzw. löschen. Der Status von Dateien oder Verzeichnissen wird mit *ls*(1) ermittelt. Verzeichnisse werden mit *mkdir*(1) erstellt und mit *rmdir* (vgl. *rm*(1)) gelöscht.

Texteingabe und -anzeige

Text wird im allgemeinen über einen Editor eingegeben. Bekannte Beispiele für Editoren des UNIX-Systems sind *ed*(1) und *vi*(1). Die am häufigsten verwendeten Kommandos zur Textausgabe am Bildschirm sind *cat*(1), *pr*(1) und *pg*(1). Mit dem Kommando *cat*(1) wird der Inhalt von ASCII- Textdateien am Bildschirm ausgegeben ohne jede Bearbeitung. Das *pr*(1) Kommando teilt einen Text in Seiten auf, liefert Überschriften und ermöglicht mehrspaltige Ausgabe. Das *pg*(1) Kommando gibt den Text in Teilen aus, die jeweils nicht größer als der Bildschirm sind.

Ein ausführbares Programm erzeugen

Wenn Sie mit Hilfe eines Editors den Text Ihres Programms in eine Datei geschrieben haben, können Sie die Datei an ein geeignetes Sprachübersetzungsprogramm übergeben. Dieses Programm akzeptiert nur Dateien, bei denen die vorgeschriebenen Namenskonventionen eingehalten wurden: alle Dateinamen von Dateien mit C-Programmen müssen mit dem Zusatz **.c** enden, von Dateien mit Fortran-Programmen mit dem Zusatz **.f**. Die Ausgabe des Übersetzungsprogramms wird in eine Datei mit dem Namen **a.out** im aktuellen Verzeichnis geschrieben, sofern Sie keine Option angegeben haben, um die Ausgabe in eine andere Datei zu schreiben. (Zum Umbenennen von **a.out** verwendet man *mv*(1).) Wenn das Programm in Assembler geschrieben ist, müssen Sie wahrscheinlich zusätzlich Bibliotheksfunktionen laden (vgl. *ld*(1) im *Programmer's Reference Manual*).

Wenn Sie diesen Prozeß ohne Fehler beendet haben, können Sie das Programm ausführen, indem Sie als Antwort auf das Bereitzeichen **$** der Shell den entsprechenden Dateinamen übergeben. Ihre Programme können genau wie die Systemprogramme Argumente aus der Kommandozeile erhalten; vgl. *exec*(2) im *Programmer's Reference Manual*. Weitere Informationen über das Erstellen und Ausführen von Programmen sind im *Programmer's Guide* zu finden.

Kommunikation mit anderen

Bestimmte Kommandos ermöglichen eine Kommunikation *zwischen Benutzern*. Auch wenn Sie diese Kommandos selbst nicht verwenden möchten, sollten Sie sich mit ihnen vertraut machen, weil es möglich ist, daß jemand anderes mit Ihnen kommunizieren möchte. *mail*(1) oder *mailx*(1) hinterlassen eine Nachricht. Ein anderer Benutzer wird beim Anmelden oder in regelmäßigen Abständen während seiner Sitzung über das Vorhandensein einer Nachricht informiert. Zur Kommunikation mit einem anderen Benutzer, der auch angemeldet ist, verwendet man das Kommando *write*(1). In den entsprechenden Einträgen in diesem Handbuch wird auch beschrieben, wie Sie auf diese beiden Kommandos reagieren können, wenn Sie der Empfänger einer Nachricht sind.

Weitere Informationen über die Kommunikation mit anderen finden Sie in den Übungen in Kapitel 8 des *User's Guide*.

BEZEICHNUNG

intro – Einführung in Kommandos und Anwenderprogramme

BESCHREIBUNG

In diesem Abschnitt werden Kommandos in alphabetischer Reihenfolge beschrieben, die für die Targon /31 zur Verfügung stehen. In den Überschriften werden zusätzlich Unterscheidungen nach gewissen Sachgebieten angegeben.

Folgende Pakete von Dienstprogrammen werden bei Auslieferung des Computers mitgeliefert:

Basis-Dienstprogramme für Netzwerke
Dienstprogramme für Magnetbänder
Dienstprogramme für Verzeichnis- und Dateiverwaltung
Editoren
Basis-Dienstprogramme
Grafik-Dienstprogramme
Dienstprogramme für Dienstprogramme für Interprozeß-Kommunikation
Dienstprogramme für Drucker
Dienstprogramme für Leistungsmessung
Dienstprogramme für Datenschutz
Spell-Dienstprogramme
Dienstprogramme zur Unterstützung von Terminals
Dienstprogramme zur Verwaltung von Terminal-Parametern
Dienstprogramme für Benutzerkonfiguration

Folgende Pakete von Dienstprogrammen können zusätzlich gekauft werden:

Dienstprogramme für Netzwerk-Unterstützung

Beschreibungssyntax für Kommandos.

Wenn keine anderen Angaben erfolgen, werden Optionen und andere Argumente für die Kommandos im Abschnitt Synopsis gemäß folgender Syntax angegeben: Die Interpretation dieser Syntax ist beschrieben:

name [*–option*...] [*cmdarg*...]
wobei

[ ]	kennzeichnet eine wahlweise Option *option* oder ein wahlweises Argument *cmdarg*.
...	kennzeichnet Wiederholungen von *option* oder *cmdarg*
name	Der Name einer ausführbaren Datei.
option	(Immer mit vorangestelltem "–".) *noargletter*... oder *argletter optarg*[,...]

noargletter Ein Einzelbuchstabe, der eine Option ohne ein Optionsargument darstellt. Es ist zu beachten, daß mehr als eine *noargletter* Option nach einem "–" in einer Gruppe zusammengefaßt sein können (Regel 5, unten).

argletter Ein Einzelbuchstabe, der eine Option darstellt, die ein Optionsargument benötigt.

optarg Ein Optionsargument (Zeichenfolge) für einen vorangehenden *argletter.* Es ist zu beachten, daß Gruppen von *optargs* nach einem *argletter* durch Kommata voneinander getrennt werden müssen oder durch Leerzeichen und dann in Anführungszeichen (Regel 8, unten).

cmdarg Pfadname (oder ein anderes Kommandoargument), das *nicht* mit "–" beginnt oder "–" alleine zur Angabe der Standardeingabe.

Kommandosyntax-Standard: Regeln

Diese Kommandosyntax-Regeln werden von den aktuellen Kommandos noch nicht vollständig befolgt, alle neuen Kommandos werden sie aber einhalten. Alle Shell-Prozeduren sollten *getopts* (1) verwenden, um Stellungsparameter und Optionen zu überprüfen. Es unterstützt die unten angegebenen Regeln 3-10. Die Einhaltung der anderen Regeln muß vom Kommando selbst überprüft werden.

1. Kommandonamen (*name* oben) müssen zwischen zwei und neun Zeichen lang sein.
2. Kommandonamen dürfen nur aus Kleinbuchstaben und Ziffern bestehen.
3. Optionsnamen (*option* oben) müssen ein Zeichen lang sein.
4. Allen Optionen muß ein "–" vorangestellt sein.
5. Optionen ohne Argumente *(noargletter* oben) können hinter einem einzelnen "–" zusammengefaßt werden.
6. Vor dem ersten Optionsargument (*optarg* oben) nach einer Option muß eine Leerstelle stehen.
7. Optionsargumente können nicht wahlweise sein.
8. Gruppen von Optionsargumenten, die einer Option folgen, müssen entweder durch Kommata voneinander getrennt werden oder durch eine Leerstelle und dann in Anführungszeichen. (z.B. –o xxx,z,yy oder –o "xxx z yy").
9. Alle Optionen müssen in der Kommandozeile vor den Argumenten (*cmdarg* oben) stehen.

10. Zur Anzeige des Endes der Optionen kann "--" verwendet werden.
11. Die Reihenfolge der Optionen relativ zueinander soll keine Rolle spielen.
12. Das jeweilige Kommando kann festlegen, ob die relative Reihenfolge seiner Argumente *(cmdarg* oben) eine Rolle spielt.
13. "-" mit vorangestellter und nachfolgender Leerstelle darf nur verwendet werden, um die Standardeingabe anzugeben.

SIEHE AUCH

getopts(1).
exit(2), wait(2), getopt(3C) im *Programmer's Reference Manual.*
Kennenlernen des Systems, am Anfang dieses Handbuchs.

DIAGNOSE

Nach Beendigung gibt jedes Kommando zwei Statusbytes zurück, wobei das eine vom System geliefert wird und den Grund für die Beendigung angibt und das andere (im Falle einer "normalen" Beendigung) vom Programm bereitgestellt wird [vgl. *wait*(2) und *exit*(2)]. Das erste Byte ist 0 bei normaler Beendigung; das letzte Byte ist gewöhnlich 0 für eine erfolgreiche Ausführung und ungleich Null zur Anzeige von Fehlern, wie z.B. falsche Parameter oder unzulässige bzw. nicht zugreifbare Daten. Es wird verschiedentlich "Exitcode", "Endestatus" oder "Rückgabe-Code" genannt; es wird nur beschrieben, wenn es sich um besondere Konventionen handelt.

ACHTUNG!

Einige Kommandos erzeugen unerwartete Ergebnisse bei der Verarbeitung von Dateien mit Nullzeichen. Diese Kommandos behandeln Texteingabezeilen häufig als Zeichenfolgen und arbeiten daher nicht korrekt, wenn sie ein Nullzeichen innerhalb einer Zeile antreffen.

BEZEICHNUNG

300, 300s – Spezialfunktionen des DASI-Terminals 300 und 300s unterstützen

ÜBERSICHT

300 [+12] [−n] [−dt,l,c]

300s [+12] [−n] [−dt,l,c]

BESCHREIBUNG

Das Kommando *300* unterstützt Sonderfunktionen und optimiert den Einsatz des DASI -Terminals-300 (GSI 300 oder DTC 300); *300s* führt die gleichen Funktionen für das DASI -Terminals-300s (GSI 300s oder DTC 300s) aus. Bewegungen für halbe Zeile vorwärts, halbe Zeile und ganze Zeilen zurück, werden in die richtigen senkrechten Bewegungen umgewandelt. Bei der folgenden Beschreibung des Kommandos *300* ist zu beachten, daß der Bezug auf bestimmte Kommandos, (wie z.B. *nroff*, *neqn*, *eqn*, usw.) nur möglich ist, falls auf Ihrem Rechner die Software der DOCUMENTER'S WORKBENCH installiert ist. Das Kommando 300 versucht auch, griechische Buchstaben und andere Sondersymbole zu zeichnen. Die Verwendung von Text mit einer Zeichendichte von 12 Zeichen pro Zoll (12-pitch) ist ebenfalls möglich. Außerdem wird die Druckzeit um 5 bis 70% verkürzt. Das Kommando *300* kann verwendet werden, um Gleichungen schön auszudrucken, und zwar durch:

neqn file ... | nroff | 300

ACHTUNG! wenn Ihr Terminal einen PLOT -Schalter besitzt, muß dieser auf *on* geschaltet werden, ehe Sie *300* verwenden.

Das Verhalten von *300* kann durch Angabe von Optionen modifiziert werden, um Texte mit einer Zeichendichte von 12 Zeichen pro Zoll (12-pitch), nicht ganzzahlige Zeilenabstände, Nachrichten und Verzögerungen zu bearbeiten.

+12 ermöglicht Verarbeitung von Texten mit 12-pitch und 6 Zeilen/Zoll. DASI -300-Terminals gestatten normalerweise nur zwei Kombinationen: 10-pitch, 6 Zeilen/Zoll oder 12-pitch, 8 Zeilen/Zoll. Um die Kombination 12-pitch, 6 Zeilen pro Zoll zu erhalten, muß der Benutzer den PITCH -Schalter auf 12 drehen und die Option **+12** verwenden.

−n kontrolliert die Größe des halbzeiligen Zeilenabstands. Eine halbe Zeile entspricht standardmäßig 4 senkrechten Plot-Inkrementen. Da jedes Inkrement 1/48 eines Zolls entspricht, erfordert ein 10-pitch-Zeilenvorschub 8 Inkremente, bei einem 12-pitch-Zeilenvorschub sind nur 6 erforderlich. Die erste Ziffer von *n* ersetzt die Standardeinstellung, wodurch Hochstellung und Tiefstellung dem individuellen Geschmack

entsprechend gestaltet werden kann. Zum Beispiel können *nroff*- halbe Zeilen als viertel Zeilen fungieren, indem man **-2** verwendet. Der Benutzer könnte auch passende halbe Zeilen für 12-pitch, 8 Zeilen/Zoll-Modus durch Benutzen der Option **-3** erhalten, wobei der PITCH -Schalter auf 12-pitch gestellt ist.

-d*t*,*l*,*c* steuert Verzögerungsfaktoren. Die Standardeinstellung ist **-d3,90,30**. DASI -300-Terminals geben manchmal eigentümliche Ausgaben aus, wenn sehr lange Zeilen, zu viele Tabulatoren-Zeichen oder lange Zeichenfolgen mit nichtidentischen Zeichen ohne Leerzeichen im Text vorkommen. Ein Null-(Verzögerungs)-Zeichen wird in eine Zeile eingefügt für jeweils *t* Tabulatoren und für jede zusammenhängende Zeichenfolge von *c* Zeichen, die nicht Leerzeichen oder Tabulator-Zeichen sind. Wenn eine Zeile länger als *l* ist, werden 1+(Gesamtlänge)/20 Nullen am Ende dieser Zeile eingefügt. Wenn Standardwerte erwünscht sind, können Angaben ausgelassen werden. Ist der Wert für *t* (*c*) Null, ergibt dies zwei Nullbytes pro Tabulator(Zeichen). Der erste Fall kann für C-Programme benutzt werden, der zweite für Dateien wie **/etc/passwd.** Da das Verhalten des Terminals von den jeweils auszudruckenden Zeichen und der Systembelastung abhängt, muß der Benutzer eventuell mit diesen Werten experimentieren, um eine korrekte Ausgabe zu erhalten. Die Option **-d** ist nur als allerletzter Ausweg für die wenigen Fälle gedacht, bei denen ein Ausdruck schwierig ist. Zum Beispiel kann die Datei **/etc/passwd** unter Verwendung von **-d3,30,5** ausgedruckt werden. Der Wert **-d0,1** wird für C-Programme empfohlen, bei denen viele Einrückebenen vorkommen. Es ist zu beachten, daß diese Verzögerungssteuerung und die für Wagenrücklauf und Zeilenvorschub eingestellten Verzögerungen so gegenseitig stark beeinflussen. Die *stty*(1)- Einstellungen **n10 cr2** oder **n10 cr3** sind für die meisten Fälle geeignet.

Wenn Sie Papier manuell einfügen bzw. Schriftarten mitten im Text ändern wollen, verwenden Sie das Kommando *300* mit der *nroff* -s Option oder **.rd** Anfrage. Anstelle der Return-Taste müssen Sie die Zeilenvorschub-Taste betätigen, um eine Antwort zu erhalten.

In vielen (aber nicht allen) Fällen sind folgende Sequenzen gleichwertig:

```
nroff -T300 files ...  and  nroff files ... | 300
nroff -T300-12 files ...  und  nroff files ... | 300 +12
```

Das Kommando *300* muß daher nur eingesetzt werden, wenn besondere Verzögerungen oder Optionen erforderlich sind; in einigen wenigen Fällen kann die zusätzliche Bewegungsoptimierung von *300* eine besser ausgerichtetete Ausgabe produzieren.

Die *neqn* Namen und die entsprechende Ausgabe für Griechisch und die von *300* unterstützten Sonderzeichen werden in *greek*(5) gezeigt.

SIEHE AUCH:

450(1), mesg(1), graph(1G), stty(1), tabs(1), tplot(1G).
eqn(1), nroff(1), tbl(1) im Handbuch Dokumentations-Tools
greek(5) im *Programmer's Reference Manual.*

FEHLER

Einige Sonderzeichen können nicht richtig in Spalte 1 gedruckt werden, weil der Schreibkopf von dort nicht nach links bewegt werden kann.
Wenn Ihre Ausgabe Griechisch und/oder Umkehrzeilenvorschübe enthält, sollten Sie eine Schreibwalze mit Reibantrieb anstelle eines Formulartraktors verwenden. Der Formulartraktor ist zwar ausreichend für Entwürfe, kann jedoch in inverser Richtung Rutschen bzw. Verzerren der griechischen Zeichen und schlechte Ausrichtung der ersten Zeile nach einer oder mehreren inversen Zeilenvorschüben verursachen.

BEZEICHNUNG

4014 – Seitenwechseleinstellung für das Tektronix-4014-Terminal

ÜBERSICHT

4014 [**-t**] [**-n**] [**-c**N] [**-p**L] [file]

BESCHREIBUNG

Die Ausgabe von *4014* ist für eine Tektronix-4014-Terminal bestimmt; *4014* ordnet 66 Zeilen passend auf den Bildschirm an, teilt den Bildschirm in *N* Spalten auf und rückt 8 Leerzeichen ein bei einspaltiger Ausgabe, die standardmäßig eingestellt ist. Tabulatoren, Leerzeichen und Rückwärtsschritte werden gesammelt und erforderlichenfalls aufgezeichnet. Halbe Zeilen und Zeile rückwärts werden für den Fernschreiber TELETYPE Model 37 unterstützt und geplottet. Am Ende jeder Seite wartet *4014* auf eine neue Zeile (Leerzeile) von der Tastatur, ehe mit der nächsten Seite fortgesetzt wird. In diesem Wartezustand bewirkt das Kommando !*cmd*, daß das Kommando *cmd* an die Shell geschickt wird.

Die Optionen der Kommandozeile lauten:

-t Kein Warten zwischen Seiten (nützlich bei Ausgabe-Umlenkung in eine Datei).

-n Ausgabe beginnt an der aktuellen Position des Cursors und kein Löschen des Bildschirms.

-c*N* Bildschirmaufteilung in *N* Spalten und Warten nach der letzten Spalte.

-p*L* Seitenlänge auf *L* setzen; *L* akzeptiert die Maßangaben **i** (Zoll) und **l** (Zeilen); Standardvorgabe ist Zeilen.

SIEHE AUCH:

pr(1), tc(1).
troff(1) im Handbuch Dokumentations-Tools

BEZEICHNUNG

450 – Sonderfunktionen des DASI-450-Terminals unterstützen

ÜBERSICHT

450

BESCHREIBUNG

Das Kommando *450* unterstützt besondere Funktionen und optimiert den Einsatz der DASI -450-Terminal bzw. anderer Terminals mit gleicher Funktionsweise wie zum Beispiel Diablo 1620 oder Xerox 1700. Umwandlung von halber Zeile vorwärts, halber Zeile rückwärts und ganzer Zeile rückwärts in die richtigen senkrechten Bewegungen wird ausgeführt. Ausgabe von griechischen Buchstaben und anderen Sondersymbolen genau wie bei *300*(1) ist ebenfalls möglich. Es ist zu beachten, daß bestimmte Kommandos (z. B. *eqn*, *nroff*, *tbl*, usw.) nur aufgerufen werden können, wenn Ihr System die Software der DOCUMENTER'S WORKBENCH enthält. *450* sollten Sie verwenden, um Gleichungen schön auszudrucken und zwar in der Reihenfolge

neqn -Datei ... | nroff | 450

ACHTUNG! Prüfen Sie, ob der PLOT -Schalter an Ihre Terminal auf ON geschaltet ist, ehe Sie *450* verwenden. Der SPACING -Schalter sollte in die gewünschte Position (entweder 10- oder 12-pitch) gestellt werden. In beiden Fällen ist der Senkrechtabstand 6 Zeilen/Zoll, sofern er nicht durch eine passende Umschaltfolge auf 8 Zeilen pro Zoll gesetzt wird. Wenn Sie Papier manuell einfügen bzw. Schriftarten mitten im Text ändern wollen, verwenden Sie *450* mit dem *nroff* –s Schalter oder den **.rd** Anfragen. Anstelle der Return-Taste müssen Sie die Zeilenvorschub-Taste betätigen, um eine Antwort zu erhalten.

In vielen (aber nicht allen) Fällen kann der Einsatz von *450* umgangen werden, indem Sie folgendes eingeben:

nroff –T450 files ...

oder

nroff –T450–12 files ...

Der Einsatz von *450* kann häufig vermieden werden, sofern nicht besondere Verzögerungen oder Optionen erforderlich sind; in einigen wenigen Fällen kann die zusätzliche Bewegungsoptimierung von *450* eine besser ausgerichtetete Ausgabe produzieren.

Die *neqn* Namen und die entsprechende Ausgabe für Griechisch und die von *450* unterstützten Sonderzeichen werden in *greek(5)* gezeigt.

SIEHE AUCH:

300(1), mesg(1), stty(1), tabs(1), graph(1G), tplot(1G),
eqn(1), nroff(1), tbl(1) im Handbuch Dokumentations-Tools
greek(5) im *Programmer's Reference Manual.*

FEHLER

Einige Sonderzeichen können nicht richtig in Spalte 1 gedruckt werden, weil der Schreibkopf von dort nicht nach links bewegt werden kann.
Wenn Ihre Ausgabe Griechisch und/oder inverse Zeilenvorschübe enthält, sollten Sie eine Schreibwalze mit Reibantrieb anstelle eines Formulartraktors verwenden. Der Formulartraktor ist zwar ausreichend für Entwürfe, kann jedoch in inverse Richtung Rutschen bzw. Verzerren der griechischen Zeichen und schlechte Ausrichtung der ersten Zeile nach einem oder mehreren inversen Zeilenvorschüben verursachen.

BEZEICHNUNG

acctcom – Prozeßabrechnungsdatei(en) analysieren und anzeigen

ÜBERSICHT

acctcom [[options] [file]] . . .

BESCHREIBUNG

acctcom liest die Datei *file* (die Standardeingabedatei oder */usr/adm/pacct*) und schreibt ausgewählte Sätze in die Standardausgabedatei. Die Eingabedateien haben den Aufbau, der in *acct*(4) beschrieben ist. Jeder Satz enthält Angaben über *einen* ausgeführten Prozeß. Folgende Angaben werden ausgegeben:

COMMAND NAME (Kommandobezeichnung), **USER** (Benutzername), **TTYNAME** (Terminalname), **START TIME** (Startzeit), **END TIME** (Endezeit), **REAL (SEC)** (tatsächliche Zeit in Sek.), **CPU (SEC)** (CPU-Zeit in Sek.), **MEAN SIZE(K)** (mittlere Speichergröße in KB) und optional *F* (*exec*-Option des *fork*-Kommandos: 1 für *fork* ohne *exec*), **STAT** (System-Endestatus), **HOG FACTOR** (siehe unten), **KCORE MIN, CPU FACTOR** (CPU-Faktor), **CHARS TRNSFD** (Anzahl der übertragenen Zeichen), und **BLOCKS READ** (Geamtzahl der gelesenen und geschriebenen Blöcke).

Wenn das Kommando mit Superuser-Privilegien ausgeführt wird, wird ein # vor die Kommandobezeichnung gestellt. Wenn ein Prozeß keinem bekannten Terminal zugeordnet ist, wird in dem Feld **TTYNAME** ein ? ausgegeben.

Normalerweise dient die Standardeingabe als Eingabedatei. Die Datei */usr/adm/pacct* hingegen wird als Eingabedatei herangezogen, wenn der Parameter *file* fehlt und die Standardeingabe mit einem Terminal verbunden oder */dev/null* ist (also wenn das Programm aus der Shell-Ebene mit & gestartet wird).

Wenn *file* angegeben ist, werden die gewünschten Dateien in der angegebenen Reihenfolge eingelesen. In der Regel werden die Dateien von vorne nach hinten gelesen, d. h. in der chronologischen Reihenfolge der Prozeß-Endezeiten. Normalerweise ist */usr/adm/pacct* die aktuelle Datei, die analysiert werden soll. Ein aktives System kann aber mehrere solche Dateien benötigen. Diese sind dann mit Ausnahme der aktuellen Datei in */usr/adm/pacct?* zu finden. Die folgenden Optionen *options* sind möglich:

–a (averages) Im Anschluß an die normalen Ausgabesätze werden zusätzlich einige statistische Durchschnittswerte für die ausgewählten Prozesse ausgegeben.

-b (backwards) Die Eingabedatei wird von hinten nach vorne gelesen, so daß die zuletzt ausgeführten Kommandos zuerst erscheinen. Diese Option ist wirkungslos, wenn die Standardeingabedatei verarbeitet wird.

-f (fork) Es werden zwei zusätzliche Spalten mit der *exec*-Option des *fork*-Kommandos und dem System-Endestatus ausgegeben. Die numerische Ausgabe für diese Option erfolgt in Oktalzahlen.

-h (hog factor) Anstatt der mittleren Speichergröße wird die relative CPU-Auslastung ausgegeben; das ist der Anteil der gesamten verfügbaren CPU-Zeit, den ein Prozeß während seiner Ausführung verbraucht hat. Der Hog-Faktor errechnet sich nach der Formel:

(gesamte CPU\-Zeit)/(verbrauchte Zeit).

-i (i/o counts) Es werden zusätzliche Spalten mit den Ein-/Ausgabezählern ausgegeben.

-k (kcore) Anstatt der Speichergröße wird die Gesamtzahl der benötigten KCore-Minuten ausgegeben.

-m (mean core) Es wird die mittlere Speichergröße ausgegeben (Standard).

-r (relative) Es wird der CPU-Faktor ausgegeben (Benutzerzeit/(Systemzeit + Benutzerzeit).

-t (times) System- und Benutzer-CPU-Zeit werden getrennt ausgewiesen.

-v Es werden keine Spaltentitel ausgegeben.

-l *line* Nur zu dem Terminal */dev/line* gehörende Prozesse werden angezeigt.

-u *user* Nur dem Benutzer *user* gehörende Prozesse werden angezeigt. Mögliche Werte für *user* sind: Benutzerkennung, Benutzername (wird in die Benutzerkennung umgesetzt), # (zur Bezeichnung von Prozessen, die mit Superuser-Privilegien ausgeführt wurden) oder ? (zur Bezeichnung von Prozessen, die mit einer unbekannten Benutzerkennung verbunden sind).

-g *group* Nur der Gruppe *group* gehörende Prozesse werden angezeigt. *group* kann durch die Gruppenkennung oder den Gruppennamen angegeben werden.

-s *time* (start) Es werden nur die Prozesse berücksichtigt, die ab der mit *time* angegebenen Uhrzeit existierten. Die Uhrzeit *time* muß in dem Format *Std* [:*Min* [:*Sek*]] angegeben werden.

-e *time* (end) Es werden nur die Prozesse berücksichtigt, die bis zu der mit *time* angegebenen Uhrzeit existierten.

-S *time* (Start) Es werden nur die Prozesse berücksichtigt, die ab der mit *time* angegebenen Uhrzeit gestartet wurden.

-E *time* (End) Es werden nur die Prozesse berücksichtigt, die bis zu der mit *time* angegebenen Uhrzeit beendet wurden. Wenn *time* für **-S** und **-E** gleich ist, werden die Prozesse angezeigt, die um die mit *time* angegebene Uhrzeit existierten.

-n *pattern* Es werden nur die Kommandos berücksichtigt, die dem Muster *pattern* entsprechen. *pattern* kann ein regulärer Ausdruck wie in *ed*(1) sein, abgesehen davon, daß + ein oder mehrere Vorkommen bezeichnet.

-q Die normale Ausgabe wird unterdrückt. Stattdessen werden, wie mit der Option **-a**, nur die statistischen Durchschnittswerte ausgegeben.

-o *ofile* Ausgewählte Prozeßabrechnungssätze werden im gleichen Format wie die Eingabedatei in die Datei *ofile* geschrieben; die standardmäßige Ausgabe wird unterdrückt.

-H *factor* Es werden nur die Prozesse berücksichtigt, die den Hog-Faktor *factor* überschreiten (siehe oben, Option **-h**).

-O *sec* Es werden nur die Prozesse berücksichtigt, deren CPU-Systemzeit *sec* Sekunden überschreitet.

-C *sec* Es werden nur die Prozesse berücksichtigt, deren gesamte CPU-Systemzeit (System- und Benutzerzeit) *sec* Sekunden überschreitet.

-I *chars* Es werden nur die Prozesse berücksichtigt, die mehr als *chars* Zeichen übertragen.

DATEIEN

/etc/passwd
/usr/adm/pacct
/etc/group

SIEHE AUCH

ps(1), su(1)
acct(2), acct(4), utmp(4) im *Programmer's Reference Manual*
acct(1M), acctcms(1M), acctcon(1M), acctmerg(1M), acctprc(1M), acctsh(1M), fwtmp(1M), runacct(1M) im *System Administrator's Reference Manual*

FEHLER

acctcom liefert nur Daten über abgeschlossene Prozesse. Auskunft über aktive Prozesse erhalten Sie mit *ps*(1). Wenn *time* größer als die momentane Uhrzeit ist, wird angenommen, daß es sich um eine Uhrzeit vom Vortag handelt.

BEZEICHNUNG

arp – ARP–Tabellen verwalten und anzeigen

ÜBERSICHT

arp *hostname*
arp -a [*unix*] [*kmem*]
arp -d *hostname*
arp -s *hostname ether_addr* [**temp**] [**pub**] [**trail**]
arp -f *filename*

BESCHREIBUNG

Das Programm *arp* dient zur Verwaltung und Anzeige der ARP-Tabellen. Die Tabellen werden zur Umsetzung von Internet- in Ethernet-Adressen verwendet, die wiederum von dem Address Resolution Protocol (ARP; Adreßumsetzungsprotokoll) benutzt werden (siehe *arp*(4)).

Wenn keine Optionen angegeben werden, zeigt das Programm den aktuellen ARP-Eintrag für den Host *hostname* an. Statt des Hostnamens kann auch die Hostnummer in der Internet-Punktschreibweise angegeben werden. Die Option **-a** (all) bewirkt, daß alle aktuellen ARP-Einträge angezeigt werden. Diese werden der Datei *kmem* (Standard: /dev/kmem) entnommen, deren Inhalt auf der Kernel-Datei *unix* (Standard: /unix) beruht.

Durch Angabe der Option **-d** (delete) kann ein Superuser einen Eintrag des Hosts *hostname* löschen.

Mit der Option **-s** (set) kann ein ARP-Eintrag für den Host *hostname* mit der Ethernet-Adresse *ether_addr* neu angelegt werden. Die Ethernet-Adresse besteht aus sechs Byte (hexadezimal), die durch Doppelpunkt voneinander getrennt sein müssen. Der neue Eintrag wird dauerhaft angelegt, außer wenn das Wort **temp** angegeben wird. Wird das Wort **pub** angegeben, dann wird der Eintrag "veröffentlicht". In diesem Fall verhält sich das System wie ein ARP-Server: es antwortet auf Anforderungen nach *hostname*, auch wenn die Hostadresse nicht seine eigene ist. Das Wort **trail** bedeutet, daß beim Senden an diesen Host die IP-Trailer-Encapsulation verwendet werden kann.

Die Option **-f** ermöglicht es, mehrere Einträge aus der Datei *filename* einzulesen und in den ARP-Tabellen anzulegen. Die Einträge in der Datei *filename* müssen das folgende Format haben:

hostname ether_addr [**temp**] [**pub**] [**trail**]

Die Bedeutung der Argumente ist die gleiche wie oben.

SIEHE AUCH

inet(3N), arp(4), ifconfig(1C)

BEZEICHNUNG

at, batch – Ausführung von Kommandos zu einem späteren Zeitpunkt

ÜBERSICHT

at *time* [*date*] [+ *increment*]
at *−r* job...
at *-l* [*job* ...]

batch

BESCHREIBUNG

at und *batch* lesen Kommandos von der Standardeingabe, die zu einem späteren Zeitpunkt ausgeführt werden sollen. Mit *at* können Sie angeben, zu welchem Zeitpunkt die Kommandos ausgeführt werden sollen, während mit *batch* in eine Warteschlange eingereihte Aufgaben dann ausgeführt werden, wenn die Belastung des Systems es zuläßt. *at* kann mit folgenden Optionen verwendet werden:

−r Entfernt Aufträge, die zuvor mit *at* geplant wurden.

−l Gibt alle Aufträge an, die für den aufrufenden Benutzer geplant sind.

Standardausgaben und Standardfehlerausgaben werden an den Benutzer abgeschickt, sofern sie nicht umgelenkt werden. Die Shell-Umgebungsvariablen, aktuelles Verzeichnis, umask und ulimit werden beibehalten, wenn die Kommandos ausgeführt werden. Offene Dateikennzahlen, nichtprogrammierte Sprünge und Priorität gehen verloren.

Benutzer dürfen *at* verwenden, wenn ihr Name in der Datei **/usr/lib/cron/at.allow** aufgeführt ist. Wenn die Datei nicht existiert, wird die Datei **/usr/lib/cron/at.deny** geprüft, um festzustellen, ob dem Benutzer Zugriff auf *at* verweigert werden soll. Wenn beide Dateien nicht vorhanden sind, kann nur unter root ein Auftrag erteilt werden. Wenn **at.deny** leer ist, darf *at* uneingeschränkt aufgerufen werden. Die Berechtigungs/Verweigerungsdateien bestehen aus einem Benutzernamen pro Zeile. Diese Dateien können nur vom Systemverwalter modifiziert werden.

time kann mit 1, 2, oder 4 Ziffern angegeben werden. Zahlen aus einer oder zwei Ziffern werden als Stunden interpretiert, vier Ziffern als Stunden und Minuten. Die Zeit kann wahlweise auch als zwei Zahlen, die durch einen Doppelpunkt getrennt sind, angegeben werden, d. h. *hour:minute*. Ein Zusatz in Form von **am** oder **pm** kann angehängt werden; sonst wird eine 24-Stunden-Uhr angenommen. Der Zusatz **zulu** kann zur Angabe von GREENWICH MEAN TIME benutzt werden. Die besonderen Namen **noon, midnight, now** und **next** werden auch erkannt.

Ein optionales *date* (Datum) kann entweder als ein Monatsname, dem eine Tagnummer folgt (und möglicherweise Jahreszahl mit vorausgehendem wahlweisen Komma) oder als ein Wochentag (ausgeschrieben oder bis auf drei Zeichen abgekürzt) angegeben werden. Zwei besondere „Tage" **today** und **tomorrow** werden erkannt. Wenn *date* nicht angegeben ist, wird **today** angenommen, falls die angegebene Uhrzeit größer als die aktuelle Uhrzeit ist und **tomorrow** wird angenommen, wenn diese kleiner ist. Wenn der Wert für den angegebenen Monat kleiner als der aktuelle Monat ist (und keine Jahreszahl vorhanden ist), wird das nächste Jahr angenommen.

Das optionale *increment* ist einfach eine Zahl, mit einem der folgenden Zusätze: **minutes, hours, days, weeks, months** oder **years.** (Singular wird ebenfalls akzeptiert.)

Korrekte Kommando-Aufrufe sind zum Beispiel:

at 0815am Jan 24
at 8:15am Jan 24
at now + 1 day
at 5 pm Friday

at und *batch* schreiben die Auftragsnummer und -zeit auf die Standardfehlerausgabe.

batch erledigt einen Batch-Auftrag. Dies ist fast gleichwertig mit "at now", jedoch nicht ganz: Der Auftrag wird in eine andere Warteschlange eingereiht; außerdem antwortet "at now" mit der Fehlermeldung **too late.**

at **-r** nimmt Aufträge, die zuvor mit *at* oder *batch* geplant wurden, heraus. Die Auftrags-Nummer ist die Nummer, die Sie vorher von *at oder batch* erhalten haben. Durch Eingeben von *at* **-l** können Sie ebenfalls die Auftrags-Nummer erfahren. Sie können nur Ihre eigenen Aufträge entfernen, es sei denn, Sie sind der Systemverwalter.

BEISPIELE

Die Kommandos *at* und *batch* lesen von der Standardeingabe die zu einem späteren Zeitpunkt auszuführenden Kommandos. *sh*(1) bietet verschiedene Möglichkeiten für die Angabe der Standardeingabe. Innerhalb Ihrer Kommandos kann ein Umlenken der Standardausgabe von Nutzen sein.

Diese Eingabefolge kann an einem Terminal verwendet werden:

batch
sort *filename >outfile*
<control-D> ('control'-Taste und 'D'-Taste gleichzeitig drücken)

Diese Folge, die das Umlenken der Standardfehlerausgabe auf eine Pipe zeigt, kann innerhalb einer Shell-Prozedur nützlich sein (die Reihenfolge der Ausgabe-Umlenkungen ist signifikant):

```
batch <<!
sort filename 2>&1 >outfile | mail loginid
!
```

Damit sich ein Auftrag immer wieder selbst einträgt, müssen Sie *at* gemäß folgendem Muster in der Shell-Prozedur aufrufen:

```
echo "sh shellfile" | at 1900 thursday next week
```

DATEIEN

/usr/lib/cron	Hauptverzeichnis für cron
/usr/lib/cron/at.allow	Liste der zugelassenen Benutzer
/usr/lib/cron/at.deny	Liste der nicht zugelassenen Benutzer
/usr/lib/cron/queue	Planungsdaten
/usr/spool/cron/atjobs	Spool-Bereich

SIEHE AUCH:

kill(1), mail(1), nice(1), ps(1), sh(1), sort(1).
cron(1M) im Administrator's Guide

DIAGNOSE

Angabe von Syntaxfehlern und falschen Uhrzeiten.

BEZEICHNUNG

awk – Programmiersprache zur Analyse und Aufbereitung von Textdateien

ÜBERSICHT

awk [**-F** *re*] [*parameter...*] ['*prog*'] [**-f** *progfile*] [*file...*]

BESCHREIBUNG

Die Option **-F** *re* definiert den regulären Ausdruck *re* (regular expression) als Eingabe-Feldtrennzeichen.

Die Parameter *parameter* können in der Form x=... y=... an *awk* übergeben werden, wobei x und y eingebaute *awk*-Variablen sind (siehe untenstehende Liste).

awk durchsucht alle Eingabedateien *file* nach Zeilen, für die eines der Textmuster zutrifft, die in dem Programm *prog* festgelegt wurden. Das Programm *prog* muß in Hochkommata (´) eingeschlossen werden, damit es von der Shell unverändert weitergegeben wird. Zu jedem Textmuster in dem Programm *prog* kann eine Aktion definiert werden. Diese wird ausgeführt, wenn in einer Zeile der Datei *file* das angegebene Textmuster gefunden wird. Muster-Aktion-Anweisungen (Regeln) können in der Kommandozeile direkt angegeben (*prog*) oder in einer Programmdatei (Option **-f** *progfile*) abgelegt werden.

Die Eingabedateien werden in der angegebenen Reihenfolge abgearbeitet. Wenn die Angabe *file* fehlt oder als Eingabedatei – angegeben wird, dann wird die Standardeingabedatei benutzt. Für jede Eingabezeile wird eine Regel nach der anderen abgearbeitet. Immer wenn ein Textmuster zutrifft, wird die entsprechende Aktion ausgeführt.

Eine Eingabezeile besteht in der Regel aus Feldern, die durch Zwischenraumzeichen voneinander getrennt sind. (Diese Vorgabe kann mit Hilfe der eingebauten Variablen FS oder der Option **-F** *re* geändert werden.) Die einzelnen Felder werden als **$1**, **$2**, ... bezeichnet; die ganze Eingabezeile kann mit **$0** angesprochen werden.

Eine Regel hat die Form:

Textmuster { Aktion }

Eines von beiden kann entfallen. Wenn das Textmuster zutrifft und die Aktion fehlt, wird die Eingabezeile ausgedruckt. Fehlt das Textmuster, wird die Aktion für jede Eingabezeile ausgeführt.

Textmuster bestehen aus relationalen und regulären Ausdrücken, die in beliebiger Weise durch logische Operatoren (!, ||, && und runde Klammern) miteinander verknüpft werden können. Relationale Ausdrücke haben die Form:

Ausdruck rel-op Ausdruck
Ausdruck enthält-op regulärer-Ausdruck

rel-op ist einer der sechs relationalen Operatoren in C, enthält-op ist entweder ~ (enthält) or !~ (enthält nicht). Eine Bedingung ist ein arithmetischer Ausdruck, ein relationaler Ausdruck, der spezielle Ausdruck

Var in Tabelle,

oder eine logische Verknüpfung dieser Ausdrücke.

Um die Steuerung zu übernehmen, kann der Benutzer die speziellen Textmuster BEGIN und END verwenden, da sie vor dem Lesen der ersten bzw. nach dem Lesen der letzten Zeile ausgeführt werden.

Reguläre Ausdrücke sind so aufgebaut wie in *egrep* [siehe *grep*(1)] und müssen innerhalb eines Textmusters in Schrägstriche (/) gesetzt werden. Textmuster, die nur aus einem regulären Ausdruck bestehen, beziehen sich auf die ganze Eingabezeile. Reguläre Ausdrücke dürfen auch in relationalen Ausdrücken vorkommen. Ein Textmuster kann aus zwei durch Komma getrennten Textmustern bestehen. Die zugehörige Aktion wird für die erste Zeile ausgeführt, in der das erste Textmuster zutrifft, und für alle Folgezeilen bis zu der Zeile einschließlich, in der das zweite Textmuster zutrifft.

Ein regulärer Ausdruck kann auch zur Angabe des Feldtrennzeichens benutzt werden. Dies geschieht mit Hilfe der Option **-F** *re* oder durch Zuweisung des Ausdrucks an die eingebaute Variable FS. Standardmäßig werden führende Leerstellen ignoriert und als Feldtrennzeichen werden Leerstellen und/oder Tabulatorzeichen verwendet. Wenn FS hingegen ein Wert zugewiesen wird, werden führende Leerstellen nicht mehr ignoriert.

Zu den eingebauten Variablen gehören auch:

ARGC	Anzahl der Argumente in der Kommandozeile
ARGV	Tabelle mit den Argumenten der Kommandozeile
FILENAME	Name der aktuellen Eingabedatei
FNR	Nummer des aktuellen Satzes in der aktuellen Eingabedatei
FS	regulärer Ausdruck für das Eingabe-Feldtrennzeichen (Standard: Leerstellen)
NF	Anzahl der Felder im aktuellen Satz
NR	Nummer des aktuellen Satzes

OFMT	Ausgabeformat von Zahlen (Standard: %.6g)
OFS	Ausgabe-Feldtrennzeichen (Standard: Leerzeichen)
ORS	Ausgabe-Satztrennzeichen (Standard: New-Line-Zeile-Zeichen)
RS	Eingabe-Satztrennzeichen (Standard: New-Line-Zeichen)

Eine Aktion ist eine Folge von Anweisungen. Die folgenden Anweisungen sind erlaubt:

```
if ( Bedingung ) Anweisung [ else Anweisung ]
while ( Bedingung ) Anweisung
do Anweisung while ( Bedingung )
for ( Ausdruck ; Bedingung ; Ausdruck ) Anweisung
for ( var in Tabelle ) Anweisung
delete Tabelle[index]
break
continue
{ [ Anweisung ] ... }
Ausdruck          # üblicherweise Variable = Ausdruck
print [ Ausdrucksliste ] [ >Ausdruck ]
printf Format [ , Ausdrucksliste ] [ >Ausdruck ]
next              # übrige Textmuster der Eingabezeile ignorieren
exit [Ausdruck]   # Rest der Eingabedatei ignorieren
                  # Endestatus ist gleich ausdruck
return [Ausdruck]
```

Anweisungen werden mit einem Semikolon, einem New-Line-Zeichen oder einer geschweiften Klammer rechts abgeschlossen. Eine leere Ausdrucksliste steht für die gesamte Eingabezeile. Ausdrücke werden je nach Verwendung als Zeichenketten oder als numerische Werte interpretiert. Sie werden mit Hilfe der Operatoren +, −, *, /, % und durch Verkettung (mit Hilfe einer Leerstelle) gebildet. Die C-Operatoren ++, −−, +=, −=, *=, /=, und %= können ebenfalls in Ausdrücken verwendet werden. Variablen können Skalare, Tabellenelemente (Schreibweise: x[i]) oder Felder sein. Variablen werden mit der leeren Zeichenkette oder mit Null initialisiert. Als Tabellenindex ist statt einer Zahl auch eine (leichter merkbare) Zeichenkette erlaubt. Zeichenkettenkonstanten müssen in Anführungszeichen (") gesetzt werden.

Mit der **print**-Anweisung werden die angegebenen Argumente in die Standardausgabedatei ausgegeben, in die durch >*expression* angegebene Datei oder in die durch | *cmd* (Kommando) angegebene Pipe. Die Argumente werden durch das aktuelle Ausgabe-Feldtrennzeichen getrennt und durch das Ausgabe-Satztrennzeichen abgeschlossen. Die **printf**-Anweisung formatiert die angegebene Ausdrucksliste genauso wie die C-Funktion *printf* [siehe *printf*(3S) im *Programmer's Reference Manual*].

awk verfügt über eine Vielzahl eingebauter Funktionen: arithmetische Funktionen, Zeichenkettenfunktionen, Ein-/Ausgabefunktionen und allgemeine Funktionen.

Die arithmetischen Funktionen sind: *atan2*, *cos*, *exp*, *int*, *log*, *rand*, *sin*, *sqrt* und *srand*. *int* liefert den ganzzahligen Teil des Arguments. *rand* liefert eine Zufallszahl zwischen 0 und 1. *srand* (expr) nimmt als Startwert für *rand* den Wert des Ausdrucks *expr* oder, wenn *expr* nicht angegeben ist, die aktuelle Uhrzeit an.

Die Zeichenkettenfunktionen sind:

gsub(*re*, *z1*, *z2*) liefert das gleiche Ergebnis wie *sub* (siehe unten), allerdings werden nacheinander alle Vorkommen des regulären Ausdrucks *re* ersetzt (wie bei dem globalen Ersetzungskommando des *ed*).

index(*z1*, *z2*)) liefert die Position des ersten Vorkommens der Zeichenkette *z2* in der Zeichenkette *z1* oder 0, wenn *z2* nicht in *z1* vorkommt.

length (*z*) liefert die Länge des als Zeichenkette interpretierten Arguments. Fehlt das Argument, wird die Länge der ganzen Zeile zurückgegeben.

match (*z*, *re*) gibt die Position des ersten Vorkommens des regulären Ausdrucks *re* in der Zeichenkette *z* zurück oder 0, wenn *re* nicht in *z* vorkommt. RSTART wird die Startposition zugewiesen (gleich der zurückgegebenen Position) und RLENGTH die Länge der gefundenen Zeichenkette.

split(*z*, *t*, *fs*) zerlegt die Zeichenkette *z* in die Tabellenelemente *t*[*1*], *t*[2], ..., *t*[*n*], und gibt *n* zurück. Bei der Zerlegung wird der reguläre Ausdruck *fs* als Feldtrennzeichen benutzt oder, wenn diese Angabe fehlt, der Inhalt der Variablen FS.

sprintf(*fmt*, *expr*, *expr*, ...) formatiert die angegebenen Ausdrücke entsprechend dem *printf*-Format *fmt* (siehe *printf*(3S)) und liefert die formatierte Zeichenkette zurück.

sub(*re*, *z1*, *z2*) ersetzt das erste Vorkommen des regulären Ausdrucks *re* durch die Zeichenkette *z1* in der Zeichenkette *z2* und gibt die Anzahl der Ersetzungen zurück. Fehlt *z2*, wird die Ersetzung im aktuellen Satz vorgenommen (**$0**).

substr(*z*, *m*, *n*) liefert die aus *n* Zeichen bestehende Teilzeichenkette, die in der Zeichenkette *z* ab der Position *m* beginnt.

Die Ein-/Ausgabefunktionen und die allgemeinen Funktionen sind:

close(*filename*) schließt die Datei oder die Pipe mit dem Namen *filename*.

cmd| *getline* übergibt die Ausgabe des Kommandos *cmd* über eine Pipe an *getline* weiter. Jeder Aufruf von *getline* liefert die jeweils nächste Zeile, die von *cmd* ausgegeben wird.

getline *getline* liefert in **$0** den nächsten Eingabesatz der aktuellen Eingabedatei zurück.

getline <*file* *getline* liefert in **$0** den nächsten Eingabesatz der Datei *file* zurück.

getline var *getline* liefert in der Variablen *var* den nächsten Eingabesatz der aktuellen Eingabedatei zurück.

getline var <*file* *getline* liefert in der Variablen *var* den nächsten Eingabesatz der Datei *file* zurück.

system(*cmd*) führt das Kommando *cmd* aus und gibt seinen Endestatus zurück.

Alle Formen der *getline*–Funktion geben als Endestatus 1 zurück, wenn die Eingabe erfolgreich gelesen werden konnte, 0 bei Dateiende und –1 bei einem Fehler.

awk bietet auch die Möglichkeit zur Definition benutzereigener Funktionen. Solche Funktionen müssen (anstelle des Textmusters in einer Regel) folgendermaßen definiert werden:

```
Funktionsname(Argumente,...) { Anweisungen }
Funk_name(Argumente,...) { Anweisungen }
```

Die Argumente einer Funktion werden bei Skalaren als Wert übergeben, bei Tabellen durch einen Zeiger. Argumentnamen gelten nur lokal innerhalb einer Funktion, alle anderen Variablennamen haben globale Geltung. Funktionsaufrufe können verschachtelt werden. Rekursive Funktionen sind erlaubt. Die **return**–Anweisung kann dazu benutzt werden, um einen Wert zurückzugeben.

BEISPIELE

Drucke alle Zeilen, die länger als 72 Zeichen sind:

```
length > 72
```

Drucke die ersten beiden Felder in umgekehrter Reihenfolge:

```
{ print $2, $1 }
```

Wie oben, aber die Eingabefelder sind durch Komma und/oder Leerstellen und Tabulatorzeichen getrennt:

```
BEGIN { FS = ",[ \t]*|[ \t]+" }
      { print $2, $1 }
```

Addiere die erste Spalte (Zahlen), drucke Summe und Durchschnitt:

```
        { s += $1 }
END     { print "Summe: ", s, " Durchschnitt: ", s/NR }
```

Drucke die Felder in umgekehrter Reihenfolge:

```
{ for (i = NF; i > 0; --i) printf("%s ",$i) }
```

Drucke alle Zeilen zwischen start/stop-Zeilenpaaren:

```
/start/, /stop/
```

Drucke alle Zeilen, deren erstes Feld sich vom ersten Feld der vorherigen Zeile unterscheidet:

```
$1 != f { print; f = $1 }
```

Simulation des *echo(1)*-Kommandos:

```
BEGIN {
      for (i = 1; i < ARGC; i++)
            printf "%s", ARGV[i]
      printf "\n"
      exit
      }
```

Drucke eine Datei mit Seitennummern, beginnend bei Seite 5:

```
/Seite/ { $2 = n++; }
        { print }
```

Kommandozeile: **awk -f programm n=5 eingabe**

SIEHE AUCH

grep(1), sed(1).
lex(1), printf(3S) im *Programmer's Reference Manual.*
Programmer's Guide.

FEHLER

Zwischenraumzeichen in der Eingabe bleiben in der Ausgabe nicht erhalten, wenn Felder in die Ausgabe mit einfließen.
Es gibt keine ausdrückliche Möglichkeit zur Umwandlung von Zahlen in Zeichenketten und umgekehrt. Umwandlungen können aber dennoch erzwungen werden: Durch Addition von 0 wird ein Ausdruck zu einer Zahl, durch Verkettung mit der leeren Zeichenkette ("") zu einer Zeichenkette.

BEZEICHNUNG

banner – Großschreiben

ÜBERSICHT

banner strings

BESCHREIBUNG

banner gibt seine Argumente (jedes mit maximal 10 Zeichen) in Großbuchstaben auf die Standardausgabe aus.

SIEHE AUCH:

echo(1).

BEZEICHNUNG

basename, dirname – Teile von Pfadnamen liefern

ÜBERSICHT

basename string [suffix]
dirname string

BESCHREIBUNG

basename entfernt jeden Präfix, der mit / endet, löscht die Endung *suffix* (falls vorhanden in *string*) in *string* und schreibt das Ergebnis auf die Standardausgabe. Es wird normalerweise in Shell-Prozeduren in die Ersetzungszeichen (` `) gesetzt.

dirname liefert alles von dem Pfadnamen *string* außer der letzten Komponente.

BEISPIELE

Das folgende Beispiel, das mit dem Argument **/usr/src/cmd/cat.c** aufgerufen wird, übersetzt die angegebene Datei und schreibt die Ausgabe in eine Datei **cat** im aktuellen Verzeichnis:

```
cc $1
mv a.out `basename $1 .\c.`
```

Das folgende Beispiel wird die Shell-Variable NAME auf **/usr/src/cmd** setzen:

```
NAME=`dirname /usr/src/cmd/cat.c`
```

SIEHE AUCH:

sh(1).

BEZEICHNUNG

bc – Sprache für Arithmetik mit beliebiger Genauigkeit

ÜBERSICHT

bc [**-c**] [**-l**] [file ...]

BESCHREIBUNG

bc ist ein interaktiver Prozessor für eine Sprache, die C ähnelt, aber Arithmetik mit beliebiger Genauigkeit bietet. *bc* liest seine Eingabe zunächst aus jeder der eingegebenen Dateien *(file)*, dann von der Standardeingabe. Das Dienstprogramm *bc*(1) ist tatsächlich ein Preprozessor für *dc*(1), das automatisch von *bc* aufgerufen wird, es sei denn, die Option **-c** ist vorhanden. In diesem Fall wird die *dc* Eingabe statt an *dc* auf die Standardausgabe geschickt. Die Optionen lauten wie folgt:

-c Nur übersetzen. Die Ausgabe wird zur Standardausgabe geschickt.

-l Argument steht für den Namen einer Bibliothek für Arithmetik mit beliebiger Genauigkeit.

Die Syntax für *bc* Programme lautet: L bedeutet Buchstaben a-z, E bedeutet Ausdruck, S bedeutet Anweisung.

Kommentare
werden in /* und */ eingeschlossen .

Namen
einfache Variablen: L
Feldelemente: L [E]
Die Worte "ibase", "obase", und "scale"

Andere Operanden
beliebig lange Zahlen wahlweise mit Vorzeichen und Komma.
(E)
sqrt (E)
length (E) Anzahl signifikanter Dezimalstellen
scale (E) Anzahl der Stellen rechts vom Komma
L (E, ... , E)

Operatoren
+ - * / % ^ (% Rest bei Division; ^ Potenz)
+ + - - (Prefix und Postfix; gilt für Namen)
= = < = > = ! = < >
= = + = - = * = / = % = ^

Anweisungen

```
E
{ S ; ... ; S }
if ( E ) S
while ( E ) S
for ( E ; E ; E ) S
leere Anweisung
break
quit
```

Funktionsdefinitionen

```
define L ( L ,..., L ) {
 auto L, ... , L
 S; ... S
 return ( E )
}
```

Funktionen in -l math-Bibliothek

s(x)	Sinus
c (x)	Kosinus
e(x)	Exponentialfunktion
l(x)	Logarithmus
a(x)	Arkustangens
j(n,x)	Bessel-Funktion

Alle Funktionsargumente werden per Wertübergabe übergeben.

Der Wert einer Anweisung, die ein Ausdruck ist, wird ausgegeben, es sei denn der Hauptoperator ist eine Zuweisung. Semikolons oder New-Line-Zeichen können Anweisungen trennen. Eine Zuweisung an *scale* beeinflußt wie bei *dc*(1) die Anzahl der Stellen, die bei Arithmetik-Operationen berücksichtigt werden. Zuweisungen an *ibase* oder *obase* setzen die jeweilige Basis für Eingabe- bzw. Ausgabe-Zahlen.

Der gleiche Buchstabe kann als Feld, Funktion und einfache Variable gleichzeitig verwendet werden. Alle Variablen sind global im Programm. "Auto"-Variablen werden während Funktionsaufrufen auf den Keller geschrieben. Wenn Felder als Funktionsargumente verwendet werden oder wenn sie als automatische Variable definiert sind, müssen leere eckige Klammern hinter dem Feldnamen stehen.

BEISPIEL

```
scale = 20
define e(x){
        auto a, b, c, i, s
        a = 1
        b = 1
        s = 1
        for(i=1; 1==1; i++){
                a = a*x
                b = b*i
                c = a/b
                if(c == o) return(s)
                s = s+c
        }
}
```

definiert eine Funktion, mit der annähernd die Werte der Exponentialfunktion berechnet werden können, und

```
for(i=1; i<=10 ; i++) e(i)
```

gibt die annähernden Werte der Exponentialfunktion für die ersten zehn ganzen Zahlen aus.

DATEIEN

/usr/lib/lib.b Mathematische Bibliothek
/usr/bin/dc Tischrechner

SIEHE AUCH

dc(1).

FEHLER

Das Kommando *bc* kennt noch nicht die Booleschen Operatoren, && und ||.
In der *for* Anweisung müssen alle drei Ausdrücke (E) vorkommen.
Quit wird beim Lesen und nicht beim Ausführen interpretiert.

BEZEICHNUNG

bdiff – big diff

ÜBERSICHT

bdiff file1 file2 [n] [–s]

BESCHREIBUNG

bdiff wird ähnlich wie *diff(1)* verwendet. Es soll herausfinden, welche Zeilen in zwei Dateien geändert werden müssen, um die Dateien identisch zu machen. Diese Funktion dient zur Verarbeitung von Dateien, die zu groß für *diff* sind.

Die Argumente für *bdiff* lauten:

file1 (file2)
: Der Name der zu vergleichenden Dateien. Wenn *file1 (file2)* – ist, wird die Standardeingabe gelesen.

n
: Die Anzahl der Zeilen pro Segment. Der Wert von *n* ist mit 3500 voreingestellt. Wenn das wahlweise dritte Argument angegeben und numerisch ist, wird es als Wert für *n* verwendet. Dies ist nützlich, wenn Segmente von 3500 Zeilen für *diff* zu groß sind, und *diff* deswegen nicht funktioniert.

–s
: gibt an, daß keine Meldungen von *bdiff* ausgegeben werden sollen. Es ist jedoch zu beachten, daß hierdurch eventuelle Meldungen von *diff*(1), das von *bdiff* aufgerufen wird, nicht unterdrückt werden.

bdiff ignoriert identische Zeilen am Anfang beider Dateien, teilt den Rest jeder Datei in Segmente mit je *n* Zeilen auf und ruft *diff* für entsprechende Segmente auf. Wenn beide wahlweisen Optionen angegeben sind, müssen sie in der oben angegebenen Reihenfolge erscheinen.

Die Ausgabe von *bdiff* ist identisch zu der von *diff,* wobei die Zeilennummern angepaßt werden, um die Segmentierung der Dateien zu berücksichtigen (d. h. es sieht so aus, als wären die Dateien als Ganzes verarbeitet worden). Es ist zu beachten, daß wegen des Segmentierens der Dateien, *bdiff* nicht unbedingt eine kleinste ausreichende Menge von Datei-Unterschieden findet.

DATEIEN

/tmp/bd?????

SIEHE AUCH

diff(1), help(1).

DIAGNOSE

help(1) für Erklärungen benutzen.

BEZEICHNUNG

bfs – durchsuchen von großen Dateien

ÜBERSICHT

bfs [–] name

BESCHREIBUNG

Das *bfs* Kommando verhält sich fast wie *ed*(1). Dateien werden jedoch nur gelesen und es können viel größere Dateien mit dieser Funktion verarbeitet werden. Dateien bis max. 1024K Bytes und 32K Zeilen werden verarbeitet. Sie können pro Zeile max. 512 Zeichen, einschließlich New-Line-Zeichen haben (255 Zeichen für 16-Bit-Rechner). *bfs* ist normalerweise für das Durchsuchen einer Datei effektiver als *ed*(1), da die Datei nicht in einen Puffer kopiert wird. Diese Funktion ist besonders gut geeignet zur Identifizierung von Abschnitten einer großen Datei, wobei *csplit*(1) verwendet werden kann, um die Datei in mehrere Einzelteile aufzuteilen, die leichter zu editieren sind.

Normalerweise wird die Größe der Datei, die durchsucht wird, ausgegeben wie bei jeder Datei, die mit dem **w** Kommando geschrieben wurde. Wahlweise kann – benutzt werden, um die Angabe der Dateigröße zu unterdrücken. Genau wie bei *ed(1)* wird die Eingabe mit einem * aufgefordert, wenn **P** und Wagenrücklauf eingegeben werden. Die Eingabe-Aufforderung kann durch erneute Eingabe von **P** und Wagenrücklauf wieder ausgeschaltet werden. Es ist zu beachten, daß bei eingeschalteter Eingabe-Aufforderung, Meldungen als Antwort auf Fehler ausgegeben werden.

Alle in *ed*(1) beschriebenen Adreßausdrücke werden unterstützt. Zusätzlich können reguläre Ausdrücke in zwei Symbolen außer / und ? eingeschlossen werden. > bedeutet nicht-zirkuläres Vorwärts-Suchen und < nicht-zirkuläres Rückwärts-Suchen. Ein geringer Unterschied besteht bei Marken-Namen: Nur die Buchstaben **a** bis **z** können verwendet werden, und alle 26 Marken werden gemerkt.

Die **e, g, v, k, p, q, w, =, !** und Nullkommandos funktionieren wie bei *ed(1)* beschrieben. Kommandos wie zum Beispiel **---**, **+++-**, **+++=**, **-12**, und **+4p** werden akzeptiert. Es ist zu beachten, daß **1,10p** und **1,10** beide die ersten zehn Zeilen ausgeben. Das **f** Kommando gibt nur den Namen der Datei, die durchsucht wird, aus; es gibt keinen *remembered* Dateinamen. Das **w** Kommando ist unabhängig von Ausgabe-Umlenkung, Abschneiden, oder Komprimieren (siehe nachstehende **xo, xt** und **xc** Kommandos). Folgende zusätzliche Kommandos stehen zur Verfügung:

xf *file*

Weitere Kommandos werden aus der angegebenen Datei *file* entnommen. Wenn Dateiende erreicht wird oder ein Unterbrechungssignal eintrifft bzw. ein Fehler auftritt, wird die Datei, die das **xf** enthält, weiter gelesen. Die **xf** Kommandos können bis zu einer Tiefe von 10 geschachtelt werden.

xn gibt momentan benutzte Marken an (Marken werden mit dem Kommando **k** gesetzt).

xo [*file*]

Weitere Ausgabe von **p** und Nullkommandos wird in die angegebene Datei *file* umgelenkt, die falls nötig mit Modus 666 erstellt wird, (lesbar und schreibbar für alle), sofern Ihre Einstellung für *umask* dies nicht anders bestimmt (vgl. *umask*(1)). Fehlt *file*, wird die Ausgabe auf die Standardausgabe umgelenkt. Es ist zu beachten, daß jede Umlenkung Zurücksetzen auf Länge 0 Wirkung oder Neuerstellung einer Datei verursacht.

: *label*

Dies setzt eine Marke *label* in eine Kommandodatei. *label* wird mit neuer Zeile abgeschlossen, und Leerzeichen zwischen **:** und dem Anfang von *label* werden ignoriert. Dieses Kommando kann auch verwendet werden, um Kommentare in eine Kommandodatei einzufügen, da man *label* nicht anspringen muß.

(.,.)xb/*regular expression*/*label*

Ein Sprung (entweder rückwärts oder vorwärts) wird zu *label* gemacht, wenn das Kommando erfolgreich ist. Es versagt bei folgenden Bedingungen:

1. Eine der Adressen ist nicht zwischen 1 und $.
2. Die zweite Adresse ist kleiner als die erste.
3. Der reguläre Ausdruck paßt zu keiner Zeile im angegebenen Bereich, einschließlich erster und letzter Zeile.

Bei Erfolg wird . auf die Zeile gesetzt, die gepaßt hat, und nach *label* gesprungen. Dieses Kommando gibt als einziges Kommando keine Fehlermeldung bei ungültigen Adressen aus, und kann verwendet werden, um zu prüfen, ob Adressen ungültig sind, ehe andere Kommandos ausgeführt werden. Es ist zu beachten, daß das Kommando

```
xb/^/ label
```

ein unbedingter Sprung ist.

Das **xb** Kommando ist nur erlaubt, wenn es nicht vom Terminal eingelesen wird. Wenn es aus einer Pipe eingelesen wird, ist nur ein Vorwärts-Sprung möglich.

xt *number*

Die Ausgabe von **p-** und Nullkommandos ist auf *number* Zeichen begrenzt. Die voreingestellte Zahl ist 255.

xv[*digit*] [*spaces*] [*value*]

Der Variablenname ist das angegebene *digit* nach **xv**. Die Kommandos **xv5100** oder **xv5 100** weisen beide den Wert **100** der Variablen **5** zu. Das Kommando **xv61,100p** weist den Wert **1,100p** der Variablen **6** zu. Der Wert einer Variablen wird ermittelt, indem man **%** vor den Variablennamen schreibt. Unter Voraussetzung der obigen Zuweisungen an die Variablen **5** und **6** werden durch

```
1,%5p
1,%5
%6
```

die ersten 100 Zeilen ausgegeben.

```
g/%5/p
```

würde global die Zeichen **100** suchen und gibt jede Zeile aus, in der sie vorkommen. Um die besonderen Bedeutung von **%** aufzuheben, muß ein \ vorausgehen.

```
g/".*\%[cds]/p
```

könnte verwendet werden, um Zeilen zu suchen und auszugeben, die *printf* -Anweisungen für Zeichen, ganzen Dezimalzahlen oder Zeichenfolgen enthalten.

Ein weiteres Leistungsmerkmal des **xv** Kommandos ist die Möglichkeit, die erste Ausgabezeile eines UNIX Kommandos in eine Variable zu speichern. Die einzige Bedingung ist, daß das erste Zeichen von *value* ein ! sein muß. Zum Beispiel würde:

```
.w junk
xv5!cat junk
!rm junk
!echo "%5"
xv6!expr %6 + 1
```

die aktuelle Zeile in die Variable **5** schreiben, ausgeben und die Variable **6** um 1 erhöhen. Um die besondere Bedeutung von ! als erstes Zeichen von *value* aufzuheben, muß man \ voranstellen.

```
xv7\!date
```

speichert den Wert **!date** in Variable **7**.

xbz *label*
xbn *label*

Diese beiden Kommandos prüfen den letzten abgespeicherten *return code* eines UNIX Kommandos (*!command*) bzw. Nicht-Null-Wert jeweils beide bis zur angegebenen Marke. Die beiden folgenden Beispiele durchsuchen die nächsten fünf Zeilen nach der Zeichenfolge **size**.

```
xv55
: l
/size/
xv5!expr %5 - 1
!if 0%5 != 0 exit 2
xbn l
xv45
: l
/size/
xv4!expr %4 - 1
!if 0%4 = 0 exit 2
xbz l
```

xc [*switch*]

Wenn *switch* **1** ist, wird die Ausgabe von **p** und Nullkommandos komprimiert; wenn *switch* **0** ist, geschieht dies nicht. Ohne Argument kehrt **xc** *switch* um. Zunächst wird *switch* auf Nicht-Komprimierung gesetzt. Bei komprimierter Ausgabe werden Tabulatorzeichenfolgen und Leerzeichen auf ein Leerzeichen reduziert und leere Zeilen entfallen.

SIEHE AUCH

csplit(1), ed(1), umask(1).

DIAGNOSE

? für Fehler in Kommandos, wenn die Eingabe-Aufforderung ausgeschaltet ist. Selbsterklärende Fehlermeldungen bei eingeschalteter Eingabe-Aufforderung.

BEZEICHNUNG

cal – Kalender ausgeben

ÜBERSICHT

cal [[month] year]

BESCHREIBUNG

cal gibt einen Kalender aus für das angegebene Jahr. Wenn auch ein Monat angegeben ist, wird ein Kalender für diesen Monat ausgegeben. Wenn weder Jahr noch Monat angegeben sind, wird ein Kalender für den aktuellen Monat ausgegeben. *year* kann zwischen 1 und 9999 liegen. *month* ist eine Zahl zwischen 1 und 12. Der ausgegebene Kalender gilt für England und die Vereinigten Staaten.

BEISPIELE

Ein ungewöhnlicher Kalender wird für September 1752 ausgegeben. Das ist der Monat, in dem 11 Tage übersprungen wurden, um Schaltjahrvariationen einzubeziehen. Zur Ausgabe dieses Kalenders geben Sie: **cal 9 1752** ein.

FEHLER

Es ist sinnvoll, das Jahr immer mit Januar zu beginnen, obwohl dies den historischen Tatsachen nicht ganz entspricht.

Beachten Sie, daß sich "cal 83" auf die frühchristliche Epoche bezieht, und nicht auf das 20. Jahrhundert.

BEZEICHNUNG

calendar – Terminkalender

ÜBERSICHT

calendar [–]

BESCHREIBUNG

calendar liest die Datei **calendar** im aktuellen Verzeichnis und druckt Zeilen, die irgendwo auf der Zeile das heutige oder morgige Datum enthalten. Die meisten annehmbaren Daten wie zum Beispiel "Aug. 24," "august 24," "8/24," usw. werden erkannt, aber nicht "24 August" oder "24/8". Bei Wochenenden gilt "tomorrow" bis Montag.

Wenn ein Argument vorliegt, sucht *calendar* in den Login-Verzeichnissen aller Benutzer nach einer Datei **calendar** und schickt jedes positive Ergebnis per *mail(1)* an die betreffenden Benutzer. Normalerweise wird dies täglich vom UNIX -Betriebssystem durchgeführt.

DATEIEN

/usr/lib/calprog	zur Feststellung des heutigen und morgigen Datums
/etc/passwd	
/tmp/cal*	

SIEHE AUCH

mail(1).

FEHLER

Der Terminkalender-Service muß für Sie wichtige, jedoch allgemein zugängliche Daten Ihres Kalenders zur Verfügung stellen.
calendar's Erweiterung auf "tomorrow" gilt nicht für Feiertage.

BEZEICHNUNG

cd – aktuelles Verzeichnis ändern

ÜBERSICHT

cd [directory]

BESCHREIBUNG

Wenn *directory* nicht angegeben ist, wird der Wert der Shell-Variablen **$HOME** als neues aktuelles Verzeichnis verwendet. Wenn *directory* einen kompletten Pfad angibt, der mit /, ., .., beginnt, wird *directory* das neue aktuelle Verzeichnis. Ansonsten versucht *cd,* das gewünschte Verzeichnis relativ zu einem der Pfade zu finden, die in **$CDPATH** angegeben sind. **$CDPATH** hat die gleiche Syntax und eine ähnliche Semantik wie **$PATH.** *cd* muß Ausführ-(Durchsuchungs)-Erlaubnis in *directory* besitzen.

Da zur Ausführung eines Kommandos ein neuer Prozeß erzeugt wird, wäre *cd* wirkungslos, wenn es als normales Kommando geschrieben wäre; daher wird es von der Shell direkt interpretiert.

SIEHE AUCH

pwd(1), sh(1).
chdir(2) im *Programmer's Reference Manual*

BEZEICHNUNG

chmod – Modus ändern

ÜBERSICHT

chmod mode file ...

chmod mode directory ...

BESCHREIBUNG

Die Berechtigungen der angegebenen Dateien *files* oder der Verzeichnisse *directories* werden entsprechend **mode** (Modus) geändert, der symbolisch oder absolut sein kann. Absolute Änderungen von Berechtigungen werden als Oktalzahlen angegeben:

chmod *nnn file(s)*

wobei *n* eine Zahl von 0 bis 7 ist. Symbolische Änderungen werden in Form von mnemotechnischen Zeichen angegeben:

chmod *a operator b file(s)*

wobei *a* für ein oder mehrere Zeichen steht, die **user** (Benutzer), **group** (Gruppe) oder **other** (Andere) angeben und *operator* +, −, oder = ist und die Erteilung oder Verweigerung der Berechtigung angibt. *b* steht für ein oder mehrere Zeichen, die die Art der Berechtigung angeben.

Ein absoluter Modus wird als eine Oktalzahl angegeben, die durch ODER-Verknüpfung der folgenden Modi konstruiert wird:

4000	Setzen des s-Bit für Eigentümer bei der Ausführung
20#0	Setzen des s-Bit für Gruppe bei der Ausführung, wenn **# 7, 5, 3,** oder **1** ist Freigabe einer obligatorischen Sperre, wenn **# 6, 4, 2,** oder **0** ist
1000	t-Bit wird eingeschaltet (siehe *chmod*(2))
0400	Leserecht für Eigentümer
0200	Schreibrecht für Eigentümer
0100	Ausführberechtigung für Eigentümer (bei Verzeichnis: Berechtigung zum Durchsuchen)
0070	Lesen, Schreiben, Ausführen (Durchsuchen) für Gruppe
0007	Lesen, Schreiben, Ausführen (Durchsuchen) für Andere

Symbolische Änderungen werden mit Buchstaben angegeben. Diese entsprechen jeweils den Zugriffsklassen und den individuellen Berechtigungen. Ihre Zugriffsberechtigungen auf eine Datei hängen von Ihrer Benutzernummer (UID) oder Gruppennummr (GID) ab. Zugriffsberechtigungen werden in drei Gruppen mit je drei Zeichen beschrieben:

Eigentümer	Gruppe	Andere
rwx	rwx	rwx

Dieses Beispiel (user, group, und others haben alle reading (Lese-), writing (Schreib-) und execution (Ausführ)-Berechtigung für eine bestimmte Datei) zeigt zwei Kategorien für die Erteilung von Zugriffsrechten: die Zugriffsklasse und die Berechtigungen.

Um die Zugriffsrechte auf eine Datei (oder ein Verzeichnis) mit der symbolischen Methode zu ändern, benutzen Sie die folgende Syntax:

[*who*] *operator* [*permission(s)*], ...

Ein Kommandoaufruf mit der symbolischen Methode lautet z.B. wie folgt:

chmod g+rw *file*

Dieses Kommando würde der Gruppe das Lesen und Schreiben von *file* erlauben.

Für *who* kann einer oder mehrere der folgenden Buchstaben angegeben werden :

u	user's permissions (Eigentümer-Berechtigungen)
g	group's permissions (Gruppen-Berechtigungen)
o	others permissions (Andere-Berechtigungen)

Der Buchstabe **a** (all) ist gleichwertig mit **ugo** und ist voreingestellt, wenn *who* ausgelassen wird.

Operator kann **+** sein, wodurch die *permission* (Berechtigung) hinzugefügt wird – wodurch die *permission* entzogen wird, oder **=**, wodurch die Berechtigung *permission* absolut gesetzt wird. (Im Gegensatz zu den anderen symbolischen Operatoren hat **=** eine absolute Wirkung, indem es alle anderen Bits zurücksetzt.) Ein Auslassen von *permission* ist nur sinnvoll im Zusammenhang mit **=**, wenn alle Berechtigungen entzogen werden.

Permission ist jede kompatible Kombination der folgenden Buchstaben:

r	reading permission (Lese-Berechtigung)
w	writing permission (Schreib-Berechtigung)
x	execution permission (Ausführ-Berechtigung)
s	s-Bit für Eigentümer oder Gruppe wird eingeschaltet
t	t-Bit wird eingeschaltet
l	obligatorisches locking (Sperren) während des Zugriffs

Mehrere durch Kommata getrennte Angaben können angegeben werden. Dabei dürfen keine Leerzeichen zwischen den Angaben auftreten.

Operationen werden in der angegebenen Reihenfolge ausgeführt. Wenn auf einen einzelnen Operator mehrere symbolische Buchstaben folgen, werden die entsprechenden Operationen gleichzeitig ausgeführt. Der Buchstabe **s** hat nur Bedeutung in Zusammenhang mit **u** oder **g** und **t** nur mit **u** .

Das obligatorische Sperren (**l**) von Dateien oder Sätzen beruht auf der Möglichkeit, die Lese- und Schreibrechte einer Datei zu sperren, während ein Programm auf diese Datei zugreift. Es ist nicht möglich, einer Gruppe Ausführungsberechtigung zu erteilen und gleichzeitig eine Datei bei Ausführung zu sperren. Es ist ebenfalls nicht möglich, das s-Bit einzuschalten und gleichzeitig eine Datei-Ausführungssperre zu aktivieren.
Folgende Beispiele

chmod g+x,+l *file*

chmod g+s,+l *file*

sind ungültig und bewirken die Ausgabe von Fehlermeldungen.

Nur der Eigentümer einer Datei oder eines Verzeichnisses (oder der Systemveralter) kann einen Dateimodus ändern. Der Systemverwalter allein darf t-Bits setzen. Wenn Sie das s-Bit für Gruppe setzen wollen, muß Ihre eigene Gruppennummer mit der Gruppennummer der Datei übereinstimmen und die Gruppe muß Ausführungsberechtigung haben.

BEISPIELE

chmod a-x *file*

chmod 444 *file*

Die ersten Beispiele verweigern Ausführungsberechtigung für alle. Das absolute (oktale) Beispiel erlaubt nur Leseberechtigung.

chmod go+rw *file*

chmod 666 *file*

In diesen Beispielen darf *file* von der Gruppe (group) und von Anderen (others) gelesen und geschrieben werden.

chmod +l *file*

Dies verursacht das Sperren von *file* während des Zugriffs.

chmod =rwx,g+s *file*

chmod 2777 *file*

Die beiden letzten Beispiele erlauben allen das Schreiben, Lesen und Ausführen der Datei; und sie schalten das s-Bit ein.

HINWEISE

Unter RFS (Remote File Sharing) haben Sie die Berechtigungen nicht immer, obwohl die Ausgabe des **ls -l** Kommandos Ihnen dies vortäuscht.

SIEHE AUCH

ls(1).
chmod(2) im *Programmer's Reference Manual.*

BEZEICHNUNG

chown, chgrp – Eigentümer oder Gruppe ändern

ÜBERSICHT

chown owner file ...

chown owner directory....

chgrp group file ...

chgrp group directory ...

BESCHREIBUNG

chown ändert den Eigentümer von *files* oder *directories* zu *owner.* Der Eigentümer kann entweder eine dezimale Benutzernummer oder ein in der Paßwortdatei gefundener Login-Name sein.

Chgrp ändert die Gruppennummer von *files* oder *directories* zu *group.* Die Gruppe kann entweder eine dezimale Gruppennummer oder ein in der Gruppendatei gefundener Gruppenname sein.

Wenn ein Benutzer, der nicht Systemverwalter ist, das Kommando aufruft, werden s-Bits für Eigentümer 04000 und für Gruppe 02000 gelöscht.

Nur der Eigentümer einer Datei (oder Systemverwalter) kann den Eigentümer oder die Gruppe dieser Datei ändern

DATEIEN

/etc/passwd
/etc/group

HINWEISE

Unter RFS (Remote File Sharing) haben Sie die Berechtigungen nicht immer, obwohl die Ausgabe des **ls -l** Kommandos Ihnen dies vortäuscht.

SIEHE AUCH

chmod(1).
chown(2), group(4), passwd(4) im *Programmer's Reference Manual.*

BEZEICHNUNG

cmp – zwei Dateien vergleichen

ÜBERSICHT

cmp [-l] [-s] file1 file2

BESCHREIBUNG

Die beiden Dateien werden verglichen. (Wenn *file1* – ist, wird die Standardeingabe verwendet.) Mit den voreingestellten Optionen gibt *cmp* nichts aus, wenn die Dateien gleich sind; falls sie sich unterscheiden, wird das Byte und die Zeilennummer angezeigt, wo der Unterschied auftritt.

Optionen:

-l Ausgabe der Byte-Nummer (dezimal) und der unterschiedlichen Bytes (oktal) für jeden Unterschied.

-s Keine Ausgabe für unterschiedliche Dateien; nur ersichtlich in dem Rückkehrcode.

SIEHE AUCH

comm(1), diff(1).

DIAGNOSE

Rückkehrwert 0 bei identischen Dateien, 1 bei unterschiedlichen Dateien, und 2 bei einem nicht zugreifbaren oder fehlenden Argument.

BEZEICHNUNG

col – Umkehrzeilenvorschübe filtern

ÜBERSICHT

col [-b] [-f] [-x] [-p]

BESCHREIBUNG

col liest die Standardeingabe und schreibt auf die Standardausgabe. Es führt die Überlagerung von Zeilen durch, die sich aus Zeilenvorschüben rückwärts (ASCII-Code **ESC-7**), und halben Zeilenvorschüben vorwärts und rückwärts (**ESC-9** und **ESC-8**) ergeben. *col* ist besonders nützlich zum Filtern von mehrspaltigen Ausgaben, die vom *nroff*- Kommando **.rt** erzeugt werden, und von Ausgaben, die ein Ergebnis des *tbl*(1) Preprozessors sind.

Wenn die Option **-b** angegeben ist, nimmt *col* an, daß das benutzte Ausgabegerät zu Backspaces nicht fähig ist. In diesem Fall wird, wenn zwei oder mehrere Zeichen an der gleichen Position erscheinen sollen, nur das zuletzt gelesene ausgegeben.

Obwohl *col* halbzeilige Bewegungen in seiner Eingabe akzeptiert, gibt es sie normalerweise nicht aus. Statt dessen wird der Text, der zwischen den Zeilen erscheinen würde, auf die nächste ganze Zeile weiter unten gesetzt. Diese Behandlung kann durch die Option **-f** (genau) unterdrückt werden; in diesem Fall kann die Ausgabe von *col* halbe Zeilenvorschübe vorwärts (**ESC-9**) enthalten, jedoch nie irgendeine Art von rückwärtigen Zeilenbewegungen.

Wenn die **-x** Option nicht gegeben ist, wandelt *col* in der Ausgabe Zwischenräume zu Tabulatorzeichen um, wodurch die Druckzeit verkürzt wird.

Die ASCII Kontrollzeichen **SO** (\017) und **SI** (\016) werden von *col* als Anfang und Ende eines Textes in einem anderen Zeichensatz angenommen. Der Zeichensatz, zu dem jedes Eingabezeichen gehört, wird gemerkt und bei Ausgabe werden die Zeichen **SI** und **SO** erforderlichenfalls erzeugt. Dadurch wird sichergestellt, daß jedes Zeichen im richtigen Zeichensatz gedruckt wird.

In der Eingabe werden nur Leerzeichen, Backspaces, Tabulator, Return, neue Zeile, **SI**, **SO**, **VT** (\013), und **ESC** gefolgt von **7**, **8** oder **9** als Kontrollzeichen akzeptiert. Das **VT**-Zeichen ist eine alternative Form des vollen Rückwärts-Zeilenvorschubs, der aus Gründen der Kompatibilität mit einigen früheren Programmen dieses Typs eingebunden wurde. Alle anderen nicht abdruckbaren Zeichen werden ignoriert.

Normalerweise ignoriert *col* alle unbekannten Umschaltfolgen, die in der Eingabe gefunden werden; die Option **-p** kann die Ausgabe dieser Sequenzen durch *col* als reguläre Zeichen bewirken, dies hängt jedoch von der Möglichkeit eines Überdrucken durch Rückwärtsbewegung ab. Der Gebrauch dieser Option wird nur empfohlen, wenn der Benutzer mit der Textposition der ESC-Sequenzen völlig vertraut ist.

SIEHE AUCH:

nroff(1), tbl(1) im *Dokumentations-Tools*

HINWEISE

Das von *col* akzeptierte Eingabeformat ist identisch mit der von *nroff* mit den Optionen **-T37** oder **-Tlp** produzierten Ausgabe. Verwenden Sie **-T37** (und die **-f** Option von *col),* wenn die allerletzte Ausgabe von *col* auf ein Gerät erfolgt, das halbzeilige Bewegungen und **-Tlp** sonst interpretieren kann.

FEHLER

Kann max. 128 Zeilen rücksetzen.

Max. 800 Zeichen, einschließlich Backspaces, auf einer Zeile möglich.

Lokale senkrechte Bewegungen, die beim Rücksetzen über die erste Zeile des Dokuments gehen, werden ignoriert. Aus diesem Grunde darf in der ersten Zeile kein hochgesetzter Index vorkommen

BEZEICHNUNG

comm – auswählen oder ablehnen von Zeilen, die in zwei sortierten Dateien gleich sind

ÜBERSICHT

comm [– [**123**]] file1 file2

BESCHREIBUNG

comm liest *file1* und *file2*, die in ASCII-Ordnung (siehe *sort*(1)) sortiert sein müssen, und gibt eine dreispaltige Ausgabe aus:
Zeilen nur von *file1*;
Zeilen nur von *file2*;
und Zeilen der beide Dateien.

Der Dateiname – bedeutet die Standardeingabe.

Option 1, 2 oder 3 unterdrückt die entsprechende Spalte. Folglich druckt **comm –12** nur die Zeilen, die in beiden Dateien vorkommen; **comm –23** druckt nur Zeilen aus der ersten Datei und nicht aus der zweiten; **comm –123** druckt nichts.

SIEHE AUCH

cmp(1), diff(1), sort(1), uniq(1).

BEZEICHNUNG

cp, ln, mv – Dateien kopieren, Verweise einrichten, Dateien umbenennen

ÜBERSICHT

cp file1 [file2 ...] target
ln [**-f**] file1 [file2 ...] target
mv [**-f**] file1 [file2 ...] target

BESCHREIBUNG

file1 wird auf *target* (Zieldatei) kopiert. Auf keinen Fall dürfen *file1* und *target* identisch sein (Vorsicht beim Gebrauch von *sh(1)* Metazeichen). Wenn *target* ein Verzeichnis ist, werden eine oder mehrere Dateien zu diesem Verzeichnis kopiert. Wenn *target* eine Datei ist, wird ihr Inhalt gelöscht.

Wenn *mv* oder *ln* feststellen, daß der Modus von *target* Schreiben nicht zuläßt, wird der Modus angegeben (siehe *chmod*(2)), eine Antwort erwartet und eine Zeile der Standardeingabe gelesen; wenn die Zeile mit y beginnt, erscheint falls zulässig das Kommando *mv* oder *ln* ; andernfalls wird das Kommando beendet. Wenn die **-f** Option verwendet wird oder die Standardeingabe kein Terminal ist, werden keine Fragen gestellt und das Kommando *mv* oder *ln* ausgeführt.

Nur bei *mv* ist es möglich, daß *file1* ein Verzeichnis ist. In diesem Fall wird das Verzeichnis nur umbenannt, wenn beide Verzeichnisse den gleichen Vater haben; *file1* wird zu *target* umbenannt. Wenn *file1* eine Datei und *target* ein Verweis auf eine andere Datei mit Verweisen ist, bleiben die anderen Verweise erhalten und *target* wird eine neue Datei.

Ist *target* keine Datei, wird bei der Benutzung von *cp* eine neue Datei erstellt, die den gleichen Modus wie *file1* hat, wobei das t-Bit nicht gesetzt ist, sofern Sie nicht die Systemverwalter sind; der Eigentümer und die Gruppe von *target* sind die des Benutzers. Wenn *target* eine Datei ist, ändert das Kopieren einer Datei in *target* weder den Modus noch den Eigentümer oder die Gruppe. Die letzte Modifikationszeit von *target* (und letzte Zugriffszeit, wenn *target* nicht existiert) und die letzte Zugriffszeit von *file1* werden auf die Zeit des Kopierens eingestellt. Wenn *target* ein Verweis auf eine Datei ist, werden alle Verweise beibehalten, und die Datei wird geändert.

SIEHE AUCH:

chmod(1), cpio(1), rm(1).

ACHTUNG!

ln verweist nicht über Dateisysteme hinweg. Diese Einschränkung ist notwendig, da Dateisysteme hinzugefügt und entfernt werden können.

FEHLER

Wenn *file1* und *target* in verschiedenen Dateisystemen sind, muß *mv* die Datei kopieren und das Original löschen. In diesem Fall gehen alle Verweise auf andere Dateien verloren.

BEZEICHNUNG

cpio – Ein/Aus-Kopieren von Datei-Archiven

ÜBERSICHT

cpio –o[acBv]

cpio –i[BcdmrtuvfsSb6] [patterns]

cpio –p[adlmuv] directory

BESCHREIBUNG

cpio –o (kopieren aus) liest die Standardeingabe, um eine Liste von Pfadnamen zu erhalten, und kopiert diese Dateien zusammen mit Pfadnamen und Status-Information auf die Standardausgabe. Ausgaben werden auf 512-Byte-Grenze aufgefüllt.

cpio –i (kopieren ein) liest Dateien aus der Standardeingabe. Es wird angenommen, daß die Standardeingabe das Ergebnis eines vorherigen **cpio –o** ist. Nur Dateien, deren Namen zu *patterns* paßt, werden ausgewählt. *Patterns* sind reguläre Ausdrücke, die in der *sh*(1) -Schreibweise für Dateinamengenerierung angegeben werden. In *pattern* passen die Meta-Zeichen ?, *, und [...] zu dem Schrägstrich-Zeichen / . Mehrere *patterns* können angegeben werden. Wenn keine *patterns* angegeben sind, gilt * (d. h. alle Dateien wählen). Jedes *pattern* sollte in Anführungszeichen stehen. Die ausgewählten Dateien werden bei Bedarf erstellt und in den aktuellen Verzeichnis-Baum kopiert, abhängig von den nachstehend aufgeführten Optionen. Die Berechtigungen der Dateien sind identisch zum letzten **cpio –o**. Der Eigentümer bzw. die Gruppe, der die Dateien gehören, ist der aktuelle Benutzer, sofern dieser nicht ein Systemverwalter ist. In diesem Fall behält *cpio* den Eigentümer bzw. die Gruppe, der die Dateien beim letzten **cpio –o** gehörten. ANMERKUNG: Wenn **cpio -i** eine Datei erstellen will, die schon existiert, und die vorhandene Datei wurde zur gleichen Zeit bzw. später erstellt, gibt **cpio** eine Warnung aus und ersetzt die Datei nicht. (Die **-u** Option dient zur bedingungslosen Überschreibung der vorhandenen Datei.)

cpio –p (durchlaufen) liest die Standardeingabe, um eine Liste von Pfadnamen der Dateien zu erhalten, die bedingt erstellt und in den Ziel- *directory* (Verzeichnis)-Baum entsprechend den nachstehenden Optionen kopiert werden.

Die zur Verfügung stehenden Optionen haben folgende Bedeutung:

a Rücksetzen der *access* (Zugriffs)-Zeiten der Eingabedateien, nachdem diese kopiert worden sind. Zugriffszeiten für Dateien, die Verweise sind, werden nicht zurückgesetzt, wenn **cpio -pla** angegeben ist.

B Ein-/Ausgabe soll in Blöcke von 5.120 Bytes pro Datensatz sein (gilt nicht für die *pass* Option; nur von Bedeutung bei Daten, die zu oder von einem zeichenorientierten Gerät z.B. /dev/rmt0 geleitet werden).

d Erstellung von *directories* (Verzeichnissen) nach Bedarf.

c Schreibt Dateikopf-Daten aus Portabilitätsgründen als ASCII- *Zeichen*. Diese Option sollte immer verwendet werden, wenn Ursprung- und Zielgeräte verschiedenen Typs sind.

r Interaktive *rename* (Umbenennung) von Dateien. Wenn der Benutzer eine Nullzeile eingibt, wird die Datei übersprungen. (Nicht möglich bei **cpio -p .**)

t Druckt ein *Inhaltsverzeichnis* der Eingabe. Es werden keine Dateien erstellt.

u *unconditionally* (ohne Vorbehalt) kopieren (normalerweise wird eine ältere Datei eine neuere mit dem gleichen Namen nicht ersetzen).

v *verbose* (wortreich): bewirkt die Ausgabe einer Dateinamenliste. Wenn diese gemeinsam mit der **t** Option benutzt wird, sieht das Inhaltsverzeichnis wie die Ausgabe eines **ls -l** -Kommandos aus (siehe *ls*(1)).

l Falls möglich sollten Verweise statt Kopien angelegt werden. Nur bei **-p** Option verwenden.

m *Modification* (Modifikationszeit) der letzten Datei wird beibehalten. Diese Option ist wirkungslos bei Verzeichnissen, die kopiert werden.

f Kopieren aller *files* (Dateien) außer den in *patterns* (Mustern).

s *Swap* (austauschen) von Bytes innerhalb jedes halben Worts. Nur mit der **-i** -Option zu verwenden.

S Austauschen *(swap)* von Halbworten innerhalb jedes Worts. Nur mit der **-i** Option zu verwenden.

b Reihenfolge der *Bytes* innerhalb jedes Worts umkehren. Nur mit der **-i** -Option zu verwenden.

6 Verarbeiten einer alten Datei (d. h. Formatieren gemäß sixth Edition des UNIX-Systems). Nur mit der **-i** -Option (einkopieren) zu verwenden.

ANMERKUNG: **cpio** geht davon aus, daß jedes Wort vier Bytes hat.

Wenn **cpio** beim Schreiben (-o) auf oder Lesen (-i) von zeichenorientierten Geräten das Mediumende (einer Diskette zum Beispiel) erreicht, druckt **cpio** folgende Nachricht aus:

If you want to go on, type device/file name when ready.

Austausch des Ausgabemediums und Eingabe des Namens des zeichenorientierten Geräts (zum Beispiel **/dev/rdiskette**) und Carriage-Return sind zur Weiterarbeit erforderlich. Sie können **cpio** auch auf ein anderes Gerät umlenken. Sie haben zum Beispiel zwei Diskettenlaufwerke und wollen von einem zum anderen wechseln, damit **cpio** weiterarbeiten kann, während Sie die Disketten austauschen. (Ein Betätigen von Carriage-Return allein genügt zum Verlassen von **cpio**.)

BEISPIELE

Die folgenden Beispiele zeigen drei Verwendungen von **cpio**.

Wenn die Standardeingabe über eine Pipe zu **cpio -o** geleitet wird, werden die Dateien so gruppiert, daß sie zu einer einzelnen Datei (../newfile) geleitet (>) werden können. Anstelle von "ls" können find, echo, cat, usw. zum Pipen von Namenslisten nach cpio verwendet werden. Anstelle einer Datei können Sie die Ausgabe an ein Gerät schikken.

ls | cpio -o >../*newfile*

cpio -i verwendet die Ausgabedatei von **cpio-o** (im Beispiel mit cat über eine Pipe geschickt), nimmt zu den Mustern (memo/a1, memo/b*) passende Dateien heraus, erstellt nach Bedarf Verzeichnisse unter dem aktuellen Verzeichnis (-d-Option) und schreibt die Dateien in die entsprechenden Verzeichnisse. Wenn keine Muster angegeben sind, werden alle Dateien von "newfile" ins Verzeichnis geschrieben.

cat newfile | cpio -id *"memo/a1" "memo/b*"*

cpio -p nimmt die durch Pipes gesandten Dateinamen und kopiert oder bindet diese Dateien (mit der -l-Option) in ein anderes Verzeichnis Ihres Geräts ein (zum Beispiel in *newdir*). Die -d-Option ermöglicht die Erstellung von Verzeichnissen je nach Bedarf. Die -m Option hält die Modifikationszeit fest. (Es ist wichtig, die -depth (Tiefen)-Option von find zur Generierung von Pfadnamen für cpio zu verwenden. Dies beseitigt Probleme für **cpio** bei der Erstellung von Dateien innerhalb von Nur-Lese-Verzeichnissen.)

find . -depth -print | cpio -pdlmv *newdir*

SIEHE AUCH:

ar(1), find(1), ls(1), tar(1).
cpio(4) im *Programmer's Reference Manual.*

HINWEISE

1) Pfadnamen sind begrenzt auf 256 Zeichen.
2) Nur der Systemverwalter darf Gerätedateien kopieren.
3) Blöcke werden in 512 Byte-Einheiten ausgegeben.

BEZEICHNUNG

crontab – crontab-Datei (für Routineaufgaben) des Benutzers

ÜBERSICHT

crontab [file]
crontab -r
crontab -l

BESCHREIBUNG

crontab kopiert die angegebene Datei oder, falls keine Datei angegeben ist, die Standardeingabe in ein Verzeichnis , das die crontabs aller Benutzer enthält. Die -r Option löscht die crontab-Datei eines Benutzers aus dem crontab-Verzeichnis. *crontab* -l listet die crontab-Datei des aufrufenden Benutzers auf.

Benutzer dürfen *crontab* verwenden, wenn ihre Namen in der Datei **/usr/lib/cron/cron.allow** aufgeführt sind. Wenn diese Datei nicht existiert, wird die Datei **/usr/lib/cron/cron.deny** überprüft, um festzustellen, ob dem Benutzer Zugriff auf *crontab* verweigert werden sollte. Wenn beide Dateien nicht vorhanden sind, darf nur Root einen Auftrag geben. Wenn **cron.allow** nicht existiert, und **cron.deny** vorhanden, jedoch leer ist, darf jeder Benutzer *crontab* verwenden. Die Dateien für Erlaubnis/Verweigerung bestehen aus Zeilen mit je einem Benutzernamen.

Eine crontab-Datei besteht aus Zeilen mit je sechs Feldern. Die Felder sind durch Leerzeichen oder Tabulatoren getrennt. Die ersten fünf sind ganzzahlige Muster, die folgendes angeben:

Minute (0-59),
Stunde (0-23),
Tag des Monats (1-31),
Monat des Jahres (1-12),
Tag der Woche (0-6 mit 0=Sonntag).

Jedes dieser Mustern kann entweder ein Stern sein (d. h. alle zulässigen Werte) oder eine Liste von Elementen, die durch Kommata getrennt sind. Ein Element ist entweder eine Zahl oder zwei Zahlen, die durch ein Minusvorzeichen getrennt sind (zur Angabe eines Bereichs, inklusive Bereichsgrenzen). Es ist zu beachten, daß Tage in zwei Feldern (Tag des Monats und Tag der Woche) angegeben werden können. Wenn beide in Form einer Elementliste angegeben sind, werden beide befolgt. Zum Beispiel würde 0 0 1,15 * 1 am ersten und fünfzehnten jedes Monats und auch jeden Montag einen Befehl ausführen. Wenn Tage nur in einem Feld angegeben werden, sollte im anderen Feld * gesetzt werden (so würde zum Beispiel durch Setzen von 0 0 * * 1 ein Kommando nur montags ausgeführt werden).

Das sechste Feld einer Zeile in einer crontab-Datei ist eine Zeichenfolge, die von der Shell zu den angegebenen Zeiten ausgeführt wird. Ein Prozentzeichen in diesem Feld (wenn es nicht durch \ entwertet wird), wird in ein Zeilenvorschub-Zeichen übersetzt. Nur die erste Zeile (bis zu einem % oder Zeilenende) des Kommandofelds wird von der Shell ausgeführt. Die anderen Zeilen werden dem Kommando als Standardeingabe bereitgestellt.

Die Shell wird von Ihrem **$HOME** Verzeichnis mit **sh** als **arg0** aufgerufen. Benutzer, die ihr *.profile* ausgeführt haben möchten, müssen dies ausdrücklich in der crontab-Datei machen. *Cron* liefert eine Standardumgebung für jede Shell, bei der **HOME, LOGNAME, SHELL(=/bin/sh)** und **PATH(=:/bin:/usr/bin:/usr/lbin)** definiert sind.

Wenn Sie die Standardausgabe und Standardfehlerausgabe Ihrer Kommandos nicht umlenken, werden alle generierten Ausgaben oder Fehler an Sie geschickt.

DATEIEN

/usr/lib/cron cron-Hauptverzeichnis
/usr/spool/cron/crontabsSpool-Bereich
/usr/lib/cron/logProtokoll-Daten
/usr/lib/cron/cron.allowListe zugelassener Benutzer
/usr/lib/cron/cron.deny Liste nicht zugelassener Benutzer

SIEHE AUCH

sh(1).
cron(1m) im *Administrator's Reference Manual.*

ACHTUNG!

Wenn Sie das **crontab** Kommando aus Versehen ohne Argument(e) eingeben, sollten Sie nicht versuchen, mit einem CTRL-d auszusteigen. Dies hätte zur Folge, daß alle Eingaben in Ihrer **crontab** -Datei gelöscht würden. Benutzen Sie stattdessen DEL zum Verlassen von crontab.

BEZEICHNUNG

csplit – Dateiaufteilung in Abschnitte

ÜBERSICHT

csplit [-s] [-k] [-f prefix] file arg1 [... argn]

BESCHREIBUNG

csplit liest die Datei *file* und teilt sie in n+1 Abschnitte auf, die durch die Argumente *arg1... argn* definiert werden. Standardmäßig werden die Abschnitte in die Dateien xx00 ... xx*n* geschrieben (*n* darf nicht größer als 99 sein). Die Abschnitte enthalten die folgenden Teile von *file:*

00: Anfang von *file* bis zu (aber nicht einschließlich) der durch *arg1* bezeichneten Zeile.

01: Von der durch *arg1* bezeichneten Zeile bis zur durch *arg2* bezeichneten Zeile.

:

:

n+1: Von der durch *argn* bezeichneten Zeile bis zum Ende von *file.*

Wenn das *file* Argument ein – ist, wird die Standardeingabe verwendet.

Die Optionen zu *csplit* lauten:

-s *csplit* standardmäßig gibt die Zeichenanzahl für jede erstellte Datei an. Wenn die **-s** Option vorhanden ist, unterdrückt *csplit* die Ausgabe der Zeichenanzahl.

-k *csplit* löscht gewöhnlich erstellte Dateien, wenn ein Fehler eintritt. Wenn die **-k** Option vorhanden ist, bleiben vorher erstellte Dateien bestehen.

-f*refix* Bei Verwendung der **-f** -Option werden die erstellten Dateien mit *prefix***00** ... *prefix*n bezeichnet. Vorgegeben ist **xx00** ... xx*n*.

Die Argumente (*arg1* ... *argn*) zu *csplit* können eine Kombination aus folgenden Angaben sein:

/rexp/ Für den Abschnitt von der aktuellen Zeile bis (aber nicht einschließlich) der Zeile, die den regulären Ausdruck *rexp* enthält, soll eine Datei erstellt werden. Die aktuelle Zeile wird die Zeile, die *rexp* enthält. Hinter diesem Argument kann wahlweise + oder - Zeilenanzahl (z. B./**Page/-5**) stehen.

%rexp% Dieses Argument ist identisch mit */rexp/*, es wird jedoch keine Datei für den Abschnitt erstellt.

lnno Für den Abschnitt soll von der aktuellen Zeile bis zu (aber nicht einschließlich) *lnno* soll eine Datei erstellt werden. Die aktuelle Zeile wird *lnno.*

{*num*} Das vorhergehende Argument wird wiederholt. Dieses Argument kann nach jedem der obigen Argumente stehen. Wenn es nach einem *rexp* Argument steht, wird dieses Argument *num* -mal angewendet. Wenn es nach *lnno* steht, wird die Datei ab der aktuellen Stelle nach je *lnno* Zeilen (*num* -mal) geteilt.

Alle *rexp* Argumente, die Leerzeichen oder andere für die Shell wichtige Zeichen enthalten, müssen zwischen den korrekten Anführungszeichen stehen. Reguläre Ausdrücke dürfen nicht Zeichen für eine neue Zeile innerhalb eines Felds enthalten. *csplit* arbeitet nicht mit der Originaldatei; es ist die Aufgabe des Benutzers, diese zu löschen.

BEISPIELE

```
csplit -f cobol file '/procedure division/' /par5./ /par16./
```

Dieses Beispiel erstellt vier Dateien, **cobol00 ... cobol03**. Nach dem Editieren der "split"-Dateien, können sie wie folgt neu kombiniert werden:

```
cat cobol0[0-3] > file
```

Es ist zu beachten, daß dieses Beispiel die Originaldatei überschreibt

```
csplit -k file 100 {99}
```

In diesem Beispiel würde die Datei bei je 100 Zeilen (bis max. 10.000 Zeilen) geteilt werden. Mit der **-k** -Option werden die erstellten Dateien beibehalten, wenn es sich um weniger als 10.000 Zeilen handelt; es würde jedoch immer noch eine Fehlermeldung ausgegeben.

```
csplit -k prog.c '%main(%' '/^}/+1' {20}
```

Wenn in **prog.c**, wie üblich in **C**, eine Funktion abgeschlossen wird durch } am Anfang der Zeile, dann wird bei diesem Beispiel für jede C-Funktion in **prog.c** (bis zu 21) eine Datei erstellt.

SIEHE AUCH:

ed(1), sh(1).
regexp(5) im *Programmer's Reference Manual.*

DIAGNOSE

Selbsterklärend mit Ausnahme von:

arg – out of range (nicht im Bereich)

Dies bedeutet, daß das angegebene Argument keine Zeile zwischen der aktuellen Position und dem Dateiende bezeichnet.

BEZEICHNUNG

ct – getty für ein Remote-Terminal erzeugen

ÜBERSICHT

ct [**-wn**] [**-xn**] [**-h**] [**-v**] [**-s***speed*] telno ...

BESCHREIBUNG

ct wählt die Telefonnummer eines Modems, das an ein Terminal angeschlossen ist, und erzeugt einen *getty* Prozeß zu diesem Terminal. *Telno* ist eine Telefonnummer, mit Gleichheitszeichen für sekundäre Wähltöne und Minuszeichen für Verzögerungen an passenden Stellen. (Der für *telno* erlaubte Zeichensatz ist 0 bis 9, -, =, *, und #. Die max. Länge von *telno* beträgt 31 Zeichen). Wenn mehrere Telefonnummern angegeben werden, arbeitet *ct* der Reihen nach bis eine antwortet; dies ist zweckmäßig zur Angabe alternativer Wahlmöglichkeiten.

ct durchläuft alle in der Datei **/usr/lib/uucp/Devices** angeführten Leitungen so lange, bis es eine verfügbare Leitung mit passenden Attributen findet oder keine Einträge mehr hat. Wenn keine freien Leitungen vorhanden sind, fragt *ct*, ob es auf eine warten soll und wenn ja, wie viele Minuten gewartet werden soll, ehe aufgegeben wird. In Abständen von einer Minute versucht *ct* immer wieder eine Verbindung herzustellen, bis die angegebene Grenze überschritten ist. Dieser Vorgang kann durch Angabe der **-w***n* Option aufgehoben werden, wobei *n* die Höchstanzahl von Minuten ist, die *ct* auf eine Leitung warten soll.

Die **-x***n* Option wird zur Fehlersuche verwendet; es wird eine detaillierte Ausgabe der Programmausführung auf stderr erzeugt. Die Testebene, *n*, ist eine einzelne Ziffer; **-x9** ist der gebräuchlichste Wert.

Normalerweise belegt *ct* die aktuelle Leitung nicht, so daß die Leitung den ankommenden Anfruf beantworten kann. Die **-h** Option verhindert dieses Verhalten. Die **-h** Option wartet auch auf die Beendigung des angegebenen *ct* Prozesses, bevor die Kontrolle an das Terminal des Benutzers zurückgegeben wird. Wenn die **-v** Option verwendet wird, sendet *ct* laufend Kommentare auf die Standardfehlerausgabe.

Die Datenrate kann mit der **-s** Option gesetzt werden, wobei *speed* in Baud ausgedrückt wird. Die Standardrate ist 1200.

Nachdem der Benutzer sich am Bestimmungsterminal abmeldet, können je nach getty-Typ (*getty* oder *uugetty*) zwei Situationen eintreten. Im ersten Fall fragt *ct* den Benutzer mit **Reconnect?** Wenn die Antwort mit dem Buchstaben **n** beginnt, wird die Leitung freigegeben; andernfalls wird wieder mit *getty* begonnen und die Eingabe-Aufforderung **login:** wird ausgegeben. Im zweiten Fall gibt es schon ein getty (*uugetty*) auf der Leitung und die **login:** Nachricht erscheint.

Zur korrekten Abmeldung muß der Benutzer **control D** eingeben.

Voraussetzung ist natürlich der Anschluß des Bestimmungsterminal an ein Modem, das einem Telephon antworten kann.

DATEIEN

/usr/lib/uucp/Devices
/usr/adm/ctlog

SIEHE AUCH

cu(1c), login(1), uucp(1c).
getty(1m), uugetty(1m) im *Administrator's Reference Manual.*

FEHLER

Bei einem gemeinsamen Anschluß für Anwählen und Empfangen, muß man bei dem *uugetty* Programm für diese Leitung die **-r** Option angeben (siehe *uugetty*(1M)).

BEZEICHNUNG

cu – ein anderes UNIX -System aufrufen

ÜBERSICHT

cu [-sspeed] [-lline] [**-h**] [**-t**] [**-d**] [**-o** | **-e**] [**-n**] telno
cu [**-s** speed] [**-h**] [**-d**] [**-o** | **-e**] **-l** line
cu [**-h**] [**-d**] [**-o** | **-e**] systemname

BESCHREIBUNG

cu ruft ein anderes UNIX -System auf, ein Terminal oder ein System, das möglicherweise kein UNIX-System ist. Ein interaktiver Dialog und Übertragungen von ASCII Dateien ist möglich.

cu akzeptiert die folgenden Optionen und Argumente:

-s*speed* Gibt die Übertragungsgeschwindigkeit (300, 1200, 2400, 4800, 9600) an. Der Standardwert ist "irgendeine" Geschwindigkeit, die von der Reihenfolge der Leitungen in der Datei **/usr/lib/uucp/Devices** abhängt. Die meisten Modems arbeiten mit 300 oder 1200 Baud. Direkt angeschlossene Leitungen können eine höhere Geschwindigkeit als 1200 Baud haben.

-l*line* Gibt einen Gerätenamen an, der als Kommunikationsleitung verwendet wird. Dadurch kann die Suche nach der ersten verfügbaren Leitung mit passender Geschwindigkeit aufgehoben werden. Wenn die **-l** Option ohne die **-s** Option benutzt wird, wird die Geschwindigkeit einer Leitung aus der Datei **Devices** entnommen. Wenn die Optionen **-l** und **-s** zusammen verwendet werden, überprüft *cu* die Datei **Devices**, um festzustellen, ob die gewünschte Geschwindigkeit für die angefragte Leitung zur Verfügung steht. Wenn ja wird der Anschluß mit der gewünschten Geschwindigkeit hergestellt; andernfalls wird eine Fehlermeldung ausgegeben und der Anruf wird nicht durchgeführt. Das angegebene Gerät ist gewöhnlich eine direkt angeschlossene asynchrone Leitung (z. B. **/dev/tty***lab*). In diesem Fall ist eine Telefonnummer (*telno*) nicht erforderlich. Das angegebene Gerät muß nicht im **/dev** Verzeichnis sein. Wenn das angegebene Gerät mit einer automatischen Wählvorrichtung ausgestattet ist, muß eine Telefonnummer bereitgestellt werden. Angabe von *systemname* anstelle von *telno* bei dieser Option führt nicht zum gewünschten Ergebnis (siehe unten *systemname*).

-h Emuliert lokales Echo, Anrufe von anderen Rechnersystemen werden unterstützt, bei denen die Terminals auf Halbduplex-Modus eingestellt werden müssen.

-t Wird zur Anwahl eines ASCII-Terminals verwendet, die auf automatische Beantwortung eingestellt wurde. Der Zeilenvorschub wird, falls nötig, auf einen Zeilenvorschub-Wagenrücklauf gesetzt.

-d Bewirkt die Ausgabe von Fehlermeldungen.

-o Für Daten die zum Remote-System geschickt werden, wird eine ungerade Parität erzeugt.

-n Diese Option dient der zusätzlichen Sicherheit und veranlaßt Benutzer, die zu wählende Telefonnummer selbst bereitzustellen anstatt sie der Kommandozeile zu entnehmen.

-e Für Daten die zum Remote-System geschickt werden, wird eine gerade Parität erzeugt.

telno Bei der Benutzung einer automatischen Wählvorrichtung ist das Argument die Telefonnummer mit Gleichheitszeichen für sekundären Wählton oder entsprechend gesetzten Minuszeichen für Verzögerungen von 4 Sekunden.

systemname Ein uucp Systemname kann anstatt einer Telefonnummer verwendet werden; in diesem Fall erhält *cu* eine passende Direktleitung oder Telefonnummer von **/usr/lib/uucp/Systems.** Anmerkung: die *systemname* Option sollte nicht in Verbindung mit den **-l** und **-s** Optionen verwendet werden, da *cu* die erste für den Systemnamen angegebene verfügbare Leitung anschließt, und dann die gewünschte Leitung und Geschwindigkeit ignoriert.

Nach Herstellung eines Anschlusses besteht *cu* aus zwei Prozessen: Der *transmit* Prozeß liest Daten von der Standardeingabe und leitet sie ans Remote-System weiter mit Ausnahme von Zeilen, die mit ~ beginnen; der *receive* Prozeß akzeptiert Daten vom Remote-System und leitet sie zur Standardausgabe weiter mit Ausnahme von Zeilen, die mit ~ beginnen. Normalerweise wird zur Kontrolle der Eingabe vom Remote-System ein automatisches DC3/DC1 Protokoll verwendet, um zu verhindern, daß der Puffer überläuft. Zeilen, die mit ~ beginnen, haben eine besondere Bedeutung.

Der *transmit* Prozeß interpretiert die folgenden vom Benutzer angegebenen Kommandos:

~. beendet den Dialog

~! wechselt in eine interaktive Shell des lokalen Systems.

~!*cmd*...
cmd auf lokalem System (über **sh -c**) ausführen.

~$*cmd*...
cmd lokal ausführen und die Ausgabe zum Remote-System schikken.

~%**cd**
Das Verzeichnis im lokalem System wechseln. Anmerkung: ~!**cd** bewirkt die Ausführung des Kommando durch eine Sub-Shell, was wahrscheinlich nicht beabsichtigt ist.

~%**take** *from* [*to*]
Kopiert Datei *from* (im entfernten System) zu Datei *to* im lokalen System. Wenn *to* ausgelassen wird, wird das *from* Argument an beiden Stellen verwendet.

~%**put** *from* [*to*]
Kopiert Datei *from* (im lokalen System) zu Datei *to* im Remote-System. Wenn *to* ausgelassen wird, wird das *from* Argument an beiden Stellen verwendet.

Bei beiden ~%**take** und **put** Kommandos werden bei der Übertragung der einzelen Dateiblöcke aufeinanderfolgende einzelne Ziffern am Bildschirm ausgegeben.

~~ *line*
Schickt die Zeile ~ *line* zum Remote-System.

~%**break**
Sendet ein **BREAK** zum Remote-System (dies kann auch als ~%b angegeben werden).

~%**debug**
Schaltet die Fehlersuche Option **-d** ein oder aus (diese kann auch durch ~%**d** angegeben werden).

~t druckt die Werte der Termio-Struktur-Variablen für das Terminal des Benutzers aus (praktisch bei Fehlersuche).

~l druckt die Werte der Termio-Struktur-Variablen für die Remote-Datenleitung aus (praktisch für die Fehlersuche).

~%nostop
Schaltet von DC3/DC1 Eingabe-Kontrollprotokoll auf keine Eingabekontrolle. Dies ist nützlich, falls es sich um ein Remote-System handelt, das nicht korrekt auf die Zeichen DC3 und DC1 reagiert.

Der *receive* Prozeß kopiert normalerweise Daten vom Remote-System auf seine Standardausgabe. Intern erreicht das Programm dies durch Einleitung einer Ausgabe-Umleitung auf eine Datei, wenn eine vom Remote-System kommende Zeile mit ~ beginnt.

Daten vom Remote-System werden in die Datei *file* im lokalen System umgelenkt (oder angehängt, wenn **>>** verwendet wird). Das nachfolgende **~>** markiert das Ende der Umlenkung.

Die Verwendung von **~%put** ist nur möglich, wenn *stty*(1) und *cat*(1) beim Remote-System vorhanden sind. Die aktuellen Zeichen für Löschen und Abbrechen müssen auf dem Remote- und dem lokalen System die gleichen sein. Backslashes werden an den entsprechenden Stellen eingefügt.

Die Verwendung von **~%take** ist nur möglich, wenn *echo*(1) und *cat*(1) auf dem Remote-System zur Verfügung stehen. Auf dem Remote-System sollte auch der *tabs* Modus (siehe *stty(1)*) gesetzt werden, wenn Tabulatoren ohne Erweiterung auf Leerzeichen kopiert werden sollen.

Wenn *cu* auf System X verwendet wird, um System Y anzuschließen und anschließend auf System Y zum Anschluß von System Z benutzt wird, können Kommandos auf System Y mit ~~ ausgeführt werden. Ausführung eines Tilde-Kommandos weist den Benutzer auf das lokale System. Zum Beispiel kann das Kommando uname auf Z, X, und Y wie folgt ausgeführt werden:

```
uname
Z
~[X]!uname
X
~~[Y]!uname
Y
```

Im allgemeinen bewirkt ~, daß das Kommando auf dem ursprünglichen Gerät ausgeführt wird, während ~~ das Kommando auf dem nächsten Gerät in der Reihe der angeschlossenen Geräte ausführt.

BEISPIELE

Anwahl eines Systems mit Telefonnummer 9 201 555 1212 unter Verwendung von 1200 Baud (wobei der Wählton nach der 9 erwartet wird):

```
cu -s1200 9=12015551212
```

Wenn die Geschwindigkeit nicht angegeben wird, ist "irgendeine" der Standardwert.

In einem System anmelden, das über eine Direktleitung angeschlossen ist:

```
cu -l /dev/tttyiXX
```

oder

```
cu -l tttyiXX
```

Anwählen eines System an der speziellen Leitung mit der speziellen Geschwindigkeit:

```
cu -s1200 -l tttyiXX
```

Anwählen eines Systems, das eine spezielle Leitung mit einer automatischen Wählvorrichtung verwendet:

```
cu -l culXX 9=12015551212
```

Verwenden eines Systemnamens:

```
cu systemname
```

DATEIEN

/usr/lib/uucp/Systems
/usr/lib/uucp/Devices
/usr/spool/locks/LCK..(tty-device)

SIEHE AUCH:

cat(1), ct(1C), echo(1), stty(1), uucp(1C), uname(1).

DIAGNOSE

Bei normaler Beendigung ist der Endecode null, sonst eins.

ACHTUNG!

Das *cu* Kommando führt keine Integritätsprüfung bei den übertragenen Daten aus. Datenfelder mit besonderen *cu* Zeichen werden eventuell nicht korrekt übertragen. Je nach der Verbindungs-Hardware muß die Konvertierung eventuell mit ~ beendet werden, selbst wenn **stty O** benutzt worden ist. Nicht abdruckbare Zeichen werden nicht immer korrekt durch **~%put** oder **~%take** Kommandos übertragen. *cu* zwischen einem IMBR1 und einem Penril-Modem gibt nicht gleich nach Anschluß ein Login-Prompt an. Sie erhalten den Prompt durch Betätigung von Wagenrücklauf.

FEHLER

Während der **~%put** Operation kann die Übertragung künstlich mit *cu* verlangsamt werden, so daß ein Datenverlust fast ausgeschlossen werden kann.

BEZEICHNUNG

cut – ausgewählte Felder auf jeder Zeile einer Datei auswählen

ÜBERSICHT

cut **–c***list* [file ...]
cut **–f***list* [**–d** *char*] [**–s**] [file ...]

BESCHREIBUNG

cut wird zum Auswählen von Spalten einer Tabelle oder von Feldern auf jeder Zeile einer Datei benutzt; d. h. gemäß der Datenbank-Fachsprache stellt cut die Projektion einer Relation dar. Die durch *list* angegebenen Felder können eine feste Länge haben, d. h. Zeichenpositionen wie bei Lochkarten (**–c**-Option) oder die Länge kann von Zeile zu Zeile variieren und mit einem Feldbegrenzer-Zeichen wie *tab* (**–f**-Option) markiert werden. *cut* kann als ein Filter verwendet werden; wenn keine Dateien angegeben sind, wird die Standardeingabe verwendet. Außerdem bezeichnet der Dateiname "–" explizit die Standardeingabe.

Die Optionen haben folgende Bedeutung:

list Eine durch Kommata getrennte Liste ganzzahliger Feldnummern (in aufsteigender Reihenfolge) mit wahlweisem – zur Anzeige von Bereichen [z.B. **1,4,7**; **1–3,8**; **–5,10** (kurz für **1–5,10**); oder **3–** (kurz für drittes bis letztes Feld)].

–c*list* *list* nach **–c** (ohne Leerzeichen) gibt Zeichenpositionen an (z.B. durchläuft **–c1–72** die ersten 72 Zeichen jeder Zeile).

–f*list* *list* nach **–f** ist eine Liste von Feldern, die in der Datei durch ein Begrenzungszeichen (siehe **–d**) getrennt sind; z.B. kopiert **–f1,7** nur das erste und siebte Feld. Zeilen ohne Feldbegrenzer werden unverändert ausgegeben (praktisch bei Zwischenüberschriften von Tabellen), sofern nicht **–s** angegeben wird.

–d*char* Das Zeichen nach **–d** ist der Feldbegrenzer (nur bei **–f** -Option). Vorgegeben ist *tab*. Leerzeichen oder andere Zeichen mit besonderer Bedeutung für die Shell müssen in Anführungszeichen stehen.

–s Unterdrückt Zeilen ohne Begrenzer-Zeichen bei der **–f** -Option. Ist die Option nicht angegeben, werden Zeilen ohne Feldbegrenzer unverändert ausgegeben.

Entweder muß die **–c** oder die **–f** -Option angegeben werden.

Um horizontal Text (durch Angabe von Kontext) aus einer Datei auszuwählen, sollten Sie *grep*(1) verwenden; um Dateien spaltenweise (horizontal) zusammenzufügen, sollten Sie *paste*(1) verwenden. Zur Neuordnung von Spalten in einer Tabelle wird *cut* und *paste* verwendet.

BEISPIELE

cut -d: -f1,5 /etc/passwd
Zuordnung der Benutzernummern zu Namen

name =`who am i | cut -f1 -d" "`
name auf aktuellen Login-Namen setzen.

DIAGNOSE

ERROR: line too long
Eine Zeile kann nicht mehr als 1023 Zeichen oder Felder haben oder es gibt kein Zeichen für neue Zeile.

ERROR: bad list for c/f option
Fehlende **-c** oder **-f** -Option oder falsch angegebene *list*. Kein Fehler tritt ein, wenn eine Zeile weniger Felder hat als für *list* gebraucht wird.

ERROR: no fields *list* ist leer.

ERROR: no delimeter
Fehlendes *char* bei **-d** -Option.

ERROR: cannot handle multiple adjacent backspaces
Nebeneinanderstehende Rücksetz-Zeichen können nicht korrekt verarbeitet werden.

WARNING!: cannot open <filename>
filename kann entweder nicht gelesen werden oder existiert nicht. Bei mehreren Dateinamen wird die Verarbeitung fortgesetzt.

SIEHE AUCH:

grep(1), paste(1).

BEZEICHNUNG

date – Datum ausgeben und setzen

ÜBERSICHT

date [+ format]
date [mmddhhmm[[yy]|[ccyy]]]

BESCHREIBUNG

Wenn kein Argument angegeben oder wenn das Argument mit + beginnt, werden das aktuelle Datum und die Zeit ausgegeben. Andernfalls wird das aktuelle Datum gesetzt (nur durch den Systemverwalter möglich). Das erste *mm* ist der Monat; *dd* ist der Tag im Monat; *hh* ist die Stunde (24 Stunden); das zweite *mm* sind die Minuten; *cc* ist das Jahrhundert minus 1; *yy* sind die beiden letzten Zahlen der Jahreszahl, die wahlweise benutzt werden können. Zum Beispiel wird bei

date 10080045

das Datum auf den 8. Oktober, 12:45 AM gesetzt. Das laufende Jahr ist voreingestellt, wenn kein Jahr angegeben ist. Das System arbeitet mit GMT (Greenwich Mean Time). *date* führt die Umwandlung in und von lokaler Standard- und Tageszeit durch. Nur der Systemverwalter kann das Datum ändern.

Wenn das Argument mit + beginnt, kann die Ausgabe von *date* vom Benutzer gesteuert werden. Alle Ausgabefelder haben eine feste Größe (gegebenenfalls mit Null aufgefüllt). Jedem Feldbezeichner ist ein % vorangestellt; er wird bei der Ausgabe durch den entsprechenden Wert ersetzt. Ein einzelnes % wird mit %% codiert. Alle andere Zeichen werden ohne Änderung auf die Ausgabe kopiert. Die Zeichenfolge wird immer mit einem Zeichen für neue Zeile beendet. Enthält das Argument Leerzeichen, so muß es quotiert werden (siehe Abschnitt **EXAMPLE**).

Spezifikationen für landessprachliche Übersetzungen für Monats- und Wochentage werden unterstützt. Welche Sprache verwendet wird, hängt vom Wert der Umgebungsvariable **LANGUAGE** ab (siehe *eviron*(5)). Die landessprachlichen Bezeichnungen für Monate und Wochentage werden Strings aus der Datei für diese Sprache im Verzeichnis **/lib/cftime** entnommen (siehe *cftime*(4)).

Wenn Datum und Zeit erfolgreich gesetzt wurden, gibt *date* das neue Datum entsprechend dem Format aus, das in der Umgebungsvariablen **CFTIME** definiert ist (siehe *eviron*(5)).

Feldbezeichner (müssen durch % eingeleitet werden):

a abgekürzter Name des Wochentags

A ausgeschriebener Name des Wochentags

b abgekürzter Monatsname

B ausgeschriebener Monatsname

d Tag des Monats – 01 bis 31

D Datum als mm/dd/yy

e Tag des Monats – 1 bis 31 (einzelne Ziffern werden durch ein Leerzeichen eingeleitet.)

h abgekürzter Monatsname (Synonym für %b)

H Stunde – 00 bis 23

I Stunde – 01 bis 12

j Tag des Jahres – 001 bis 366

m Monat des Jahres – 01 bis 12

M Minute – 00 bis 59

n fügt ein Zeichen für neue Zeile ein

p Zeichenkette mit dem Kennzeichen für Vormittag oder Nachmittag (Standard ist AM or PM)

r Zeit in der Form *hh:mm:ss pp* wobei *pp* das Kennzeichen für Vormittag bzw. Nachmittag ist (Standard ist AM or PM)

R Zeit in der Form hh:mm

S Sekunde – 00 bis 59

t fügt ein Tabulatorzeichen ein

T Zeit in der Form *hh:mm:ss*

U Nummer der Kalenderwoche (Sonntag ist der erste Tag der Woche) – 01 bis 52

w Tag der Woche – Sonntag = 0

W Nummer der Kalenderwoche (Montag ist der erste Tag der Woche) – 01 bis 52

x Länderspezifisches Format des Datums

X Länderspezifisches Format der Zeit

y Jahr innerhalb des Jahrhunderts – 00 bis 99

Y Jahr in der Form *ccyy* (4 Ziffern)

Z Name der Zeitzone

BEISPIEL

date '+DATE: %m/%d/%y%nTIME: %H:%M:%S'

würde folgende Ausgabe ergeben:

DATE: 08/01/76
TIME: 14:45:05

DIAGNOSE

No permission	Wenn Sie nicht der Systemverwalter sind und versuchen, das Datum zu ändern;
bad conversion	Wenn das Setzen des Datums syntaktisch falsch ist;

DATEIEN

/dev/kmem

ACHTUNG!

Wenn Sie das Datum bei Multi-User-Betrieb ändern müssen, ist *sysadm (1) datetime* zu verwenden.

BEMERKUNG

Systemadministratoren sollten folgendes berücksichtigen: Unvorhergesehene Ergebnisse können auftreten, wenn Sie versuchen sollten, die aktuelle Zeit auf einen Zeitpunkt zu setzen, an dem ein Wechsel von Zeitzonen stattfindet (z. B. in dem Intervall zwischen Sommerzeit und Winterzeit).
Grundsätzlich sollte die Systemzeit während des laufenden Betriebs der Anlage nicht zurückgesetzt werden, da in diesem Fall Probleme bei der Verwendung bestimmter Tools (z. B. bei Kommandos, die SCCS-Dateien verändern) auftreten können.

SIEHE AUCH:

sysadm(1),
cftime(4), environ(5) im *Programmer's Reference Manual.*

BEZEICHNUNG

dc – Tischrechner

ÜBERSICHT

dc [file]

BESCHREIBUNG

dc ist ein Programmpaket für Arithmetik mit beliebiger Genauigkeit. Normalerweise verarbeitet es ganze Dezimalzahlen, es kann jedoch eine Eingabebasis, Ausgabebasis und eine Anzahl der beizubehaltenden Dezimalstellen angegeben werden. (Siehe *bc*(1), ein Preprozessor für *dc*, der Infix-Schreibweise und eine C-ähnliche Syntax zur Implementierung von Funktionen bietet. *bc* stellt auch geeignete Ablaufstrukturen für Programme bereit.) Die wesentliche Struktur von *dc* ist ein Kellerautomat (umgekehrte Polnische Notation). Wenn ein Argument angegeben ist, wird bis zum Erreichen des Dateiendes die Eingabe aus dieser Datei und dann von Standardeingabe gelesen. Folgende Konstruktionen werden erkannt:

number

Der Wert der *number* wird auf den Keller geschrieben. Eine Zahl ist eine nicht unterbrochene Zeichenfolge der Ziffern 0–9. Ein Unterstrich (_) kann zur Eingabe einer negativen Zahl vorangestellt werden. Zahlen können Dezimalstellen enthalten.

+ - / * % ^

Addition (+), Subtraktion (–), Multiplikation (*), Division (/), ganzzahlige Division (%) oder Potenzierung (^) der beiden obersten Werte im Keller. Die beiden Einträge werden aus dem Keller herausgeholt; an der gleichen Stelle erscheint dann das Ergebnis. Dezimalstellen werden bei Exponenten ignoriert.

s*x* Das oberste Element wird aus dem Keller geholt und in einem *x* Register gespeichert; *x* kann ein beliebiges Zeichen sein. Wenn **s** großgeschrieben ist, wird *x* wie ein Keller behandelt und der Wert auf den Keller geschrieben.

l*x* Der Wert im Register *x* wird auf den Keller geschrieben. Das Register *x* wird nicht verändert. Alle Register haben anfangs den Wert Null. Wenn das **l** großgeschrieben ist, wird das Register *x* wie ein Keller behandelt, und sein oberster Wert wird auf den Hauptkeller geschrieben.

d Der oberste Wert im Keller wird dupliziert.

p Der oberste Wert im Keller wird ausgegeben, er bleibt unverändert.

P Interpretiert den obersten Wert des Kellers als eine ASCII Zeichenfolge, entfernt diese und druckt sie aus.

f Alle Werte im Keller werden ausgegeben.

q Beendet das Programm. Bei Ausführung einer Kommandofolge wird die Rekursion um zwei reduziert.

Q Beendet das Programm. Der oberste Wert wird aus dem Keller geholt und die Rekursionstiefe der Kommandofolge um diesen Wert reduziert.

x Interpretiert das oberste Element des Kellers als Zeichenreihe und führt sie als Folge von *dc* -Kommandos aus.

X Tauscht die an oberster Stelle im Keller befindliche Zahl gegen ihren Skalierungsfaktor aus.

[...] Schreibt die in Klammern eingeschlossene ASCII-Zeichenfolge an die oberste Stelle im Keller.

<*x* >*x* =*x*
Die beiden obersten Elemente werden aus dem Keller geholt und verglichen. Register *x* wird ausgewertet, wenn die Elemente der angegebenen Relation entsprechen.

v Ersetzt das oberste Element im Keller durch seine Quadratwurzel. Alle vorhandenen Dezimalstellen des Arguments werden berücksichtigt, ansonsten wird der Skalierungsfaktor ignoriert.

! Interpretiert die restliche Zeile als ein Kommando des UNIX-Systems .

c Alle Werte werden aus dem Keller geholt.

i Der oberste Wert wird aus dem Keller geholt und als Basis für weitere Eingaben verwendet. **I** Schreibt die Eingabebasis auf den Keller.

o Der oberste Wert wird aus dem Keller geholt und als Basis für weitere Ausgaben verwendet.

O Schreibt die Ausgabebasis auf den Keller.

k Der oberste Wert des Kellers wird geholt; dieser Wert wird als nicht-negativer Skalierungsfaktor verwendet; die entsprechende Stellenanzahl wird in der Ausgabe ausgegeben und bei Multiplikation, Division und Potenzierung beibehalten. Die Wechselbeziehung zwischen Skalierungsfaktor, Eingabe- und Ausgabebasis bleibt zufriedenstellend, wenn alle zusammen geändert werden.

z Die Kellergröße wird in den Keller geschrieben.

Z Ersetzt die Zahl oben im Keller durch ihre Länge.

? Eine Eingabezeile wird aus der Eingabequelle (normalerweise dem Terminal) genommen und ausgeführt.

; : Werden von *bc(1)* für Feldoperationen verwendet.

BEISPIEL

Dieses Beispiel gibt die ersten zehn Werte von n! aus:

```
[la1+dsa *pla10>y]sy
0sa1
lyx
```

SIEHE AUCH:

bc(1).

DIAGNOSE

x is unimplemented
wobei *x* eine Oktalzahl ist.

stack empty
es sind nicht genügend Elemente im Keller, um die gewünschte Funktion auszuführen

Out of space
wenn die freie Liste voll ist (zu viele Stellen).

Out of headers
wenn zu viele Zahlen aufbewahrt werden

Out of pushdown
wenn zu viele Elemente im Keller sind.

Nesting Depth
zu viele geschachtelte Ausführungsebenen.

BEZEICHNUNG

dd – eine Datei umwandeln und kopieren

ÜBERSICHT

dd [option =value] ...

BESCHREIBUNG

dd kopiert die angegebene Eingabedatei mit möglichen Umwandlungen auf die angegebene Ausgabe. Die Benutzung der Standardeingabe und -Ausgabe ist voreingestellt. Die Blockgrößen der Eingabe und Ausgabe können festgesetzt werden, um den Vorteil der rohen (raw) physikalischen Ein-/Ausgabe zu nutzen.

Option	*Value*
if=*file*	Eingabedateiname; Standardeingabe ist voreingestellt
of=*file*	Ausgabedateiname; Standardausgabe ist voreingestellt
ibs=*n*	Eingabe - Blockgröße ist *n* Bytes (Voreinstellung 512)
obs=*n*	Ausgabe - Blockgröße (Voreinstellung 512)
bs=*n*	Sowohl Eingabe- als auch Ausgabeblockgröße festsetzen, die die *ibs* und *obs* ersetzt; auch wenn keine Umwandlung angegeben ist, ist diese Option besonders effizient, da keine Kopien im Kern gemacht werden müssen
cbs=*n*	Puffergröße zur Umwandlung
skip=*n*	Überspringen von *n* Eingabeblöcken vor Beginn des Kopiervorgangs
seek=*n*	Suchen von *n* Blöcken vom Anfang der Ausgabedatei vor dem Kopieren
count=*n*	Nur *n* Eingabeblöcke kopieren
conv=**ascii**	Konvertieren von EBCDIC in ASCII
ebcdic	Konvertieren von ASCII in EBCDIC
ibm	eine leicht unterschiedliche Abbildung von ASCII in EBCDIC
lcase	Abbilden des Alphabets auf Kleinbuchstaben
ucase	Abbilden des Alphabets auf Großbuchstaben
swab	Je zwei benachbarte Bytes vertauschen
noerror	Verarbeitung bei Fehler nicht beenden
sync	Auffüllen jedes Eingabeblocks bis zu *ibs*
...,...	Mehrere durch Komma getrennte Umwandlungen

Überall, wo Größen festgelegt sind, wird eine Anzahl von Bytes erwartet. Eine Zahl kann mit **k**, **B**, oder **w** enden, um Multiplikation mit jeweils 1024, 512 oder 2 anzugeben; ein Zahlenpaar kann durch **x** getrennt werden, um eine Multiplikation anzuzeigen.

cbs wird nur verwendet, wenn *conv=ascii* oder *conv=ebcdic* angegeben sind. Im ersten Fall werden *cbs* Zeichen in den Umwandlungspuffer geschrieben (wo Umwandlung zu ASCII erfolgt). Abschließende Leerzeichen werden entsprechend *cbs* gesetzt und eine neue Zeile wird hinzugefügt, bevor die Zeile ausgegeben wird. Im zweiten Fall werden die ASCII Zeichen in den Umwandlungspuffer gelesen (wo Umwandlung zu EBCDIC erfolgt). Zur Bildung eines Ausgabeblocks der Größe *cbs* werden Leerzeichen hinzugefügt.

Nach Beendigung gibt *dd* die Anzahl der vollständigen und der unvollständigen Ein- und Ausgabeblöcke an.

DIAGNOSE

f+p blocks in(out) Anzahl der vollständigen und unvollständigen gelesenen (geschriebenen) Blöcke".

BEZEICHNUNG

deroff - entfernen der Formatierungsanweisungen von nroff/troff, tbl und eqn

ÜBERSICHT

deroff [**-mx**] [**-w**] [files]

BESCHREIBUNG

deroff liest jede der Dateien *files* der Reihe nach und entfernt alle *troff*(1) Anweisungen, Makroaufrufe, Backslash-Anweisungen, *eqn(1)* Anweisungen (zwischen **.EQ** und **.EN** Zeilen und zwischen Begrenzern) und *tbl*(1) Beschreibungen, und ersetzt sie möglicherweise durch Zwischenraumzeichen (Leerzeichen und leere Zeilen), und schreibt dann die restliche Datei auf die Standardausgabe. *deroff* berücksichtigt Ketten bei eingebundenen Dateien (**.so** und **.nx** *troff* Kommandos); wenn eine Datei bereits eingebunden wurde, wird ein **.so** für diese Datei ignoriert, und ein **.nx** für diese Datei beendet die Ausführung. Wenn keine Eingabedatei angegeben ist, liest *deroff* die Standardeingabe.

Auf die **-m** -Option kann ein **m, s** oder **l** folgen. Mit der **-mm** -Option werden Makros so interpretiert, daß nur der laufende Text (d. h. kein Text aus Makrozeilen) ausgegeben wird. Die **-ml** -Option stellt die **-mm** -Option ein und löscht zusätzlich Listen, die zu den **mm** -Makros gehören.

Bei Angabe der **-w** -Option besteht die Ausgabe aus einer Wortliste, die pro Zeile ein "Wort" enthält, während alle anderen Zeichen gelöscht wurden. Andernfalls entspricht die Ausgabe dem Original, und zwar mit den oben erwähnten Löschungen. Im Text ist ein "Wort" eine beliebige Zeichenfolge, die mindestens zwei Buchstaben enthält und aus Buchstaben, Ziffern, Und-Zeichen (&) und Apostroph (´) besteht; bei einem Makroaufruf ist ein "Wort" jedoch eine Zeichenfolge, die mit mindestens zwei Buchstaben *beginnt* und insgesamt mindestens drei Buchstaben enthält. Begrenzer sind beliebige Zeichen außer Buchstaben, Ziffern, Apostroph und Und-Zeichen. Abschließende Apostroph und Und-Zeichen werden aus dem "Wort" entfernt.

SIEHE AUCH:

eqn(1), nroff(1), tbl(1), troff(1) im Handbuch Dokumentations-Tools

FEHLER

deroff ist kein vollständiger *troff* Interpreter und kann möglicherweise bei sehr spezielle Anweisungen falsche Ausgaben erzeugen. Ee wird dann meistens mehr Ausgabe als erforderlich ist erzeugt.
Die **-ml** -Option bearbeitet verschachtelte Listen nicht korrekt.

BEZEICHNUNG

df – gibt Anzahl der freien Blöcke und I-Knoten-Einträge auf der Platte an

ÜBERSICHT

df [-lt] [-f] [*file-system* | *directory* | *mounted-resource*]

BESCHREIBUNG

Das Kommando **df** gibt die Anzahl freier Blöcke und I-Knoten-Einträge in eingehängten Dateisystemen, Verzeichnissen oder eingehängten Ressourcen an, indem die in den Superblöcken spezifizierten Zahlenangaben überprüft werden.

file-system kann entweder durch einen Gerätenamen (z. B./**dev/wd02**) oder durch den Namen des Verzeichnisses, daß die Einhängestelle ist (z. B./**usr**), angegeben werden.

directory kann der Name eines Verzeichnisses sein, wenn Information über das Gerät, daß dieses Verzeichnis enthält, ausgegeben werden soll.

mounted-resource kann der Name von Remote-Betriebsmitteln sein, wenn Information über das Gerät, daß die Ressource enthält, ausgegeben werden soll.

Wenn keine Argumente verwendet werden, wird der freie Speicherplatz in allen lokalen und entfernt eingehängten Dateisystemen ausgegeben.

Das Kommando **df** verwendet die folgenden Optionen:

-l gibt nur Informationen über lokale Dateisysteme aus.

-t bewirkt die Ausgabe der Zahlen für alle zugewiesenen Blöcke und I-Knoten-Einträge sowie die Anzahl der freien Blöcke und I-Knoten-Einträge.

-f Die aktuellen Blöcke in der freien Liste werden gezählt, anstatt die Zahlen dem Superblock zu entnehmen (freie I-Knoten-Einträge werden nicht angegeben). Diese Option gibt keine Daten über eingehängte Remote-Betriebsmittel aus.

ANMERKUNG

Wenn mehrere Remote-Betriebsmittel, die sich in dem gleichen Dateisystem in einem Remote-Gerät befinden, aufgelistet werden, werden alle Listen nach der ersten mit einem Stern versehen.

DATEIEN

/dev/wd/*, /etc/mnttab

SIEHE AUCH:

mount(1M).

fs(4), mnttab(4) im *Programmer's Reference Manual.*

BEZEICHNUNG

diff – Datei-Vergleich

ÜBERSICHT

diff [**-efbh**] file1 file2

BESCHREIBUNG

diff gibt die Zeilen an, die in zwei Dateien geändert werden müssen, um die Dateien identisch zu machen. Wenn *file1* (*file2*) – ist, wird die Standardeingabe verwendet. Wenn *file1* (*file2*) ein Verzeichnis ist, dann wird in diesem Verzeichnis eine Datei mit dem Namen *file2* (*file1*) verwendet. Die Ausgabe enthält normalerweise Zeilen in folgendem Format:

n1 **a** *n3,n4*
n1,n2 **d** *n3*
n1,n2 **c** *n3,n4*

Diese Zeilen sind den *ed* Kommandos ähnlich, um *file1* in *file2* umzuwandeln. Die Zahlen nach den Buchstaben betreffen *file2*. Durch Austausch von **a** durch **d** und durch Rückwärtslesen kann man feststellen, wie *file2* in *file1* umgewandelt wird. Wie bei *ed*, werden identische Paare, bei denen *n1* = *n2* oder *n3* = *n4* ist, durch eine einzige Zahl abgekürzt.

Bei Betrachtung jeder dieser Zeilen erscheinen erst alle Zeilen, die sich auf die erste Datei beziehen, gekennzeichnet durch < , dann alle Zeilen, die sich auf die zweite Datei beziehen und mit > gekennzeichnet sind.

Mit der **-b** -Option werden abschließende Leerzeichen (Leerzeichen und Tabulatoren) ignoriert und andere Folgen von Leerzeichen als gleich betrachtet.

Die **-e** -Option erstellt für den Editor *ed* ein Skript aus den Kommandos *a, c* und *d* das *file2* wieder aus *file1* erstellt. Die **-f** -Option erzeugt ein ähnliches Skript in umgekehrter Reihenfolge, das jedoch für *ed* nichts nützt. In Zusammenhang mit **-e** kann das folgende Shell-Programm mehrere Versionen einer Datei verwalten. Nur eine Vorgängerdatei ($1) und die Folge von mit *diff* erstellten Version-zu-Version-*ed*-Skripts wird benötigt. Die "letzte Version" erscheint auf der Standardausgabe.

```
(shift; cat $*; echo '1,$p') | ed - $1
```

Bis auf wenige Ausnahmen findet *diff* die kleinste notwendige Menge von Unterschieden zwischen den Dateien.

Die Option **-h** führt einen schnellen, nicht ganz so gründlichen Vergleich durch. Dies funktioniert nur bei Dateien mit verhältnismäßig kurzen und gut voneinander getrennten Unterschieden; kann jedoch Dateien mit unbegrenzter Länge bearbeiten. Die Optionen **-e** und **-f** stehen bei **-h** nicht zur Verfügung.

DATEIEN

/tmp/d?????
/usr/lib/diffh für **-h**

SIEHE AUCH

bdiff(1), cmp(1), comm(1), ed(1).

DIAGNOSE

Der Endestatus ist 0 bei Gleichheit, 1 bei Ungleichheit und 2 bei Problemfällen.

FEHLER

Die mit Hilfe der Optionen **-e** oder **-f** erstellten ed-Skripts, tendieren manchmal dazu, Zeilen mit einem einzigen Punkt (.) zu erstellen.

ACHTUNG!

Missing newline at end of file X
gibt an, daß die letzte Zeile der Datei X kein New-Line-Zeichen hatte. Wenn die Zeilen unterschiedlich sind, werden sie gekennzeichnet und ausgegeben. Die Ausgabe sieht jedoch so aus, als ob sie identisch wären.

BEZEICHNUNG

diff3 – Datei-Vergleich mit 3 Dateien

ÜBERSICHT

diff3 [**–ex3**] file1 file2 file3

BESCHREIBUNG

diff3 vergleicht drei Versionen einer Datei und gibt nicht übereinstimmende Textbereiche, wie folgt gekennzeichnet, an:

====	Alle drei Dateien sind unterschiedlich
====1	*file1* ist unterschiedlich
====2	*file2* ist unterschiedlich
====3	*file3* ist unterschiedlich

Die erforderlichen Änderungen, um einen angegebenen Bereich einer Datei in einen anderen Bereich oder eine Datei umzuwandeln, werden wie folgt angezeigt:

f : **n1** *a*	Text soll nach Zeilennummer *n1* in die Datei *f* angehängt werden, wobei *f* = 1, 2 oder 3 ist.
f : *n1* , *n2* **c**	Text soll im Bereich der Zeile *n1* bis Zeile *n2* geändert werden. Wenn *n1* = *n2* ist, wird abkürzend nur *n1* angegeben.

Der Originalinhalt des Bereichs folgt sofort nach einer **c** Angabe. Wenn der Inhalt von zwei Dateien identisch ist, wird der Inhalt der Datei mit der niedrigeren Nummer unterdrückt.

Bei der **–e** Option gibt *diff3* ein Skript für den Editor *ed* aus, das in *file1* alle Unterschiede zwischen *file2* und *file3* einbezieht, d. h. die Unterschiede, die normalerweise mit = = = = und ====3 gekennzeichnet sind. Die Option **–x** (**–3**) generiert ein Skript, wobei nur mit = = = = (====3) gekennzeichnete Unterschiede berücksichtigt werden. Durch die folgende Kommandofolge wird das erzeugte Skript dann auf *file1* angewandt.

(cat script ; echo ''1,$p'') | ed – file1

DATEIEN

/tmp/d3*, /usr/lib/diff3prog

SIEHE AUCH

diff(1).

FEHLER

Textzeilen, die aus einem einzelnen . bestehen, machen die Ausführung von **–e** unmöglich.

Dateien mit mehr als 64K Bytes können nicht bearbeitet werden.

BEZEICHNUNG

dircmp – Verzeichnisse vergleichen

ÜBERSICHT

dircmp [**-d**] [**-s**] [**-w***n*] dir1 dir2

BESCHREIBUNG

dircmp prüft *dir1* und *dir2* und erzeugt verschiedene tabellarisch erfaßte Informationen über den Inhalt der Verzeichnisse. Listen der Dateien, die nur in einem Verzeichnis vorkommen, werden bei allen Optionen erzeugt. Wenn keine Option angegeben ist, wird eine Liste ausgegeben, die die in beiden Verzeichnissen vorkommenden Dateinamen enthält und angibt, ob deren Inhalt identisch ist.

-d Vergleicht den Inhalt von Dateien mit gleichem Namen in beiden Verzeichnissen und gibt eine Liste mit Angaben zu Änderungen in beiden Dateien aus, damit sie identisch werden. Das Format der Liste wird in *diff(1)* beschrieben.

-s Unterdrückt Meldungen über identische Dateien.

-w*n* Ändert die Breite der Ausgabezeile auf *n* Zeichen. Die Standardbreite ist 72.

SIEHE AUCH:

cmp(1), diff(1).

BEZEICHNUNG

du – Übersicht über Plattenbelegung

ÜBERSICHT

du [**–sar**] [names]

BESCHREIBUNG

du ermittelt die Anzahl der Blöcke aller Dateien und (rekursiv) allen Verzeichnissen die in den durch das *names* Argument angegebenen Verzeichnissen und Dateien enthalten sind. Die Anzahl der Blöcke schließt die indirekten Blöcke der Datei ein. Wenn *names* fehlt, wird das aktuelle Verzeichnis verwendet.

Die wählbaren Argumente lauten

–s hat nur die Ausgabe der Gesamtsumme (für jede der mit *names* bezeichneten Dateien) zur Folge.

–a verursacht, daß eine Ausgabezeile für jede Datei erzeugt wird.

Wenn weder **–s** noch **–a** angegeben sind, wird nur für jedes Verzeichnis eine Ausgabezeile erzeugt.

–r bewirkt, daß *du* Meldungen ausgibt über Verzeichnisse, die nicht gelesen werden können bzw. von Dateien, die nicht geöffnet werden können, anstatt keine Angabe zu machen (dies ist die Voreinstellung).

Eine Datei mit zwei oder mehreren Verweisen wird nur einmal gezählt.

FEHLER

Wenn die **–a** Option nicht benutzt wird, werden nur Verzeichnis-Argumente aufgeführt.
Wenn Verweise zwischen Dateien in verschiedenen Verzeichnissen bestehen, wobei sich die Verzeichnisse in getrennten Zweigen des Dateisystem-Baumes befinden, zählt *du* die Dateien mehr als einmal.
Dateien mit Löchern erzeugen eine falsche Blockzählung.

BEZEICHNUNG

echo – Argumente ausgeben

ÜBERSICHT

echo [arg] ...

BESCHREIBUNG

echo schreibt seine Argumente, die durch Leerzeichen und durch ein Zeichen für neue Zeile abgeschlossen sind, auf die Standardausgabe. Es versteht auch Escape-Symbole ähnlich wie bei C; es ist darauf zu achten, daß diese Escape-Symbole nicht mit dem Gebrauch von \ bei der Shell in Konflikt kommen:

\b	Backspace
\c	Ausgabe der Zeile ohne New-Line-Zeichen
\f	Formularvorschub
\n	New-Line-Zeichen
\r	Wagenrücklauf
\t	Tabulator
\v	Vertikaltabulator
****	Backslash
\0*n*	wobei *n* das 8-Bit-Zeichen ist, dargestellt durch seinen ASCII-Code, der eine 1-, 2- oder 3-stellige Oktalzahl ist.

echo ist hilfreich für die Erstellung von Meldungen in Kommandodateien und für das Senden bekannter Daten an eine Pipe.

SIEHE AUCH:

sh(1).

EINSCHRÄNKUNGEN

Bei der Darstellung eines 8-Bit-Zeichens unter Verwendung der Escape-Symbol-Konvention **\0*n***, muß vor *n* **immer** die Ziffer Null (0) stehen.

Zum Beispiel wird durch **echo** ´**WARNING:\07**´ am Bildschirm **WARNING:** und der Piepston ausgegeben. Die Verwendung von einzelnen (oder doppelten) Anführungszeichen (oder zwei Backslashes) ist zum Schutz des "\" erforderlich, der vor der "07" steht.

Bezüglich der oktalen Codierung eines Zeichens können Sie sich in ascii(5), im *Programmer's Reference Manual* informieren.

BEZEICHNUNG

ed, red – Texteditor

ÜBERSICHT

ed [**-s**] [**-p** string] [**-x**] [file]

red [**-s**] [**-p** string] [**-x**] [file]

BESCHREIBUNG

ed ist der Standardtexteditor. Wenn das Argument *file* (Datei) angegeben wird, simuliert *ed* ein *e* Kommando (siehe unten) auf der angegebenen Datei; das heißt, die Datei wird in den Puffer von *ed* gelesen, so daß sie editiert werden kann.

-s Unterdrückt die Ausgabe der Zeichenanzahl bei den Kommandos *e*, *r* und *w* sowie die Ausgabe von Fehlermeldungen bei den Kommandos *e* und *q* und die Ausgabe des Eingabe-Aufforderungszeichens ! nach einem ! *shell command.*

-p erlaubt dem Benutzer die Angabe einer Eingabe-Aufforderungszeichenfolge.

-x Verschlüsselungsoption; wird diese Option benutzt, dann simuliert *ed* ein **X**-Kommando und verlangt vom Benutzer die Eingabe eines Schlüssels. Dieser Schlüssel wird benutzt, um den Text mit Hilfe des Algorithmuses *crypt (1))* zu ver- oder entschlüsseln. Das **X**-Kommando führt eine Abschätzung durch um zu erkennen, ob der gelesene Text entschlüsselt ist oder nicht. Der temporäre Datei-Puffer wird ebenfalls entschlüsselt, wobei eine übertragene Form des eingegebenen Schlüssels der **-x** -Option benutzt wird. (vgl. *crypt (1)).* Vgl. auch den Abschnitt **ACHTUNG!** am Ende dieser Beschreibung.

-C Verschlüsselungsoption; ähnlich wie die **-x** -Option außer, daß der *ed* ein **C**-Kommando benutzt. Das **C**-Kommando ist dem **X**-Kommando ähnlich bis auf die Tatsache, daß der gelesene Text als verschlüsselt gilt.

ed arbeitet auf einer Kopie der Datei, die editiert wird; Änderungen auf der Kopie treten erst in Kraft, wenn sie mit dem Kommando *w* (schreiben) in die Datei geschrieben wurden. Die Kopie des editierten Texts befindet sich in einer temporären Datei, die mit *buffer* (Puffer) bezeichnet wird. Es gibt nur einen Puffer.

red ist eine eingeschränkte Version von *ed*. Mit diesem Kommando können nur Dateien, die sich im aktuellen Verzeichnis befinden, editiert werden. Die Ausführung von Shell-Kommandos mit !*shell command* ist untersagt. Wenn versucht wird, diese Einschränkung zu umgehen, erfolgt die Ausgabe einer Fehlermeldung (*restricted shell*).

ed und *red* unterstützen beide die Formatierungsmöglichkeit von *fspec*(4). Bei Eingabe einer Formatangabe als erste Zeile von *file* und Aufruf von *ed* mit dem Bildschirm im **stty -tabs** oder **stty tab3** Modus (siehe *stty*(1)), werden die angegebenen Tabulatorpositionen automatisch beim Durchsuchen von *file* verwendet. Wenn zum Beispiel die erste Zeile einer Datei

<:t5,10,15 s72:>

enthält, würden die Tabulatoren in Spalte 5, 10 und 15 gesetzt und die max. Zeilenlänge wäre 72.
Anmerkung: Bei Texteingabe in eine Datei ist dieses Format nicht aktiv; dagegen werden, ist man im **stty -tabs** oder **stty tab3** Modus, die Tabulatorzeichen standardmäßig jede achte Spalte gesetzt.

Die Kommandos für *ed* haben eine einfache und regelmäßige Struktur: Keine, eine oder zwei Adressen, gefolgt von einem *command* (Kommando), das aus einem einzelnen Zeichen besteht. Dieses Kommando kann möglicherweise Parameter haben. Diese Adressen geben eine oder mehrere Zeilen im Puffer an. Jedes Kommando, das Adressen erfordert, hat Standardadressen, so daß die Adressen sehr oft ausgelassen werden können.

Im allgemeinen kann nur ein Kommando in einer Zeile erscheinen. Bestimmte Kommandos gestatten die Eingabe von Text. Dieser Text wird an die passende Stelle im Puffer geschrieben. Während *ed* den Text akzeptiert, befindet es sich im *input mode* (Eingabemodus). In diesem Modus werden:
no (keine) Kommandos erkannt; die Eingabe wird nur gesammelt. Man verläßt den Eingabemodus, indem am Anfang der Zeile ein Punkt (.) und gleich dahinter ein Wagenrücklauf eingegeben wird.

ed unterstützt eine eingeschränkte Form von *regular expression* (regulärer Ausdruck); reguläre Ausdrücke werden zur Angabe von Zeilen in Adressen verwendet und bei einigen Kommandos (z.B. *s*) zur Angabe von Teilen einer Zeile, die ersetzt werden sollen. Ein regulärer Ausdruck (RE) gibt eine Menge von Zeichenfolgen an. Man sagt, daß der RE die Elemente dieser Menge bezeichnet. REs, die bei *ed* erlaubt sind, werden wie folgt konstruiert:

Folgende Einzelzeichen bezeichnen jeweils ein einzelnes Zeichen:

1.1 Ein gewöhnliches Zeichen
(außer den in 1.2 (unten) beschriebenen) ist ein Einzelzeichen RE , der sich selbst bezeichnet.

1.2 Ein Backslash (\) , gefolgt von einem beliebigen Sonderzeichen, ist ein Einzelzeichen RE, das das Sonderzeichen bezeichnet. Die Sonderzeichen lauten wie folgt:

a. ., *, [, und \ (Punkt, Asterisk, linke eckige Klammer und Backslash). Dieses sind immer Sonderzeichen, es sei denn sie stehen in eckigen Klammern ([]; siehe 1.4 unten).

b. ^ (Zirkumflex), das eine besondere Bedeutung hat, wenn es am Anfang eines kompletten RE steht (vgl. 3.1 und 3.2 unten) oder direkt nach der linken Klammer eines []-Paares (vgl. 1.4 unten).

c. $ (Dollarzeichen), das eine besondere Bedeutung am Ende eines kompletten RE (vgl. 3.2 unten) hat.

d. Das Zeichen, das zum Begrenzen eines kompletten RE verwendet wird und speziell für diesen RE gilt (bitte beachten Sie zum Beispiel wie nachstehend der Slash (/) bei einem *g* Kommando benutzt wird.)

1.3 Ein Punkt (.) ist ein Einzelzeichen RE, der jedes Zeichen, außer dem New-Line-Zeichen bezeichnet.

1.4 Eine nichtleere Zeichenfolge, die in eckigen Klammern ([]) steht, ist ein Einzelzeichen RE, das jedes einzelne Zeichen in dieser Zeichenfolge bezeichnet. Wenn jedoch das erste Zeichen der Zeichenfolge ein Zirkumflex (^) ist, bezeichnet der RE alle Zeichen außer dem New-Line-Zeichen und den in der Zeichenfolge aufgeführten Zeichen. Das Zirkumflex ^ hat diese besondere Bedeutung nur wenn es am Anfang einer Zeichenfolge erscheint. Das Minuszeichen (-) kann zur Angabe eines Bereichs aufeinanderfolgender ASCII-Zeichen verwendet werden; so ist zum Beispiel **[0-9]** gleichwertig mit **[0123456789]**. Das Minuszeichen - verliert diese besondere Bedeutung, wenn es als erstes oder letztes Zeichen in der Zeichenfolge (nach einem ersten Zirkumflex ^, falls vorhanden) erscheint. Die rechte eckige Klammer (]) verliert ihre besondere Bedeutung (Abschluß der Zeichenfolge), wenn sie das erste Zeichen innerhalb der Zeichenfolge ist (nach einem erstem Zirkumflex ^, falls vorhanden); z.B. erkennt **[]a-f]** entweder eine rechte eckige Klammer (]) oder einen der Buchstaben **a** bis **f** einschließlich. Die vier oben im Abschnitt 1.2.a erwähnten Zeichen stehen innerhalb einer solchen Zeichenfolge für sich (keine besondere Bedeutung) .

Folgende Regeln können zum Aufbau von *REs* aus Einzelzeichen RE verwendet werden:

2.1 Ein Einzelzeichen RE ist ein RE, der die Zeichen bezeichnet, die der Einzelzeichen RE bezeichnet.

2.2 Ein Einzelzeichen RE gefolgt von einem Asterisk (*) ist ein RE, der kein oder mehrere Vorkommnisse des Einzelzeichen RE bezeichnet. Wenn mehrere Möglichkeiten bestehen, wird die längste Zeichenfolge von links, die eine Übereinstimmung ermöglicht, gewählt.

2.3 Ein Einzelzeichen RE, gefolgt von \{*m*\}, \{*m*,\}, oder \{*m*,*n*\} ist ein RE, der in Folge von Vorkommnissen des Einzelzeichen RE bezeichnet. Die Werte von *m* und *n* müssen nicht-negative ganze Zahlen und kleiner als 256 sein; \{*m*\} erkennt genau *m* Vorkommnisse; \{*m*,\} erkennt mindestens *m* Vorkommnisse; \{*m*,*n*\} erkennt eine beliebige Anzahl von Vorkommnissen zwischen *m* und *n* einschließlich. Immer wenn mehrere Möglichkeiten bestehen, erkennt RE so viele Vorkommnisse wie möglich.

2.4 Die Verkettung von REs ist ein RE, der die Verkettung der Zeichenfolgen bezeichnet, die zu den einzelnen Komponenten des RE passen.

2.5 Ein RE, der zwischen den Zeichenfolgen \(und \) steht, ist ein RE, der alles bezeichnet, was der nicht geklammerte RE bezeichnet.

2.6 Der Ausdruck *n* bezeichnet die gleiche Zeichenfolge, die zu einem Ausdruck zwischen \(und \) vorher im gleichen RE paßt. Hier ist *n* eine Ziffer; der angegebene Unterausdruck ist derjenige, der mit dem n-ten Auftreten von \(von links gerechnet beginnt. Zum Beispiel erkennt der Ausdruck ^\(.*\)\1$ eine Zeile, die aus zwei wiederholten Vorkommnissen der gleichen Zeichenfolge besteht.

Schließlich kann in einem kompletten *RE* angegeben werden, daß er nur den Anfang oder das Ende (oder beide) der Zeile betrifft.

3.1 Ein Zirkumflex (^) am Anfang eines kompletten RE beschränkt den RE auf den Anfang der Zeile.

3.2 Ein Dollarzeichen ($) am Ende eines ganzen RE beschränkt den RE auf das Ende der Zeile.

Die Konstruktion ^RE$ erfordert, daß der RE zu der ganzen Zeile paßt.

Der leere RE (d. h. //) ist gleichwertig mit dem letzten aufgetretenen RE. Vgl. den letzten Absatz vor *DATEIEN*.

Um die Adressierung mit *ed* zu verstehen, muß man wissen, daß es immer eine aktuelle Zeile gibt. Im allgemeinen ist die aktuelle Zeile die letzte Zeile, die durch ein Kommando betroffen war; die genaue Auswirkung auf die aktuelle Zeile wird bei der Beschreibung jedes Kommandos besprochen. Adressen werden wie folgt konstruiert:

1. Das Zeichen . adressiert die aktuelle Zeile.
2. Das Zeichen $ adressiert die letzte Zeile des Puffers.
3. Eine Dezimalzahl *n* adressiert die *n* -te Zeile des Puffers.
4. *'x* adressiert die Zeile, die mit der Marke *x* markiert ist, wobei x immer kleingeschrieben werden muß. Zeilen werden mit dem unten beschriebenen *k* Kommando markiert.
5. Ein RE zwischen Schrägstrichen (/) adressiert die erste Zeile, die beim Vorwärtssuchen, das bei der Zeile beginnt, die auf die aktuelle Zeile in Richtung Pufferende folgt, gefunden wird, und bei der ersten Zeile, die eine zu dem RE passende Zeichenfolge enthält, stoppt. Falls nötig läuft die Suche zyklisch über den Anfang des Puffers weiter bis zur (einschließlich) der aktuellen Zeile, so daß der ganze Puffer durchsucht wird. Vgl. den letzten Absatz vor *DATEIEN*.
6. Ein RE zwischen Fragezeichen (?) adressiert die erste Zeile, die beim Rückwärtssuchen, das bei der Zeile beginnt, die der aktuellen Zeile in Richtung Pufferanfang vorausgeht, gefunden wird, und bei der ersten Zeile, die eine zu dem RE passende Zeichenfolge enthält, stoppt. Falls nötig läuft die Suche zyklisch zum Ende des Puffers weiter bis zur (einschließlich) der aktuellen Zeile. Vgl. auch unten den letzten Absatz vor *DATEIEN*.
7. Eine Adresse, gefolgt von einem Pluszeichen (+) oder ein Minuszeichen (−), gefolgt von einer Dezimalzahl, gibt diese Adresse, plus oder minus der angezeigten Zeilenanzahl, an. Das Pluszeichen kann ausgelassen werden.
8. Wenn eine Adresse mit + oder − beginnt, wird die Addition oder Subtraktion in Hinblick auf die aktuelle Zeile verstanden; z.B. soll **−5 .−5** bedeuten.
9. Wenn eine Adresse mit + oder − endet, dann wird jeweils 1 hinzugefügt bzw. subtrahiert. Als Folge dieser Regel und der oben bei 8 angeführten Regel bezieht sich die Adresse − auf die Zeile, die der aktuellen Zeile vorausgeht. (Um mit früheren Versionen des Editors kompatibel zu bleiben, ist in Adressen das Zirkumflex-Zeichen ^ völlig gleichwertig mit −). Außerdem haben abschließende + und − Zeichen eine kumulative Wirkung, so daß − − sich auf die aktuelle Zeile weniger 2 bezieht.
10. Aus praktischen Gründen steht ein Komma (,) für das Adreßpaar **1,$**, während ein Semikolon (;) für das Paar **.,$** steht.

Kommandos können keine, eine oder zwei Adressen erfordern. Kommandos, die keine Adressen erfordern, sehen die Anwesenheit einer Adresse als einen Fehler an. Kommandos, die eine oder zwei Adressen akzeptieren, nehmen Standardadressen an, wenn eine unzureichende Anzahl von Adressen angegeben wurde; wenn mehr Adressen angegeben sind, als bei einem solchen Kommando erwartet wird, werden die letzte(n) benutzt.

Normalerweise werden Adressen mit einem Komma (,) voneinander getrennt. Sie können auch mit einem Semikolon (;) getrennt werden. Im letzten Fall wird die aktuelle Zeile (.) auf die erste Adresse gesetzt, und erst dann wird die zweite Adresse kalkuliert. Diese Einrichtung kann zur Bestimmung der Anfangszeilen für Vorwärts- und Rückwärtssuchen verwendet werden (vgl. Regeln 5 und 6 oben). Die zweite Adresse von zwei beliebigen Adreßfolgen muß mit einer Zeile übereinstimmen, die im Puffer der Zeile folgt, die mit der ersten Adresse übereinstimmt.

In der folgenden Liste von *ed* Kommandos werden die Standardadressen in runden Klammern gezeigt. Die runden Klammern sind nicht Teil der Adresse; sie zeigen nur an, daß die bestimmten Adressen voreingestellt sind.

Es ist gewöhnlich nicht zulässig, daß mehr als ein Kommando auf einer Zeile steht. Jedoch kann jedes Kommando (außer *e*, *f*, *r*, oder *w)* mit dem Zusatz **l**, **n** oder **p** angegeben werden. In diesem Fall wird die aktuelle Zeile entweder aufgelistet, numeriert oder ausgegeben, wie nachstehend bei den Kommandos *l*, *n*, und *p* erklärt wird.

(.)a

<text>

.

Das Kommando *a*ppend (anhängen) liest den angegebenen Text und hängt ihn hinter der adressierten Zeile an; . bleibt auf der letzten eingefügten Zeile, oder wenn keine vorhanden war, auf der letzten adressierten Zeile. Die Adresse 0 ist gültig für dieses Kommando: sie bewirkt, daß der "angehängte" Text an den Anfang des Puffers gestellt wird. Die maximale Anzahl von Zeichen, die pro Zeile von einem Terminal eingegeben werden können, ist 256 (einschließlich des New-Line-Zeichens).

(.)c

<text>

.

Das Änderungskommando *c*hange (ändern) löscht die adressierten Zeilen und akzeptiert dann Eingabetext, der diese Zeilen ersetzt; . bleibt auf der letzten Zeileneingabe, oder falls es keine gab, auf der ersten nicht gelöschten Zeile.

C

Wie das X-Kommando außer, daß *ed* den gesamten für die e- und r-Kommandos eingelesenen Text als verschlüsselt annimmt, bis daß ein Null-Schlüssel eingegeben ist.

(.,.)d

Das Löschkommando *d*elete (löschen) löscht die adressierten Zeilen aus dem Puffer. Die Zeile nach der letzten gelöschten Zeile wird die aktuelle Zeile; falls die gelöschten Zeilen sich ursprünglich am Ende des Puffers befanden, wird die letzte Zeile die aktuelle Zeile.

e *file*

Das Editierkommando *e*dit (editieren) bewirkt das Löschen des gesamten Inhalts des Puffers, und anschließend das Einlesen der angegebenen Datei; . wird auf die letzte Zeile des Puffers gesetzt. Wenn kein Dateiname angegeben ist, wird der momentan vermerkte Dateiname verwendet (vgl. *f* Kommando). Die Anzahl der gelesenen Zeichen wird angezeigt; die Datei mit dem Namen *file* wird vermerkt, da sie eventuell als Standard-Dateiname bei den nachfolgenden Kommandos *e*, *r*, und *w* verwendet wird. Wenn *file* durch ! ersetzt ist, wird die restliche Zeile als ein Shell-Kommando (*sh*(1)) angesehen, dessen Ausgabe gelesen werden soll. Ein solches Shell-Kommando wird nicht als der aktuelle Dateiname gemerkt. Vgl. nachstehend *DIAGNOSE*

E *file*

Das Editierkommando *E*dit ist dasselbe wie *e*, außer daß der Editor nicht überprüft, ob Änderungen im Puffer seit dem letzten Kommando *w* ausgeführt wurden.

f *file*

Wenn die Datei *file* angegeben ist, ändert das Kommando *f*ilename (Dateiname) den momentan gemerkten Dateinamen auf *file*; andernfalls wird der momentan gemerkte Dateiname ausgegeben.

(1,$)g/*RE***/***command list*

Bei dem Kommando *g*lobal wird zunächst jede Zeile, die zu dem RE paßt, markiert. Dann wird bei jeder dieser Zeilen die angegebene Kommandoliste *command list* ausgeführt, wobei . vor Ausführung auf diese Zeile gesetzt wird. Ein einzelnes Kommando oder das erste Kommando einer Kommandoliste erscheint auf der gleichen Zeile wie das globale Kommando. Alle Zeilen einer mehrzeiligen Liste mit Ausnahme der letzten Zeile müssen mit einem \ abschließen; *a*, *i*, und *c* Kommandos mit zugeordneter Eingabe sind erlaubt. Der ., der den Eingabemodus abschließt, kann

ausgelassen werden, falls es sich um die letzte Zeile der Kommandoliste handelt. Eine leere Kommandoliste ist gleichbedeutend mit dem Kommando *p*. Die Kommandos *g*, *G*, *v*, und *V* sind nicht in der *command list* zugelassen. Vgl. nachstehend *FEHLER* und den letzten Absatz vor *DATEIEN*.

(1,$)G/*RE***/**

Bei dem interaktiven Kommando *G*lobal wird zunächst jede Zeile, die zu dem gegebenen RE paßt, markiert. Dann wird für jede Zeile, die markierte Zeile angegeben, dann . auf diese Zeile gesetzt und jedes Kommando (mit Ausnahme von *a*, *c*, *i*, *g*, *G*, *v* und *V*) kann eingegeben werden und wird aufgeführt. Nach der Kommandoausführung wird die nächste markierte Zeile angezeigt, usw. Ein New-Line-Zeichen bedeutet ein Null-Kommando; ein & bewirkt die Wiederausführung des zuletzt im aktuellen Aufrufen von *G* aufgeführten Kommandos. Es ist zu beachten, daß die Kommandos, die während der Ausführung des Kommandos *G* eingegeben wurden, jede Zeile im Puffer adressieren und diese beeinflußen können. Das Kommando *G* kann von einem Unterbrechungssignal (ASCII DEL oder BREAK) beendet werden.

h

Das Kommando *h*elp (Hilfe) gibt eine kurze Fehlermeldung aus, die die Ursache der allerletzten ? Diagnose erklärt.

H

Das Kommando *H*elp stellt für *ed* einen Modus ein, bei dem Fehlermeldungen für alle nachfolgenden ? Diagnosen ausgegeben werden. Es erklärt auch das vorhergehende ?, falls eins ausgegeben wurde. Das Kommando *H* schaltet diesen Modus ein und aus, ursprünglich ist er ausgeschaltet.

(.)i
<text>
.

Das Einfügekommando *i*nsert (einfügen) fügt den angegebenen Text vor der adressierten Zeile ein; . bleibt auf der letzten eingefügten Zeile, oder, falls keine vorhanden ist, auf der adressierten Zeile. Dieses Kommando unterscheidet sich vom Kommando *a* nur in der Plazierung des eingegebenen Texts. Die Adresse 0 ist für dieses Kommando nicht gültig. Die Höchstanzahl von Zeichen, die von einem Terminal eingegeben werden können, ist 256 Zeichen pro Zeile (einschließlich des New-Line- Zeichens).

(.,.+1)j

Das Verbindungskommando *j*oin verbindet aufeinanderfolgende Zeilen durch Löschen der New-Line-Zeichen. Wenn nur eine Adresse angegeben ist, bleibt dieses Kommando untätig.

(.)k*x*

Das Markierungskommando mar*k* markiert die adressierte Zeile mit der Marke *x*; diese muß kleingeschrieben werden. Die Adresse *'x* adressiert dann diese Zeile; . bleibt unverändert.

(.,.)l

Das Kommando *l*ist (auflisten) gibt die adressierten Zeilen eindeutig aus: einige nicht abdruckbare Zeichen (z.B. *tab, backspace*) werden sichtbar dargestellt. Alle anderen nicht abdruckbaren Zeichen werden oktal angegeben, und lange Zeilen werden umgebrochen. Ein Kommando *l* kann an jedes andere Kommando außer *e, f, r* oder *w* angehängt werden.

(.,.)m*a*

Das Verlagerungskommando *m*ove (verlagern) verlagert die adressierte(n) Zeile(n) hinter die mit *a* adressierte Zeile. Die Adresse 0 ist bei *a* gültig und veranlaßt, daß die adressierte(n) Zeile(n) an den Dateianfang verlagert wird (werden). Ein Fehler ist eingetreten, falls die Adresse *a* in den Bereich der verlagerten Zeilen kommt; . bleibt auf der letzten verlagerten Zeile.

(.,.)n

Das Numerierungskommando *n*umber (numerieren) gibt vor jeder angezeigten, adressierten Zeile die Zeilennummer und ein Tabulatorzeichen an; . bleibt auf der letzten angezeigten Zeile. Das Kommando *n* kann an jedes andere Kommando außer *e, f, r* oder *w* angehängt werden.

(.,.)p

Das Druckkommando *p*rint (ausgeben) gibt die adressierten Zeilen aus; der . bleibt auf der letzten ausgegebenen Zeile. Das Kommando *p* kann an jedes andere Kommando außer *e, f, r* oder *w* angehängt werden. Zum Beispiel löscht *dp* die aktuelle Zeile und gibt die neue aktuelle Zeile aus.

P

Der Editor gibt bei allen nachfolgenden Kommandos das Eingabe-Aufforderungszeichen * aus. Das Kommando *P* kann diesen Modus ein- und ausschalten. Ursprünglich ist es ausgeschaltet.

q

Das Abbruchkommando *q*uit (beenden) veranlaßt die Beendigung den Abbruch von *ed*. Der Puffer wird nicht automatisch in die Datei geschrieben. Vgl. jedoch nachstehend *DIAGNOSE*.

Q

Der Editor bricht ohne vorherige Überprüfung, ob seit dem letzten Kommando *w* Änderungen im Puffer eingetreten sind, ab.

(**$**)**r** *file*

Das Lesekommando *r*ead (lesen) liest die angegebene Datei hinter der adressierten Zeile. Wenn kein Dateiname angegeben ist, wird der momentan gemerkte Dateiname, falls vorhanden, verwendet (vgl. Kommandos *e* und *f*). Der momentan gemerkte Dateiname wird nicht geändert, es sei denn, *file* ist der allererste Dateiname, der seit dem Aufruf von *ed* angegeben wird. Die Adresse 0 ist für *r* gültig und veranlaßt, daß die Datei am Anfang des Puffers gelesen wird. Wenn das Lesen erfolgreich ist, wird die Anzahl der gelesenen Zeichen angegeben; . wird auf die letzte eingelesene Zeile gestellt. Wenn *file* durch ! ersetzt ist, wird die restliche Zeile als ein Shell-Kommando (*sh*(1)) angesehen, dessen Ausgabe gelesen werden soll. Zum Beispiel hängt "$r !ls" das aktuelle Verzeichnis an das Ende der editierten Datei. Ein solches Shell-Kommando wird nicht als der aktuelle Dateiname gemerkt.

(.,.)**s**/*RE*/*replacement*/ oder
(.,.)**s**/*RE*/*replacement*/**g** **oder**
(.,.)**s**/*RE*/*replacement*/**n** n = 1-512

Das Substitutionskommando *substitute* (ersetzen) durchsucht jede adressierte Zeile nach einem Auftreten des angegebenen RE. Auf jeder Zeile, auf der eine Übereinstimmung gefunden wird, werden alle (sich nicht überschneidenden) passenden Zeichenfolgen durch die Substitution *replacement* ersetzt, wenn die globale Substitutionsoption **g** hinter dem Kommando steht. Wenn die globale Option nicht verwendet wird, wird nur das erste Auftreten der passenden Zeichenfolge ersetzt. Wenn hinter dem Kommando eine Zahl n erscheint, wird nur das *n-te* Auftreten der passenden Zeichenfolge auf jeder adressierten Zeile ersetzt. Für die Substitution ist es ein Fehler, wenn sie bei *all* (allen) adressierten Zeilen versagt. Als Begrenzungszeichen für RE und *replacement* können anstelle von / alle Zeichen außer dem Leerzeichen oder dem New-Line-Zeichen verwendet werden; . bleibt auf der letzten Zeile, auf der eine Substitution stattgefunden hat. Vgl. auch nachstehend den letzten Absatz vor *DATEIEN*.

Wenn ein Und-Zeichen (&) in der Substitution *replacement* auftritt, wird dieses Zeichen durch die Zeichenfolge ersetzt, die zu dem RE auf der aktuellen Zeile paßt. Diese besondere Bedeutung von & in diesem Zusammenhang kann durch ein vor diesem Zeichen stehenden \ unterdrückt werden. Ein etwas allgemeineres Leistungsmerkmal ist die Möglichkeit, daß die Zeichen *n* wenn *n* eine Ziffer ist, durch den Text ersetzt werden, der zu dem n-ten regulären Unterausdruck des RE paßt, der zwischen \(und \) steht. Wenn verschachtelte, in runden Klammern eingeschlossene Unterausdrücke vorhanden sind, wird *n* bestimmt, indem das

Auftreten von \(von links aus gezählt wird. Wenn das Zeichen % das einzige Zeichen in der Ersetzungszeichenfolge ist, wird die zuletzt im Ersetzungskommando verwendete Zeichenfolge *replacement* als *replacement* bei dem aktuellen Substitutionskommando verwendet. % verliert diese Sonderbedeutung, wenn in der Ersetzungszeichenfolge mehr als ein Zeichen vorkommen oder wenn ein \ davor steht.

Eine Zeile kann getrennt werden, indem ein New-Line-Zeichen in die Zeile gesetzt wird. Das New-Line-Zeichen muß durch eine vorangehendes \ Ersetzungszeichenfolge entwertet werden. Obige Substitution kann nicht als Teil einer Kommandoliste *g* oder *v* ausgeführt werden.

(.,.)t*a*

Dieses Kommando funktioniert genau wie das Kommando *m* , außer, daß eine Kopie *copy* der adressierten Zeile hinter die Adresse *a* (die 0 sein kann) geschrieben wird; . bleibt auf der letzten Zeile der Kopie.

u

Das Kommando *u*ndo (rückgängig machen) macht die Auswirkungen des allerletzten Kommandos rückgängig, das beliebige Änderungen im Puffer ausgeführt hat; hierbei handelt es sich um ein *a*, *c*, *d*, *g*, *i*, *j*, *m*, *r*, *s*, *t*, *v*, *G* oder *V* Kommando.

(1,$)v/*RE/command list*

Dieses Kommando ist mit dem globalen Kommando *g* identisch, außer daß vor Ausführung der Kommandoliste *command list* . nacheinander auf jede Zeile gesetzt wird, die nicht zu RE paßt.

(1,$)V/*RE/*

Dieses Kommando ist mit dem interaktiven globalen Kommando *G* identisch, außer daß die Zeilen, die anfangs markiert werden, die Zeilen sind, die nicht zu RE passen.

(1,$)w *file*

Das Schreibkommando *w*rite (schreiben) schreibt die adressierten Zeilen in die angegebene Datei. Wenn die Datei nicht vorhanden ist, wird sie mit dem Modus 666 erstellt (von allen lesbar und beschreibbar), sofern nicht Ihre Einstellung bei *umask* (vgl. *umask*(1)) dies anders bestimmt. Der aktuell gemerkte Dateiname wird nicht geändert, es sei denn, *file* ist der allererste Dateiname, der seit dem Aufruf von *ed* angegeben wurde. Wenn kein Dateiname angegeben ist, wird der aktuell gemerkte Dateiname, falls vorhanden, benutzt; (vgl. Kommandos *e* und *f)*; . bleibt unverändert. Wenn das Kommando erfolgreich ist, wird die Anzahl der geschriebenen Zeichen angegeben. Wenn *file* durch ! ersetzt wird,

wird die restliche Zeile als ein Shell-Kommando (*sh*(1)) angesehen, dessen Standardeingabe die adressierten Zeilen sind. Dieses Shell-Kommando wird nicht als der aktuelle Dateiname gemerkt.

x

Ein Codierschlüssel wird von der Standardeingabe angefordert. Nachfolgende Kommandos *e*, *r* und *w* werden diesen Schlüssel zum Verschlüsseln und Entschlüsseln von Text benutzen (vgl. *crypt*(1)). Eine speziell leer gelassener Schlüssel schaltet die Verschlüsselung aus. Vgl. auch die Option **-x** von *ed*.

(**$**)**=**

Die Zeilennummer der adressierten Zeile wird angegeben; . wird durch dieses Kommando nicht verändert.

!*shell command*

Die restliche Zeile nach **!** wird an die Shell des UNIX-Systems (*sh*(1)) zur Interpretation als Kommando geschickt. Ein nicht entwertetes Zeichen **%** im Text dieses Kommandos wird durch den gemerkten Dateinamen ersetzt; falls ein **!** als erstes Zeichen des Shell-Kommandos erscheint, wird es durch das vorhergehende Shell-Kommando ersetzt. **!!** wiederholt also das letzte Shell-Kommando. Wenn solche Erweiterungen vorgenommen werden, wird die erweiterte Zeile ausgegeben; . bleibt unverändert.

(**.+1**)<new-line>

Eine Adresse allein auf einer Zeile bewirkt, daß die adressierte Zeile ausgegeben wird. Ein New-Line-Zeichen ist gleichwertig mit **.+1p** ; dies ist praktisch beim Vorwärtslaufen durch den Puffer.

Falls ein Unterbrechungssignal (ASCII DEL oder BREAK) geschickt wird, druckt *ed* ein **?** und geht auf die Kommandoebene von *ed* zurück.

Grenzwerte für einige Größen: 512 Zeichen pro Zeile, 256 Zeichen pro globaler Kommandoliste und 64 Zeichen pro Dateiname. Die Grenzwerte bezüglich der Zeilenanzahl hängen ab von der Speicherkapazität, die dem Benutzer zur Verfügung steht. Jede Zeile beansprucht 1 Wort.

Beim Lesen einer Datei entfernt *ed* ASCII NUL-Zeichen.

Falls eine Datei nicht mit einem New-Line-Zeichen abgeschlossen ist, fügt *ed* ein New-Line-Zeichen hinzu und erklärt dies in einer Ausgabemeldung.

Wenn das abschließende Begrenzungszeichen eines RE oder einer Ersatzzeichenfolge (z.B. /) das letzte Zeichen vor einem New-Line-Zeichen ist, kann dieses Begrenzungszeichen ausgelassen werden, die adressierte Zeile wird dann ausgegeben.

Folgende Kommandopaare sind gleichwertig:

s/s1/s2	s/s1/s2/p
g/s1	g/s1/p
?s1	?s1?

DATEIEN

$TMPDIR
wenn diese Umgebungsvariable nicht Null ist, wird der Wert dieser Variablen anstelle von /usr/tmp als Verzeichnisname für die temporäre Arbeitsdatei verwendet.

/usr/tmp
Standard-Dateiverzeichnis für temporäre Arbeitsdatei.

/tmp Falls die Umgebungsvariable **TMPDIR** nicht existiert oder gleich Null ist und **/usr/tmp** nicht existiert, dann wird **/tmp** als Verzeichnisname der temporären Arbeitsdatei benutzt.

ed.hup
Arbeit wird hier gesichert, wenn das Terminal abgemeldet wurde.

BEMERKUNG

Die **-** Option wurde, obwohl sie weiterhin unterstützt wird, in der Dokumentation durch die **-s**-Option ersetzt, um dem Standard der Kommando-Syntax zu entsprechen (siehe *intro(1)*).

SIEHE AUCH

edit(1), ex(1), grep(1), sed(1), sh(1), stty(1), umask(1), vi(1).
fspec(4), regexp(5) im *Programmer's Reference Manual*.

DIAGNOSE

? bei Kommandofehlern.

?*filebei***einer***Datei,* **auf***die*
(für detaillierte Erläuterungen sind die Kommandos *h*elp und *H*elp zu benutzen).

Wenn im Puffer seit dem letzten Kommando *w* bei dem der ganze Puffer geschrieben wurde, Veränderungen eingetreten sind, wird der Benutzer von *ed* gewarnt, falls er versucht, den Puffer von *ed* über die Kommandos *e* oder *q* zu zerstören. Ein **?** wird angezeigt und macht es möglich, mit dem Editieren fortzufahren. Ein zweites Kommando *e* oder *q* an dieser Stelle wird effektiv. Die Option **-s** auf der Kommandozeile verhindert diese Funktion.

WARNUNG

Die Verschlüsselungsoptionen und -kommandos werden mit dem Security-Administration-Utilities-Package vertrieben, das nur in den Vereinigten Staaten erhältlich ist.

FEHLER

Ein Kommando *!* kann nicht bei *g* oder *v* Kommandos angegeben werden.

Das Kommando *!* und die Verwendung von **!** bei den Kommandos *e*, *r* und *w* kann nicht benutzt werden, wenn der Editor von einer eingeschränkten Shell (vgl. *sh*(1)) aufgerufen wird.

Die Folge **\n** in einem RE ist nicht identisch mit einem New-Line-Zeichen.

Wenn die Editoreingabe von einer Kommandodatei stammt (z.B.
ed file < ed-cmd-file),
bricht der Editor ab, sobald ein Kommando nicht erfolgreich war.

BEZEICHNUNG

edit – Texteditor (Variante von ex für gelegentliche Benutzer)

ÜBERSICHT

edit [**-r**] [**-x**] *name* ...

BESCHREIBUNG

edit ist eine Variante des Texteditors *ex*. Diese Funktion wird für neue oder gelegentliche Benutzer empfohlen, die einen kommandoorientierten Editor verwenden möchten. Er arbeitet genauso wie *ex(1)*, wobei die folgenden Variablen automatisch gesetzt sind:

novice	EIN
report	EIN
showmode	EIN
magic	AUS

Diese Optionen können über das **set** -Kommando in *ex*(1) ein- und ausgeschaltet werden.

-r Wiederherstellen einer Datei nach einem Editor- oder Systemabsturz.

-x Verschlüsselungsoption; wird diese Option benutzt, dann simuliert *ed* ein X-Kommando und verlangt vom Benutzer die Eingabe eines Schlüssels. Dieser Schlüssel wird benutzt, um den Text mit Hilfe des Algorithmuses *crypt (1))* zu ver- oder entschlüsseln. Das X-Kommando führt eine Abschätzung durch um zu erkennen, ob der gelesene Text entschlüsselt ist oder nicht. Der temporäre Datei-Puffer wird ebenfalls entschlüsselt, wobei eine übertragene Form des eingegebenen Schlüssels der -x -Option benutzt wird. (vgl. *crypt (1))*. Vgl. auch den Abschnitt **ACHTUNG!** am Ende dieser Beschreibung.

-C Verschlüsselungsoption; ähnlich wie die -x -Option außer, daß der *ed* ein C-Kommando benutzt. Das C-Kommando ist dem X-Kommando ähnlich bis auf die Tatsache, daß der gelesene Text als verschlüsselt gilt.

Die folgende kurze Einführung soll Ihnen beim erstmaligen Einsatz von *edit* behilflich sein. Wenn Sie ein Terminal benutzen, möchten Sie eventuell Näheres über den Bildschirm-Editor *vi* erfahren.

Zum Editieren des Inhalts einer existierenden Datei geben Sie zunächst das Kommando "edit name" auf Shell-Ebene ein. *edit* macht eine Kopie der Datei, die Sie dann editieren können, und Sie erfahren, wie viele Zeilen und Zeichen in der Datei sind. Zur Erstellung einer neuen Datei beginnen Sie ebenfalls mit dem Kommando *edit* und dem Dateinamen: *edit* Name. Der Editor teilt Ihnen dann mit, daß es sich um eine neue Datei `[New File]` handelt.

Mit dem Zeichen ':', das Sie nach dem Start des Editors sehen sollten, fordert *edit* die Eingabe von Kommandos an. Wenn Sie eine existierende Datei editieren, dann haben Sie einige Zeilen im Puffer von *edit* (der Puffer ist eine Kopie der von Ihnen editierten Datei). Wenn Sie mit dem Editieren beginnen, wird die letzte Zeile der Datei mit Hilfe von *edit* zur ersten aktuellen Zeile gemacht. Die meisten Kommandos für *edit* verwenden die "aktuelle Zeile", sofern Sie nicht angeben, welche Zeile zu benutzen ist. Wenn Sie **print** (kann auf **p** abgekürzt werden) eingeben und Carrige-Return betätigen (dies sollte nach allen *edit* Kommandos geschehen), wird diese aktuelle Zeile ausgegeben. Wenn Sie die aktuelle Zeile löschen (**delete(d)**) , gibt *edit* die neue aktuelle Zeile aus, welche normalerweise die nächste Zeile der Datei ist. Wenn Sie diese letzte Zeile löschen, wird die letzte neue Zeile die aktuelle Zeile.

Wenn Sie mit einer leeren Datei beginnen oder einige neue Zeilen hinzufügen möchten, kann das Kommando **append** (**a**) verwendet werden. Nachdem Sie dieses Kommando eingegeben haben (betätigen von Carrige-Return nach dem Wort **append**), wird *edit* Zeilen von Ihrem Bildschirm lesen, bis Sie eine Zeile eingeben, die nur aus einem "." besteht, und schreibt diese Zeilen hinter die aktuelle Zeile. Die letzte von Ihnen eingegebene Zeile wird die aktuelle Zeile. Das Kommando **insert** (**i**) arbeitet wie **append,** schreibt die eingegebenen Zeilen jedoch vor und nicht hinter die aktuelle Zeile.

edit numeriert die Zeile im Puffer, wobei die erste Zeile die Nummer 1 hat. Wenn Sie das Kommando "1" eingeben, gibt *edit* diese erste Zeile aus. Wenn Sie anschließend das Kommando **delete** eingeben, löscht *edit* die erste Zeile, Zeile 2 wird Zeile 1, und *edit* gibt die aktuelle Zeile (die neue Zeile 1) aus, so daß Sie genau verfolgen können, wo Sie im Text sind. Im allgemeinen ist die aktuelle Zeile immer die letzte Zeile, die durch ein Kommando betroffen war.

Sie können den Text der aktuellen Zeile ändern, indem Sie das Kommando **substitute** (**s**) verwenden. Sie geben "s/*old* /*new*/" ein, wobei bei *old* die alten Zeichen eingesetzt werden, die Sie austauschen wollen, und bei *new* die neuen Zeichen eingegeben werden, die Sie ersetzen wollen.

Das Kommando **file** (**f**) gibt an, wie viele Zeilen sich in dem Puffer befinden, den Sie editieren wollen, und gibt aus"[Modified]"(geändert), wenn Sie Änderungen vorgenommen haben. Nachdem Sie eine Datei geändert haben, können Sie mit dem Kommando **write** (**w**) den geänderten Puffertext schreiben, wodurch die alte Datei mit dem neuen Text überschrieben wird. Verlassen des Editors geschieht dann mit dem Kommando **quit** (**q**). Wenn Sie *edit* auf einer Datei ausführen, jedoch keine Änderungen vorgenommen haben, ist es nicht notwendig (kann aber nicht schaden), die Datei wieder in den Puffer zu schreiben (wri-

te). Wenn Sie *edit* verlassen wollen (quit), nachdem Sie Änderungen vorgenommen haben, ohne den Puffer rauszuschreiben, werden Sie mit dem Satz "No **write** since last change" (kein Schreiben seit der letzten Änderung) gewarnt und *edit* wartet auf das nächste Kommando. Wenn Sie den Puffer nicht in die Datei schreiben wollen, geben Sie das Kommando **quit** ein. Der Puffer geht dann verloren und Sie kehren zur Shell zurück.

Mit Hilfe der Kommandos **delete** und **append** und Eingabe der Zeilennummern, um bestimmte Zeilen in der Datei anzusprechen, können Sie beliebige Änderungen vornehmen. Sie sollten jedoch etwas genauer Bescheid wissen, wenn Sie *edit* häufiger benutzen wollen.

Mit dem Kommando **change** (c) wird die aktuelle Zeile in die von Ihnen eingebene Folge von Zeilen geändert (wie bei **append** geben Sie die Zeilen an bis einschließlich einer Zeile, die aus "." besteht). Sie können dem Kommando **change** mitteilen, daß es mehr als eine Zeile ändern soll, indem Sie die Zeilennummern der zu ändernden Zeilen eingeben, z.B. "3,5change". Genauso können Sie auch Zeilen ausgeben. "1,23p" gibt die ersten 23 Zeilen der Datei aus.

Das Kommando **undo** (u) macht die Auswirkung des letzten von Ihnen eingegeben Kommandos, das den Puffer geändert hat, rückgängig. So können Sie, wenn ein Kommando **substitute** nicht den gewünschten Effekt hat, mit dem Kommando **undo** den alten Inhalt der Zeile wiederherstellen. Sie können das Kommando **undo** mit **undo** wieder aufheben. *edit* gibt eine Warnung aus, wenn von Ihnen eingegebene Kommandos mehr als eine Zeile des Puffers betreffen. Es ist zu beachten, daß Kommandos wie *write* und *quit* nicht wieder rückgängig gemacht werden können.

Wenn Sie sich die nächste Zeile im Puffer ansehen wollen, betätigen Sie die Carrige-Return-Taste. Wenn Sie mehrere Zeilen sehen wollen, betätigen Sie anstelle von Carrige-Return ^D (Kontrolltaste und Taste D gleichzeitig betätigen und dann freigeben). Dadurch werden auf einem Terminal der halbe Bildschirm angezeigt oder 12 Zeilen auf einer Schreibstation ausgegeben. Sie können sich den Text davor anschauen, indem Sie das Kommando "z" eingeben. Die aktuelle Zeile wird dann als letzte Zeile ausgegeben; mit dem Kommando "´´" können Sie zu der Zeile zurückkehren, auf der Sie sich vor der Eingabe von "z" befanden. Das Kommando z kann mit folgenden Zeichen geschrieben werden: "z-" druckt eine Bildschirmseite (oder 24 Zeilen) und endet mit der aktuellen Zeile. Wenn Sie nicht die ganze Bildschirmseite anzeigen wollen, geben Sie "z.11" ein, um 5 Zeilen vor und 5 Zeilen nach der aktuellen Zeile zu erhalten. Allgemein werden bei der Angabe **z.***n* für *n* ungerade *n* Zeilen, für *n* gerade *n-1* Zeilen am Bildschirm angezeigt. Sie können hinter jedem Kommando eine Zeilenangabe machen, um

den Wirkungsbereich des Kommandos zu bestimmen. So können Sie mit dem Kommando "delete 5" 5 Zeilen löschen, wobei an der aktuellen Zeile begonnen wird.

Wenn Sie eine bestimmte Stelle in der Datei finden möchten, können Sie die Zeilennummern, sofern Sie diese kennen, verwenden; da die Zeilennummern sich ändern, wenn Sie Zeilen einfügen und löschen, ist diese Methode nicht sehr zuverlässig. In der Datei können Sie rückwärts und vorwärts nach Zeichenfolgen suchen, indem Sie das Kommando **/text/** zur Vorwärtssuche von *text* eingeben oder **?text?**, um rückwärts nach *text* zu suchen. Wenn beim Suchen das Dateiende erreicht wird, ohne daß der Text gefunden wird, läuft die Suche bis zur aktuellen Zeile zurück. Ein weiteres Leistungsmerkmal ist das Suchen mit dem Kommando **/^text/**, das nach *text* am Anfang einer Zeile sucht, während **/text$/** nach *text* am Ende einer Zeile sucht. Bei diesen Kommandos können abschließende / oder ? ausgelassen werden.

Die aktuelle Zeile hat einen symbolischen Namen "."; dies ist zur Angabe eines Bereichs wie bei ".,$p" praktisch, durch das die restlichen Zeilen in der Datei ausgegeben werden. Die letzte Zeile der Datei können Sie mit ihrem symbolischen Namen "$" bezeichnen. So löscht das Kommando "$d" die letzte Zeile in der Datei, ganz gleich welche Zeile vorher die aktuelle Zeile war. Es ist auch möglich, Arithmetik in den Zeilen-Referenzen zu benutzen. So ist die Zeile "$-5" die fünfte vor der letzten Zeile, und ".+20" bedeutet 20 Zeilen nach der aktuellen Zeile.

Sie können feststellen, auf welcher Zeile Sie sich befinden, indem Sie ".=" eingeben. Dieses Kommando wird speziell für das Verlagern oder Kopieren eines Textabschnitts innerhalb einer Datei oder von einer Datei zu einer anderen empfohlen. Zunächst stellen Sie die erste und letzte Zeilennummer, die Sie kopieren oder verschieben (z.B. 10 bis 20) wollen, fest. Bei einer Verlagerung geben Sie dann "10,20delete a" ein, woraufhin diese Zeilen von der Datei gelöscht und in einen Puffer mit der Bezeichnung *a* geschrieben werden. *edit* hat 26 solcher Puffer, die von *a* bis *z* gehen. Zu einem späteren Zeitpunkt können Sie diese Zeilen mit "put a" zurückholen, wodurch der Inhalt des Puffers *a* hinter die aktuelle Zeile geschrieben wird. Wenn Sie diese Zeilen von einer Datei in eine andere verschieben oder kopieren wollen, geben Sie ein **edit** (e) Kommando nach dem Kopieren der Zeilen, gefolgt vom Namen der zu editierenen Datei ein wie z.B. "edit chapter2". Um Zeilen zu kopieren, ohne sie zu löschen, benutzen Sie *yank* anstatt *delete*. Wenn der Text, den Sie verschieben oder kopieren wollen, in der gleichen Datei ist, geben Sie zum Beispiel nur "10,20m $" ein. In diesem Fall ist es nicht erforderlich, benannte Puffer zu verwenden.

SIEHE AUCH

ed(1), ex(1), vi(1).

ACHTUNG!

Die Verschlüsselungsoptionen werden mit dem Security-Administration-Utilities-Package vertrieben, das nur in den Vereinigten Staaten erhältlich ist.

BEZEICHNUNG

egrep – Eine Datei nach einem Muster durchsuchen, unter Verwendung von vollständigen regulären Ausdrücken

ÜBERSICHT **egrep** [options] full regular expression [file ...]

BESCHREIBUNG

egrep (*expression grep*) durchsucht Dateien nach Zeichenmustern und gibt alle Zeilen aus, die dieses Muster enthalten. Zur Mustererkennung verwendet *egrep* vollständige reguläre Ausdrücke (Ausdrücke mit Zeichenfolgen, die alle alphanumerischen Zeichen und Sonderzeichen verwenden). Es wird ein schneller deterministischer Algorithmus verwendet, der manchmal exponentiell Platz benötigt.

egrep akzeptiert reguläre Ausdrücke wie in *ed*(1), mit Ausnahme von \(und \), wobei folgende Formen zusätzlich benutzt werden können:

1. Ein vollständiger regulärer Ausdruck, gefolgt von +, steht für ein oder mehrere Vorkommnisse des regulären Ausdrucks.
2. Ein vollständiger regulärer Ausdruck, gefolgt von ?, steht für kein oder 1 Vorkommnis des vollständigen regulären Ausdrucks.
3. Vollständige reguläre Ausdrücke, getrennt durch | oder durch ein New-Line-Zeichen, bezeichnen Zeichenfolgen, die zu einem der Ausdrücke passen.
4. Ein vollständiger regulärer Ausdruck kann, um Unterausdrücke zusammenzufassen, in runde Klammern () eingeschlossen werden.

Mit der Verwendung der Zeichen $, *, [, ^, |, (,), und \, in regulären Ausdrücken sollte man vorsichtig umgehen, da sie auch eine Bedeutung für die Shell haben. Es ist am sichersten, das gesamte Argument *full regular expression* in einfache Anführungszeichen einzuschließen ´ ... ´.

Die Prioritätsreihenfolge der Operatoren ist [ ], *?+, Verkettung, | und New-Line-Zeichen.

Wenn keine Dateien angegeben werden, nimmt *egrep* die Standardeingabe an. Normalerweise wird jede gefundene Zeile auf die Standardausgabe kopiert. Der Dateiname wird vor jeder gefundenen Zeile angegeben, wenn es sich um mehr als eine Eingabedatei handelt.

Die Optionen der Kommandozeile lauten:

-b Jeder Zeile wird die Block-Nummer, mit der sie gefunden wurde, vorangestellt. Diese Option wird beim Kontext-Suchen von Block-Nummern im Kontext empfohlen (erster Block ist O).
-c Gibt nur die Anzahl der Zeilen an, die das Muster enthalten.
-i Ignoriert während des Vergleichs Groß-/Kleinschreibung

-l Gibt die Namen der Dateien mit übereinstimmenden Zeilen einmal aus, wobei diese durch New-Line-Zeichen getrennt sind. Wiederholt nur einmal die Namen der Dateien, wenn das Muster mehrmals gefunden wurde.

-n Jeder Zeile wird die entsprechende Zeilennummer in der Datei vorangestellt (die erste Zeile ist 1).

-v Gibt alle Zeilen aus außer denen, die das Muster enthalten.

-e *special_expression*
Sucht nach einem *special expression* (*full regular expression*, der mit einem - beginnt).

-f *file* Entnimmt der Datei *file* die Liste der *full regular expressions.*

SIEHE AUCH

ed(1), fgrep(1), grep(1), sed(1), sh(1).

DIAGNOSE

Endestatus ist 0, wenn Übereinstimmungen gefunden wurden, 1 wenn keine gefunden wurden, 2 für Syntaxfehler (selbst wenn Übereinstimmungen gefunden wurden) oder nicht zugreifbare Dateien.

FEHLER

Am besten wäre nur ein Kommando *grep,* jedoch gibt es leider keinen einzelnen Algorithmus, der einen ausreichenden Bereich abdeckt, ohne zu große Platz-/Zeit-Verluste. Zeilen werden auf BUFSIZ-Zeichen begrenzt; längere Zeilen werden abgeschnitten. BUFSIZ ist in **/usr/include/stdio.h** definiert.

BEZEICHNUNG

enable, disable – Drucker einschalten/abschalten

ÜBERSICHT

enable printers
disable [**-c**] [**-r**[reason]] printers

BESCHREIBUNG

enable aktiviert die durch *printers* angegebenen Drucker und ermöglicht den Ausdruck der von *lp(1)* angenommenen Aufträge. Es wird die Verwendung von *lpstat*(1) zur Feststellung des Druckerstatus empfohlen.

disable deaktiviert die durch *printers* angegebenen Drucker und verhindert so den Ausdruck der von *lp*(1) angenommenen Aufträge. Standardmäßig werden Aufträge, die gerade auf dem angegebenen Drucker ausgedruckt werden, entweder vollständig auf dem gleichen Drucker oder auf einem anderen Drucker gleichen Typs von Neuem ausgedruckt. Es wird empfohlen, *lpstat*(1) zur Feststellung des Druckerstatus zu benutzen. Folgende Optionen sind bei *disable* möglich:

-c Abbrechen aller Aufträge, die gerade auf einem der angegebenen Drucker ausgedruckt werden.

-r[*reason*] Verbindet einen Grund *reason* mit dem Abschalten der Drucker. Dieser Grund gilt für alle erwähnten Drucker bis zur nächsten Option **-r** . Wenn die Option **-r** nicht vorhanden ist oder die Option **-r** ohne Argument *reason* angegeben wird, wird eine Standard-Meldung (*reason*) verwendet. *reason* wird mit *lpstat*(1) angezeigt.

DATEIEN

/usr/Spool/lp/ *

SIEHE AUCH:

lp(1), lpstat(1).

BEZEICHNUNG

env – Umgebung für Kommandoausführung setzen

ÜBERSICHT

env [–]. [name =value] ... [command args]

BESCHREIBUNG

env bestimmt die aktuelle Umgebung, modifiziert diese entsprechend den angegebenen Argumenten und führt dann das Kommando mit der modifizierten Umgebung aus. Argumente in der Form *name* =*value* werden in die Umgebung, die vererbt wird, eingebracht, ehe das Kommando ausgeführt wird. Die Option – bewirkt, daß die gewöhnlich vererbte Umgebung vollkommen ignoriert wird, damit das Kommando mit der Umgebung ausgeführt wird, die durch die Argumente angegeben wird.

Wenn kein Kommando angegeben ist, wird die resultierende Umgebung mit einem Name-Wert-Paar pro Zeile angezeigt.

SIEHE AUCH:

sh(1).

exec(2), profile(4), environ(5) im *Programmer's Reference Manual.*

BEZEICHNUNG

ex – Texteditor

ÜBERSICHT

ex [**-s**] [**-v**] [**-t** *tag*] [**-r file**] [**-L**] [**-R**] [**-x**] [**-C**] [**-c** *command*] *file ...*

BESCHREIBUNG

ex ist die Wurzel einer Editorenfamilie: *ex* und *vi*. *ex* ist ein übergeordnetes Kommando von *ed*, dessen bemerkenswerteste Erweiterung der Bildschirm-Editor ist. Das Hauptmerkmal von *vi* ist das Editieren am Bildschirm.

Wenn Sie ein Terminal haben, möchten Sie wahrscheinlich einen Bildschirm-Editor verwenden; in diesem Fall vgl. *vi* (1), da dieses Kommando speziell auf Editieren am Bildschirm mit *ex* zugeschnitten ist.

Für Benutzer des Kommandos ed

Wenn Sie *ed* schon benutzt haben, werden Sie feststellen, daß zusätzlich zu allen *ed(1)* - Kommandos *ex* eine Anzahl zusätzlicher Leistungsmerkmale besitzt, die bei CRT-Terminals praktisch sind. Intelligente und Hochgeschwindigkeitsterminals eignen sich außerordentlich gut für den Einsatz von *vi*. Im allgemeinen nutzt der Editor sowohl das Leistungsvermögen der CRT-Terminals in weit größerem Umfang als *ed* als auch das Leistungsvermögen der Terminal-Datenbasis (vgl. *Anleitungen zu Dienstprogrammen für Bildschirmdaten*) und ersieht den Terminal-Typ aus der Variablen TERM in der Umgebung, um zu bestimmen, wie Ihr Bildschirm leistungsfähig eingesetzt werden kann. Der Editor verwendet Leistungsmerkmale wie zum Beispiel Zeichen und Zeile einfügen und löschen mit dem **visual** (visuellen) Kommando (wird mit **vi** abgekürzt), das der Haupteditiermodus bei *vi* (1) ist.

ex enthält eine Anzahl neuer Leistungsmerkmale zur bequemen Textanzeige von Dateien. Das Kommando **z** erleichtert Zugriff auf Textfenster. Die Eingabe von ^D bewirkt, daß der Editor den Text um ein halbes Fenster weiterrollt und wird bei schnellem Durchsuchen einer Datei anstelle einer Betätigung der Eingabetaste empfohlen. Natürlich ermöglicht es der bildschirmorientierte **visual** (visuelle) Modus, fortlaufend in die Editiervorgänge einzugreifen.

ex bietet Ihnen mehr Hilfe, wenn Sie Fehler machen. Das Kommando **undo** (**u**) ermöglicht Ihnen, jede einzelne falsche Änderung rückgängig zu machen. *ex* gibt Ihnen viel Feedback; normalerweise werden geänderte Zeilen ausgegeben; wenn mehr Zeilen von einem Kommando betroffen sind, wird dies angezeigt, so daß leicht festzustellen ist, wenn durch ein Kommando mehr Zeilen als erwartet betroffen wurden.

Der Editor verhindert normalerweise auch Überschreiben von vorhandenen Dateien, wenn Sie zufällig durch ein *write* eine andere als die ge-

rade editierte Datei beschreiben würden. Wenn das System (oder Editor) plötzlich nicht funktioniert oder Sie legen zufällig den Telefonhörer auf, können Sie mit dem Editor-Kommando **recover** (oder der *-r-file*-Option) Ihre Arbeit wiederfinden. Sie können in etwa an der Stelle wieder aufsetzen, an der zuvor abgebrochen wurde.

ex bietet mehrere Leistungsmerkmale um mehrere Dateien gleichzeitig zu bearbeiten. Sie können eine Dateiliste auf der Kommandozeile eingeben und dann mit dem Kommando **next** (**n**) jede Datei entsprechend bearbeiten. Bei dem **next**- Kommando kann auch eine Liste von Dateinamen oder ein Dateinamen-Muster wie bei der Shell angegeben werden. Im allgemeinen können Dateinamen im Editor mit der vollen Shell-Metasyntax gebildet werden. Das Metazeichen '%' steht auch bei der Bildung von Dateinamen zur Verfügung und wird durch den Namen der aktuellen Datei ersetzt.

Zum Austauschen von Text zwischen Dateien und innerhalb einer Datei besitzt der Editor eine Reihe von Puffern, die mit *a* bis *z* bezeichnet werden. Sie können Text in die angegebenen Puffer schreiben und den Text beim Editieren einer anderen Datei in diese Datei übernehmen.

Der Editor verfügt über eine Gruppe von Puffern, die mit kleinen Buchstaben (**a** - **z**). Text kann in diese benamten Puffer geschrieben werden, um ihn an anderer Stelle im Text einzufügen. Der Inhalt der Puffer bleibt erhalten, wenn Sie eine neue Datei mit dem **edit** (**e**)-Kommando editieren.

Das Kommando **&** wiederholt das letzte **substitute** -Kommando. Außerdem gibt es ein Ersatzkommando, das Bestätigung verlangt. Sie geben einen Bereich von auszuführenden Substitutionen ein, worauf der Editor Sie bei jeder Substitution fragt, ob diese gewünscht ist.

Beim Suchen und Ersetzen kann Groß-/Kleinschreibung ignoriert werden. *ex* gestattet auch die Konstruktion von regulären Ausdrücken, um Worte zu suchen. Dies ist zum Beispiel praktisch, wenn Sie das Wort "edit" suchen, wenn in Ihrem Text auch das Wort "editor" vorkommt.

Bei *ex* stehen eine Reihe von Optionen *options* zur Verfügung, die Sie nach Ihren eigenen Wünschen einsetzen können. Eine sehr nützliche Option ist *autoindent*, die dem Editor gestattet, automatisch führende Leerzeichen zum Ausrichten eines Textes einzufügen. Sie können dann zur bequemen Ausrichtung von neuen Texten die **^D**-Taste verwenden für Tabulator vorwärts, rückwärts und Leerzeichen.

Verschiedene neue nützliche Leistungsmerkmale schließen ein intelligentes Kommando **join** (**j**) ein, das automatisch Zwischenraumzeichen zwischen zusammengeschlossenen Zeilen liefert, die Kommandos < und > zur Verschiebung von Zeilengruppen und die Möglichkeit, Teile des Pufferinhalts durch Programme wie *sort* zu „filtern".

AUFRUFOPTIONEN

Folgende Aufrufoptionen werden von *ex* interpretiert (Optionen, die früher dokumentiert wurden, werden im **NOTES**-Abschnitt am Ende der Manual-Seite besprochen):

-s Unterdrückt interaktives Feedback des Benutzers. Dies ist nützlich bei der Verarbeitung von Editorskripten.

-v Ruft *vi* auf

-t *tag* Editiert die Datei, die das *tag* enthält und positioniert den Editor auf die Definition des *tag*.

-r *file* Editieren von *file* nach einem Editor- oder System-Absturz. (Es wird die Version wiederhergestellt, die im Puffer war, als der Absturz auftrat.)

-L Auflisten aller Dateien, die gesichert wurden, als der Editor- oder System-Absturz auftrat.

-R Setzt **readonly**-Modus; das **readonly**-Flag wird gesetzt, der zufälliges Überschreiben der Datei verhindert.

-x Verschlüsselungsoption; wird diese Option benutzt, dann simuliert *ex* ein **X**-Kommando und verlangt vom Benutzer die Eingabe eines Schlüssels. Dieser Schlüssel wird benutzt, um mit dem *crypt(1)*-Algorithmus den Text zu ver- oder entschlüsseln. Das **X**-Kommando untersucht, ob der gelesene Text verschlüsselt ist oder nicht. Die temporäre Pufferdatei wird ebenfalls entschlüsselt, wobei eine übertragene Version des eingegebenen Schlüssels für die **x**-Option benutzt wird. (vgl. *crypt (1)*). Siehe auch den Abschnitt **ACHTUNG!** am Ende dieser Beschreibung.

-C Verschlüsselungsoption; ähnlich wie die **-x** -Option außer, daß *ex* ein **C**-Kommando simuliert. Das **C**-Kommando ist dem **X**-Kommando ähnlich bis auf die Tatsache, daß der gelesene Text als verschlüsselt gilt.

-c command Bei Start des Editors das angegebene Kommando *command* (meist ein Such- oder Positionierungs-Kommando) ausführen.

Das Argument *file* gibt die zu editierenden Dateien an.

Modi von ex

Kommando	Normaler und Anfangs-Status. Eingabe wird mit : angefordert. Ihr Abbruchzeichen löscht Teilkommandos.
Einfügen	Einschalten mit **a**, **i** oder **c**. Beliebiger Text kann eingegeben werden. Einfügen wird normalerweise beendet, wenn eine Zeile nur . enthält, oder mit einer Unterbrechung bei Fehlerzustand.
Öffnen/Visuell	Visuell Einschalten mit **vi**, beenden mit **Q** oder ^\.

Kommandonamen und Abkürzungen von ex

abbrev	**ab**	map	**n**	set	**se**
append	**a**	mark	**ma**	shell	**sh**
args	**ar**	move	**m**	source	**so**
change	**c**	next	**n**	substitute	**s**
copy	**co**	number	**nu**	unabbrev	**una**
delete	**d**	preserve	**pre**	undo	**u**
edit	**e**	print	**p**	unmap	**unm**
file	**f**	put	**pu**	version	**ve**
global	**g**	quit	**q**	visual	**vi**
insert	**i**	read	**re**	write	**w**
join	**j**	recover	**rec**	xit	**x**
list	**l**	rewind	**rew**	yank	**ya**

ex-Kommandos

Shell Maskierung	**!**
Erzwungene Verschlüsselung	**C**
Heuristische Verschlüsselung	**X**
Linksshift	**<**
Schreib nächste Zeile	CR
Zurücksubstituieren	**&**
Rechtsshift	**>**
Scroll	**^D**
Window	**z**

Kommandoadressen von ex

n	Zeile *n*	*/pat*	nächste Zeile vorwärts, die *pat* enthält
.	aktuelle Zeile	*?pat*	letzte Zeile zurück, die *pat* enthält
$	letzte Zeile	*x-n*	*n* vor *x*
+	nächste Zeile	*x,y*	*x* bis *y*
–	vorige Zeile	*'x*	Markiert mit *x*
+n	*n* vorwärts	*''*	voriger Kontext
%	1,$		

Initialisierungsoptionen

EXINIT	**set**-Anweisungen für Umgebungsvariable setzen.
$HOME/.exrc	Editor-Initialisierungsdatei
./.exrc	Editor-Initialisierungsdatei
set *x*	Option einschalten
set no*x*	Option ausschalten
set *x*=*val*t	Wert *val* x zuweisen
set	Alle geänderten Optionen anzeigen
set all	Alle Optionen anzeigen
set *x*?	Wert der Option *x* anzeigen

Häufig verwendete Optionen und ihre Abkürzungen

autoindent	ai	Einrücken
autowrite	aw	Datei sichern, bevor sie geändert wird
directory		Pfadname der temporären Arbeitsdatei
ignorecase	ic	Groß-/Kleinschrift beim Suchen ignorieren
list		^für Tabulator, $ am Ende ausgeben
magic		. [* Sonderzeichen in Mustern
modelines		ersten 5 und letzten 5 Zeilen ausführen als **vi**/**ex** Kommandos, falls sie in der Form **ex:***command*: oder **vi:***command*: sind.
number	nu	Zeilen numerieren
paragraphs	para	Makronamen, die Paragraphen einleiten.
redraw		Intelligente Terminals simulieren
report		informiert Sie, falls die Anzahl der Zeilen, die durch das letzte Kommando geändert wurden, größer ist, als der Wert der **report**-Variablen.
scroll		Kommandomodus-Zeilen
sections	sect	Makronamen, die Abschnitte einleiten
shiftwidth	sw	Für < >, und ^D-Eingabe
showmatch	sm	Bis zu) und } wie angegeben
showmode	smd	Einfügemodus in *vi* anzeigen
slowopen	slow	Änderung während Einfügen stoppen

term		übergibt an den **vi** den Terminal-Typ (standardmäßig wird dieser Wert über die **TERM**-Variable übergeben).
window		Textzeilen
wrapmargin	wm	Automatische Zeilentrennung
wrapscan	ws	ungefähr am Pufferende?

Bildung der Suchmuster

^	Zeilenanfang
$	Zeilenende
.	Beliebiges Zeichen
\<	Wortanfang
\>	Wortende
[*str*]	Beliebiges Zeichen aus *str*
[^*str*]	... nicht aus *str*
[*x*–*y*]	... zwischen *x* und *y*
*	Beliebige Wiederholung des vorausgehenden

AUTOR

vi und *ex* basieren auf Software, die von der University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science entwickelt wurde.

DATEIEN

/usr/lib/ex?.?strings	Fehlermeldungen
/usr/lib/ex?.?recover	Kommando wiederherstellen
/usr/lib/ex?.?preserve	Kommando reservieren
/usr/lib/*/*	Beschreiben der Leistungsmerkmale der Terminals
$HOME/.exrc	Editor-Startdatei
./.exrc	Editor-Startdatei
/tmp/Ex*nnnnn*	Editor temporär
/tmp/R*xnnnnn*	Ausgegebener Puffer temporär
/usr/preserve/*login*	Geschütztes Verzeichnis (wobei *login* das Login des Benutzers ist)

NOTES

Verschiedenen Optionen wurden, obwohl sie weiterhin unterstützt werden, in der Dokumentation durch Optionen ersetzt, die dem Standard der Kommandobeschreibungssyntax entsprechen (see *intro(1)*). Die – Option wurde durch **-s**, eine **-r**-Option ohne ein weiteres Argument wurde durch 7-**L** und +*command* wurde durch **-c** *command* ersetzt.

SIEHE AUCH:

crypt(1), ed(1), edit(1), grep(1), sed(1), sort(1), vi(1).
curses(3X) im *Programmer's Reference Manual*, term(4), terminfo(4) im System *Administrator's Reference Manual*,
User's Guide.
Das curses/terminfo Kapitel des Programmer's Guide.

ACHTUNG!

Die Verschlüsselungsoptionen und -kommandos werden mit dem Security-Administration-Utilities-Package vertrieben, das nur in den Vereinigten Staaten erhältlich ist.

FEHLER

Das Kommando *z* gibt logische Zeilen aus anstelle der physikalischen Zeilen. Bei langen Zeilen kann die Ausgabe eventuell über einen Bildschirm gehen.

Datei-Ein-/Ausgabefehler geben keinen Namen an, wenn die Option `-s` auf der Kommandozeile verwendet wird.

Es gibt keine einfache Methode, wenn die Groß-Kleinschreibung ignoriert werden soll.

Der Editor warnt nicht, falls Text in Puffer geschrieben wird und nicht vor Verlassen des Editors benutzt wird.

Nullzeichen werden in Eingabedateien gelöscht und können nicht in den Ausgabedateien erscheinen.

BEZEICHNUNG

expr – Argumente als Ausdruck auswerten

ÜBERSICHT

expr arguments

BESCHREIBUNG

Die Argumente werden als ein Ausdruck behandelt. Nach der Auswertung wird das Ergebnis auf die Standardausgabe geschrieben. Teile des Ausdrucks müssen durch Leerzeichen getrennt werden. Vor Zeichen, die bei der Shell verwendet werden, muß ein Entwertungszeichen stehen. Es ist zu beachten, daß **0** nicht die leere Zeichenfolge, sondern den Wert Null anzeigt. Zeichenfolgen, die Leerzeichen oder andere Sonderzeichen enthalten, sollten in Anführungszeichen stehen. Ein einstelliges Minuszeichen kann vor ganzzahligen Argumenten stehen. Intern werden ganze Zahlen als 32-Bit, Zweierkomplement-Zahlen behandelt.

Die Operatoren und Kennworte sind unten aufgeführt. Vor Zeichen, die entwertet werden müssen, steht \. Die Liste ist nach zunehmender Priorität geordnet; Operatoren mit gleicher Priorität sind innerhalb { } zusammengefaßt.

expr \| *expr*
liefert den ersten Ausdruck *expr*, wenn er weder die leere Zeichenfolge noch **0** ist, sonst wird der zweite *expr* ausgegeben.

expr \& *expr*
liefert den ersten *expr* , wenn kein *expr* die leere Zeichenfolge oder **0** ist, sonst wird **0** zurückgegeben.

expr { =, \>, \> =, \<, \< =, != } *expr*
gibt das Ergebnis eines ganzzahligen Vergleichs zurück, wenn beide Argumente ganze Zahlen sind, sonst wird das Ergebnis eines lexikalischen Vergleichs geliefert.

expr { +, – } *expr*
Addition oder Subtraktion der als ganze Zahlen bewerteten Argumente.

expr { *, /, % } *expr*
Multiplikation, Division oder ganzzahlige Division der als ganze Zahlen bewerteten Argumente.

expr : *expr*

Der Mustervergleichs-Operator : vergleicht das erste Argument mit dem zweiten Argument, das ein regulärer Ausdruck sein muß. Die Syntax für reguläre Ausdrücke ist genau wie bei *ed*(1) mit der Ausnahme, daß alle Textmuster am Anfang vorkommen müssen (d. h. mit einem ^ beginnen); ^ ist also hier kein Sonderzeichen. Normalerweise gibt der Mustervergleichs-Operator die Anzahl der übereinstimmenden Zeichen zurück (**0** im Fehlerfall). Alternativ dazu können die Symbole \(...\) verwendet werden, um einen Teil des ersten Arguments zurückzugeben.

BEISPIELE

1. a=`expr $a + 1`

   addiert 1 zur Shell-Variablen **a**.

2. Wenn $a entweder "/usr/abc/file" oder "file" ist, liefert
   expr $a : '.*/\(.*\)' \| $a

   das letzte Segment des Pfadnamens (d. h. file). Vorsicht bei einem alleinstehenden / als Argument: *expr* erkennt dieses Zeichen als Divisionsoperator (vgl. FEHLER unten).

3. Eine bessere Darstellung des zweiten Beispiels.
   expr //$a : '.*/\(.*\)'

   Hinzufügung von // eliminiert Zweideutigkeit hinsichtlich des Divisionsoperators und vereinfacht den gesamten Ausdruck.

4. expr $VAR : '.*'

   liefert die Anzahl der Zeichen in **$VAR**.

SIEHE AUCH

ed(1), sh(1).

DIAGNOSE

expr gibt die nachstehenden Rückgabewerte zurück:

0	falls der Ausdruck weder die leere Zeichenfolge noch **0** ist
1	falls der Ausdruck die leere Zeichenfolge oder **0** *ist*
2	bei ungültigen Ausdrücken.

syntax error	bei Operator/Operand Fehlern
non-numeric argument	wenn bei einer solchen Zeichenfolge Arithmetik durchgeführt werden soll

FEHLER

Nach der Verarbeitung der Argumente durch die Shell kann *expr* den Unterschied zwischen einem Operator und einem Operanden nur am Wert erkennen. Wenn **$a** ein = ist, sieht das Kommando:

```
expr $a = '='
```

so:

```
expr = = =
```

aus, wenn die Argumente an *expr* übergeben werden (und dann als = Operator angesehen werden). Die korrekt arbeitende Anweisung ist:

```
expr X$a = X=
```

BEZEICHNUNG

factor – die Primfaktoren einer Zahl ausgeben

ÜBERSICHT

factor [integer]

BESCHREIBUNG

Wenn Sie das Kommando *factor* ohne ein Argument verwenden, wird die Eingabe einer ganzen Zahl erwartet. Nachdem Sie eine positive ganze Zahl, die kleiner oder gleich 10^{14} ist, angegeben haben, wird die ganze Zahl in Faktoren zerlegt und die Primfaktoren werden angezeigt. Anschließend wird eine neue Eingabe erwartet. *factor* hört auf, wenn eine Null oder nicht numerischen Zeichen eingegeben werden.

Wenn Sie *factor* mit einem Argument aufrufen, wird – wie oben beschrieben – die ganze Zahl in Faktoren zerlegt und dann das Programm beendet.

factor benötigt maximal eine Zeit proportional zu $\sqrt{n}$. *factor* benötigt diese Zeit, wenn n eine Primzahl oder das Quadrat einer Primzahl ist.

DIAGNOSE

factor druckt die Fehlermeldung „Ouch!" , wenn die Eingabe nicht im zulässigen Bereich liegt oder inkorrekt ist.

BEZEICHNUNG

fgrep – eine Zeichenfolge in einer Datei suchen

ÜBERSICHT

fgrep [options] string [file ...]

BESCHREIBUNG

fgrep (schnelles *grep* sucht eine Zeichenfolge in einer Datei und gibt alle Zeilen aus, die diese Zeichenfolge enthalten. *fgrep* unterscheidet sich von *grep(1)* und *egrep(1)*, da es eine Zeichenfolge sucht anstelle eines Textmusters, das zu einem Ausdruck paßt. Es verwendet einen schnellen und kompakten Algorithmus.

Die Zeichen $, *, [, ^, |, (,), und \ werden von *fgrep* wörtlich interpretiert, d. h. *fgrep* erkennt keine vollen regulären Ausdrücke wie *egrep*. Da diese Zeichen besondere Bedeutung bei der Shell haben, wird empfohlen, die ganze Zeichenfolge *string* in einfache Anführungszeichen zu setzen ´ ... ´.

Wenn keine Dateien angegeben werden, nimmt *fgrep* die Standardeingabe an. Normalerweise wird jede gefundene Zeile auf die Standardausgabe kopiert. Der Dateiname wird vor jeder gefundenen Zeile angezeigt, wenn mehrere Eingabedateien vorhanden sind.

Die Optionen der Kommandozeile lauten:

- **-b** Vor jede Zeile die Block-Nummer stellen, in der sie gefunden wurde. Dies kann bei der Suche von Blocknummern (erster Block ist 0) im Kontext nützlich sein
- **-c** Nur Anzahl der Zeilen drucken, die das Muster enthalten
- **-i** Groß-/Kleinschreibung bei Vergleich ignorieren.
- **-l** Dateinamen von Dateien mit übereinstimmenden Zeilen einmal ausgeben, getrennt durch New-Line-Zeichen. Dateinamen werden nicht wiederholt, wenn das Textmuster häufiger gefunden wird.
- **-n** Vor jede Zeile ihre Zeilennummer in der Datei schreiben (erste Zeile ist 1).
- **-v** Alle Zeilen außer den Zeilen, die das Muster enthalten, ausgeben
- **-x** Nur identische Zeilen ausgeben.
- **-e** *special_string*
  special string suchen (*string* beginnt mit einem –).
- **-f** *file*
  Aus der Datei *file* die Liste der zu suchenden Zeichenfolgen *string* entnehmen

SIEHE AUCH:

ed(1)egrep(1) grep(1) sed(1) sh(1)

DIAGNOSE

Endestatus ist 0, wenn Übereinstimmungen gefunden werden, 1 wenn es keine Übereinstimmungen gibt, 2 für Syntaxfehler oder nicht zugreifbare Dateien (selbst wenn Übereinstimmungen gefunden wurden).

FEHLER

Am besten wäre nur ein Kommando *grep*, jedoch gibt es leider keinen einzelnen Algorithmus, der einen ausreichenden Bereich abdeckt, ohne zu große Platz-/Zeit-Verluste. Zeilen sind auf BUFSIZ-Zeichen begrenzt; längere Zeilen werden abgeschnitten. BUFSIZ ist in **/usr/include/stdio.h.** definiert.

BEZEICHNUNG

file – Dateiart bestimmen

ÜBERSICHT

file [-c] [-f ffile] [-m mfile] arg ...

BESCHREIBUNG

file überprüft jedes Argument, um es klassifizieren zu können. Wenn ein Argument ASCII Code enthält, untersucht *file* zur Sprachfeststellung die ersten 512 Bytes. Wenn ein Argument ein ablauffähiges **a.out** ist, gibt das Kommando *file* die Versionsnummer aus, vorausgesetzt sie ist größer als 0.

-c Die Option -c veranlaßt das Kommando *file* , die "magic" Datei auf Formatfehler zu prüfen. Diese Validierung wird normalerweise aus Effizienzgründen nicht vorgenommen. Bei der Option -c werden keine Dateien ausgegeben.

-f Bei der Option **-f** wird das nächste Argument als eine Datei angesehen, die die Namen der zu überprüfenden Dateien enthält.

-m Die Option **-m** gibt dem Kommando *file* die Anweisung, eine andere "magic" Datei zu verwenden.

file verwendet die Datei **/etc/magic** zur Identifizierung von Dateien, die eine Art *magic number* haben, d. h. Dateien, die eine numerische oder Zeichenfolgekonstante enthalten, die den Typ angibt. Kommentar am Anfang von **/etc/magic** erklärt das Format.

DATEIEN

/etc/magic

SIEHE AUCH:

filehdr(4) im *Programmer's Reference Manual*

BEZEICHNUNG

find – Dateien suchen

ÜBERSICHT

find path-name-list expression

BESCHREIBUNG

find durchläuft die Verzeichnis-Hierarchie rekursiv für jeden Pfadnamen in der *path-name-list* (d. h. ein oder mehrere Pfadnamen), um die Dateien zu suchen, die einem booleschen Ausdruck *expression* entsprechen, der aus den nachstehenden Elementarausdrücken aufgebaut ist. In den Beschreibungen wird das Argument *n* als eine dezimale ganze Zahl verwendet, wobei *+n* mehr als *n*, *–n* weniger als *n* und *n* genau *n* bedeutet. Gültige Ausdrücke sind:

–name *file* — Wahr, wenn das Argument *file* mit dem aktuellen Dateinamen übereinstimmt. Die normale Shell-Argumente-Syntax kann verwendet werden, wenn, falls nötig, durch \ entwertet wird (Vorsicht bei [, ? und *).

[–perm] *–onum* — Wahr, wenn die Zugriffs-Bits genau mit der Oktalzahl *onum* übereinstimmen (vgl. *chmod*(1)). Wenn *onum* mit einem Minuszeichen beginnt, werden nur die Bits, die in *onum* gesetzt sind, mit den Zugriffs-Bits verglichen, und der Ausdruck wird wahr, wenn sie übereinstimmen

–type *c* — Wahr, wenn der Typ der Datei *c* ist, wobei *c* die Werte **b, c, d, p** oder **f** annehmen kann für blockorientierte Datei, zeichenorientierte Gerätedatei, Verzeichnis, fifo (benannte Pipe) oder einfache Datei.

–links *n* — Wahr, wenn die Datei *n* Verweise hat.

–user *uname* — Wahr, wenn die Datei dem Benutzer *uname* gehört. Wenn *uname* numerisch ist und nicht als ein Login-Namen in der Datei **/etc/passwd** erscheint, wird der Ausdruck als eine Benutzernummer angesehen.

–group *gname* — Wahr, wenn die Datei der Gruppe *gname* gehört. Wenn *gname* numerisch ist und nicht in der Datei **/etc/group** erscheint, wird der Ausdruck als eine Gruppennummer angesehen.

–size *n*[**c**] — Wahr, wenn die Datei *n* Blöcke groß ist (512 Bytes pro Block). Wenn auf *n* ein **c** folgt, ist die Anzahl der Zeichen gemeint.

-atime *n*	Wahr, wenn Zugriff auf die Datei innerhalb *n* Tagen erfolgte. Die Zugriffszeit der Verzeichnisse in *pathname-list* wird von *find* selbst geändert.
-mtime *n*	Wahr, wenn die Datei innerhalb *n* Tagen modifiziert wurde.
-ctime *n*	Wahr, wenn die Datei innerhalb *n* Tagen geändert wurde.
-exec *cmd*	Wahr, wenn das ausgeführte Kommando *cmd* Null als Endestatus liefert. Das Ende des Kommandos *cmd* muß durch ein quotiertes Semikolon gekennzeichnet sein. Das Kommando-Argument { } wird durch den aktuellen Pfadnamen ersetzt.
-ok *cmd*	Wie **-exec**, außer daß die erzeugte Kommandozeile zuerst mit einem Fragezeichen ausgegeben wird und nur ausgeführt wird, wenn der Benutzer y eingibt.
-print	Immer wahr; veranlaßt die Ausgabe des aktuellen Pfadnamens.
-cpio *device*	Immer wahr; schreibt die aktuelle Datei auf *device* in *cpio* (1) Format (5120-Byte-Datensätze)
-newer *file*	Wahr, wenn die aktuelle Datei später als das Argument *file* modifiziert wurde.
-depth	Immer wahr; die Datenverzeichnis-Hierarchie wird durchlaufen und zwar so, daß alle Einträge des Verzeichnisses vor dem Verzeichnis selbst durchlaufen werden. Dieses Kommando wird empfohlen, wenn *find* zusammen mit *cpio*(1) zur Übertragung von Dateien verwendet wird, die in Verzeichnissen stehen, für die keine Schreibberechtigung besteht.
-mount	Immer wahr; beschränkt die Suche auf das Dateisystem, das das bezeichnete Verzeichnis enthält, oder wenn kein Verzeichnis angegeben wurde, auf das aktuelle Verzeichnis.
-local	Wahr, wenn die Datei sich physikalisch auf dem lokalen System befindet.
(*Ausdruck*)	Wahr, wenn der in Klammern stehende Ausdruck wahr ist (runde Klammern sind Spezialsymbole der Shell und müssen quotiert werden).

Die Elementarausdrücke können durch Benutzung der folgenden Operatoren (nach abnehmender Priorität aufgelistet) miteinander kombiniert werden:

1) Die Negation eines Elementarausdrucks (! ist der einstellige Operator *not*).
2) Verkettung von Elementarausdrücken (die Operation *and* wird durch die Nebeneinanderstellung von zwei Elementarausdrücken implizit dargestellt).
3) Alternative von Elementarausdrücken (**-o** ist der Operator *or*).

BEISPIEL

Entfernung aller Dateien mit Namen **a.out** oder ***.o**, auf die eine Woche lang kein Zugriff erfolgte.

```
find / \( -name a.out -o -name '*.o' \) -atime +7 -exec rm {} \;
```

DATEIEN

/etc/passwd, /etc/group

SIEHE AUCH

chmod(1), cpio(1), sh(1), test(1).
stat(2), umask(2), fs(4) im *Programmer's Reference Manual*.

FEHLER

find/-depth mißlingt immer mit der Meldung: "find: stat failed: : No such file or directory".

BEZEICHNUNG

ftp – ARPANET-Dateiübertragungsprogramm

ÜBERSICHT

ftp [**-v**] [**-d**] [**-i**] [**-n**] [**-g**] [*host*]

BESCHREIBUNG

Ftp ist die Benutzerschnittstelle für das ARPANET Standard File Transfer Protocol. Das Programm ermöglicht die Dateiübertragung von und zu einem entfernten (remote) Netzwerkrechner.

Der Client-Host, mit dem *ftp* kommunizieren soll, kann in der Kommandozeile angegeben werden. In diesem Fall versucht *ftp* sofort, eine Verbindung mit einem FTP-Server auf diesem Host herzustellen. Andernfalls startet *ftp* seinen Kommando-Interpreter und wartet auf Anweisungen des Benutzers. Wenn *ftp* auf Kommandoeingaben wartet, zeigt es dies dem Benutzer durch die Eingabe-Aufforderung `ftp>` an. *Ftp* unterstützt die folgenden Kommandos:

! [*command* [*args*]]
: Auf dem lokalen Rechner wird eine interaktive Shell gestartet. Falls angegeben, wird das erste Argument als direkt auszuführendes Kommando aufgefaßt. Die übrigen Argumente werden als Argumente dieses Kommandos interpretiert.

$ *macro-name* [*args*]
: Das Makro *macro-name*, das mit dem Kommando **macdef** definiert wurde, wird ausgeführt. Die Argumente werden unverändert übergeben.

account [*passwd*]
: Sobald eine Anmeldung in einem entfernten Rechner erfolgreich abgeschlossen wurde, wird das zusätzliche Paßwort *passwd* bei Bedarf an diesen Rechner übergeben. Wenn kein Argument angegeben wird, fordert *ftp* den Benutzer zur Eingabe des Konto-Paßworts auf. Das eingegebene Paßwort wird nicht angezeigt.

append *local-file* [*remote-file*]
: Die lokale Datei wird an eine Datei in dem entfernten Rechner angehängt. Wenn *remote-file* fehlt, erhält die entfernte Datei den Namen der lokalen Datei (nach Umformung gemäß den *ntrans*- bzw. *nmap*-Regeln). *Ftp* benutzt dabei die momentan gültigen Werte von *type*, *form*, *mode* und *structure*.

ascii Der Dateiübertragungstyp (*type*) wird auf Netzwerk-ASCII-Übertragung eingestellt. Dies ist der Standardtyp. - Das Kommando ist in dieser Form nicht verfügbar. Bitte benutzen Sie das Kommando "type ascii".

bell Nach der Ausführung eines Dateiübertragungskommandos soll ein akustisches Signal ausgegeben werden.

binary
Der Dateiübertragungstyp (*type*) wird auf binäre Übertragung eingestellt. - Das Kommando ist in dieser Form nicht verfügbar. Bitte benutzen Sie das Kommando "type binary".

bye Die FTP-Sitzung mit dem entfernten Server wird beendet, desgleichen *ftp* selbst. Ein Dateiendezeichen bewirkt das gleiche.

case Die Umwandlung von Dateinamen entfernter Rechner bei **mget**-Kommandos wird ein- bzw. ausgeschaltet (Standard: ausgeschaltet). Wenn *case* eingeschaltet ist, werden die Dateinamen entfernter Rechner, sofern sie nur aus Großbuchstaben bestehen, im lokalen Verzeichnis in Kleinbuchstaben geschrieben.

cd *remote-directory*
remote-directory wird zum Arbeitsverzeichnis auf dem entfernten Rechner gemacht.

cdup Das Elternverzeichnis des aktuellen Arbeitsverzeichnisses auf dem entfernten Rechner wird zum neuen Arbeitsverzeichnis gemacht.

close *Ftp* beendet die Sitzung mit dem entfernten Server und kehrt in die Ebene des Kommando-Interpreters zurück. Alle definierten Makros werden gelöscht.

cr Das Ausblenden des Carriage-Return-Zeichens beim Einlesen von ASCII-Dateien wird ein- bzw. ausgeschaltet (Standard: eingeschaltet). Bei der Übertragung von ASCII-Dateien wird das Ende eines Satzes an einer Zeichenfolge erkannt, die aus einem Carriage-Return-Zeichen und einem New-Line-Zeichen besteht. Wenn *cr* eingeschaltet ist, wird das Carriage-Return-Zeichen aus diesen Zeichenfolgen ausgeblendet. Dies ist notwendig, weil das UNIX-Betriebssystem voraussetzt, daß ein Satz durch ein einzelnes New-Line-Zeichen abgeschlossen wird. Auf entfernten Rechnern, die nicht unter einem UNIX-Betriebssystem laufen, können Dateien *einzelne* New-Line-Zeichen enthalten. Bei der Übertragung von ASCII-Dateien können diese New-Line-Zeichen nur dann von einem Satzende-Kennzeichen unterschieden werden, wenn *cr* ausgeschaltet ist.

delete *remote-file*
Die Datei *remote-file* wird auf dem entfernten Rechner gelöscht.

debug [*debug-value*]
Der Testhilfemodus wird ein- bzw. ausgeschaltet. Das optionale Argument *debug-value* bezeichnet die Testhilfestufe, die einge-

schaltet werden soll. Wenn der Testhilfemodus aktiv ist, gibt *ftp* jedes an den entfernten Rechner gesendete Kommando aus. Einer solchen Zeile geht die Zeichenfolge --> voran.

dir [*remote-directory*] [*local-file*]
Es wird eine Liste des Verzeichnisses *remote-directory* ausgegeben, und falls *local-file* angegeben ist, in dieser lokalen Datei gespeichert. Fehlt die Verzeichnisangabe, wird das aktuelle Arbeitsverzeichnis in dem entfernten Rechner benutzt. Wenn keine lokale Datei angegeben wird oder - als *local-file*, geht die Ausgabe auf das Terminal.

disconnect
Gleichbedeutend mit **close**.

form *format*
Das Dateiübertragungsformat wird auf *format* gesetzt. Das Standardformat ist file (Datei).

get *remote-file* [*local-file*]
Die entfernte Datei *remote-file* wird eingelesen und auf dem lokalen Rechner gespeichert. Wenn *local-file* fehlt, erhält die lokale Datei den Namen der entfernten Datei (nach Umformung gemäß den *case*–, *ntrans*– bzw. *nmap*–Regeln). *Ftp* benutzt für die Dateiübertragung die momentan gültigen Werte von *type*, *form*, *mode* und *structure*.

glob Die Dateinamen-Expansion wird bei den Kommandos **mdelete**, **mget** und **mput** ein- bzw. ausgeschaltet. Wenn die Dateinamen-Expansion mit **glob** ausgeschaltet wird, werden Dateinamen-Argumente nicht expandiert und bleiben daher unverändert. Bei dem Kommando **mput** werden Dateinamen nach dem Muster von **csh**(1) expandiert. Bei den Kommandos **mdelete** und **mget** wird jeder Name einer entfernten Datei auf dem entfernten Rechner getrennt expandiert. Die dabei entstehenden Listen von Dateinamen werden nicht zu einer Liste zusammengefaßt. Die Expansion von Verzeichnisnamen unterscheidet sich möglicherweise von der Expansion eines normalen Dateinamens: das genaue Ergebnis hängt von dem fremden Betriebssystem und dem FTP-Server ab. Mit Hilfe des Kommandos '**mls** *remote-files* -' können Sie sich aber das Ergebnis einer Expansion vorher ansehen. Hinweis: Mit **mget** und **mput** kann nicht ein ganzer Teilbaum eines Verzeichnisses mit allen Unterverzeichnissen und Dateien übertragen werden. Wenn Sie dies wollen, müssen Sie mit **tar**(1) eine Sicherungskopie eines solchen Teilbaums erstellen und diese im Binärmodus übertragen.

hash Das Hash-Zeichen ("#") wird ein- bzw. ausgeschaltet. Wenn es eingeschaltet ist, wird nach der Übertragung jedes Blocks (1024 Byte) das Hash-Zeichen ausgegeben.

help [*command*]
Es werden Hilfeinformationen über die Funktion des Kommandos *command* ausgegeben. Wenn kein Argument angegeben wird, zeigt *ftp* eine Liste der erlaubten Kommandos an.

lcd [*directory*]
Das Arbeitsverzeichnis auf dem lokalen Rechner wird gewechselt. Wenn *directory* fehlt, wird in das Home-Verzeichnis gewechselt.

ls [*remote-directory*] [*local-file*]
Es wird eine kurze Liste eines Verzeichnisses des entfernten Rechners ausgegeben. Wenn *remote-directory* nicht angegeben ist, wird das aktuelle Arbeitsverzeichnis benutzt. Wenn keine lokale Datei angegeben wird oder - als *local-file*, geht die Ausgabe auf das Terminal.

macdef *macro-name*
Mit diesem Kommando wird die Definition eines Makros eingeleitet. Alle folgenden Zeilen werden unter dem Makronamen *macro-name* abgespeichert. Eine Nullzeile beendet die Makrodefinition. Sie besteht aus zwei aufeinanderfolgenden New-Line-Zeichen (wenn die Eingabe aus einer Datei kommt) oder aus zwei Carriage-Return-Zeichen (bei Eingabe vom Terminal). Es sind maximal 16 Makros erlaubt und höchstens 4096 Zeichen in allen Makros zusammen. Makrodefinitionen bleiben bis zu einem **close**-Kommando erhalten. Die Zeichen '$' und '\' haben für den Makroprozessor eine besondere Bedeutung. Ein Dollarzeichen '$', dem ein oder mehrere Ziffern folgen, wird durch das entsprechende Argument im Makroaufruf ersetzt. Die Zeichenfolge '$i' veranlaßt den Makroprozessor dazu, das Makro in einer Schleife zu durchlaufen. Im ersten Durchlauf wird '$i' durch das erste Argument aus dem Makroaufruf ersetzt, im zweiten Durchlauf durch das zweite Argument usw. Ein '\', gefolgt von einem Zeichen, wird durch dieses Zeichen ersetzt. Mit '\$' können Sie die Sonderbehandlung des '$'-Zeichens umgehen.

mdelete [*remote-files*]
Die Dateien *remote-files* auf dem entfernten Rechner werden gelöscht.

mdir *remote-files local-file*
Wie **dir**, es können aber mehrere entfernte Dateien angegeben werden. Wenn der interaktive Abfragemodus (*prompt*) aktiv ist, wird der Benutzer zur Sicherheit noch einmal gefragt, ob das letz-

te Argument wirklich die lokale Datei ist, in die die Ausgabe von **mdir** geschrieben werden soll.

mget *remote-files*
Die Namen der entfernten Dateien *remote-files* werden expandiert. Dann wird das Kommando **get** mit jedem erzeugten Dateinamen durchgeführt. Weitere Informationen zur Dateinamen-Expansion sind unter **glob** zu finden. Die geänderten Dateinamen werden dann nach den *case*–, *ntrans*– bzw. *nmap*–Regeln umgeformt. Die Dateien werden in das aktuelle Arbeitsverzeichnis übertragen. Dieses kann mit dem Kommando '**lcd** directory' geändert werden; neue lokale Verzeichnisse können mit dem Kommando '**!** mkdir directory' eingerichtet werden.

mkdir *directory-name*
Es wird ein neues Verzeichnis auf dem entfernten Rechner eingerichtet.

mls *remote-files local-file*
Wie **ls**, es können aber mehrere entfernte Dateien angegeben werden. Wenn der interaktive Abfragemodus (*prompt*) aktiv ist, wird der Benutzer zur Sicherheit noch einmal gefragt, ob das letzte Argument wirklich die lokale Datei ist, in die die Ausgabe von **mls** geschrieben werden soll.

mode [*mode-name*]
Der Dateiübertragungsmodus wird auf *mode-name* gesetzt. Der Standardmodus ist der stream-Modus.

mput *local-files*
Die Namen der lokalen Dateien *local-files* werden expandiert. Dann wird das Kommando **put** mit jedem erzeugten Dateinamen durchgeführt. Weitere Informationen zur Dateinamen-Expansion sind unter **glob** zu finden. Die sich ergebenden Dateinamen werden dann nach den *ntrans*– bzw. *nmap*–Regeln umgeformt.

nmap [*inpattern outpattern*]
Die Dateinamen-Umsetzung wird ein- bzw. ausgeschaltet. Wenn keine Argumente angegeben werden, wird die Umsetzung ausgeschaltet. Falls Argumente angegeben sind, werden entfernte Dateinamen umgesetzt, wenn **mput**- und **put**-Kommandos ohne Angabe eines entfernten Zieldateinamens abgesetzt werden. Desgleichen werden lokale Dateinamen umgesetzt, wenn **mget**- und **get**-Kommandos ohne Angabe eines lokalen Zieldateinamens abgesetzt werden. Dieses Kommando wird bei der Kommunikation mit einem entfernten Rechner ohne UNIX-Betriebssystem und mit abweichenden Regeln und Praktiken für die Vergabe von Dateinamen benötigt. Die Umsetzung erfolgt anhand der angegebe-

nen Muster *inpattern* und *outpattern*. *Inpattern* ist ein Muster für die umzusetzenden Dateinamen, die ggf. bereits nach den **ntrans**- und **case**-Regeln umgeformt wurden. Variablen werden in einem Muster durch eine der folgenden Zeichenfolgen in *inpattern* angegeben:

@\$1', '$2', ..., '$9

Mit '\$' können Sie die Sonderbehandlung des '$'-Zeichens ausschließen. Alle anderen Zeichen werden unverändert übernommen und dienen dazu, die Variablenwerte in *inpattern* zu bestimmen. Beispiel: Gegeben sind *inpattern* mit $1.$2 und der entfernte Dateiname "xydaten.daten". Dann hat $1 den Wert "xydaten", und $2 hat den Wert "daten". Das Ausgabemuster *outpattern* legt fest, wie die Umsetzung aus dem Eingabemuster erfolgen soll. Die Zeichenfolgen

@\$1', '$2', ..., '$9

werden durch den entsprechenden Wert aus dem Eingabemuster ersetzt. Die Zeichenfolge '$0' wird durch den ursprünglichen Dateinamen ersetzt. Darüber hinaus wird die Zeichenfolge

[*seq1*,*seq2*]

durch *seq1* ersetzt, wenn *seq1* nicht leer ist, andernfalls durch *seq2*. Beispiel: Das Kommando

nmap $1.$2.$3 [$1,$2].[$2,datei]

setzt die Eingabedateinamen "xydatei.daten" und "xydatei.daten.alt" um in den Ausgabdateinamen "xydatei.daten", den Eingabedateinamen "xydatei" in "xydatei.datei" und den Eingabedateinamen ".xydatei" in "xydatei.xydatei". *Outpattern* kann Leerstellen enthalten, wie in dem folgenden Beispiel:

nmap $1 |sed s/ *$// > $1

Durch Voranstellen von '\' können Sie die Sonderbehandlung der Zeichen '$', '[', ']' und ',' ausschließen.

ntrans [*inchars* [*outchars*]]

Die zeichenweise Dateinamen-Übersetzung wird ein- bzw. ausgeschaltet. Wenn keine Argumente angegeben werden, wird die Übersetzung ausgeschaltet. Falls Argumente angegeben sind, werden entfernte Dateinamen zeichenweise übersetzt, wenn **mput**- und **put**-Kommandos ohne Angabe eines entfernten Zieldateinamens abgesetzt werden. Desgleichen werden lokale Dateinamen übersetzt, wenn **mget**- und **get**-Kommandos ohne Angabe eines lokalen Zieldateinamens abgesetzt werden. Dieses Kommando wird bei der Kommunikation mit einem entfernten Rechner ohne UNIX-Betriebssystem und mit abweichenden Regeln und Praktiken für die Vergabe von Dateinamen benötigt. Wenn ein Zeichen in einem Dateinamen in *inchars* enthalten ist, wird es durch das entsprechende Zeichen in *outchars* ersetzt. Ist die Posi-

tionsnummer eines Zeichens in *inchars* größer als die Länge von *outchars*, wird das Zeichen aus dem Dateinamen entfernt.

open *host* [*port*]
Es wird eine Verbindung zu dem FTP-Server auf dem Host *host* hergestellt. Wenn die optionale Anschlußnummer angegeben wird, versucht *ftp*, einen FTP-Server über diesen Anschluß zu erreichen. Wenn die Option *auto-login* aktiv ist (Standard), versucht *ftp* auch, den Benutzer automatisch bei dem FTP-Server anzumelden (siehe unten).

prompt
Der interaktive Abfragemodus wird ein- bzw. ausgeschaltet (Standard: eingeschaltet). Die interaktive Abfrage bei der Übertragung von mehreren Dateien ermöglicht es dem Benutzer, ausgewählte Dateien einzulesen oder zu speichern. Wenn der interaktive Abfragemodus ausgeschaltet ist, werden mit **mget** oder **mput** alle Dateien übertragen und mit **mdelete** alle Dateien gelöscht.

proxy *ftp-command*
Das angegebene FTP-Kommando wird nicht über die bestehende primäre, sondern über eine sekundäre Steuerverbindung ausgeführt. Dieses Kommando ermöglicht die gleichzeitige Verbindung zu zwei entfernten FTP-Servern und damit die Übertragung von Dateien zwischen den beiden Servern. Das erste **proxy**-Kommando sollte ein **open** sein, damit die sekundäre Steuerverbindung hergestellt werden kann. Wenn Sie das Kommando **proxy ?** eingeben, werden alle FTP-Kommandos angezeigt, die über die sekundäre Kontrollverbindung ausgeführt werden können. Die folgenden Kommandos unterscheiden sich in ihrer Funktionsweise, wenn sie mit **proxy** aufgerufen werden: **open** führt bei der automatischen Anmeldung keine neuen Makrodefinitionen durch; **close** löscht nicht die vorhandenen Makrodefinitionen; **get** und **mget** übertragen Dateien vom primären Server zum sekundären Server; **put, mput** und *append* übertragen Dateien vom sekundären Server zum primären Server. Die Dateiübertragung zwischen zwei FTP-Servern ist nur möglich, wenn der sekundäre Server das PASV-Kommando des FTP-Protokolls unterstützt.

put *local-file* [*remote-file*]
Die lokale Datei *local-file* wird auf dem entfernten Rechner gespeichert. Wenn *remote-file* fehlt, erhält die entfernte Datei den Namen der lokalen Datei (nach Umformung gemäß den *ntrans*- bzw. *nmap*-Regeln). *Ftp* benutzt für die Dateiübertragung die momentan gültigen Werte von *type*, *form*, *mode* und *structure*.

pwd Der Name des aktuellen Arbeitsverzeichnisses auf dem entfernten Rechner wird ausgegeben.

quit Gleichbedeutend mit **bye**.

quote *arg1 arg2 ...*
Die angegebenen Argumente werden unverändert an den entfernten FTP-Server übergeben.

recv *remote-file* [*local-file*]
Gleichbedeutend mit **get**.

remotehelp [*command-name*]
Der entfernte FTP-Server wird aufgefordert, Hilfeinformationen auszugeben. Wenn ein Kommandoname *command-name* angegeben ist, wird auch er an den Server übergeben.

rename [*from*] [*to*]
Die Datei *from* auf dem entfernten Rechner wird in *to* umbenannt.

reset Die Antwort-Warteschlange wird zurückgesetzt. Dieses Kommando synchronisiert wieder den Austausch von Kommandos und Antworten mit dem entfernten FTP-Server. Die erneute Synchronisierung kann nach einer Verletzung des FTP-Protokolls durch den entfernten Server erforderlich werden.

rmdir *directory-name*
Das angegebene Verzeichnis auf dem entfernten Rechner wird gelöscht.

runique
Die Vergabe eindeutiger Dateinamen auf dem lokalen Rechner wird ein- bzw. ausgeschaltet (Standard: ausgeschaltet). Wenn bei **get** oder **mget** eine Datei mit dem Namen der lokalen Zieldatei bereits vorhanden ist, wird ".1" an den Namen angehängt. Wenn der neue Name ebenfalls vorhanden ist, wird ".2" an den ursprünglichen Namen angehängt usw. Wenn auch ".99" zu einer Kollision führt, wird eine Fehlermeldung ausgegeben und die Übertragung nicht durchgeführt. Der erzeugte eindeutige Dateiname wird angezeigt. Beachten Sie, daß **runique** lokale Dateinamen nicht verändert, wenn sie durch ein Shell-Kommando erzeugt werden (siehe unten).

send *local-file* [*remote-file*]
Gleichbedeutend mit **put**.

sendport
Die Anwendung von PORT-Kommandos wird ein- bzw. ausgeschaltet. *Ftp* versucht standardmäßig für alle Dateiübertragungen

die Verbindung mit Hilfe eines PORT-Kommandos herzustellen. Durch den Einsatz des PORT-Kommandos können Verzögerungen beim Übertragen mehrerer Dateien vermieden werden. Wenn die Ausführung des PORT-Kommandos scheitert, benutzt *ftp* den Standard-Datenanschluß. Wenn die Anwendung von PORT-Kommandos ausgeschaltet ist, unterläßt *ftp* den Versuch, für alle Dateiübertragungen ein PORT-Kommando einzusetzen. Dies kann sich für bestimmte FTP-Implementierungen als sinnvoll erweisen, wenn das PORT-Kommando zwar ignoriert, aber fälschlicherweise die erfolgreiche Durchführung gemeldet wird.

status
Der momentane Status des *ftp* wird angezeigt.

struct [*struct-name*]
Die Dateiübertragungsstruktur wird auf *struct-name* gesetzt. Die Standardstruktur ist die stream-Struktur.

sunique
Die Vergabe eindeutiger Dateinamen auf dem entfernten Rechner wird ein- bzw. ausgeschaltet (Standard: ausgeschaltet). Für die erfolgreiche Ausführung des Kommandos ist es erforderlich, daß der entfernte FTP-Server das STOU-Kommando des FTP-Protokolls unterstützt. Der entfernte Server zeigt den erzeugten eindeutigen Dateinamen an.

tenex Der Dateiübertragungstyp (*type*) wird so eingestellt, daß die Dateiübertragung mit TENEX-Rechnern möglich ist. - Das Kommando ist in dieser Form nicht verfügbar. Bitte benutzen Sie das Kommando "type tenex".

trace Die Protokollierung der übertragenen Pakete wird ein- bzw. ausgeschaltet.

type [*type-name*]
Der Dateiübertragungstyp wird auf *type-name* (*ascii*, *binary*, *image*, *ebcdic*, *tenex*) eingestellt. Wenn kein Typ angegeben ist, wird der momentan gültige Typ angezeigt. Standardmäßig wird als Typ Netzwerk-ASCII angenommen.

user *user-name* [*password*] [*account*]
Mit dem *user*–Kommando können Sie sich gegenüber dem entfernten FTP-Server identifizieren. Wenn kein Paßwort angegeben ist, aber vom Server verlangt wird, fordert *ftp* den Benutzer zur Eingabe des Paßworts auf (das lokale Eingabe-Echo wird ausgeschaltet). Wenn kein Kontofeld (*account*) angegeben ist, aber vom Server verlangt wird, fordert *ftp* den Benutzer zur Eingabe des Kontofeldes auf. Wenn ein Kontofeld angegeben ist, wird nach Abschluß der Anmeldung an den entfernten Server ein

ACCT-Kommando übergeben. Dies trifft allerdings nur dann zu, wenn der entfernte Server das Kontofeld nicht bereits zur Anmeldung benötigt. Wenn die Option *auto-login* aktiv ist, läuft der beschriebene Vorgang bei der erstmaligen Verbindung mit dem FTP-Server automatisch ab.

verbose
Der Protokollmodus wird ein- bzw. ausgeschaltet (Standard: eingeschaltet). Im Protokollmodus werden dem Benutzer alle Antworten des FTP-Servers angezeigt. Außerdem wird im Protokollmodus nach Abschluß einer Dateiübertragung eine Übersicht mit statistischen Daten über die Durchsatzleistung der Übertragung ausgegeben.

? [*command*]
Gleichbedeutend mit **help.**

Kommando-Argumente, die Leerstellen enthalten, müssen in Anführungszeichen (") gesetzt werden.

ABBRUCH EINER DATEIÜBERTRAGUNG

Eine Dateiübertragung kann mit der Unterbrechungstaste (in der Regel Ctrl-C) von einem Terminal aus abgebrochen werden. Sendevorgänge werden sofort angehalten. Zum Abbruch von Empfangsvorgängen wird das ABOR-Kommando des FTP-Protokolls an den entfernten Server gesendet. Weitere ankommende Daten werden nicht mehr verarbeitet. Wie schnell in diesem Fall der Abbruch erfolgt, hängt davon ab, ob der entfernte Server das ABOR-Kommando verarbeiten kann. Falls er nicht dazu imstande ist, erscheint die nächste "ftp>"-Eingabe-Aufforderung erst dann wieder, wenn der entfernte Server die angegebene Datei fertig übertragen hat.

Die Betätigung der Unterbrechungstaste an einem Terminal wird ignoriert, wenn *ftp* nach dem Abschluß der lokalen Verarbeitung auf Antwort vom entfernten Server wartet. Eine längere Verzögerung in dieser Situation kann aufgrund der oben beschriebenen Durchführung des ABOR-Kommandos auftreten oder wegen eines unerwarteten Verhaltens des entfernten Servers (auch Verletzungen des FTP-Protokolls). Falls die Verzögerung auf einem unerwarteten Verhalten des entfernten Servers beruht, muß das lokale *ftp*–Programm von Hand abgebrochen werden.

REGELN FÜR DIE VERGABE VON DATEINAMEN

Dateinamen, die an *ftp* als Argumente übergeben werden, werden nach folgenden Regeln verarbeitet:

1) Wenn als Dateiname – angegeben wird, werden **stdin** (zum Lesen) bzw. **stdout** (zum Schreiben) benutzt.

2) Wenn das erste Zeichen eines Dateinamens ein | ist, wird der Rest des Arguments als Shell-Kommando interpretiert. *Ftp* startet dann eine Shell durch Aufruf von *popen*(3) mit dem angegebenen Argument und liest von stdin bzw. schreibt auf stdout. Wenn das Shell-Kommando Leerstellen enthält, muß es in Anführungszeichen gesetzt werden, z. B. "**| ls -lt**". Beispiel: "**dir |more**".

3) Wenn keine der obigen Regeln zutrifft, wird die Dateinamen-Expansion aktiviert, und lokale Dateinamen werden nach den Regeln von *csh*(1) expandiert (siehe auch das *glob*--Kommando). Wenn *ftp* eine einzelne lokale Datei erwartet, (z. B. bei **put**), wird nur der erste der expandierten Dateinamen benutzt.

4) Bei **mget**- und **get**-Kommandos ohne Angabe lokaler Dateinamen wird der entfernte Dateiname als lokaler Dateiname übernommen (nach Umformung gemäß den *case*-, *ntrans*- bzw. *nmap*-Regeln). Der so entstandene Dateiname kann noch einmal verändert werden, wenn **runique** eingeschaltet ist.

5) Bei **mput**- und **put**-Kommandos ohne Angabe entfernter Dateinamen wird der lokale Dateiname als entfernter Dateiname übernommen (nach Umformung gemäß den *ntrans*- bzw. *nmap*-Regeln). Der so entstandene Dateiname kann von dem entfernten Server noch einmal verändert werden, wenn **sunique** eingeschaltet ist.

DATEIÜBERTRAGUNGS-PARAMETER

Die FTP-Spezifikation legt viele Parameter fest, die eine Dateiübertragung beeinflussen können. Der Dateiübertragungstyp *type* kann einen der folgenden Werte annehmen: ascii, image (binary), ebcdic und local byte size (hauptsächlich für PDP-10 und PDP-20). *Ftp* unterstützt die Dateiübertragungstypen *ascii* und *image* sowie *local byte size 8* (lokale Bytegröße 8) für die Dateiübertragung im **tenex**-Modus.

Ftp unterstützt nur die Standardwerte der übrigen Dateiübertragungsparameter: *mode*, *form* und *struct*.

OPTIONEN

Optionen können in der Kommandozeile oder im Kommando-Interpreter angegeben werden.

Die Option **-v** (verbose) bewirkt, daß *ftp* alle Antworten des entfernten Servers protokolliert und statistische Angaben über die Dateiübertragung ausgibt.

Die Option **-n** (no auto-login) bewirkt, daß *ftp* bei der erstmaligen Verbindung nicht versucht, eine automatische Anmeldung (auto-login) durchzuführen. Wenn die automatische Anmeldung eingeschaltet ist, sucht *ftp* in der Datei *netrc* im Home-Verzeichnis des Benutzers nach ei-

nem Eintrag mit einer Beschreibung des Kontos auf dem entfernten Rechner. Gibt es keinen solchen Eintrag, fordert *ftp* zur Eingabe des Benutzernamens für den entfernten Rechner auf (Standard: Benutzername auf dem lokalen Rechner). Wenn nötig, wird auch zur Eingabe eines Paßwortes und eines Kontos für die Anmeldung aufgefordert.

Die Option **-i** schaltet den interaktiven Abfragemodus beim Übertragen mehrerer Dateien aus.

Die Option **-d** schaltet den Testhilfemodus ein.

Die Option **-g** (globbing) schaltet die Dateinamen-Expansion aus.

DIE DATEI .netrc

Die Datei .netrc enthält Angaben für die Anmeldung und zur Initialisierung, die von dem Auto-Login-Prozeß benötigt werden. Sie befindet sich im Home-Verzeichnis des Benutzers. In der Datei können die folgenden Klauseln vorkommen; sie müssen durch Leerstellen, Tabulatorzeichen oder New-Line-Zeichen voneinander getrennt werden:

machine *name*

Diese Klausel definiert den Namen eines entfernten Rechners. Der Auto-Login-Prozeß durchsucht die Datei .netrc nach einer **machine**-Klausel mit einem passenden Rechnernamen. Er muß mit dem Namen des entfernten Rechners übereinstimmen, der in der *ftp*-Kommandozeile oder in einem **open**-Kommando angegeben wurde. Wenn ein passender Eintrag gefunden wird, werden die nachfolgenden .netrc-Klauseln verarbeitet, solange bis das Dateiende erreicht ist oder eine andere **machine**-Klausel gefunden wird.

login *name*

Diese Klausel definiert den Namen eines Benutzers auf dem entfernten Rechner. Wenn diese Klausel vorhanden ist, veranlaßt der Auto-Login-Prozeß die Anmeldung des Benutzers mit dem angegebenen Namen.

password *string*

Diese Klausel definiert das Paßwort des Benutzers. Wenn diese Klausel vorhanden ist, übergibt der Auto-Login-Prozeß die angegebene Zeichenfolge *string* an den entfernten Server, sofern dieser ein Paßwort für die Anmeldung benötigt. Beachten Sie: Wenn diese Klausel in der Datei .netrc vorhanden ist, und wenn die Datei nicht nur vom Benutzer, sondern auch von anderen gelesen werden kann, bricht *ftp* den Auto-Login-Prozeß ab.

account *string*

Diese Klausel definiert ein zusätzliches Konto-Paßwort. Wenn diese Klausel vorhanden ist, übergibt der Auto-Login-Prozeß die

angegebene Zeichenfolge *string* an den entfernten Server, sofern dieser ein zusätzliches Konto-Paßwort benötigt. Ist letzteres nicht der Fall, setzt der Auto-Login-Prozeß ein ACCT-Kommando ab.

macdef *name*

Mit dieser Klausel wird die Definition eines Makros eingeleitet. Diese Klausel funktioniert genauso wie das *ftp*-Kommando **macdef**. Das Makro wird unter dem angegebenen Namen *name* definiert. Die eigentliche Makrodefinition beginnt mit der nächsten Zeile der Datei .netrc und wird durch eine Nullzeile beendet (zwei aufeinanderfolgende New-Line-Zeichen). Ein Makro mit dem Namen *init* wird automatisch als letzter Schritt des Auto-Login-Prozesses ausgeführt.

FEHLER

Die ordnungsgemäße Ausführung vieler Kommandos hängt von dem einwandfreien Verhalten des entfernten Servers ab.

In der ASCII-Dateiübertragungsroutine des 4.2BSD UNIX-Betriebssystems wurde ein Fehler bei der Behandlung von Carriage-Return-Zeichen berichtigt. Diese Änderung kann zur fehlerhaften Übertragung binärer Dateien zwischen 4.2BSD-Servern führen, wenn der Dateiübertragungstyp *ascii* benutzt wird. Zur Umgehung dieses Problems sollten Sie statt dessen den Dateiübertragungstyp *binary image* einsetzen.

BEZEICHNUNG

gdev: hpd, erase, hardcopy, tekset, td – Routinen und Filter für Grafikvorrichtungen

ÜBERSICHT

hpd [–options] [GPS file ...]
erase
hardcopy
tekset
td [**–ern***n*] [GPS file . . .]

BESCHREIBUNG

Alle nachstehend beschriebenen Kommandos befinden sich in **/usr/bin/graf** (vgl. *graphics*(1G)).

hpd *hpd* übersetzt eine GPS (grafische primitive Zeichenfolge; vgl. *gps*(4)) in Anweisungen für den Hewlett-Packard 7221A Grafik Plotter. Ein Fenster wird aus den Maximum und Minimum Punkten in *file* berechnet, sofern nicht die *Optionen* **–u** oder **–r** zur Verfügung stehen. Wenn *file* nicht angegeben ist, wird die Standardeingabe angenommen. *options* sind:

- **c***n* Zeichensatz *n* wählen, *n* zwischen 0 und 5 (vgl. das *HP7221A Plotter Operating and Programming Manual, Appendix A*).
- **p***n* Mit *n* numerierten Schreiber wählen, *n* zwischen 1 und 4 einschließlich.
- **r***n* Fenster auf GPS-Bereich *n*, *n* zwischen 1 und 25 einschließlich.
- **s***n* Zeichen *n* Grad rechts von der Senkrechten schrägstellen.
- **u** Fenster für das ganze GPS-Universum.
- **xd***n* x-Verschiebung der linken unteren Fensterecke auf *n* Zoll setzen.
- **xv***n* Fensterbreite auf *n* Zoll setzen.
- **yd***n* y-Verschiebung der linken unteren Fensterecke auf *n* Zoll setzen.
- **yv***n* Fensterhöhe auf *n* Zoll setzen.

erase *Erase* sendet zum Löschen des Bildschirms Zeichen an einen Tektronix-Bildspeicher der Serie 4010.

hardcopy Wenn ein Tektronix-Terminal eine Hartkopie-Einheit hat, generiert *hardcopy* darauf eine Bildschirmkopie.

tekset *tekset* sendet einem Tektronix-Terminal Zeichen zum Löschen der Bildschirmanzeige, zum Setzen des Anzeigemodus auf Alpha und zur Einstellung von Zeichen auf die kleinste Schriftart.

td *td* übersetzt ein GPS für einen Tektronix-Bildspeicher der Serie 4010 in Bildschirm-Code. Aus den Maximum und Minimum Punkten in *file* wird ein Fenster berechnet, sofern nicht die *Optionen* **-u** oder **-r** zur Verfügung stehen. Wenn kein *file* angegeben ist, wird die Standardeingabe angenommen. Optionen lauten:

- **e** Bildschirm nicht löschen vor der Ausgabe.
- **r***n* Anzeige des GPS-Bereichs *n*, *n* zwischen 1 und 25 einschließlich.
- **u** Anzeige des ganzen GPS-Universums.

SIEHE AUCH:

ged(1g), graphics(1g).
gps(4) im *Programmer's Reference Manual.*

BEZEICHNUNG

ged – Grafik-Editor

ÜBERSICHT

ged [**–eruR**]*n*] [GPS file ...]

BESCHREIBUNG

ged ist ein interaktiver Grafik-Editor zur Anzeige, Konstruktion und zum Editieren von GPS-Dateien auf Tektronix-Termianls der Serie 4010. Wenn GPS *file*(s) angegeben werden, liest *ged* sie in einen internen Bildpuffer und zeigt den Puffer an. Die gepufferte GPS kann dann editiert werden. Wenn – als Dateiname angegeben ist, liest *ged* eine GPS aus der Standardeingabe.

ged akzeptiert auf der Kommandozeile die folgenden Optionen

- **e** Bildschirm nicht vor der ersten Anzeige löschen.
- **r***n* Feld-Nummer *n* anzeigen.
- **u** Das ganze GPS-*universe* anzeigen.
- **R** Bei Verwendung von ! erfolgt Aufruf der eingeschränkten Shell.

Eine GPS-Datei enthält drei Grafikobjekte: *lines* (Linien), *arc* (Kreisbogen) und *text* (Text). Die Objekte *arc* und *lines* haben einen Startpunkt, oder *object-handle*, gefolgt von Null oder mehreren Punkten oder *point-handles* (Gitterpunkte). *Text* hat nur einen Startpunkt. Die Objekte sind in einer Kartesischen Ebene (*universe*-Universum) angeordnet, die auf jeder Achse 64K (–32K bis +32K) Punkte (*universe-units*) hat. Das Universum ist in 25 gleich große Bereiche aufgeteilt, die Felder (*regions*) genannt werden. Diese Felder bestehen aus fünf Reihen mit je fünf Quadraten, die auf dem *universe* von unten links bis oben rechts von 1 bis 25 durchnumeriert sind.

ged bildet die mit Fenster (*windows*) bezeichneten rechteckigen Felder des *universe* auf dem Bildschirm ab. Fenster gestatten dem Benutzer, Bilder von verschiedenen Stellen und in unterschiedlichen Vergrößerungen zu betrachten. Das *universe-window* ist das Fenster mit der geringsten Vergrößerung, d. h. das Fenster sieht das ganze Universum. Das *home-window* ist das Fenster, das den Inhalt des Bildpuffers ganz anzeigt.

KOMMANDOS

ged Kommandos werden in *stages* (Stufen) eingegeben. Normalerweise wird jede Stufe mit einem **<CR>** (return) beendet. Vor dem endgültigen **<CR>** kann ein Kommando durch Eingabe von **rubout** abgebrochen werden. Die Eingabe einer Stufe kann innerhalb einer Stufe unter Verwendung der Lösch- und Abbruchzeichen der aufrufenden Shell

editiert werden. Das Eingabe-Aufforderungszeichen * zeigt an, daß *ged* in Stufe 1 wartet.

Jedes Kommando besteht aus einer Teilmenge der folgenden Stufen:

1. *Command line*
   Eine Kommandozeile *command line* besteht aus einem Kommandonamen *name*, gefolgt von *argument*(s), gefolgt von **<CR>**. Ein Kommandoname *name* ist ein einzelnes Zeichen. Kommandoargumente *arguments* sind entweder Optionen *option*(s) oder ein Dateiname *file-name*. Die Optionen *options* werden mit einem führenden - angezeigt.

2. *Text*
   Text ist eine Folge von Zeichen, die durch ein nicht quotiertes **<CR>** abgeschlossen sind (max.120 Textzeilen).

3. *Points*
   Punkte *points* ist eine Folge eines oder mehrerer Bildschirm-Positionen (maximal 30), die entweder in dem Punktraster oder mit Namen angegeben sind. Das Eingabe-Aufforderungszeichen zur Eingabe von *points* ist das Erscheinen des Punktraster. Wenn das Punktraster sichtbar ist, wird bei der Eingabe von:

   sp (Leerzeichen) die aktuelle Position als Punkt eingegeben. Der Punkt wird mit einer Zahl identifiziert.

   $*n* der vorige *point* eingegeben, der mit *n* numeriert ist.

   >*x* der zuletzt eingegebene *point* mit dem Großbuchstaben *x* markiert.

   $*x* der mit *x* markierte *point* eingegeben.

   . die vorherigen *points* zu den aktuellen *points* gemacht. Bei Kommandobeginn sind die vorigen *points* die durch das vorige Kommando eingegebenen Positionen.

   = die aktuellen *points* angezeigt.

   $*n* der *point* der vorigen *points* eingegeben, der mit *n* numeriert war.

   # wird der letzte eingegebene *point* gelöscht.

   @ werden alle eingegebenen *points* gelöscht.

4. *Pivot*
   pivot ist eine einzelne Position, die durch Betätigung von **<CR>** oder unter Verwendung des Operators $ eingegeben und mit einem * angezeigt wird.

5. *Destination*

destination ist eine einzelne Position, die durch Betätigung von **<CR>** oder unter Verwendung von **$** eingegeben wird.

KOMMANDOZUSAMMENFASSUNG

In der Zusammenfassung werden Zeichen, die vom Benutzer eingegeben werden, **bold** (halbfett) angezeigt. Kommandostufen sind *italic* (kursiv) angegeben. In Klammern eingeschlossene Argumente "[]" können wahlweise eingegeben werden. Runde Klammern "()" bei Argumenten, die durch "or" getrennt sind, bedeuten, daß genau eines der Argumente angegeben werden muß.

Kommandos zum Zeichnen:

arc	[-echo,style,weight] *points*
Box	[-echo,style,weight] *points*
Circle	[-echo,style,weight] *points*
Hardware	[-echo] *text points*
Lines	[-echo,style,weight] *points*
Text	[-angle,echo,height,mid-point,right-point,text,weight] *text points*

Editierkommandos:

Delete	(- (universe oder view) oder *points*)
Edit	[-angle,echo,height,style,weight] (- (universe oder view) oder *points*)
Kopy	[-echo,points,x] *points pivot destination*
Move	[-echo,points,x] *points pivot destination*
Rotate	[-angle,echo,kopy,x] *points pivot destination*
Scale	[-echo,factor,kopy,x] *points pivot destination*

Einblendkommandos:

coordinates	*points*
erase	
new-display	
object-handles	(- (universe oder view) oder *points*)

point-handles	(- (labelled-points oder universe oder view) oder *points*)
view	(- (home oder universe oder region) oder [-x] *pivot destination*)
x	[-view] *points*
zoom	[-out] *points*

Andere Kommandos:

quit oder Quit

read	[-angle,echo,height,mid-point,right-point, text,weight *file-name* [*destination*]
set	[-angle,echo,factor,height,kopy, mid-point,points, right-point,style,text,weight,x]
write	*file-name*
!*command*	
?	

Optionen:

Die Optionen *options* geben Parameter an, die zum Erstellen, zum Editieren und Einblenden von grafischen Objekten benutzt werden. Falls ein Parameter, der von einem Kommando verwendet wird, nicht als *option* angegeben ist, wird der Standardwert für den Parameter verwendet (vgl. unten set). Das Format der Kommandooptionen *options* lautet:

-*option* [,*option*]

wobei *option* von der Form *keyletter*[*value*] ist. Optionen können die Werte (value) gesetzt oder nicht gesetzt annehmen, was jeweils durch + oder - angezeigt wird. Wenn bei einer Option kein *value* angegeben ist, wird gesetzt angenommen.

Objekt-Optionen:

angle*n*	Winkel von *n* Grad.
echo	Wenn gesetzt, werden Hinzufügungen im Bildpuffer vermerkt.
factor*n*	Maßstabfaktor ist *n* Prozent.
height*n*	Höhe von *text* ist *n* Universum-Einheiten ($0 \leq n < 1280$).

kopy — Wenn gesetzt, wird Kopieren anstelle von Verlagern ausgeführt.

mid-point — Wenn gesetzt, wird mid-point zum Positionieren der *text*- Zeichenfolge verwendet.

points — Wenn gesetzt, wird mit Punkten gearbeitet; andernfalls wird mit Objekten gearbeitet.

right-point — Wenn gesetzt, wird right-point zum Positionieren der *text*-Zeichenfolge verwendet.

style*type* — Der Linientyp wird auf einen der folgenden *types* gesetzt:

so	ununterbrochene Linie
da	gestrichelte Linie
dd	punktierte/gestrichelte Linie
do	punktierte Linie
ld	lang gestrichelte Linie

text — Wenn nicht gesetzt, werden *text* Zeichenfolgen skizziert anstatt gezeichnet.

weight*type* — Setzt die Strichstärke auf einen der folgenden *types*:

n	fein
m	mittel
b	grob

Feldoptionen:

home — Bezieht sich auf das Home-Fenster.

out — Reduziert die Vergrößerung

region*n* — Bezieht sich auf das Feld *n*.

universe — Bezieht sich auf das Universum-Fenster.

view — Bezieht sich auf die Objekte, die gerade eingeblendet sind.

x — Gibt den Mittelpunkt des verwiesenen Felds an.

KOMMANDOBESCHREIBUNGEN

Kommandos zum Zeichnen:

Arc und Lines

verhalten sich ähnlich. Beide bestehen aus einer *command line* , gefolgt von *points*. Der erste eingegebene *point* ist der Startpunkt. Aufeinanderfolgende *points* sind Gitterpunkte. Lines verbinden die Punkte gemäß numerischer Reihenfolge. Arc bildet eine Kurve aus den Punkten (momentan können höchstens 3 Punkte zu einem Kreisbogen verbunden werden; Kurvenverbindungslinien werden bei einer späteren Version hinzugefügt).

Box und **Circle**
sind Sonderformen von **Lines** und **Arc**. **Box** erzeugt ein Rechteck mit Seiten, die parallel zu den Achsen des Universums sind. Eine Diagonale des Rechtecks verbindet den ersten eingegebenen *point* mit dem letzten *point*. Der erste *point* ist der Startpunkt. Gitterpunkte werden an jedem der Scheitelpunkte erzeugt. **Circle** erzeugt einen Kreisbogen, dessen Mittelpunkt am *point* mit Nummer Null liegt und durch den letzten *point* geht. Der Startpunkt des Kreises fällt zusammen mit dem letzten *point*. Ein Gitterpunkt wird 180 Grad vom Startpunkt am Kreis erzeugt.

Text und **Hardware**
erzeugen *text*-Objekte. Jedes besteht aus einer Kommandozeile *command line*, *text* und *points*. *Text* ist eine Folge von Zeichen, die durch **<CR>** abgeschlossen ist. Mehrere Textzeilen können eingegeben werden, indem ein **cr** durch einen Backslash (z.B. **\cr**) entwertet wird. Das Kommando **Text** erstellt softwaremäßig erzeugte Zeichen. Jede Zeile mit Software-Text wird als ein separates *text* Objekt behandelt. Der erste eingegebene *point* ist der Startpunkt für die erste Textzeile. Das Kommando **Hardware** sendet die Zeichen von *text* in nicht interpretierter Form an das Terminal.

Editierkommandos:
Editierkommandos arbeiten auf Teilen des Bildpuffers, die *defined areas* (definierte Felder) heißen. Ein definiertes Feld wird entweder interaktiv oder durch eine *area option* angesprochen. Wenn keine *area option* angegeben ist, wird der Umfang des definierten Felds durch *points* angegeben. Wenn kein *point* eingegeben ist, wird um **<CR>** herum ein kleines definiertes Feld erzeugt. Dies ist nützlich bei der Zuordnung eines einzigen *point*. Wenn nur ein *point* eingegeben ist, wird **<CR>** zusammen mit *point* zur Anzeige einer Diagonalen eines Rechtecks verwendet. Ein definiertes Feld, das durch *points* anzeigt wird, wird mit punktierten Strichen umrissen.

Delete
entfernt alle Objekte, deren Startpunkt innerhalb eines definierten Felds liegt. Die Option **universe** entfernt alle Objekte und löscht den Bildschirm.

Edit ändert die Parameter der Objekte innerhalb eines definierten Felds. Die folgenden Parameter können geändert werden:

angle	Winkel von *text*
height	Höhe von *text*
style	Stil von *lines* und *arc*
weight	Stärke von *lines*, *arc* und *text*.

Kopy (oder **Move**)
kopiert (oder verschiebt) Start- und/oder Gitterpunkte innerhalb eines definierten Felds durch die Verschiebung von *pivot* zur *destination*.

Rotate
rotiert Objekte innerhalb eines definierten Felds um den *pivot*. Wenn die Option **kopy** gesetzt ist, werden die Objekte kopiert und nicht verschoben.

Scale
Bei Objekten, deren Startpunkte innerhalb eines definierten Felds liegen, werden Punktverschiebungen vom *pivot* durch factor-Prozente maßstabgetreu vorgenommen. Wenn die Option **kopy** gesetzt ist, werden die Objekte kopiert und nicht verschoben.

Einblendkommandos:

coordinates
gibt die Position von *point*(s) in Universum- und Bildschirmeinheiten an.

erase löscht den Bildschirm (jedoch nicht den Bildpuffer).

new-display
löscht den Bildschirm und zeigt dann den Bildpuffer an.

object-handles (oder **point-handles**) markiert Startpunkte (und/oder Gitterpunkte), die innerhalb eines definierten Feldes liegen mit **O** (oder **P**). **Point-handles** identifiziert markierte Punkte, wenn die Option **labelled-points** gesetzt ist.

view verschiebt das Fenster, so daß der zu *pivot* gehörige Universum-Punkt mit dem Bildschirm-Punkt, der zu *destination* gehört, übereinstimmt. Die Optionen für **home**, **universe**, und **region** zeigen besondere Fenster in dem Universum an.

x gibt den Mittelpunkt eines definierten Felds an. Die Option **view** zeigt den Mittelpunkt des Bildschirms an.

zoom
reduziert (zoom out) oder erhöht die Größe des Einblendfensters, basierend auf dem definierten Feld. Beim Erhöhen wird das Fenster so eingestellt, daß es das definierte Feld umfaßt. Beim Reduzieren liegt das aktuelle Fenster innerhalb des definierten Felds.

Andere Kommandos:

quit oder Quit
beendet *ged*. Quit antwortet mit einem ?, falls der Bildpuffer seit der letzten Modifikation nicht weggeschrieben wurde.

read liest den Inhalt einer Datei ein. Wenn die Datei ein GPS enthält, wird dies direkt gelesen. Wenn die Datei Text enthält, wird dieser in *text*-Objekt(e) konvertiert. Die erste Zeile einer Textdatei beginnt bei *destination*.

set bei Vorgabe von Optionen *option*(s) wird bei diesem Kommando eine Rückstellung der Standardparameter ausgeführt, andernfalls werden die Standardwerte angegeben.

write schreibt den Inhalt des Bildpuffers in eine Datei.

! verläßt *ged* zur Ausführung eines UNIX System-Kommandos.

? listet *ged* Kommandos auf.

SIEHE AUCH

gdev(1G), graphics(1G), sh(1).
gps(4) im *Programmer's Reference Manual*.

BEZEICHNUNG

getopt – Syntaxanalyse der Kommando-Optionen

ÜBERSICHT

set – – `getopt** optstring **$*`

BESCHREIBUNG

ACHTUNG! Benutzung des neuen Kommandos *getopts* (1) anstelle von *getopt* (1) wird empfohlen. *getopt* (1) wird in der nächsten Version nicht mehr unterstützt. Weitere Erläuterungen finden Sie nachstehend im Abschnitt **ACHTUNG!**.

getopt wird zur Einteilung der Optionen in den Kommandozeilen verwendet, um in Shell-Prozeduren leichter Syntaxprüfungen durchführen und auf korrekte Optionen prüfen zu können. *optstring* ist eine Zeichenfolge von Optionsbuchstaben (vgl. *getopt*(3C)); wenn auf einen Buchstaben ein Doppelpunkt folgt, enthält die Option ein Argument, das durch Zwischenraumzeichen getrennt sein kann oder nicht. Die besondere Option – – wird zur Angabe des Endes der Optionen verwendet. Wenn diese Angabe ausdrücklich verwendet wird, erkennt *getopt* sie; andernfalls wird sie von *getopt* erzeugt; in beiden Fällen wird *getopt* sie ans Ende der Optionen stellen. Die Positionsparameter ($1 $2 ...) der Shell werden zurückgesetzt, so daß vor jeder Option ein – steht und jede Option für sich in einem Positionsparameter steht; jedes Optionsargument wird ebenso einem eigenen Positionsparameter zugeordnet.

BEISPIEL

Der folgende Code-Auszug zeigt, wie die Argumente bei einem Kommando verarbeitet werden können, das die Optionen **a** oder **b** sowie die Option **o**, die ein Argument erfordert, hat:

```
set -- `getopt abo: $*`
if [ $? ! = O ]
then
 echo $USAGE
 exit 2
fi
for i in $*
do
 case $i in
 -a | -b) FLAG=$i; shift;;
 -o) OARG=$2; shift 2;;
        --)             shift; break;;
        esac
done
```

Der angegebene Code behandelt die folgenden Aufrufe alle gleich:

```
cmd -aoarg file file
cmd -a -o arg file file
cmd -oarg -a file file
cmd -a -oarg -- file file
```

SIEHE AUCH:

getopts(1), sh(1).
getopt(3c) im *Programmer's Reference Manual.*

DIAGNOSE

getopt gibt eine Fehlermeldung auf die Standardfehlerausgabe aus, wenn es auf einen Optionsbuchstaben trifft, der nicht in *optstring* enthalten ist.

ACHTUNG!

getopt (1) unterstützt nicht den Teil in Regel 8 der Kommandosyntaxregeln (vgl. *intro*(1)), der bei Gruppen von Argumenten nach einer Option die Abgrenzung durch Zwischenraumzeichen oder Anführungszeichen gestattet. Zum Beispiel wird

```
cmd -a -b -o "xxx z yy" file
```

nicht richtig bearbeitet. Zur Umgehung dieser mangelnden Funktionsflexibilität verwenden Sie das neue Kommando *getopts* (1) anstelle von *getopt* (1)

getopt (1) wird in der nächsten größeren Version nicht unterstützt. Für diese Version wurde die Umwandlungsfunktion *getoptcvt* bereitgestellt. Weitere Erläuterungen zu *getopts* und *getoptcvt* finden Sie in *getopts* (1) in diesem Handbuch.

Wenn auf eine Option mit einem Optionsargument ein Wert folgt, der identisch mit einer der in *optstring* angeführten Optionen ist (unter Bezugnahme auf das obenerwähnte BEISPIEL, aber mit folgendem Aufruf: cmd -o -a file), wird *getopt* **-a** immer als ein Optionsargument für **-o** behandeln; es wird nie **-a** als eine Option erkennen. In diesem Fall geht in dem Beispiel die **for** Schleife am Argument *file* vorbei.

BEZEICHNUNG

getopts, getoptcvt – Syntaxanalyse für Kommando-Optionen

ÜBERSICHT

getopts optstring name [arg ...]

/usr/lib/getoptcvt [**-b**] file

BESCHREIBUNG

getopts wird von Shell-Prozeduren zur Analyse der Positionsparameter und zur Prüfung auf zulässige Optionen verwendet. Das Kommando unterstützt alle dafür vorgesehenen Kommandosyntaxregeln (vgl. Regeln 3-10, *intro* (1)). Es sollte anstelle des Kommandos *getopt* (1) verwendet werden. (Vgl. nachstehende Hinweise **ACHTUNG!**)

optstring muß die Optionsbuchstaben enthalten, die das Kommando *getopts* erkennt; wenn auf einen Buchstaben ein Doppelpunkt folgt, wird erwartet, daß die Option ein Argument oder eine Gruppe von Argumenten, die durch Zwischenraumzeichen getrennt sind, enthält.

Bei jedem Aufruf von *getopts* schreibt dieses Kommando die nächste Option in die Shell-Variable *name* und den Index des als nächstes zu verarbeitenden Arguments in die Shell-Variable **OPTIND.** Bei jedem Aufruf der Shell oder einer Shell-Prozedur wird **OPTIND** auf **1** initialisiert.

Wenn eine Option ein Optionsargument erfordert, schreibt es *getopts* in die Shell-Variable **OPTARG.**

Wenn eine ungültige Option vorkommt, wird **?** in *name* geschrieben.

Bei Optionsende endet *getopts* mit einem Nicht-Null-Endestatus. Die besondere Option "--" kann zur Angabe des Endes der Optionen verwendet werden.

Standardmäßig analysiert *getopts* die Positionsparameter. Wenn zusätzliche Argumente (*arg ...*) beim Aufruf von *getopts* angegeben sind, wird *getopts* diese statt dessen analysieren.

/usr/lib/getoptcvt liest das Shell-Skript in *file,* wandelt es so um, daß *getopts* (1) anstelle von *getopt* (1) verwendet wird und schreibt das Ergebnis auf die Standardausgabe.

-b die Ergebnisse von */usr/lib/getoptcvt* können auf frühere Versionen des UNIX-Systems übertragen werden. */usr/lib/getoptcvt* modifiziert das Shell-Skript in *file* so, daß erst bei Ausführung des resultierenden Shell-Skripts bestimmt wird, ob *getopts* (1) oder *getopt* (1) aufgerufen wird.

Auf diese Weise werden sich alle neuen Kommandos an die in *intro* (1) beschriebenen Kommandosyntaxregeln halten; sie sollten also zur Ana-

lyse von Positionsparametern und zur Prüfung auf zulässige Optionen für das angeführte Kommando *getopts* (1) oder *getopt* (3C) verwenden (vgl. nachstehende Hinweise bei **ACHTUNG!**).

BEISPIEL

Der folgende Auszug eines Shell-Programms zeigt, wie die Argumente für ein Kommando verarbeitet werden können, das die Optionen **a** oder **b** sowie die Option **o**, die ein Optionsargument erfordert, enthält:

```
while getopts abo: c
do
        case $c in
        a | b)          FLAG=$c;;
        o)              OARG=$OPTARG;;
        \?)             echo $USAGE
                        exit 2;;
        esac
done
shift `expr $OPTIND - 1`
```

Die folgenden Aufrufe werden von diesem Code als gleichwertig angesehen:

```
cmd file -a -b -o "xxx z yy"
cmd file -a -b -o "xxx z yy" --
cmd file -ab -o xxx,z,yy
cmd file -ab -o "xxx z yy"
cmd file -o xxx,z,yy -b -a
```

SIEHE AUCH:

intro(1), sh(1).
getopts(3C) im *Programmer's* Reference *Manual*.
Anmerkungen zu UNIX System V Ausgabe 3.0.

ACHTUNG!

Obwohl bei der aktuellen Implementierung Abweichungen von den Regeln der Kommandosyntax (vgl. *intro*(1)) möglich sind, ist diese nicht zu empfehlen, da dies bei zukünftigen Versionen des Systems möglicherweise nicht unterstützt wird. Wie im obigen Abschnitt **BEISPIEL** sind **a** und **b** Optionen, und die Option **o** erfordert ein Optionsargument:

cmd -aboxxx file (Verstoß gegen Regel 5: Optionen mit Optionsargumenten dürfen nicht mit anderen Optionen zusammenstehen)

cmd -ab -oxxx file (Verstoß gegen Regel 6: nach einer Option, die ein Optionsargument enthält, muß eine Leerstelle folgen)

Änderung des Werts der Shell-Variablen **OPTIND** oder Analyse von verschiedenen Argument-Sätzen kann zu unerwarteten Ergebnissen führen.

DIAGNOSE

getopts gibt eine Fehlermeldung auf die Standardfehlerausgabe aus, wenn ein Optionsbuchstabe vorkommt, der nicht in *optstring* enthalten ist.

BEZEICHNUNG

glossary – Definitionen von allgemeinen UNIX-System-Begriffen und -Symbolen

ÜBERSICHT

[help] glossary [term]

BESCHREIBUNG

glossary , ein Help-Kommando des UNIX Systems, gibt Definitionen technischer Begriffe und Symbole, die bei UNIX üblich sind.

Ohne ein Argument zeigt *glossary* ein Menü, das die Begriffe und Symbole angibt, die momentan im *glossary* enthalten sind. Ein Benutzer kann einen der Begriffe wählen oder durch Eingabe von q (für "quit") in die Shell zurückkehren. Wenn ein Begriff ausgewählt wird, wird die Definition herausgesucht und angezeigt. Bei Auswahl des entsprechenden Menüs wird das Verzeichnis der Begriffe und Symbole wieder angezeigt.

Die Definition eines Begriffs kann auch direkt von der Shell-Ebene (wie oben gezeigt) angefordert werden, wobei die Definition herausgesucht das Verzeichnis der Begriffe und Symbole aber nicht angezeigt wird. Einige der Symbole müssen, wenn sie von der Shell-Ebene aus angefordert werden, quotiert werden, damit die Funktion das Symbol versteht. Nachfolgend ist eine Liste angegeben, die die Symbole und die entsprechende Quotierung enthält.

SYMBOL	ESCAPE-FOLGE
""	\"\"
''	\'\'
[]	\[\]
` `	\` \`
#	\#
&	\&
*	*
\	\\
|	\\|

Durch Eingabe von ! und dem auszuführenden Kommando kann der Benutzer von jedem Bildschirm des Help-Kommandos ein Kommando über die Shell (*sh*(1)) ausführen. Wird das ausgeführte Kommando in der ersten Eingabe-Aufforderungs-Stufe eingegeben, wird der Bildschirm neu geschrieben. Bei Eingabe von anderen Eingabe-Aufforderungs-Stufen wird nur die Eingabe-Aufforderung neu ausgegeben.

Standardmäßig scrollt das Help-Kommando die Daten, die dem Benutzer gezeigt werden. Bevorzugen Sie einen leeren Bildschirm vor Ausgabe der Daten (kein Scrollen), muß die Shell-Variable **SCROLL** auf **no** gesetzt und exportiert werden, so daß sie Teil Ihrer Umgebung wird. Dies geschieht durch Hinzufügung der folgenden Zeile zu Ihrer Datei *.profile* (vgl. *profile* (4)): " export SCROLL ; SCROLL=no". Sollten Sie später Scrollen bevorzugen, muß **SCROLL** auf **yes** gesetzt werden.

Informationen über die Help-Kommando (*starter*, *locate*, *usage*, *glossary*, und *help*) finden Sie in den entsprechenden Handbuchseiten.

SIEHE AUCH: help(1), helpadm(1M),
locate(1), sh(1), starter(1), usage(1).
term(5) im *Programmer's Reference Manual.*

ACHTUNG!
Wird die Shell-Variable **TERM** (vgl. *sh* (1)) in der Datei *.profile* des Benutzers nicht gesetzt, hat **TERM** standardmäßig den Wert des Terminal-Typs 450 (ein Hartkopie-Terminal). Eine Liste der gültigen Terminal-Typen ist in *term* (5) enthalten.

BEZEICHNUNG

graph – einen Graph zeichnen

ÜBERSICHT

graph [options]

BESCHREIBUNG

Ohne Optionen liest *graph* aus der Standardeingabe Zahlenpaare als Abszissen und Ordinaten eines Graphs. Aufeinanderfolgende Punkte werden durch gerade Linien verbunden. Der Graph wird codiert auf die Standardausgabe ausgegeben, um von den *tplot* (1G) Filtern gezeichnet werden zu können.

Wenn den Koordinaten eines Punkts eine nicht numerische Zeichenfolge folgt, wird diese Zeichenfolge als Beschriftung, die bei diesem Punkt anfängt, ausgegeben. Beschriftungen können mit Anführungszeichen (") versehen werden. In diesem Fall können sie leer sein oder Leerzeichen und Zahlen enthalten; Beschriftungen enthalten nie New-Line-Zeichen.

Die folgenden Optionen werden je als ein getrenntes Argument erkannt:

-a Liefert die Abszissen automatisch (sie fehlen bei der Eingabe); der Abstand wird durch das nächste Argument (Standard ist 1) angegeben. Ein zweites wahlweises Argument ist der Startpunkt für automatische Abszissen (Standardwert 0 oder eine niedrigere Grenze wird durch **-x** angegeben).

-b Nach jeder Beschriftungsangabe wird der Graph unterbrochen.

-c Die im nächsten Argument angegebene Zeichenreihe ist die Standardbeschriftung für jeden Punkt.

-g Nächstes Argument ist Rasterart, 0 kein Raster, 1 Rahmen, 2 volles Raster (standardmäßig).

-l Nächstes Argument ist Beschriftung für den Graph.

-m Nächstes Argument ist die Art der Verbindungslinien: 0 nicht verbunden, 1 verbunden (Standard). Einige Geräte erzeugen unterscheidbare Linienarten bei weiteren kleinen ganzen Zahlen (z. B. bedeutet bei Tektronix 4014: 2=gepunktet, 3=gestrichelt/gepunktet, 4=klein gestrichelt, 5=groß gestrichelt).

-s Bildschirm speichern, nicht vor dem Plotten löschen.

-x [l] Wenn **l** anwesend ist, ist die x-Achse logarithmisch. Nächstes (oder die nächsten zwei) Argument(e) ist (sind) untere (und obere) *x* Grenzen. Drittes Argument, wenn vorhanden, ist Rasterabstand auf der *x* Achse. Normalerweise werden diese Angaben automatisch bestimmt.

-y [l] wie bei *x*.

-h Nächstes Argument ist die Höhenangabe (prozentual).
-w Dasselbe für Breite.
-r Nächstes Argument gibt prozentual an, wieviel vor dem Plotten nach rechts gegangen werden soll.
-u Wie oben, Bewegung jedoch nach oben vor dem Plotten.
-t Überlagern der horizontalen und senkrechten Achsen. (Die Option **-x** bezieht sich nun auf die senkrechte Achse.)

Eine Legende zur Anzeige des Rasterbereichs wird bei Rastern angezeigt, es sei denn, die Option **-s** ist anwesend. Wenn eine angegebene untere Grenze die obere Grenze überschreitet, wird die Achse umgekehrt.

SIEHE AUCH:

graphics(1g), spline(1g), tplot(1g).

FEHLER

graph speichert alle Punkte intern und entfernt diejenigen, für die kein Platz vorhanden ist.
Segmente, die Grenzen überschreiten, werden entfernt und nicht im Fenster angezeigt.
Logarithmische Achsen können nicht umgekehrt werden.

BEZEICHNUNG

graphics – Zugriff auf grafische und numerische Kommandos

ÜBERSICHT

graphics [-r]

BESCHREIBUNG

graphics setzt den Pfadnamen **/usr/bin/graf** vor den aktuellen Wert **$PATH**, ändert das Shell-Eingabe-Aufforderungszeichen auf ^ und führt eine neue Shell aus. Das Verzeichnis **/usr/bin/graf** enthält alle Kommandos des Graphik-Subsystems. Wenn die Option **-r** angegeben ist, wird der Zugriff auf die grafischen Kommandos in einer eingeschränkten Umgebung eingerichtet; d. h. **$PATH** wird auf

:/usr/bin/graf:/rbin:/usr/rbin

gesetzt und die eingeschränkte Shell, *rsh*, wird aufgerufen. Zur Wiederherstellung der Umgebung, wie sie vor Aufruf des Kommandos *graphics* war, muß **EOT** (Control-d an den meisten Terminals) eingegeben werden. Zum Abmelden aus dem Grafik-Subsystem wird **quit** eingegeben.

Das Format der Kommandozeile bei einem *graphics* Kommando ist *command name*, gefolgt von *argument*(s). Ein Argument *argument* kann ein Dateiname *file name* oder eine Folge von Optionen *option string* sein. Ein *file name* ist der Name einer beliebigen UNIX-System-Datei, außer den Dateien, die mit – beginnen. Der *file name* – ist der Name für die Standardeingabe. Ein *option string* besteht aus –, gefolgt von einer oder mehreren Optionen *option*(s). Eine *option* besteht aus einem Schlüsselbuchstaben, dem möglicherweise ein Wert folgt. *options* können durch Kommas getrennt werden.

Die grafischen Kommandos sind in vier Gruppen untergliedert.

Kommandos, die numerische Daten manipulieren und zeichnen; vgl. *stat*(1G).

Kommandos, die Inhaltsverzeichnisse erzeugen; vgl. *toc*(1G).

Kommandos, die sich auf grafische Geräte beziehen; vgl. *gdev*(1G) und *ged*(1G).

Eine Sammlung von grafischen Dienstprogramm-Kommandos; vgl. *gutil*(1G).

Eine Liste der *graphics* Kommandos kann durch Eingabe von **whatis** in der *graphics* Umgebung erzeugt werden.

SIEHE AUCH:

gdev(1G) ged(1G) gutil(1G) stat(1G) toc(1G)
gps(4) im *Programmer's Reference Manual.*

BEZEICHNUNG

greek - Filter für die Auswahl des Terminals

ÜBERSICHT

greek [-Tterminal]

BESCHREIBUNG

greek ist ein Filter, der den erweiterten Zeichensatz sowie die halbzeiligen und Rückwärtsbewegungen des Teletype, Modell 37 mit 128 Zeichen für bestimmte andere Terminals neu interpretiert. Wenn notwendig und möglich, werden Sonderzeichen durch Überschreiben simuliert. Wenn das Argument ausgelassen wird, versucht *greek* die Umgebungsvariable **$TERM** (vgl. *environ*(5)) zu benutzen. Zur Zeit werden die folgenden Terminal-Typen erkannt:

300	DASI 300.
300-12	DASI 300 mit einem Zeichenabstand von 12 Zeichen pro Zoll (12-pitch).
300s	DASI 300s.
300s-12	DASI 300s mit einem Zeichenabstand von 12 Zeichen pro Zoll (12-pitch).
450	DASI 450.
450-12	DASI 450 mit einem Zeichenabstand von 12 Zeichen pro Zoll (12-pitch).
1620	Diablo 1620 (alias DASI 450).
1620-12	Diablo 1620 (alias DASI 450) mit einem Zeichenabstand von 12 Zeichen pro Zoll (12-pitch).
2621	Hewlett-Packard 2621, 2640 und 2645.
2640	Hewlett-Packard 2621, 2640 und 2645.
2645	Hewlett-Packard 2621, 2640 und 2645.
4014	Tektronix 4014.
hp	Hewlett-Packard 2621, 2640 und 2645.
tek	Tektronix 4014.

DATEIEN

/usr/bin/300
/usr/bin/300s
/usr/bin/4014
/usr/bin/450
/usr/bin/hp

SIEHE AUCH:

300 (1), 4014(1), 450(1), hp(1), tplot(1G).
eqn(1), mm(1), nroff(1) im Handbuch Dokumentations-Tools
greek(5), term(5) im *Programmer's Reference Manual.*

BEZEICHNUNG

grep – eine Datei nach einem Textmuster durchsuchen

ÜBERSICHT

grep [options] limited regular expression [file ...]

BESCHREIBUNG

grep durchsucht Dateien nach einem Textmuster und gibt alle Zeilen an, die dieses Muster enthalten. *grep* verwendet eingeschränkte reguläre Ausdrücke (Ausdrücke, die Zeichenfolgen über einer Teilmenge aller möglichen alphanumerischen Zeichen und Sonderzeichen bezeichnen,) im Stil von *ed (1)*, um Muster zu suchen. Es wird ein kompakter nichtdeterministischer Algorithmus verwendet.

Bei Verwendung der Zeichen $, *, [, ^, |, (,), und \ wird zu Vorsicht geraten, da sie auch Bedeutung für die Shell haben. Es ist am sichersten, den ganzen *limited regular expression* in einfache Anführungszeichen zu setzen ´...´.

Wenn keine Dateien angegeben sind, nimmt *grep* die Standardeingabe an. Normalerweise wird jede gefundene Zeile auf die Standardausgabe kopiert. Wenn es sich um mehr als eine Eingabedatei handelt, wird der Dateiname vor jeder gefundenen Zeile angegeben.

Die Optionen der Kommandozeile lauten:

- **-b** Vor jeder Zeile steht die Blocknummer, bei der sie gefunden wurde. Dies kann zum Auffinden von Blocknummern im Kontext (erster Block ist O) zweckmäßig sein.
- **-c** Es wird nur die Anzahl der Zeilen, die das Muster enthalten, angegeben.
- **-i** Ignoriert Groß-/Kleinschreibung während der Vergleiche.
- **-l** Die Namen der Dateien mit übereinstimmenden Zeilen werden nur einmal angezeigt und sind durch New-Line-Zeichen getrennt. Wiederholt die Namen der Dateien nicht, wenn das Muster mehr als einmal gefunden wird.
- **-n** Vor jeder Zeile wird ihre entsprechende Zeilennummer in der Datei geschrieben (erste Zeile ist 1).
- **-s** Unterdrückt Fehlermeldungen bei nicht vorhandenen oder unlesbaren Dateien
- **-v** Gibt alle Zeilen aus außer denen, die das Muster enthalten

SIEHE AUCH:

ed(1), egrep(1), fgrep(1), sed(1), sh(1).

DIAGNOSE

Der Endestatus ist 0, wenn Übereinstimmungen gefunden werden, 1 wenn keine gefunden werden, 2 für Syntaxfehler oder nicht zugreifbare Dateien (selbst wenn Übereinstimmungen gefunden wurden).

FEHLER

Zeilen sind auf BUFSIZ-Zeichen begrenzt; längere Zeilen werden getrennt. BUFSIZ ist in **/usr/include/stdio.h** definiert.
Wenn eine Zeile Nullen enthält, sucht *grep* nur Übereinstimmungen bis zur ersten Null; wenn Übereinstimmung festgestellt wird, wird die ganze Zeile angegeben.

BEZEICHNUNG

gutil – grafische Dienstprogramme

gutil: Dienstprogramme – bel, cvrtopt, gd, gtop, pd, ptog, quit, remcom, whatis, yoo

ÜBERSICHT

command-name [options] [files]

BESCHREIBUNG

Nachstehend wird eine Liste verschiedener geräteunabhängiger Dienstprogramm-Kommandos aufgeführt, die in **/usr/bin/graf** aufzufinden sind. Wenn keine Dateien *files* angeben sind, erfolgt die Eingabe von der Standardeingabe. Ausgabe erfolgt auf die Standardausgabe. Grafische Daten werden im GPS-Format gespeichert; vgl. *gps*(4).

bel schickt Bell-Zeichen zum Terminal

cvrtopt [=s*string* f*string* i*string* t*string*] [*args*] – Optionsumwandler

Cvrtopt formatiert *args* (gewöhnlich die Argumente der Kommandozeile einer aufrufenden Shell-Prozedur) neu, um die Verarbeitung durch Shell-Prozeduren zu erleichtern. Ein *arg* ist entweder ein Dateiname (eine Zeichenfolge, die nicht mit einem – beginnt und nicht – allein) oder eine Optionszeichenfolge (eine Zeichenfolge von Optionen, die mit einem – beginnt). Die Ausgabe ist in der Form:

–option –option . . . file name(s)

Alle Optionen erscheinen einmalig und stehen vor Dateinamen. Optionen, die Werte (z.B. –r1.1) oder zwei Buchstaben enthalten, müssen in Form von Optionen für *cvrtopt* beschrieben werden.

cvrtopt wird gewöhnlich in nachstehender Form mit *set* in der ersten Zeile einer Shell-Prozedur verwendet:
set – cvrtopt = [*options*] **$@**
Options für *cvrtopt* sind:

s*string*	*string* akzeptiert Zeichenfolgen als Argumente.
f*string*	*string* akzeptiert Gleitkommazahlen als Argumente.
i*string*	*string* akzeptiert ganze Zahlen als Argumente.
t*string*	*string* ist eine Option mit zwei Buchstaben, die kein Argument hat
string	ist ein Optionsname mit einem oder zwei Buchstaben.

gd [GPS *files*] – GPS dump

Gd gibt eine lesbare Form von GPS aus.

gtop [**-r***n* **u**] [GPS *files*] – GPS für *plot*(4) Filter
Gtop formt ein GPS in *plot*(4)-Kommandos um, die durch *plot*-Filter angezeigt werden können. GPS Objekte werden übersetzt, wenn sie innerhalb des Fensters liegen, das die erste Datei *file* eingrenzt, sofern nicht eine Option *option* angegeben ist.
Optionen:

r*n* übersetzt Objekte im GPS-Feld (region) *n*.

u übersetzt alle Objekte im GPS-Universum (universe).

pd [*plot*(5) *files*] – *plot*(4) dump
Pd gibt eine lesbare Form der grafischen Kommandos im *plot*(4)-Format heraus.

ptog [*plot*(5) *files*] Filter von – *plot*(4) nach GPS
Ptog formt *plot*(4)-Kommandos in ein GPS um.

quit – Sitzung beenden

remcom [*files*] – entfernt Kommentare
Remcom kopiert die Eingabe auf die Ausgabe, wobei Kommentare entfernt werden. Kommentare sind wie in C definiert, (d. h. /* Kommentar */).

whatis [**-o**] [*names*] – kurze rechnerabhängige Dokumentation
Whatis druckt eine kurze Beschreibung für jeden angegebenen Namen *name*. Wenn kein *name* angegeben ist, wird das aktuelle Beschreibungsverzeichnis *names* ausgegeben. Das Kommando **whatis *** gibt alle Beschreibung aus.
Option:

o gibt nur Kommandooptionen (options) an

yoo *file* – Pipeverbindung
Yoo ist eine Pipe-Basisprozedur, die die Ausgabe einer Pipe in eine Datei *file* ablegt, die in der Pipe verwendet wird. Es ist zu beachten, daß dies ohne *yoo* gewöhnlich nicht möglich ist, da dieselbe Datei gleichzeitig gelesen und beschrieben wird.

SIEHE AUCH

graphics(1G).
gps(4), plot(4) im *Programmer's Reference Manual.*

BEZEICHNUNG

hostid – setzen oder ausgeben der ID des aktuellen Host-Systems

ÜBERSICHT

hostid [identifier]

BESCHREIBUNG

Das *hostid* Kommando gibt die ID des aktuellen Hosts hexadezimal aus. Dieser numerische Wert sollte über allen Hosts eindeutig sein und wird normalerweise als Host-Internet-Adresse beutzt. Der Super-User kann durch Angabe einer hexadezimalen Zahl die Host-ID setzen; dies wird normalerweise im Startscript /etc/rc.local durchgeführt.

SIEHE AUCH

gethostid(2), sethostid(2)

BEZEICHNUNG

hostname – setzen oder ausgeben des aktuellen Host-Namen

ÜBERSICHT

hostname [nameofhost]

BESCHREIBUNG

Das *hostname* Kommando gibt den Namen des aktuellen Hosts aus, wie er vor dem "Login"-Prompt eingegeben wurde. Der Super-User kann über das Argument den Host-Namen setzen; normalerweise wird dies vom Startscript/etc/rc durchgeführt.

BEMERKUNG

Der *hostname* gehört zum DOD-Standard RFC 810:

"a" - "z"
"0" - "9"
"-"
"."
kein Platz
keine Fallunterscheidung

SEE ALSO

gethostname(2), sethostname(2)

BEZEICHNUNG

hp – Sonderfunktionen des Hewlett-Packard-Terminals behandeln

ÜBERSICHT

hp [-e] [-m]

BESCHREIBUNG

hp unterstützt Sonderfunktionen der Hewlett-Packard-Terminals der Serie 2640, die in erster Linie zur exakten Wiedergabe von *nroff* Ausgabe dienen. Typischerweise wird *hp* in Verbindung mit der Software der DOCUMENTER'S WORKBENCH eingesetzt:

nroff -h files ... | hp

Unabhängig von den Hardware-Eigenschaften Ihres Termials versucht *hp* die Funktionen Unterstreichen und Rückpositionierung richtig auszuführen. Wenn Ihr Terminal über "erweiterte Anzeigefunktionen" verfügt, können Indizes und hochgesetzte Indizes unterschiedlich angezeigt werden. Wenn das Terminal "mathematische Symbole" kennt, können griechische und andere Sonderzeichen angezeigt werden.

Die Optionen sind wie folgt:

-e Es wird angenommen, daß Ihr Terminal über die "erweiterten Anzeigefunktionen" verfügt. Die zusätzlichen Anzeigemodi werden voll genutzt. Zeichen für Überschreiben werden mit dem Modus Unterstreichen dargestellt. Hochgesetzte Indizes werden im halbhellen Modus angezeigt, und Indizes im halbhellen Unterstreich-Modus. Wenn diese Option ausgelassen wird, nimmt *hp* an, daß Ihr Terminal die "erweiterten Anzeigefunktionen" nicht besitzt. In diesem Fall werden alle Zeichen für Überschreiben, Indizes und hochgesetzte Indizes im Invers-Modus, d. h. Dunkel auf Hell anstelle des üblichen Hell auf Dunkel angezeigt.

-m Bewirkt Ausgabereduzierung durch Entfernen von New-Line-Zeichen. 3 oder mehr aufeinanderfolgende New-Line-Zeichen werden in eine Folge von nur 2 New-Line-Zeichen umgewandelt; d. h. aufeinanderfolgende leere Zeilen ergeben nur eine einzige leere Ausgabezeile. Dies ermöglicht Ihnen, mehr tatsächlichen Text am Terminal anzuzeigen.

Hinsichtlich griechischer und anderer Sonderzeichen stellt *hp* den gleichen Zeichensatz wie *300*(1) bereit, außer daß "nicht" nur angenähert durch den rechten Pfeil dargestellt wird und nur die oberste Hälfte des Integralzeichens angezeigt wird.

DIAGNOSE

"Zeile zu lang", wenn die Darstellung einer Zeile 1.024 Zeichen überschreitet.

Die Endecodes sind **0** bei normalem Ende, **2** für alle Fehler.

SIEHE AUCH:

300(1), greek(1).
col(1), eqn(1), nroff(1), tbl(1) im Handbuch Dokumentations-Tools.

FEHLER

Eine "Überschreibfolge" ist definiert als ein abdruckbares Zeichen, gefolgt von einem Rückschritt und danach wieder ein abdruckbares Zeichen. Wenn bei solchen Folgen eines der Zeichen ein Unterstreich-Zeichen ist, wird das andere Zeichen unterstrichen oder invers dargestellt; andernfalls wird nur das erste Zeichen angezeigt (unterstrichen oder invers). Wenn ein Rückschrittzeichen neben einem ASCII- Kontrollzeichen steht, geschieht nichts. Kontrollzeichenfolgen (z. B. Umkehrzeilenvorschübe, Rückschrittzeichen) können Text "verschwinden" lassen; speziell bei Tabellen, die mit *tbl*(1) erstellt wurden und senkrechte Zeilen enthalten, fehlen häufig die Textzeilen,die den "Fuß" einer vertikalen Zeile enthalten, sofern nicht die Eingabe für *hp* zuerst mit *col*(1) aufbereitet wurde (und dann über eine Pipe an *hp* geschickt wurde).

Obwohl einige Bildschirmgeräte numerische, hochgesetzte Indizes bereitstellen, werden diese nicht angezeigt.

BEZEICHNUNG

hpio – Datei-Archivierung auf Band für Hewlett-Packard-Terminals der Serie 2645A

ÜBERSICHT

hpio **-o[rc]** file ...

hpio **-i[rta]** [**-n** count]

BESCHREIBUNG

hpio wurde zur Nutzung der Magnetbandlaufwerke der Hewlett-Packard-Terminals der Serie 2645A entwickelt. Auf einem Magnetband können max. 255 UNIX Systemdateien zur rechnerunabhängige Aufbewahrung oder zur Übertragung auf ein anderes UNIX System gespeichert werden. Die jeweilige Anzahl der Dateien hängt von der Größe der Dateien ab. Eine Datei mit ca. 115.000 Bytes füllt fast ein ganzes Magnetband. Ca. 300 1-Byte Dateien passen auf ein Band, das Terminal kann jedoch nur die ersten 255 Dateien wiederauffinden. Diese Handbuchinformation sollte nicht als Anleitung für den Einsatz von Magnetbändern bei Hewlett-Packard-Terminals der Serie 2645A dienen. Es soll vielmehr ausreichend Information zur Verfügung gestellt werden, um Band-Archive erstellen und lesen zu können und um ein Band für den Zugriff auf eine gewünschte Datei zu positionieren.

hpio **-o** (copy out) kopiert die angegebenen Dateien *file*(s), zusammen mit Pfadnamen und Status-Daten auf ein Magnetband an Ihrem Terminal (es wird angenommen, daß das Gerät auf Bandanfang oder direkt nach einer Bandabschnittsmarke positioniert ist). Das linke Magnetbandlaufwerk wird standardmäßig verwendet. Jede Datei *file* wird in eine eigene Banddatei geschrieben und mit einer Bandabschnittsmarke abgeschlossen. Wenn *hpio* endet, wird das Magnetband hinter die letzte geschriebene Bandabschnittsmarke positioniert.

hpio **-i** (copy in) holt eine Datei(en) von einem Magnetband (es wird angenommen, daß es am Anfang einer Datei positioniert ist, die vorher von einem **hpio** **-o** geschrieben wurde). Standardmäßig wird die nächste Datei vom Band im linken Magnetbandlaufwerk geholt.

hpio positioniert das Magnetband am Ende immer hinter die letzte Datei, die vom Band gelesen oder aufs Band geschrieben wurde. Bänder müssen immer rückgespult werden, ehe das Gerät ausgeschaltet wird. Zum Rückspulen eines Magnetbands wird der grüne Funktionsknopf betätigt, dann Funktionstaste 5, und danach Auswahl des entsprechenden Magnetbandlaufwerks durch Betätigung von Funktionstaste 5 für das linke Magnetbandlaufwerk oder Funktionstaste 6 für das rechte Magnetbandlaufwerk. Wenn mehrere Dateien auf ein Band archiviert wurden, kann das Band an den Anfang einer speziellen Datei positioniert werden, indem der grüne Funktionsknopf betätigt wird, dann Funkti-

onstaste 8, gefolgt von der Eingabe der gewünschten Dateinummer (1–255) ohne RETURN und schließlich Funktionstaste 5 für das linke Band oder Funktionstaste 6 für das rechte Band. Die gewünschte Dateinummer kann auch durch eine Zahl mit Vorzeichen relativ zu der aktuellen Dateinummer angegeben werden.

Die Bedeutung der verfügbaren Optionen ist:

r Das rechte Magnetbandlaufwerk benutzen.

c Enthält eine Prüfsumme am Ende von jeder Datei *file*. Die Prüfsumme wird immer von **hpio −i** für jede Datei, die mit dieser Option von **hpio −o** geschrieben wurde, geprüft .

n *count* Die Anzahl der vom Band zuholenden Eingabedateien ist auf *count* gesetzt. Wenn diese Option nicht angegeben ist, ist *count* standardmäßig 1. Um alle Dateien vom Band zu holen, kann ein beliebig großes *count* angegeben werden. *hpio* stoppt an der Datenendemarke auf dem Band.

t Gibt nur ein Inhaltsverzeichnis aus. Es werden keine Dateien erstellt. Die Ausgabe gibt die Dateigröße in Bytes, den Dateinamen, die Dateizugriffsmoden und Vorhandensein einer Prüfsumme an.

a Fragt vor Erstellung einer Datei. **hpio −i** gibt normalerweise die Dateigröße und den Namen an, erstellt und liest die Datei ein und gibt eine Status-Meldung aus, wenn die Datei eingelesen worden ist. Wenn die Datei eine Prüfsumme enthält, wird angegeben, ob die Prüfsumme mit dem berechneten Wert übereinstimmt. Bei dieser Option wird nach der Angabe der Dateigröße und dem Namen ein **?** ausgegeben. Antworten mit **y** oder **Y** bewirken ein Kopieren der Datei wie oben beschrieben. Bei allen anderen Antworten wird die Datei übersprungen.

DATEIEN

/dev/tty?? zum Blockieren von Meldungen während auf ein Band zugegriffen wird

SIEHE AUCH:

cu(1C).

DIAGNOSE

BREAK
Ein Unterbrechungssignal hat den Prozeß abgebrochen

Can't create '*file*'.
Zugriffsberechtigungen des Dateisystems erlauben keine Erstellung von *file*.

Can't get tty options on stdout.
Es war *hpio* nicht möglich, die Eingabe-Ausgabe-Kontrolleinstellungen für das Bildschirmgerät zu erhalten.

Can't open *'file'*.
Kein Zugriff auf *file* zum Kopieren auf Band.

End of Tape.
Auf dem Band stand bei Leseanforderung kein Datensatz zur Verfügung. Eine Datenendemarke ist der übliche Grund hierfür; diese Meldung kann auch ausgegeben werden, wenn auf das falsche Magnetbandlaufwerk zugegriffen wird und wenn kein Band vorhanden ist.

'file' not a regular file. *File* ist ein Verzeichnis oder eine andere Gerätedatei. Nur normale Dateien werden auf Band kopiert.

Readcnt = *rc*, termcnt = *tc*.
hpio erwartete, beim nächsten Block auf dem Band *rc* Bytes zu lesen, aber der Block enthielt *tc* Bytes. Dies kann vorkommen, wenn das Band nicht ordnungsgemäß positioniert ist oder durch einen Fehler im Block, der durch Störung von anderen Ein/Ausgaben am Bildschirm verursacht wurde.

Skip to next file failed.
Überspringen einer Bandabschnittsmarke nicht erfolgreich.

Tape mark write failed.
Schreiben einer Bandabschnittsmarke am Ende einer Datei nicht erfolgreich.

Write failed.
Schreiben auf Band nicht erfolgreich. Der Grund für diese Meldung ist meistens eine falsche Angabe des Magnetbandlaufwerks bzw. das Band ist ganz abgespult bzw. bei einem Versuch, auf ein schreibgeschütztes Band zu schreiben.

ACHTUNG!

Ein/Ausgabe-Operationen auf Band können ungültige Daten kopieren, wenn andere Ein/Ausgaben am Bildschirm vorgenommen werden. Es sollten keine Eingaben vorgenommen werden, während *hpio* läuft. Während es läuft, schaltet *hpio* Schreiberlaubnis für andere Systembenutzer aus; Prozesse, die asynchron mit Ihrem Bildschirmgerät gestartet wurden, können jedoch immer noch Störungen verursachen. Wenn diese Art von Störung beim Schreiben eines Bands eintritt, erscheinen meistens die Zeichen, die kopiert werden sollten, am Bildschirm.

Die Tastatur einschließlich der BREAK -Taste des Bildschirmgeräts wird bei Band-Schreib-Operationen gesperrt; die BREAK -Taste ist nur zwischen Schreiboperationen funktionsfähig.

Für den Dialog mit den Magnetbandlaufwerken muß *hpio* vollständige Kontrolle der Bildschirmattribute besitzen. Wechselwirkung mit Kommandos wie zum Beispiel *cu*(1C) kann eine erfolgreiche Ausführung verhindern.

FEHLER

Einige Binärdateien enthalten Folgen, die das Gerät nicht handeln kann.

Ein **hpio -i** , das auf die Datenendemarke auf dem Band trifft (z. B. beim Durchsuchen des ganzen Bands mit **hpio -itn 300**), positioniert das Band hinter der Datenendemarke. Wenn danach **hpio -o** ausgeführt wird, können die Daten nicht wieder aufgefunden werden. Eine manuelle Neupositionierung des Bandes mit der Bildschirmoperation FIND FILE -1 (Betätigen des grünen Funktionsknopfes, Funktionstaste 8, und dann Funktionstaste 5 für das linke Band oder Funktionstaste 6 für das rechte Band) ist dann vor Starten von **hpio -o** erforderlich.

Wenn *hpio* während des Schreibvorgangs auf Band eine Unterbrechung empfängt, kann die Tastatur am Bildschirmgerät gesperrt bleiben. In diesem Fall kann die Tastatur mit der Bildschirmtaste RESET TERMINAL wieder freigegeben werden.

BEZEICHNUNG

id – Benutzer- und Gruppennummern und Namen ausgeben

ÜBERSICHT

id

BESCHREIBUNG

id gibt die Benutzer- und die Gruppenummern und die entsprechenden Namen des aufrufenden Prozesses aus. Wenn die effektiven und realen Nummern unterschiedlich sind, werden beide ausgegeben.

SIEHE AUCH:

logname(1) im *User's Reference Manual*,
getuid(2) im *Programmer's Reference Manual*.

BEZEICHNUNG

ipcrm – Kennung einer Message-Queue (Nachrichten-Warteschlange), eines Semaphor-Satzes oder einer Shared-Memory-ID (gemeinsam benutzter Speicherplatz) entfernen

ÜBERSICHT

ipcrm [*options*]

BESCHREIBUNG

ipcrm entfernt eine oder mehrere angegebene Kennungen für Messages (Nachrichten), Semaphore oder Shared-Memory (gemeinsam benutzter Speicherplatz). Die Kennungen werden durch folgenden *options* angegeben:

-q *msqid* entfernt die Message-Queue-Kennung *msqid* vom System und entfernt die Message-Queue und die entsprechende Datenstruktur.

-m *shmid* entfernt die Shared-Memory-Kennung *shmid* vom System. Das gemeinsam benutzte Speicherplatzsegment und die entsprechende Datenstruktur werden nach der letzten Freigabe entfernt.

-s *semid* entfernt die Semaphor-Kennung *semid* vom System und entfernt den Semaphordatensatz und die entsprechende Datenstruktur.

-Q *msgkey* entfernt die Message-Queue-Kennung, die mit key (Schlüssel) *msgkey* erstellt wurde, und entfernt die Message-Queue und die entsprechende Datenstruktur.

-M *shmkey* entfernt die Shared-Memory-Kennung Speicherkennung, die mit key (Schlüssel) *shmkey* erstellt wurde. Das gemeinsam benutzte Speichersegment und die entsprechende Datenstruktur werden nach der letzten Freigabe entfernt.

-S *semkey* entfernt die Semaphor-Kennung, die mit key (Schlüssel) *semkey* erstellt wurde, und entfernt den Semaphor-Datensatz und die entsprechende Datenstruktur.

Einzelheiten über diese Entfernungen sind in *msgctl*(2), *shmctl*(2) und *semctl*(2) beschrieben. Die Kennungen und Schlüssel (keys) können mit Hilfe von *ipcs*(1) erfragt werden.

SIEHE AUCH

ipcs(1).
msgctl(2), msgget(2), msgop(2), semctl(2), semget(2), semop(2), shmctl(2), shmget(2), shmop(2) im *Programmer's Reference Manual*.

BEZEICHNUNG

ipcs – Status der Interprozeß-Kommunikation anzeigen

ÜBERSICHT

ipcs [options]

BESCHREIBUNG

ipcs gibt bestimmte Daten über aktive Interprozeß Kommunikation aus. Ohne *options* wird Information über Message-Queues, Shared-Memory und Semaphoren, die momentan aktiv im System sind, im Kurzform ausgegeben. Andernfalls wird die Ausgabe durch folgenden *options* gesteuert:

-q Daten über aktive Message-Queues ausgegeben.

-m Daten über aktive, Shared-Memory-Segmente ausgeben.

-s Daten über aktive Semaphoren ausgegeben.

Wenn eine der Optionen **-q**, **-m** oder **-s** angegeben ist, werden nur die Daten der angegebenen Optionen ausgegeben. Wenn keine dieser drei Optionen angegeben ist, werden Daten über alle drei Optionen entsprechend den nachstehenden Optionen ausgegeben:

-b Größte zulässige Größe angeben. (Für Message-Queues Höchstanzahl an Bytes für Messages in einer queue für Shared-Memory, Größe des Segments und Anzahl von Semaphoren in jedem Semaphor-Datensatz.) Vgl. nachstehende Bedeutung der Spalten in einer Liste.

-c Login-Namen des Erstellers und Gruppenname ausgeben. Siehe unten.

-o Daten über nicht fertige Anwendungen. (Anzahl der Messages in einer Queue, Gesamtzahl von Bytes in Messages in einer Queue für Message Queues und Anzahl der angeschlossenen Prozesse bei Shared-Memory Segmenten).

-p Prozeßnummern ausgeben. (Bei Message-Queues die Prozeßnummer des letzten Prozesses, der eine Nachricht geschickt hat, und Prozeßnummer des letzten Prozesses, der eine Nachricht empfangen hat, bei Shared-Memory die Prozeßnummer des Vaterprozesses und Prozeßnummer des letzten Prozesses, der das gemeinsam benutzte Speicherplatzsegment benutzt oder freigegeben hatte). Siehe unten.

-t Zeitangaben ausgeben. (Zeitpunkt der letzten Kontrolloperation ausgeben, die die Zugriffsberechtigungen für alle Einrichtungen geändert hat. Zeit der letzten *msgsnd* und *msgrcv* bei Message-

Queues, letzte *shmat* und *shmdt* bei Shared-Memory, letztes *semop*(2) bei Semaphoren.) Siehe unten.

-a Alle Ausgabe *Optionen* verwenden. (Dies ist eine abgekürzte Schreibweise für **-b**, **-c**, **-o**, **-p** und **-t**.).TP **-C** *corefile* Die Datei *corefile* anstelle von **/dev/kmem** verwenden.

-N *namelist*
Das Argument wird als der Name einer alternativen *namelist* angesehen (**/unix** ist Standard).

Die Spaltenüberschriften und die Bedeutung der Spalten in einer *ipcs* -Liste werden nachstehend angegeben; die Buchstaben in runden Klammern zeigen die *options* an, die das Erscheinen der entsprechenden Überschrift veranlassen; **all** bedeutet, daß die Überschrift immer erscheint. Es ist zu beachten, daß diese *options* nur bestimmen, welche Daten für jede Kommunikationsart bereitgestellt werden; sie entscheiden *nicht,* welche Kommunikationsarten aufgelistet werden.

T (all) Kommunikationsart:
- **q** Message-Queue;
- **m** shared memory segment;
- **s** Semaphor.

ID (all) Die Kennung für die Kommunikationsart.

KEY (all) Key (Schlüssel), der bei *msgget*, *semget*, oder *shmget* als Argument benutzt wurde um den Eintrag für die jeweilige Kommunikationsart zu erstellen. (Anmerkung: bei Shared-Memory wird key auf **IPC_PRIVATE0** geändert, wenn das Segment entfernt worden ist, bis alle an das Segment angeschlossenen Prozesse es freigeben.)

MODE (all) Die Zugriffs-Modi: Der Modus besteht aus 11 Zeichen, die wie folgt interpretiert werden:
Die ersten beiden Zeichen lauten:
- **R** wenn ein Prozeß auf ein *msgrcv* wartet;
- **S** wenn ein Prozeß auf ein *msgsnd* wartet;
- **D** wenn das zugeordnete Shared-Memory-Segment entfernt worden ist. Es verschwindet, wenn es der letzte ans Segment angeschlossene Prozeß freigibt;
- **C** falls das zugeordnete Shared-Memory-Segment gelöscht werden soll, beim ersten Anschluß;
- **-** Wenn der entsprechende Sonderschalter nicht gesetzt ist.

Die nächsten 9 Zeichen werden als drei Einheiten mit je drei Bits interpretiert. Die erste Einheit bezieht sich auf die Berechtigung des Eigentümers; die nächste auf die Berechtigung für andere in der Benutzergruppe der Kommunikationsart; und die letzte auf alle anderen. Bei allen Datensätzen zeigt das erste Zeichen die Leseberechtigung an, das zweite Zeichen zeigt die Berechtigung zum Schreiben oder Ändern des Eintrags für die jeweilige Kommunikationsart an, und das letzte Zeichen ist im Moment unbenutzt.

Die Berechtigungen werden wie folgt angezeigt:

- **r** wenn Leseberechtigung gewährt wird;
- **w** wenn Schreibberechtigung gewährt wird;
- **a** wenn Veränderungsberechtigung gewährt wird;
- **–** wenn die jeweilige Berechtigung *nicht* gewährt wird.

OWNER (all) Der Login-Name des Eigentümers des Eintrags für die jeweilige Kommunikationsart.

GROUP (all) Der Gruppenname der Gruppe des Eigentümers des Eintrags für die jeweilige Kommunikationsart.

CREATOR (a,c) Der Login-Name des Erstellers des Eintrags für die jeweilige Kommunikationsart.

CGROUP (a,c) Der Gruppenname der Gruppe des Erstellers des Eintrags für die jeweilige Kommunikationsart.

CBYTES (a,o) Die Anzahl Bytes von momentan unerledigten Nachrichten in der entsprechenden Message-Queue.

QNUM (a,o) Die momentan unerledigte Anzahl von Nachrichten in der entsprechenden Message-Queue.

QBYTES (a,b) Maximale Byteanzahl bei unerledigten Nachrichten in der entsprechenden Message-Queue.

LSPID (a,p) Die Prozeßnummer des letzten Prozesses, der eine Nachricht an die entsprechende Queue geschickt hat.

LRPID (a,p) Die Prozeßnummer des letzten Prozesses, der eine Nachricht von der entsprechenden Queue empfangen hat.

STIME (a,t) Der Zeitpunkt, an dem die letzte Nachricht an die entsprechende Queue geschickt wurde.

RTIME (a,t) Der Zeitpuntk, an dem die letzte Nachricht von der entsprechenden Queue empfangen wurde.

CTIME (a,t) Der Zeitpunkt, zu dem der entsprechende Eintrag erstellt oder geändert wurde.

NATTCH (a,o) Die Anzahl der an das entsprechend gemeinsam benutzte Speicherplatzsegment angeschlossenen Prozesse.

SEGSZ (a,b) Die Größe des entsprechenden gemeinsam benutzten Speicherplatzsegments.

CPID (a,p) Die Prozeßnummer des Erstellers des Shared-Memory Eintrags.

LPID (a,p) Die Prozeßnummer des letzten Prozesses, der das gemeinsam benutzte Speicherplatzsegment belegt oder freigibt.

ATIME (a,t) Der Zeitpunkt, zu dem der letzte Anschluß an das entsprechende gemeinsam benutzte Speicherplatzsegment beendet wurde.

DTIME (a,t) Der Zeitpunkt, zu dem die letzte Freigabe vom entsprechende gemeinsam benutzten Speicherplatzsegment beendet wurde.

NSEMS (a,b) Die Anzahl Semaphoren in dem Semaphor-Satz, der dem Semaphoreintrag zugeordnet ist.

OTIME (a,t) Der Zeitpunkt, zu dem die letzte Semaphoroperation beendet war für den entsprechenden Semaphoreintrag.

DATEIEN

/unix	Systemnamensliste
/dev/kmem	Speicherplatz
/etc/passwd	Benutzernamen
/etc/group	Gruppennamen

SIEHE AUCH:

msgop(2), semop(2), shmop(2) im *Programmer's Reference Manual.*

FEHLER

Während *ipcs* läuft, können Veränderungen eintreten; die Ausgabe entspricht der Realität nicht immer ganz.

BEZEICHNUNG

ismpx – Statusangabe eines window-fähigen Terminals ausgeben

ÜBERSICHT

ismpx [-s]

BESCHREIBUNG

Das Kommando *ismpx* gibt an, ob die Standardeingabe an einen Multiplexed-Kanal *xt* (7) angeschlossen ist; d. h. ob sie mit *layers*(1) läuft oder nicht. Es wird für Shell-Skripte empfohlen, die Programme zu einem windowfähigen Terminal weitersenden oder von der Bildschirmgröße abhängig sind.

ismpx druckt **yes** und gibt **0** zurück, wenn es unter *layers*(1) aufgerufen wird, andernfalls wird **no** gedruckt und **1** zurückgegeben.

-s keine Ausgabe; nur Zurückgabe des entsprechenden Endestatus.

ENDESTATUS

Gibt **0** zurück, wenn es unter *layers*(1) aufgerufen wird, **1** sonst.

SIEHE AUCH:

layers(1), jwin(1).
xt(7) im *Administrator's Reference Manual.*

BEISPIEL

```
if ismpx -s
        then
                jwin
        fi
```

BEZEICHNUNG

join – (relationaler) Datenbank-Operator

ÜBERSICHT

join [options] file1 file2

BESCHREIBUNG

join bildet auf der Standardausgabe eine Verbindung von zwei Relationen, die durch die Zeilen der Dateien *file1* und *file2* bestimmt sind. Falls *file1* das Zeichen – ist, wird die Standardeingabe verwendet.

Die Felder von *file1* und *file2*, über die die Dateien verbunden werden sollen, müssen aufsteigend gemäß ASCII sortiert sein; normalerweise ist dies das erste Feld in jeder Zeile [vgl. *sort*(1)].

Für jedes Zeilenpaar aus *file1* und *file2*, die identische Verbindungsfelder haben, gibt es eine Zeile in der Ausgabe. Die Ausgabezeile besteht normalerweise aus dem gemeinsamen Feld, anschließend die restlichen Zeilen aus *file1* und dann die restlichen Zeilen aus *file2*.

Die Standardtrennsymbole der Eingabefelder sind Leerzeichen, Tabulator oder New-Line-Zeichen. In diesem Fall zählen mehrere Trennsymbole als ein Feld-Trennsymbol, und führende Trennsymbole werden ignoriert. Das Standardtrennsymbol der Ausgabefelder ist ein Leerzeichen.

Einige der unten aufgeführten Optionen verwenden das Argument *n*. Dieses Argument soll eine **1** oder eine **2** sein, wobei es sich jeweils auf *file1* oder *file2* bezieht. Die folgenden Optionen werden erkannt:

–a*n* Erzeugt zusätzlich zur normalen Ausgabe eine Ausgabezeile für jede Zeile in der Datei *n*, für die es keine entsprechende Zeile in der anderen Datei gibt, wobei *n* den Wert 1 oder 2 hat.

–e *s* Ersetzt leere Ausgabefelder durch die Zeichenfolge *s*.

–j*n m* Stellt die Verknüpfung auf dem *m-ten* Feld der Datei *n* her. Falls *n* fehlt, wird das *m-te* Feld in jeder Datei benutzt. Feldnumerierung beginnt mit **1**.

–o *list* Jede Ausgabezeile umfaßt die in *list* bezeichneten Felder, bei denen jedes Element die Form *n.m* besitzt, wobei *n* eine Dateinummer und *m* eine Feldnummer ist. Das gemeinsame Feld wird nicht angezeigt, sofern es nicht ausdrücklich verlangt wird.

–t*c* Das Zeichen *c* ist als Trennsymbol (Tabulatorzeichen) zu verwenden. Jedes Vorkommen von *c* in einer Zeile ist signifikant. Das Zeichen *c* wird als Feld-Trennsymbol für Eingabe und Ausgabe verwendet.

BEISPIEL

Die folgende Kommandozeile bewirkt die Verbindung der Paßwortdatei und der Gruppendatei, wenn sie in der numerischen Gruppennummer übereinstimmen und gibt dann den Login-Namen, -Gruppennamen und das -Datenverzeichnis aus. Es wird angenommen, daß die Dateien gemäß ASCII in den Gruppennummer-Feldern sortiert sind.

```
join -j1 4 -j2 3 -o 1.1 2.1 1.6 -t: /etc/passwd /etc/group
```

SIEHE AUCH

awk(1), comm(1), sort(1), uniq(1).

FEHLER

Bei Standardfeld-Trennung ist die Sortierordnung wie bei **sort -b**; bei **-t** wird voll sortiert.

Die Konventionen von *join*, *sort*, *comm*, *uniq* und *awk*(1) passen überhaupt nicht zusammen.

Numerische Dateinamen können Konflikte hervorrufen, wenn die **-o** Option direkt vor den Dateinamen angegeben wird.

BEZEICHNUNG

jterm – Layer für Terminals mit Fenstertechnik zurücksetzen

ÜBERSICHT

jterm

BESCHREIBUNG

Das Kommando *jterm* wird zum Rücksetzen eines Layer für windowfähige Terminals verwendet nach dem Weitersenden eines Terminal-Programms, das die Terminal-Attribute des Layers ändert. Dieses Kommando ist sinnvoll im Zusammenhang mit *layers*(1). In der Praxis wird es normalerweise benutzt, um den Standarddatenstations-Emulator wieder zu starten, nachdem ein spezielles Anwendungsprogramm einen anderen verwendet hatte. Zum Beispiel wird beim AT&T Teletype 5620 DMD nach Ausführung von *hp2621*(1) in einem Layer durch Aufruf von *jterm* der Standardemulator in diesem Layer wieder gestartet.

ENDESTATUS

Gibt bei erfolgreicher Beendigung **0** zurück, andernfalls **1**.

HINWEIS

Das zurückgesetzte Layer ist das an die Standardfehlerausgabe angeschlossene; d. h., das Fenster, in dem Sie sich befinden, wenn Sie das Kommando *jterm* eingeben.

SIEHE AUCH:

layers(1).

BEZEICHNUNG

jwin – Größe des Layers ausgeben

ÜBERSICHT

jwin

BESCHREIBUNG

jwin läuft nur unter *layers*(1) und dient zur Bestimmung der Größe des Layers, das dem aktuellen Prozess zugeordnet ist. Es gibt die Breite und Höhe des Layers in Bytes aus (jeweils Anzahl der Zeichen pro Zeile und Zeilenanzahl). Bei bit-mapped Bildschirmen wird die Breite und Höhe des Layers auch in Bits angegeben.

ENDESTATUS

Gibt **0** bei erfolgreicher Beendigung zurück, andernfalls **1** .

DIAGNOSE

Wenn *layers*(1) nicht aufgerufen wurde, wird eine Fehlermeldung ausgegeben:

jwin: not mpx

HINWEIS

Das Layer, dessen Größe ausgegeben wird, ist das an die Standardeingabe angeschlossene; d. h. das Fenster, in dem Sie sich befinden, wenn Sie das Kommando *jwin* eingeben.

SIEHE AUCH:

layers(1).

BEISPIEL

```
jwin
bytes: 86 25
bits: 780 406
```

BEZEICHNUNG

kill – einen Prozeß abbrechen

ÜBERSICHT

kill [–signo] PID ...

BESCHREIBUNG

kill schickt das Signal 15 (beenden) an die angegebenen Prozesse. Dieses Kommando bricht normalerweise Prozesse ab, die das Signal nicht abfangen oder es ignorieren. Die Prozeß-Nummer jedes asynchronen mit & gestarteten Prozesses wird von der Shell gemeldet (es sei denn, mehr als ein Prozeß wird in einer Pipe gestartet, dann wird die Nummer des letzten Prozesses in der Pipe gemeldet). Prozeß-Nummern können auch mit Hilfe von *ps*(1) erfragt werden.

Einzelheiten über den Abbruch von Prozessen sind in *kill*(2) beschrieben. Zum Beispiel wird bei Angabe der Prozeß-Nummer 0 allen Prozessen in der Prozeßgruppe das Signal geschickt.

Der abzubrechende Prozeß muß dem aktuellen Benutzer gehören, es sei denn, er ist der Systemverwalter.

Wenn als erstes Argument eine Signalnummer mit vorangestelltem – angegeben ist, wird dieses Signal anstelle des Endesignals geschickt (vgl. *signal*(2)). Speziell "kill –9 ..." ist ein sicherer Abbruch.

SIEHE AUCH:

ps(1), sh(1).
kill(2), signal(2) im *Programmer's Reference Manual.*

BEZEICHNUNG

layers – Layer-Verwaltung für window-fähige Terminals

ÜBERSICHT

layers [**-s**] [**-t**] [**-d**] [**-p**] [**-f** file] [layersys-prgm]

BESCHREIBUNG

layers verwaltet asynchrone Windows (vgl. *layers*(5)) bei einem window-fähigen Terminal. Bei Auruf des Kommandos sucht *layers* eine unbenutzte *xt*(7) -Kanalgruppe und verbindet diese mit der Terminalleitung für die Standardausgabe. Es wartet dann auf Kommandos von dem Terminal.

Optionen der Kommandozeile:

-s Meldet am Ende der Sitzung, nachdem *layers* verlassen wurde, eine Protokollstatistik auf der Standardfehlerausgabe. Die Statistik kann während einer Sitzung durch Aufrufen des Programms *xts*(1M) ausgegeben werden.

-t Schaltet Tracing für *xt*(7) Treiber ein und gibt am Ende der Sitzung, nachdem *layers* verlassen wurde, einen Trace-Dump auf der Standardfehlerausgabe aus. Der Trace-Dump kann während einer Sitzung durch Aufrufen des Programms *xtt*(1M) ausgegeben werden.

-d Falls eine Firmware-Korrekturroutine weitergesendet wurde, werden die Größen von Text-, Daten- und bss-Segment der Firmware-Korrekturroutine über Standardfehlerausgabe ausgegeben.

-p Wenn eine Firmware-Korrekturroutine weitergesendet wurde, wird die Protokoll-Statistik und ein Trace auf der Standardfehlerausgabe ausgegeben.

-f *file* Startet *layers* mit einer in *file* angegebenen Anfangskonfiguration. Jede Zeile der Datei beschreibt ein zu erstellendes Window und hat das folgende Format:

origin_x origin_y corner_x corner_y command_list

Die Koordinaten geben die Größe und die Position des Layers auf dem Bildschirm an, in dem Bildschirm-Koordinatensystem. Wenn alle vier *0* sind, muß der Benutzer das Layer interaktiv definieren. *command_list* ist eine Liste mehrerer Kommandos, die bereitgestellt werden muß. Sie wird in dem neuen Layer unter Verwendung der Shell des Benutzers ausgeführt (durch Ausführen von: $SHELL-i-c" *command_list*"). Dies bedeutet daß das letzte Kommando eine Shell wie zum Beispiel /bin/sh aufrufen sollte. (Wenn das letzte Kommando keine Shell ist, wird nach Beendigung des letzten Kommandos das Layer nicht funktionsfähig sein.)

layersys-prgm
Eine Datei, die eine Firmware-Korrekturroutine enthält, die das Kommando *layers* zum Bildschirm weitersendet, ehe Layers erstellt werden und *command_list* ausgeführt wird.

Jedes Layer ist in den meisten Fällen funktionell identisch mit einem separaten Bildschirm. Zeichen, die auf der Tastatur eingegeben werden, werden zur Standardeingabe des UNIX Systemprozesses geschickt, der an das aktuelle Layer (genannt Host-Prozeß) angeschlossen ist; Zeichen, die von dem Hostprozeß auf die Standardausgabe geschrieben werden, erscheinen in diesem Layer. Wenn ein Layer erstellt ist, wird eine separate Shell eingerichtet und mit diesem Layer verbunden. Wenn die Umgebungsvariable **SHELL** gesetzt ist, erhält der Benutzer diese Shell andernfalls wird */bin/sh* verwendet. Zur Kommunikation mit anderen Benutzern über *write*(1) ruft *layers* das Kommando *relogin*(1M) auf, wenn das erste Layer erstellt ist. *relogin*(1M) trägt dieses Layer jetzt als Login-Terminal des Benutzers ein. Ein anderes Layer kann ausgewählt werden, wenn *relogin*(1M) direkt verwendet wird. Bei Beendigung stellt *layers* die ursprüngliche Einstellung wieder her.

Wie Layers erstellt, entfernt, verändert und bearbeitet werden, hängt von dem jeweiligen Terminal ab. Zum Beispiel verfügt das AT&T Teletype 5620 DMD über ein Pop-Up-Menü mit Layer-Operationen, die mit der Maus ausgewählt werden können. Wie eine *layers*-Sitzung beendet wird, wird ebenfalls vom Terminal definiert.

Wenn ein Benutzer ein bildschirm-spezifisches Anwendersoftware-Paket benutzen möchte, muß die Umgebungsvariable **DMD** auf den Pfadnamen des Verzeichnisses gesetzt werden, in dem das Paket installiert wurde. Anderfalls sollte **DMD** nicht gesetzt werden.

BEISPIEL

layers -f startup

wobei **startup**

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

HINWEISE

Der *xt*(7) Treiber unterstützt eine alternative Datenübertragungsmethode, die als ENCODING MODE bezeichnet wird. Dieser Modus ermöglicht die Ausführung von *layers* auch bei Datenleitungen, die Kontrollzeichen abfangen oder 8-Bit-Zeichen nicht übertragen. ENCODING MODUS wird vor Ausführung von *layers* eingestellt entweder durch Setzen einer Konfigurationsoption an Ihrem windowfähigen Terminal oder durch Setzen der Umgebungsvariablen **DMDLOAD** auf den Wert *hex*:

export DMDLOAD; DMDLOAD=hex

Wenn, nach Ausführung von **layers -f** *file*, das Terminal in einem oder mehreren Layers nicht reagiert, hat meistens das letzte Kommando in der *command-list* für diesen Layer keine Shell aufgerufen.

ACHTUNG!

Um auf diese Version von *layers* zugreifen zu können, müssen Sie darauf achten , daß */usr/bin* vor jedem anderen Datenverzeichnis in Ihrem PATH erscheint, das ein layers-Programm enthält (wie z. B. *$DMD/bin*). (Weitere Informationen zur Definition der Shell-Umgebungsvariablen **PATH** in Ihrem *.profile* finden Sie in *profile*(4)). Andernfalls ist es möglich, daß Sie, falls eine bildschirm-abhängige Version von *layers* vorliegt, diese anstelle der richtigen erhalten.

Wenn *layers* mit den Optionen **-s**, **-t**, **-d** oder **-p** aufgerufen wird, wird die Standardfehlerausgabe am besten in eine andere Datei umgelenkt (z.B.**layers -s 2>stats**) um Statistik- und Trace-Ausgaben zu sichern; andernfalls könnte die Ausgabe ganz oder teilweise verloren gehen.

DATEIEN

$DMD/lib/layersys/lsys.8;?;?

SIEHE AUCH:

sh(1), write(1).
layers(5), libwindows(3X) im *Programmer's Reference Manual.*
relogin(1M), xt(7), xts(1M), xtt(1M), wtinit(1M) im Administrator's Reference Manual.

BEZEICHNUNG

line – eine Zeile lesen

ÜBERSICHT

line

BESCHREIBUNG

line kopiert eine Zeile (bis zu einem New-Line-Zeichen) aus der Standardeingabe und schreibt sie auf die Standardausgabe. Bei **EOF** wird der Endecode 1 zurückgegeben; es wird immer mindestens ein New-Line-Zeichen angegeben. Dieses Kommando wird häufig zum Einlesen vom Terminal des Benutzers in Shell-Skripten verwendet.

SIEHE AUCH:

sh(1).
read(2) im *Programmer's Reference Manual.*

BEZEICHNUNG

locate – ein UNIX Systemkommando mit Hilfe von Stichworten identifizieren

ÜBERSICHT

[help] locate
[help] locate [keyword1 [keyword2] ...]

BESCHREIBUNG

Das Kommando *locate* ist ein Help-Kommando des UNIX Systems und bietet On-Line Unterstützung für die Identifizierung von UNIX Systemkommandos.

Ohne Argumente wird der Anfangsbildschirm von *locate* angezeigt; nun kann der Benutzer Stichworte eingeben, die sich funktionell auf die Aktion der gewünschten UNIX Systemkommandos beziehen, die erkannt werden sollen. Ein Benutzer kann Stichworte eingeben und erhält dann eine Liste von UNIX Systemkommandos, deren funktionelle Attribute mit denen in der Stichwort-Liste übereinstimmen, oder er kann durch Eingabe von q (für "quit") zur Shell zurückkehren. Wenn Sie zum Beispiel den Inhalt einer Datei ausgeben wollen, geben Sie die Stichworte "print" (Ausgabe) und "file" (Datei) ein. Das Kommando *locate* würde dann die Namen aller Kommandos ausgeben, die sich auf diese Stichworte beziehen.

Stichworte können auch direkt von der Shell aus, wie oben gezeigt, eingegeben werden. In diesem Fall wird der Anfangsbildschirm nicht angezeigt, und das Ergebnis der Kommando-Liste wird ausgegeben.

Weitere Informationen über ein Kommando in der mit *locate* erstellten Liste erhalten Sie, wenn Sie den Modul *usage* verwenden, der zu den Help-Kommandos des UNIX Systems gehört. Dies geschieht durch Eingabe der entsprechenden Menü-Auswahl, nachdem die Kommando-Liste angezeigt wurde. Durch Eingabe von ! und dem auszuführenden Kommando kann der Benutzer von jedem Bildschirm des Help-Kommandos ein Kommando über die Shell (*sh*(1)) ausführen. Wird das ausgeführte Kommando in der ersten Eingabe-Aufforderungs-Stufe eingegeben, wird der Bildschirm neu geschrieben. Bei Eingabe von anderen Eingabe-Aufforderungs-Stufen wird nur die Eingabe-Aufforderung neu ausgegeben.

Standardmäßig scrollt das Help-Kommando die Daten, die dem Benutzer gezeigt werden. Bevorzugen Sie einen leeren Bildschirm vor Ausgabe der Daten (kein Scrollen), muß die Shellvariable **SCROLL** auf **no** gesetzt und exportiert werden, so daß sie Teil Ihrer Umgebung wird. Dies geschieht durch Hinzufügung der folgenden Zeile zu Ihrer Datei *.profile* (vgl. *profile* (4)): " export SCROLL ; SCROLL=no". Sollten Sie später Scrollen bevorzugen, muß **SCROLL** auf **yes** gesetzt werden.

Informationen über die Help-Kommandos (*starter*, *locate*, *usage*, *glossary*, und *help*) finden Sie in den entsprechenden Handbuchseiten.

SIEHE AUCH:

glossary(1), help(1), sh(1), starter(1), usage(1).
term(5) im *Programmer's Reference Manual.*

ACHTUNG!

Wird die Shellvariable **TERM** (vgl. *sh* (1)) in der Datei *.profile* des Benutzers nicht gesetzt, hat **TERM** standardmäßig den Wert des Terminl-Typs 450 (ein Hartkopie-Terminal). Eine Liste der gültigen Terminal-Typen ist in *term* (5) enthalten.

BEZEICHNUNG

login – Anmeldung im System

ÜBERSICHT

login [name [env-var ...]]

BESCHREIBUNG

Das Kommando *login* wird am Anfang jeder Bildschirm-Sitzung benutzt und ermöglicht Ihnen, sich im System anzumelden. Login kann als Kommando aufgerufen oder, wenn eine Verbindung das erste Mal aufgebaut wird, vom System vollzogen werden. Außerdem wird es vom System aufgerufen, wenn der vorherige Benutzer seine Ausgangsshell durch *cntrl-d* zur Anzeige von "Dateiende" beendet hat. (Vgl. *Kennenlernen des Systems* zu Beginn dieses Handbuchs bezüglich der Startanweisungen.)

Wenn *login* als Kommando aufgerufen wird, muß es den vorherigen Kommando-Interpreter ersetzen. Das erreichen Sie durch Eingabe von:

exec login

aus der Ausgangs-Shell.

login fragt nach Ihrem Benutzernamen (wenn dieser nicht als Argument angegeben wurde) und ggf. nach Ihrem Paßwort. Echo wird bei Eingabe Ihres Paßworts (sofern möglich) ausgeschaltet und erscheint deshalb nicht in der Ausgabe.

Bei einigen Installationen kann eine Option aufgerufen werden, bei der ein zweites "dialup" (Anwähl-) Paßwort verlangt wird. Diese Option gibt es nur für Wählleitungen und das zweite Paßwort wird mit der Nachricht "dialup password:" angefordert. Für ein erfolgreiches Anmelden sind beide Paßwörter erforderlich.

Wenn Sie das Login nicht erfolgreich innerhalb eines bestimmten Zeitraums (z.B. einer Minute) beendet haben, werden Sie ohne Nachricht abgehängt.

Nach einem erfolgreichen Login werden die Betriebsstatistik-Dateien aktualisiert, die Prozedur **/etc/profil** ausgeführt, die Tagesmeldung, wenn vorhanden, ausgegeben, die Benutzernummer, die Gruppennummer, das aktuelle Verzeichnis und der Kommando-Interpreter (normalerweise *sh*(1)) initialisiert und die Datei **.profile** im aktuellen Verzeichnis wird ausgeführt, wenn sie vorhanden ist. Diese Angaben sind in dem Eintrag der Datei **/etc/passwd** für den Benutzer zu finden. Der Name des Kommando-Interpreters ist – , gefolgt von der letzten Komponenten des Pfadnamens für den Interpretierer (z.B. –**sh**). Wenn dieses Feld in der Paßwortdatei leer ist, wird der Standardkommandointerpreter **/bin/sh** verwendet. Wenn dieses Feld "*" ist, wird das angegebene Verzeichnis zum Root-Verzeichnis, das ist der Startpunkt die für Pfadnamen, die mit / beginnen. Daraufhin wird *login* erneut auf der

neuen Stufe ausgeführt, die eine eigene Root-Struktur einschließlich **/etc/login** und **/etc/passwd** haben muß.

Die Basis-Umgebung *environment* wird initialisiert zu:

HOME=*your-login-directory*
PATH=:/bin:/usr/bin
SHELL=*last-field-of-passwd-entry*
MAIL=/usr/mail/*your-login-name*
TZ=*timezone-specification*

Die Umgebung kann erweitert oder verändert werden durch Angabe von zusätzlichen Argumenten für *login*, entweder beim Aufruf oder wenn *login* nach Ihrem Login-Namen fragt. Die Argumente können die Form *xxx* oder *xxx=yyy* haben. Argumente ohne ein Gleichheitszeichen werden als

l*n*=xxx

in die Umgebung übernommen, wobei *n* eine Zahl ist, die mit 0 beginnt und jedes Mal erhöht wird, wenn eine neue Variable erforderlich ist. Variablen, die ein = enthalten, werden ohne Änderungen in die Umgebung übernommen. Wenn sie schon in der Umgebung sind, wird der alte Wert ersetzt. Es gibt zwei Ausnahmen. Die Variablen **PATH** und **SHELL** können nicht geändert werden. Dadurch wird verhindert, daß Benutzer, die mit eingeschränkten Shell-Umgebungen arbeiten, sekundäre Shells erzeugen, die nicht eingeschränkt sind. *Login* und *getty* verstehen beide einfache Fluchtsymbolkonventionen von Einzelzeichen: Der Backslash vor einem Zeichen quotiert es und ermöglicht damit Zeichen wie Leerzeichen und Tabulatoren zu benutzen.

DATEIEN

/etc/utmp	Betriebsdaten
/etc/wtmp	Betriebsdaten
/usr/mail/*your-name*	Briefkasten für Benutzer *your-name*
/etc/motd	Tagesmeldung
/etc/passwd	Paßwortdatei
/etc/profile	Systemprofil
.profile	Login-Profil des Benutzers

SIEHE AUCH

mail(1), newgrp(1), sh(1), su(1M).
passwd(4), profile(4), environ(5) im *Programmer's Reference Manual.*

DIAGNOSE

login incorrect wenn der Benutzername oder das Paßwort nicht gefunden werden.
No shell,cannot open password file, oder *no directory*: Wenden Sie sich bitte an einen UNIX System-Fachmann.

No utmp entry.You must exec "login" from the lowest level "sh" wird bei einem Versuch ausgegeben, *login* als Kommando aufzurufen, ohne das interne Shell-Kommando *exec* zu benutzen oder Login nicht aus der Ausgangs-Shell aufzurufen.

BEZEICHNUNG

logname – Login-Namen abfragen

ÜBERSICHT

logname

BESCHREIBUNG

logname gibt den Inhalt der Umgebungsvariablen **$LOGNAME** zurück, die gesetzt wird, wenn ein Benutzer sich am System anmeldet.

DATEIEN

/etc/profile

SIEHE AUCH:

env(1), login(1).
logname(3X), environ(5) im *Programmer's Reference Manual.*

BEZEICHNUNG

lp, cancel – Aufträge an einen LP-Zeilendrucker schicken/abbrechen

ÜBERSICHT

lp [**-c**] [**-d***dest*] [**-m**] [**-n***number*] [**-o***option*] [**-s**] [**-t***title*] [**-w**] files
cancel [ids] [printers]

BESCHREIBUNG

lp veranlaßt den Druck der angegebenen Dateien und zusätzlicher Information auf einem Zeilendrucker (der gesamte Vorgang wird als *request* (Auftrag) bezeichnet). Wenn keine Dateinamen angegeben sind, wird die Standardeingabe angenommen. Der Dateiname – steht für die Standardeingabe und kann auf der Kommandozeile in Verbindung mit anderen Dateien *files* angegeben werden. Die Reihenfolge, in der *files* erscheinen, entspricht der Reihenfolge, in der sie gedruckt werden.

lp ordnet eine eindeutige *id* (Nummer) jedem Auftrag zu und gibt diese auf die Standardausgabe aus. Diese *id* kann später zum Abbruch (vgl. *cancel*) oder zur Feststellung des Auftrags-Status (vgl. *lpstat*(1)) verwendet werden.

Die folgenden Optionen können in beliebiger Reihenfolge erscheinen und mit Dateinamen vermischt sein:

-c Anfertigung von Kopien der zu druckenden Dateien *files* sofort nach Aufruf von *lp*. Normalerweise wird *files* nicht kopiert, sondern, wenn möglich, wird ein Verweis eingerichtet. Wenn die Option **-c** nicht angegeben ist, sollte der Benutzer, ehe der Auftrag ganz gedruckt worden ist, vorsichtig beim Entfernen von *files* sein. Es ist auch zu beachten, daß ohne den Einsatz der Option **-c**, Änderungen in den angegebenen *files*, die nach dem Druckauftrag, doch vor dem Ausdruck gemacht wurden, in der gedruckten Ausgabe erscheinen.

-d*dest* Wählt *dest* als den Drucker oder den Druckertyp, der das Drucken vornehmen soll. Wenn *dest* ein Drucker ist, wird der Auftrag nur auf diesem speziellen Drucker gedruckt. Wenn *dest* ein Druckertyp ist, wird der Auftrag auf dem ersten verfügbaren Drucker dieses Typs gedruckt. Bei bestimmten Bedingungen (Drucker nicht verfügbar, Platzbeschränkungen usw.) können Aufträge für angegebene Ziele nicht angenommen werden (vgl. *accept*(1M) und *lpstat*(1)). Standardmäßig wird *dest* der Umgebungsvariablen **LPDEST** entnommen (falls diese gesetzt ist). Sonst wird ein Standardziel (falls vorhanden) für das Rechnersystem verwendet. Zielnamen variieren bei Systemen (vgl. *lpstat*(1)).

-m Sendet Post (vgl. *mail(1)*), nachdem die Dateien gedruckt worden sind. Standardmäßig wird keine Post bei normaler Beendigung des Druckauftrages geschickt.

-n*number* Ausdruck der *number* (Anzahl) an Ausgabekopien (1 ist voreingestellt).

-o*option* Angabe der druckerabhängigen oder typabhängigen *options* (Optionen). Mehrere dieser Angaben können angegeben werden, indem die **-o** Option öfter angegeben wird. Weitere Informationen über *options* finden Sie unter **Models** in *lpadmin* (1M).

-s Unterdrückung von Nachrichten von *lp*(1) wie zum Beispiel "request id is ..." ("Auftragsnummer ist ...").

-t*title* Ausdruck von *title* auf die erste Seite (mit großen Buchstaben geschriebener Text) der Ausgabe.

-w Eine Nachricht an das Terminal des Benutzers schicken, nachdem die Dateien *files* gedruckt worden sind. Falls der Benutzer nicht angemeldet ist, wird statt dessen Post geschickt.

Cancel bricht Druckaufträge, die mit dem Kommando *lp*(1) abgesetzt wurden, ab. Die Argumente der Kommandozeile können entweder Auftragsnummern *ids* (wie von *lp*(1) geliefert) oder *printer* Namen (für eine vollständige Liste *lpstat*(1) verwenden) sein. Angabe einer Auftrags-*id* (Nummer) bricht den entsprechenden Auftrag ab, der momentan auf dem Drucker gedruckt wird. Angabe eines Druckers *printer* bricht der Auftrag ab, der momentan auf diesem Drucker gedruckt wird. In beiden Fällen gibt der Abbruch eines Auftrages, der momentan ausgedruckt wird, den Drucker frei, so daß dieser den nächsten verfügbaren Auftrag drucken kann.

DATEIEN

/usr/spool/lp/ *

SIEHE AUCH:

enable(1), lpstat(1), mail(1).
accept(1M), lpadmin(1M), lpsched(1M) im *Administrator's Reference Manual*.

BEZEICHNUNG

lpstat – LP-Status-Information ausgeben

ÜBERSICHT

lpstat [options]

BESCHREIBUNG

lpstat gibt Informationen über den aktuellen Status des LP Spool aus.

Wenn keine Optionen *options* angegeben sind, gibt *lpstat* den Status aller vom Benutzer an *lp*(1) angegebenen Aufträge aus. Argumente, die keine *options* sind, werden als Auftragsnummern *ids* (von *lp* geliefert) angesehen. *lpstat* gibt den Status solcher Aufträge aus. *options* können in beliebiger Reihenfolge erscheinen und mit anderen Argumenten wiederholt und vermischt werden. Unten angeführten Schlüsselbuchstaben kann wahlweise eine Liste *list* folgenden Formats nachgestellt werden: eine Liste von Elementen, die durch ein Komma voneinander getrennt sind, oder eine Liste mit Elementen, die in Doppelanführungszeichen eingeschlossen sind und durch Komma und/oder mehrere Leerzeichen voneinander getrennt sind. Zum Beispiel:

```
-u"user1, user2, user3"
```

Läßt man *list* nach diesen Schlüsselbuchstaben aus, so bewirkt dies die Ausgabe aller für die Schlüsselbuchstaben wichtigen Daten wie zum Beispiel:

```
lpstat -o
```

druckt den Status aller Ausgabeaufträge.

-a[*list*] Ausgabe des Annahmestatus (in bezug auf *lp*) von Zielen für Aufträge. *List* ist eine Liste von Druckernamen und Typnamen.

-c[*list*] Ausgabe der Typnamen und entsprechender Geräte. *List* ist eine Liste der Typnamen.

-d Ausgabe des Standardsystemziel für *lp*.

-o[*list*] Ausgabe des Status von Ausgabeaufträgen. *List* ist eine Liste von Druckernamen, Typnamen und Auftragsnummern *ids*.

-p[*list*] Ausgabe des Drucker-Status. *List* ist eine Liste von Druckernamen.

-r Ausgabe des Status des LP-Auftrags-Schedulers

-s Ausgabe einer Statuszusammenfassung, das Standardsystemziel, Liste von Typnamen und entsprechenden Geräten sowie eine Liste von Druckern und entsprechenden Geräten.

-t Ausgabe aller Status-Daten.

-u[*list*] Ausgabe des Status der Ausgabeaufträge für Benutzer. *List* ist eine Liste der Login-Namen.

-v[*list*] Ausgabe der Namen von Druckern und die Pfadnamen der entsprechenden Geräte. *List* ist eine Liste von Druckernamen.

DATEIEN

/usr/Spool/lp/ *

SIEHE AUCH:

enable(1), lp(1).

BEZEICHNUNG

ls – Inhalt des Verzeichnisses auflisten

ÜBERSICHT

ls [-RadCxmlnogrtucpFbqisf] [names]

BESCHREIBUNG

Für jedes Verzeichnis-Argument listet *ls* den Inhalt des Verzeichnisses; für jedes Dateiargument wiederholt *ls* den Namen und alle anderen gewünschten Informationen. Die Ausgabe wird standardmäßig alphabetisch sortiert. Wenn kein Argument angegeben ist, wird das aktuelle Verzeichnis aufgeführt. Wenn mehrere Argumente angegeben sind, werden die Argumente zuerst entsprechend sortiert, jedoch erscheinen Dateiargumente vor den Verzeichnissen und deren Inhalt.

Es gibt drei Hauptlistenformate. Das Standardformat ist ein Eintrag pro Zeile, die Optionen **-C** und **-x** schalten mehrspaltige Formate ein, und die Option **-m** aktiviert das Ausgabeformat ohne Spalten. Zur Bestimmung der Ausgabeformate für die Optionen **-C**, **-x** und **-m** verwendet *ls* eine Umgebungsvariable **COLUMNS**, die die auf einer Ausgabezeile verfügbare Anzahl an Zeichenstellen festlegt. Wenn diese Variable nicht gesetzt ist, wird die Datendatei *terminfo*(4) zur Festlegung der Anzahl Spalten aufgrund der Umgebungsvariablen **TERM** verwendet. Wenn diese Daten nicht zur Verfügung stehen, werden 80 Spalten angenommen.

Das Kommando *ls* hat die folgenden Optionen:

-R Rekursive Auflistung der vorkommenden Unterdateiverzeichnisse.

-a Auflistung aller Einträge einschließlich der mit Punkt beginnenden Einträge (.), die normalerweise nicht aufgelistet werden.

-d Falls ein Argument ein Verzeichnis ist, wird nur der Name (nicht der Inhalt) ausgegeben; meistens mit **-l** verwendet, um den Status eines Verzeichnisses zu erhalten.

-C Mehrspaltige Ausgabe mit Einträgen, die nach Spalten sortiert sind.

-x Mehrspaltige Ausgabe mit Einträgen, die horizontal anstelle von vertikal sortiert sind.

-m Ausgabeformat ohne Spalten; Dateien sind hintereinander aufgeführt und mit Kommata voneinander getrennt.

-l Ausführliche Ausgabe, die für jede Datei den Modus, die Anzahl der Verweise, Eigentümer, Gruppe, Größe in Bytes und Zeit der letzten Modifikation angibt (siehe unten). Wenn die Datei eine Gerätedatei ist, enthält das Größenfeld Geräte-Nummern anstatt einer Größe.

-n Wie bei **-l**, außer daß die Eigentümer-Benutzernummer **UID** und die Gruppennummer **GID** anstelle des zugehörigen Strings ausgegeben werden.

-o Wie bei **-l**, außer daß die Gruppe nicht ausgegeben wird.

-g Wie bei **-l**, außer daß der Eigentümer nicht ausgegeben wird.

-r Kehrt die Sortierfolge um, so daß Daten in umgekehrter alphabetischer Reihenfolge oder älteste Daten zuerst angeführt werden.

-t Sortiert nach der Zeitangabe (späteste zuerst) anstatt wie üblich nach Namen. Der Standard ist die letzte Modifikationszeit. (Vgl. **-n** und **-c**.)

-u Verwendet die Zeit des letzten Zugriffs anstelle der letzten Modifikation zur Sortierung (mit der Option **-t**) oder zum Ausgeben (mit der Option **-l**).

-c Verwendet die Zeit der letzten Modifikation des I-Knoten-Eintrags (erstellte Datei, geänderter Modus, usw.) zur Sortierung (mit Option **-t**) oder zum Ausgeben (mit Option **-l**).

-p Schreibt einen Slash (/) nach jedem Dateinamen, falls diese Datei ein Verzeichnis ist.

-F Schreibt einen Slash (/) nach jedem Dateinamen, falls diese Datei ein Verzeichnis ist und setzt einen Asterisk (*) nach jedem Dateinamen, falls diese Datei ablauffähig ist.

-b Nicht druckbare (darstellbare) Zeichen werden in oktaler **\ddd** Schreibweise dargestellt.

-q Nicht druckbare (darstellbare) Zeichen in Dateinamen werden durch das Zeichen (?) dargestellt.

-i Gibt für jede Datei den I-Knoten in der ersten Spalte der Ausgabe an.

-s Gibt für jeden Eintrag die Größe in Blöcken, einschließlich indirekter Blöcke.

-f Jedes Argument wird als Verzeichnis aufgefaßt und der Name, der in jedem Eintrag gefunden wird, wird ausgegeben. Diese Option schaltet **-l, -t, -s** und **-r** aus und schaltet **-a** ein; die Ausgabe erfolgt in der Reihenfolge, in der die Einträge im Verzeichnis erscheinen.

Die Ausgabe der Option **-l** besteht aus zehn Zeichen. Das erste Zeichen kann eines der folgenden sein:

d der Eintrag ist ein Verzeichnis;
b der Eintrag ist eine blockorientierte Gerätedatei;
c der Eintrag ist eine zeichenorientierte Gerätedatei;
p der Eintrag ist eine FIFO-Gerätedatei ("benannte Pipe");
- der Eintrag ist eine normale Datei.

Die nächsten 9 Zeichen werden als je drei Einheiten mit drei Bits interpretiert. Die erste Einheit bezieht sich auf die Berechtigung des Eigentümers; der nächste auf die Berechtigungen von anderen innerhalb der Benutzergruppe der Datei; und der letzte auf alle anderen. Innerhalb jeder Einheit zeigen die drei Zeichen jeweils die Lese-/Schreib- oder Ausführungberechtigung der Datei an. Bei einem Verzeichnis bedeutet Ausführberechtigung die Berechtigung, das Verzeichnis nach einer bestimmten Datei zu durchsuchen.

Bei **ls -l** erscheint die Ausgabe wie folgt:

```
-rwxrwxrwx  1 smith   dev     10876  May 16 9:42 part2
```

Diese horizontale Anordnung stellt eine Menge Informationen bereit. Wenn Sie von rechts nach links lesen, können Sie sehen, daß das aktuelle Verzeichnis eine mit "part2" bezeichnete Datei enthält. Danach wird der Zeitpunkt angegeben, zu dem der Dateiinhalt das letzte Mal geändert wurde, und zwar am 16. Mai um 9:42 A.M. Die Datei ist durchschnittlich groß, sie enthält 10.876 Zeichen oder Bytes. Der Eigentümer der Datei oder der Benutzer gehört zur Gruppe "dev" (bedeutet vielleicht "development") und sein oder ihr Login-Name ist "smith." Die Zahl, in diesem Fall "1," zeigt die Anzahl Verweise auf die Datei "part2" an. Schließlich bedeutet die aus einem Gedankenstrich und Buchstaben bestehende Folge, daß der Benutzer, die Gruppe und andere Schreib-/, Lese- und Ausführberechtigung für "part2" haben.

Das Ausführsymbol (**x**) belegt hier die dritte Position in der Dreier-Gruppe. Ein **-** an dritter Stelle bedeutet die Verweigerung der Ausführungsberechtigung.

Die Berechtigungen werden wie folgt angezeigt:

r die Datei ist lesbar
w die Datei ist schreibbar
x die Datei ist ablauffähig
- die angezeigte Berechtigung ist *nicht* gewährt
l obligatorisches Sperren bei Zugriff (das s-Bit der Gruppe ist eingeschaltet und das Gruppenausführungsbit ist ausgeschaltet).
s das s-Bit des Eigentümers oder der Gruppen ist eingeschaltet, und das entsprechende Benutzer- oder Gruppenausführungsbit ist auch eingeschaltet

S unbestimmter Bit-Zustand (s-Bit des Eigentümers ist eingeschaltet und das Benutzerausführungsbit ist ausgeschaltet)

t das 1000 (oktal) Bit oder t-Bit ist eingeschaltet (vgl. **chmod(1)**) und Ausführung ist eingeschaltet

T das 1000 Bit ist eingeschaltet, und Ausführung ist ausgeschaltet (unbestimmter Bit-Zustand)

Bei Benutzer- und Gruppenberechtigung steht an dritter Position manchmal ein anderes Zeichen als **x** oder **-**. Auch **s** kann diese Position belegen. Es zeigt den Status des s-Bits beim Eigentümer oder bei der Gruppe an. Die Fähigkeit, bei Ausführung die gleiche Nummer wie der Benutzer anzunehmen, wird zum Beispiel beim Login verwendet, wenn Sie als Root beginnen, aber die Identität des Benutzers, der bei "Login" angegegeben wird, annehmen müssen.

Bei den Gruppenberechtigungen kann **l** die dritte Position belegen. **l** bezieht sich auf obligatorische Datei- und Datensatzsperrung (locking). Diese Berechtigung beschreibt die Fähigkeit einer Datei, anderen Dateien zu erlauben, die Lese- und Schreibberechtigungen der Datei während des Zugriffs zu sperren.

Bei den Berechtigungen für Andere kann die dritte Position mit **t** oder **T** belegt werden. Diese beziehen sich auf den Status des t-Bits und der Ausführungsberechtigung.

BEISPIELE

Der erste Beispielsatz bezieht sich auf Berechtigungen:

```
-rwxr--r--
```

Hier wird eine Datei beschrieben, die vom Benutzer lesbar, schreibbar und ausführbar ist und von der Gruppe und anderen lesbar ist.

```
-rwsr-xr-x
```

Das zweite Beispiel beschreibt eine Datei, die vom Benutzer lesbar, schreibbar und ausführbar ist und von der Gruppe und anderen lesbar und ausführbar; die Benutzernummer darf während der Ausführung von dem Benutzer, der sie gegenwärtig ausführt, angenommen werden.

```
-rw-rwl---
```

Dieses Beispiel beschreibt eine Datei, die nur vom Benutzer und der Gruppe lesbar und schreibbar ist und die während des Zugriffs gesperrt werden kann.

```
ls -a
```

Dieser Aufruf wird die Namen aller Dateien im aktuellen Verzeichnis ausgeben, einschließlich der Dateien, die mit einem Punkt beginnen (.) und die normalerweise nicht ausgegeben werden.

ls -aisn

Dieser Aufruf gibt Ihnen ausführliche Information über alle Dateien (hierzu gehören auch nicht druckbare Dateienamen (**a**)), die I-Knoten-Nummer – die Adresse des I-Knoten-Eintrags der Datei – in der linken Spalte (**i**); die size (Größe) der Dateien (in Blöcken) in der Spalte rechts von der I-Knoten-Nummer (i-numbers = **s**); schließlich wird die Nachricht in der numerischen (**n**umeric) Version der langen Liste gedruckt, wobei die Benutzernummern (anstelle von Benutzernamen) und die Gruppennummern (anstelle von Gruppennamen) der entsprechenden Dateien angegeben sind.

Wenn die Größen der Dateien in einem Datenverzeichnis ausgegeben werden, wird die Gesamtanzahl an Blöcken einschließlich der indirekten Blöcke ausgegeben.

DATEIEN

/etc/passwd Benutzernummern für **ls -l** und **ls -o**
/etc/group Gruppennummern für **ls -l** und **ls -g**
/usr/lib/terminfo/?/* Informationsdatendatei der Terminals

SIEHE AUCH:

chmod(1), find(1).

HINWEISE

Unter Remote File Sharing haben Sie eventuell nicht die Berechtigungen, die die Ausgabe des Kommandos **ls -l** Ihnen vorgibt.

FEHLER

Nicht druckbare Zeichen in Dateinamen können die spaltenweisen Ausgabeoptionen verwirren.

BEZEICHNUNG

machid: pdp11, u3b, u3b2, u3b5, vax, m68k – Prozessortyp abfragen

ÜBERSICHT

pdp11

u3b

u3b2

u3b5

vax

m68k

BESCHREIBUNG

Die folgenden Kommandos geben den Wert wahr (Endecode 0) zurück, wenn Sie mit einem Prozessor arbeiten, der mit dem Kommandonamen angezeigt wird.

pdp11	Wahr bei PDP-11/45 oder PDP-11/70 .
u3b	Wahr bei 3B20 Rechner.
u3b2	Wahr bei 3B2 Rechner.
u3b5	Wahr bei 3B5 Rechner.
vax	Wahr bei VAX-11/750 oder VAX-11/780 .
m68k	Wahr, wenn Sie auf einer Targon /31 arbeiten.

Die Kommandos, die nicht gültig sind, liefern den Wert falsch (ungleich Null). Diese Kommandos werden häufig in Makefiles (vgl. *make*(1)) und Shell-Prozeduren (vgl. *sh*(1)) verwendet, um eine erhöhte Flexibilität zu erreichen.

SIEHE AUCH:

sh(1), test(1), true(1).
make(1) im *Programmer's Reference Manual.*

BEZEICHNUNG

mail, rmail – Post an Benutzer schicken oder Post lesen

ÜBERSICHT

Sending mail:

mail [**-wt**] persons

rmail [**-wt**] persons

Reading mail:

mail [**-ehpqr**] [**-f** file] [**-F** persons]

BESCHREIBUNG

Sending mail: Die nachfolgenden Argumente der Kommandozeile betreffen SENDING mail (das Senden von Post)

-w schickt einen Brief an einen Remote-Benutzer ohne auf die Beendigung des Remote-Übertragungsprogramms zu warten.

-t fügt eine **To:** -Zeile an den Brief an, die den beabsichtigten Empfänger angibt.

Eine Person *person* ist gewöhnlich ein Benutzername, der von *login*(1) erkannt wird. Wenn *persons* genannt sind, nimmt *mail* an, daß eine Nachricht geschickt wird (außer bei der -F Option). Es liest von der Standardeingabe bis zu einem Dateiende-Zeichen (Control-d) oder bis es eine Zeile liest, die nur aus einem Punkt besteht. Wenn eines dieser Endezeichen empfangen wird, fügt *mail* den Brief *letter* der Postdatei *mailfile* jeder Person *person* hinzu. Ein *letter* ist eine Nachricht *message,* der ein Poststempel *postmark* vorausgeht. Der Nachricht gehen der Name des Absenders und der Poststempel *postmark* voraus. Ein *postmark* besteht aus einer oder mehreren 'From'-Zeilen, auf die eine Leerzeile folgt.

Wenn ein Brief nicht zustellbar ist, wird er zum Absender mit Diagnoseangaben zurückgeschickt, die den Ort und die Art des Versagens anzeigen. Falls *mail* während der Eingabe unterbrochen wird, wird die Datei **dead.letter** gesichert, damit diese neu editiert und wieder gesendet werden kann. Die Datei **dead.letter** wird jedesmal, wenn sie benötigt wird, neu erstellt und löscht den Inhalt der vorigen **dead.letter** -Datei.

rmail erlaubt nur das Schicken von Post; *uucp*(1C) benutzt *rmail* als eine Sicherheitsmaßnahme.

Wenn bei dem lokalen System die Basis-Dienstprogramme für Netzwerke installiert sind, kann Post zu einem Empfänger an einem Remote-System geschickt werden. Hierzu wird der Person *person* der Systemname und das Ausrufzeichen vorangestellt. Eine Reihe von Systemnamen, die durch Ausrufzeichen getrennt sind, können zum Senden eines Briefs in einem erweiterten Netzwerk verwendet werden.

Reading mail:

Die nachfolgenden Argumente der Kommandozeile betreffen READING mail (Post lesen):

-e verhindert die Ausgabe von Post. 0 wird zurückgegeben, wenn der Benutzer Post hat; andernfalls wird 1 zurückgegeben.

-h Die Ausgabe eines Ausschnitts von Nachrichtenköpfen anstatt Anzeige der letzten Nachricht. Der Anzeige folgt das Eingabe-Aufforderungszeichen '?' .

-p Die Ausgabe aller Nachrichten ohne Bearbeitungsaufforderung.

-q Die Beendigung von *mail* nach Unterbrechungen. Normalerweise verursacht eine Unterbrechung nur den Abbruch der gerade ausgegebenen Nachricht.

-r Die Ausgabe von Nachrichten, wobei die zuerst eingegebene Nachricht als erste ausgegeben wird.

-f*file*

mail wird veranlaßt, die Datei *file* (z. B. **mbox**) statt der Standard-Datei *mailfile* zu benutzen.

-F*persons*

Wenn in einen leeren Briefkasten eingegeben, bewirkt *mailbox* die Zustellung aller ankommenden Post an *persons*.

mail gibt Postmeldungen eines Benutzers in der Reihenfolge "zuletzt eingegeben/zuerst ausgegeben" an, sofern nicht Argumente der Kommandozeile eine andere Reihenfolge bestimmen. Bei jeder Nachricht wird der Benutzer mit dem Eingabe-Aufforderungszeichen ?, zur Eingabe aufgefordert, und von der Standardeingabe wird eine Zeile gelesen. Die folgenden Kommandos stehen zur Bestimmung der Nachrichtenbearbeitung zur Verfügung:

<new-line>, **+** oder **n**	Weiter zur nächsten Nachricht.
d, oder **dp**	Nachricht löschen und weiter zur nächsten Nachricht.
d #	Nachrichtennummer # löschen. Nicht weiter zur nächsten Nachricht.
dq	Nachricht löschen und *mail* verlassen.
h	Einen Ausschnitt von Nachrichtenköpfen aus der Umgebung der aktuellen Nachricht anzeigen.
h #	Kopf der Nachrichtennummer # anzeigen.
h a	Köpfe aller Nachrichten im *mailfile* des Benutzers anzeigen.

h d	Köpfe der zur Löschung bestimmten Nachrichten anzeigen.
p	Aktuelle Nachricht nochmal ausgeben.
–	Vorherige Nachricht ausgeben.
a	Nachricht ausgeben die während der Postsitzung *mail* ankam.
#	Nachrichtennummer # ausgeben.
r [*users*]	Absender und anderen Benutzern *user(s)* antworten, dann Nachricht löschen.
s [*files*]	Nachricht in den angegebenen Dateien *files* sichern (**mbox** ist Standard).
y	Genau wie sichern.
u [**#**]	Löschen der Nachrichtennummer # rückgängig machen (Standard ist die zuletzt gelesene Nachricht).
w [*files*]	Nachricht ohne Kopf in den angegebenen Dateien *files* sichern (**mbox** ist Standard).
m [*persons*]	Die Nachricht an die bezeichneten Personen *persons* senden.
q oder **ctl-d**	Post, deren Löschung rückgängig gemacht wurde, in die Datei *mailfile* schreiben und *mail* beenden.
x	Sämtliche Post unverändert in die Datei *mailfile* schreiben und *mail* beenden.
!*command*	In die Shell wechseln, um ein Kommando *command* auszuführen.
?	Eine Kommandozusammenfassung ausgeben.

Wenn sich ein Benutzer anmeldet, wird angezeigt, daß Post vorhanden ist. Es wird ebenfalls darauf hingewiesen, daß während *mail* neue Post angekommen ist.

Die Datei *mailfile* kann auf zwei Arten zur Änderung der *mail* Funktion manipuliert werden. Die Berechtigungen der Datei für andere kann Lese-/Schreib-, Nur-Lese- oder gar keine Berechtigung sein, wodurch verschiedene Ebenen der Geheimhaltung gesichert sind. Wenn die Berechtigung anders als die Standardeinstellung eingestellt wird, wird die Datei sogar erhalten, wenn sie leer ist, um die gewünschten Berechtigungen zu erhalten.

Die Datei kann auch folgende erste Zeile enthalten:

weitergeleitet an *person*

wodurch alle an den Eigentümer der *mailfile* gesendeten Post an die Person *person* weitergeleitet wird. Eine Nachricht "Weitergesendet von..." wird dem Nachrichtenkopf hinzugefügt. Dies ist besonders nützlich in einer Mehrplatz-Umgebung, um die Post einer Person an ein einziges Gerät weiterzuleiten und den Empfänger zu informieren, daß die Post weitergeleitet wurde. Ein- und Ausschalten der Weitersende-Funktion wird mit der Option **-F** ausgeführt.

Zur Weitersendung sämtlicher Post an systema!user wird folgendes eingegeben:

mail -Fsystema!user

Zur Weitersendung an mehrere Benutzer:

mail -F"user1,systema!user2,systema!systemb!user3"

Es ist zu beachten, daß bei mehreren Benutzern, die ganze Liste in Anführungszeichen stehen sollte, so daß alles als Argument der **-F** Option interpretiert wird. Die Liste kann max. 1024 Bytes groß sein; Kommata oder Zwischenraumzeichen können zur Abgrenzung der Benutzer verwendet werden.

Weitersenden wird ausgeschaltet durch:

mail -F ""

Die beiden Anführungszeichen sind bei der Option -F obligatorisch um ein NULL-Argument anzugeben.

Damit die Weitersendung richtig läuft, sollte *mailfile* die Gruppenkennung "mail" haben und die Gruppenberechtigung sollte Lesen/Schreiben zulassen.

DATEIEN

/etc/passwd	zur Identifizierung des Absenders und zur Lokalisierung von Personen
/usr/mail/*user*	ankommende Post für den Benutzer *user*; d. h. die *mailfile*
$HOME/mbox	gespeicherte Post
$MAIL	Variable, die den Pfadnamen von *mailfile* enthält
/tmp/ma*	temporäre Datei
/usr/mail/*.lock	Sperrung des Postverzeichnises
dead.letter	nicht zustellbarer Text

SIEHE AUCH:

login(1), mailx(1), write(1) im *User's Reference Manual.*

ACHTUNG!

Die Funktion "Weitersenden an Person" kann in einer Schleife enden, falls *sys1/userb* an *sys2/userb* und *sys2/userb* an *sys1/userb* schickt. Es wird durch eine Nachricht angezeigt, die wie folgt lautet: "unbounded...saved mail in dead.letter" (unbegrenzt.....Post in dead.letter gespeichert").

FEHLER

Gewisse Bedingungen führen zu einem fehlerhaften Entfernen einer gesperrten Datei.

Nach einer Unterbrechung kann die nächste Nachricht eventuell nicht angezeigt werden; durch Eingabe von einem **p**, kann die Ausgabe erzwungen werden.

BEZEICHNUNG

mailx – interaktives System zur Nachrichtenverarbeitung

ÜBERSICHT

mailx [*options*] [*name...*]

BESCHREIBUNG

Das Kommando *mailx* bietet eine bequeme, flexible Umgebung zum elektronischen Schicken und Empfangen von Nachrichten. Beim Lesen von Post stellt *mailx* Kommandos zur bequemen Sicherung, Löschung und Nachrichtenbeantwortung bereit. Beim Schicken von Post erlaubt *mailx* das Editieren, Überprüfen und Modifizieren von Nachrichten bei deren Eingabe.

Einige der Leistungsmerkmale von *mailx* für Remote-Betrieb funktionieren nur, wenn die Basis-Dienstprogramme für Netzwerke auf Ihrem System installiert sind.

Ankommende Post wird in einer Standard-Datei für jeden Benutzer gespeichert, die als der Briefkasten *mailbox* des Benutzers bezeichnet wird. Wenn *mailx* zum Lesen von Nachrichten aufgerufen wird, ist *mailbox* die Standardstelle, um sie zu finden. Beim Lesen der Nachrichten werden sie markiert und dann in einer sekundären Datei aufbewahrt, außer wenn spezielle Maßnahmen getroffen werden, daß die die Nachrichten nicht wieder angesehen werden müssen. Diese sekundäre Datei wird als *mbox* bezeichnet und steht normalerweise im HOME-Verzeichnis des Benutzers (vgl. "MBOX" (UMGEBUNGSVARIABLEN) hinsichtlich einer Beschreibung dieser Datei). Nachrichten können in anderern sekundären Dateien, die vom Benutzer angegeben werden, gesichert werden. Nachrichten bleiben in einer sekundären Datei, bis sie zwangsweise entfernt werden.

Der Benutzer kann auf eine sekundäre Datei mit der Option **-f** des Kommandos *mailx* zugreifen. Nachrichten in der sekundären Datei können dann gelesen oder anderweitig verarbeitet werden, indem die gleichen KOMMANDOS wie beim primären Briefkasten *mailbox* verwendet werden. Aus diesem Grunde wird nachstehend auch von einem aktuellen Briefkasten *mailbox* gesprochen.

Auf der Kommandozeile beginnen *options* mit einem Gedankenstrich (-), alle anderen Argumente werden als Ziele (Empfänger) angesehen. Wenn keine Empfänger angegeben werden, liest *mailx*-Nachrichten von der *mailbox* . Optionen der Kommandozeile lauten:

-e Prüfen, ob Post vorhanden ist. *mailx* gibt nichts aus und endet mit einem erfolgreichen Rückkehrcode, wenn Post gelesen werden muß.

-f [*filename*]	Liest Nachrichten von *filename* anstelle von *mailbox* . Wenn kein *filename* angegeben ist, wird die *mbox* verwendet.
-F	Zeichnet die Nachricht in einer Datei auf, die nach dem ersten Empfänger benannt wird. Sie hebt die "record"-Variable auf, falls diese gesetzt ist (vgl. UMGEBUNGSVARIABLEN).
-h *number*	Die Anzahl der bis zu diesem Zeitpunkt durchgeführten "Sprünge" im Netzwerk. Diese Option wird für Netzwerk-Software geliefert, um endlose Zustellungsschleifen zu vermeiden. (Vgl. **addsopt** bei UMGEBUNGSVARIABLEN)
-H	Nur Nachrichtenköpfe ausgeben.
-i	Ignoriert Unterbrechungen. Vgl. auch "ignore" (UMGEBUNGSVARIABLEN).
-n	Nicht aus der System-Standarddatei *mailx.rc.* initialisieren.
-N	Ausgabe der Nachrichtenköpfe unterdrücken.
-r address	Gibt Adresse *address* weiter an Netzwerkzustellungs-Software. Alle Tilde-Kommandos werden abgeschaltet. (Vgl. **addsopt** bei UMGEBUNGSVARIABLEN)
-s *subject*	Setzt das Betrifft (Subject)-Feld im Nachrichtenkopf auf *subject*.
-u *user*	Liest *user's mailbox* . Dies ist nur effektiv, wenn *user's mailbox* nicht lesegeschützt ist.
-U Wandelt	*uucp* -Adressen in Internet-Standardadressen um. Überschreibt die "conv" Umgebungsvariablen. (Vgl. **addsopt** bei UMGEBUNGSVARIABLEN)

Beim Lesen von Post befindet sich *mailx* im Kommando-Modus (*command mode*). Die Köpfe der ersten Nachrichten werden angezeigt. Dann folgt ein Eingabe-Aufforderungszeichen, das anzeigt, daß *mailx* normale Kommandos akzeptieren kann (vgl. nachstehend KOMMANDOS). Beim Schicken von Post befindet sich *mailx* im Eingabe-Modus (*input-mode*). Wenn kein Betrifft (Subject) auf der Kommandozeile angegeben ist, wird ein Eingabe-Aufforderungszeichen zur Eingabe von Betrifft ausgegeben. (Wenn Betrifft länger als 1024 Zeichen ist, erzeugt *mailx* einen Speicherabzug). Nach Eingabe der Nachricht liest *mailx* die Nachricht und speichert sie in einer temporären Datei. Kommandos können eingegeben werden, indem am Anfang einer Zeile das Escape-Zeichen Tilde (~) eingegeben wird, gefolgt von einem einzelnen Kommandobuchstaben und optionalen Argumenten. Vgl. TILDE-ESCAPE-FOLGEN bezüglich einer Zusammenfassung dieser Kommandos.

Das Verhalten von *mailx* hängt von Umgebungsvariablen (*environment variables*) ab. Diese sind Optionen und Parameter mit Werten, die mit den **set** und **unset** Kommandos gesetzt und gelöscht werden. Vgl. nachstehende UMGEBUNGSVARIABLEN bezüglich einer Zusammenfassung dieser Parameter.

Auf der Kommandozeile können drei verschiedene Empfängertypen angegeben werden: Login-Namen, Shell-Kommandos oder Abkürzungs-Gruppen. Login-Namen können Netzwerkadressen sein, inklusive gemischter Netzwerkadressierung. Wenn Post nicht zustellbar ist, wird sie an die *mailbox* des Absenders zurückgeschickt. Wenn der Empfängername mit einem Pipe-Symbol (|) beginnt, wird der restliche Name als ein Shell- Kommando angesehen, an das die Nachricht über eine Pipe geschickt werden soll. Dies ist eine automatische Schnittstelle zu Programmen, die die Standardeingabe lesen wie zum Beispiel *lp*(1). Abkürzungsgruppen (alias groups) werden mit dem Kommando **alias** gesetzt (vgl. nachstehende KOMMANDOS) und sind Listen beliebiger Empfängertypen.

Normale Kommandos haben folgendes Format

[**command**] [*msglist*] [*arguments*]

Wenn kein Kommando im Kommando-Modus *command mode* angegeben ist, wird **print** (Ausgabe) angenommen. Bei *input mode* werden Kommandos am Escape-Zeichen erkannt, und Zeilen, die nicht wie Kommandos behandelt wurden, werden als Eingabe für die Nachricht angesehen.

Jeder Nachricht wird eine laufende Nummer zugeordnet, und die aktuelle Nachricht wird immer mit einer rechten spitzen Klammer (>) in der Zusammenfassung der Nachrichtenköpfe markiert. Viele Kommandos akzeptieren optional eine Liste von Nachrichten (*msglist*) als Argument. Die Voreinstellung für *msglist* ist die aktuelle Nachricht. Eine Nachrichtenliste *msglist* ist eine Liste von Nachrichtidentifizierungen, die durch Leerzeichen getrennt sind, und folgende Zeichen enthalten können:

n	Nachrichtennummer **n.**
.	Die aktuelle Nachricht.
^	Die erste ungelöschte Nachricht.
$	Die letzte Nachricht.
*	Alle Nachrichten.
n-m	Ein Bereich (inklusive) von Nachrichtennummern.
user	Alle Nachrichten von **user.**

/string Alle Nachrichten mit **string** in der Betrifft-Zeile (Groß-/Kleinschreibung wird ignoriert).

:c Alle Nachrichten des Typs *c*, wobei *c* eine der folgenden Angaben ist:

- **d** gelöschte Nachrichten
- **n** neue Nachrichten
- **o** alte Nachrichten
- **r** gelesene Nachrichten
- **u** nicht gelesene Nachrichten.

Es ist zu beachten, daß der Kontext des Kommandos bestimmt, ob diese Nachrichtentyp-Angabe sinnvoll ist.

Andere Argumente sind gewöhnlich beliebige Zeichenfolgen, deren Angabe vom aktuellen Kommando abhängt. Dateinamen werden erforderlichenfalls mit den üblichen Shell-Konventionen erweitert (vgl. *sh*(1)). Sonderzeichen werden von angegebenen Kommandos erkannt und sind nachstehend unter Kommandos dokumentiert.

Beim Starten versucht *mailx*, Kommandos aus der optionalen Systemdatei (**/usr/lib/mailx/mailx.rc**) auszuführen, um gewisse Parameter zu initialisieren, und anschließend aus einer privaten Startdatei (**$HOME/.mailrc**) für individuelle Variablen. Abgesehen von nachstehenen Ausnahmen sind normale Kommandos in Startdateien erlaubt. Eine Startdatei wird meistens zum Einrichten von Anfangs-Anzeigeoptionen und -Listen von Abkürzungen verwendet. Die folgenden Kommandos sind in einer Startdatei nicht erlaubt: **!**, **Copy**, **edit**, **followup**, **Followup**, **hold**, **mail**, **preserve**, **reply**, **Reply**, **shell**, und **visual**. Ein Fehler in der Startdatei bewirkt, daß die verbleibenden Zeilen in der Datei ignoriert werden. Die **.mailrc** Datei ist optional, und muß lokal angelegt werden.

KOMMANDOS

Nachstehend finden Sie eine vollständige Liste der *mailx* Kommandos:

!shell-command

Wechsel in die Shell. Vgl. "SHELL" (UMGEBUNGSVARIABLEN).

comment

Nullkommando (Kommentar). Dies kann bei *.mailrc* Dateien nützlich sein.

=

Gibt die aktuelle Nachrichtennummer aus.

?

Gibt eine Kommando-Zusammenfassung aus.

alias *alias name ...*
group *alias name ...*
Einen Abkürzung für die angegebenen Namen angeben. Die Namen werden eingesetzt, wenn *alias* als ein Empfänger verwendet wird; praktisch in der *.mailrc* Datei.

alternates *name ...*
Gibt eine Liste alternativer Namen für Ihr Login an. Wenn auf eine Nachricht geantwortet wird, werden diese Namen von der Empfängerbeantwortungsliste entfernt. Ohne Argumente gibt **al**ternates die aktuelle Liste alternativer Namen aus. Vgl. auch "allnet" (UMGEBUNGSVARIABLEN).

cd [*directory*]
chdir [*directory*]
Verzeichnis ändern. Wenn *directory* nicht angegeben ist, wird $HOME verwendet.

copy [*filename*]
copy [*msglist*] *filename*
Kopieren von Nachrichten in die Datei, ohne die Nachrichten als gesichert zu markieren. Sonst gleichwertig mit dem save Kommando.

Copy [*msglist*]
Sichern der angegebenen Nachrichten in eine Datei, deren Name vom Verfasser der zu sichernden Nachricht abgeleitet wird, ohne die Nachrichten als gesichert zu markieren. Sonst gleichwertig mit dem **S**ave Kommando.

delete [*msglist*]
Löschen von Nachrichten in der *mailbox* . Wenn "autoprint" gesetzt ist, wird die nächste Nachricht hinter der zuletzt gelöschten ausgegeben (vgl. UMGEBUNGSVARIABLEN).

discard [*header-field ...*]
ignore [*header-field ...*]
Unterdrückt die Ausgabe der angegebenen Nachrichtenkopf-Felder beim Anzeigen von Nachrichten am Bildschirm. Beispiele für ignorierbare Nachrichtenkopf-Felder sind "status" und "cc". Die Felder werden einbezogen, wenn die Nachricht gesichert ist. Die **P**rint- und **T**ype-Kommandos heben dieses Kommando auf.

dp [*msglist*]
dt [*msglist*]
Löschen der angegebenen Nachrichten aus der *mailbox* und Ausgabe der nächsten Nachricht nach der letzten gelöschten Nachricht. Fast gleichwertig mit einem **delete** Kommando, gefolgt von einem **print** Kommando.

echo *string* ...
Die angegebenen Zeichenfolgen ausgeben (wie *echo*(1)).

edit [*msglist*]
Die angegebenen Nachrichten editieren. Die Nachrichten werden in eine temporäre Datei geschrieben, und die "EDITOR"-Variable wird verwendet, um den Namen des Editors zu erhalten (vgl. UMGEBUNGSVARIABLEN). Standard-Editor ist *ed*(1).

exit
xit
mailx beenden, ohne die *mailbox* zu ändern. Nachrichten werden nicht in der *mbox* gesichert (vgl. auch **quit**).

file [*filename*]
folder [*filename*]
Verlassen der aktuellen Nachrichtendatei und Einlesen der angegebenen Datei. Mehrere Sonderzeichen werden erkannt, wenn sie als Dateinamen verwendet werden; sie haben folgende Bedeutung:
% die aktuelle *mailbox* .
%**user**
die *mailbox* für **user**
die vorige Datei.
& die aktuelle *mbox*.
Standard-Datei ist die aktuelle *mailbox* .

folders
-Ausgabe der Dateinamen des Verzeichnisses, das in der "folder"-Variablen gesetzt ist (vgl. UMGEBUNGSVARIABLEN).

followup [*message*]
Antworten auf eine Nachricht, Aufzeichnen der Antwort in einer Datei, deren Name vom Verfasser der Nachricht abgeleitet ist. Überschreibt die "record"-Variable, falls diese gesetzt ist. Vgl. auch die **Followup**, **Save** und **Copy** Kommandos und "outfolders" (UMGEBUNGSVARIABLEN).

Followup [*msglist*]
Antworten auf die erste Nachricht in der *msglist* , Schicken der Nachricht zum Verfasser jeder Nachricht in der *msglist* . Die Betrifft-Zeile wird von der ersten Nachricht genommen und die Antwort wird in einer Datei, deren Name vom Verfasser der ersten Nachricht abgeleitet wird, aufgezeichnet. Vgl. auch die **followup**, **Save** und **Copy** Kommandos und "outfolders" (UMGEBUNGSVARIABLEN).

from [*msglist*]
Gibt die Zusammenfassung der Nachrichtenköpfe für die angegebenen Nachrichten aus.

group *alias name ...*
alias *alias name ...*
Bestimmen einer Abkürzung für die angegebenen Namen. Die Namen werden eingesetzt, wenn *alias* als ein Empfänger verwendet wird. Nützlich in der *.mailrc* -Datei.

headers [*message*]
Gibt die Seite des Nachrichtenkopfs aus, die die angegebene Nachricht enthält. Die "screen"-Variable setzt die Anzahl der Nachrichtenköpfe pro Seite (vgl. UMGEBUNGSVARIABLEN). Vgl. auch das **z** Kommando.

help
Gibt eine Zusammenfassung der Kommandos aus.

hold [*msglist*]
preserve [*msglist*]
Aufbewahrung der angegebenen Nachrichten im *mailbox* .

if *s* | *r*
mail-commands
else
mail-commands
endif
Bedingte Ausführung, bei der *s* bewirkt, daß die folgenden Kommandos *mail-commands* bis zu einem **else** oder **endif** ausführt werden, wenn das Programm im *send*-Modus ist; *r* führt die Kommandos nur im Empfang-Modus *(receive)* aus; nützlich in der *.mailrc*-Datei.

ignore *header-field ...*
discard *header-field ...*
Unterdrückt Ausgabe der angegebenen Nachrichtenkopf-Felder beim Anzeigen von Nachrichten am Bildschirm. Beispiele für ignorierbare Nachrichtenkopf-Felder sind "status" und "cc." Alle Felder werden berücksichtigt, wenn die Nachricht gesichert wird. **Print** und **Type** Kommandos heben dieses Kommando auf.

list
Gibt alle verfügbaren Kommandos aus. Es wird keine Erklärung gegeben.

mail *name ...*
Schicken einer Nachricht zum angegebenen Benutzer.

Mail *name*
Schicken einer Nachricht zum angegebenen Benutzer und Aufzeichnen einer Nachrichtenkopie in einer Datei, die nach diesem Benutzer genannt wird.

mbox [*msglist*]
Leiten der angegebenen Nachrichten in die Standardsicherungsdatei *mbox*, wenn *mailx* normal beendet wird. Vgl. "MBOX" (UMGEBUNGSVARIABLEN) hinsichtlich einer Beschreibung dieser Datei. Vgl. auch die **exit** und **quit** Kommandos.

next [*message*]
Zur nächsten Nachricht gehen, die mit *message* übereinstimmt. Eine *msglist* kann angegeben werden, aber in diesem Fall ist die erste gültige Nachricht in der Liste die einzige, die verwendet wird. Dies ist nützlich beim "Springen" auf die nächste Nachricht eines bestimmten Benutzers, da sonst der Name in Abwesenheit eines tatsächlichen Kommandos als Kommando angesehen werden würde. Vgl. die oben angeführten Erläuterungen über *msglist* bezüglich einer Beschreibung möglicher Angaben für Nachrichten.

pipe [*msglist*] [*shell-command*]
| [*msglist*] [*shell-command*]
Die Nachricht über eine Pipe an das angegebene *shell-command* schicken. Die Nachricht wird behandelt, als wäre sie gelesen. Wenn keine Argumente angegeben sind, wird die aktuelle Nachricht an das Kommando geschickt, das durch die "cmd"-Variablen angegeben wird. Wenn die "page"-Variable gesetzt ist, wird nach jeder Nachricht ein Formularvorschub-Zeichen eingefügt (vgl. UMGEBUNGSVARIABLEN).

preserve [*msglist*]
hold [*msglist*]
Aufbewahren der angegebenen Nachrichten in *mailbox* .

Print [*msglist*]
Type [*msglist*]
Ausgabe der angegebenen Nachricht am Bildschirm, einschließlich aller Nachrichtenkopf-Felder; hebt die durch das **ignore** Kommando gesetzte Unterdrückung von Feldern wieder auf.

print [*msglist*]
type [*msglist*]
Ausgabe der angegebenen Nachrichten. Wenn "terminal" gesetzt ist, werden die Nachrichten, die länger als die in der "terminal"-Variablen angegebenen Zeilenzahl sind, mit dem Kommando seitenweise ausgegeben, das in der "PAGER"-Variablen angegeben ist. Das Standardkommando ist *pg*(1) (vgl. UMGEBUNGSVARIABLEN).

quit
Beenden von *mailx*, wobei die gelesenen Nachrichten in *mbox* und die nicht gelesenen Nachrichten in *mailbox* abgespeichert werden. Nachrichten, die explizit in einer Datei gesichert worden sind, werden gelöscht.

Reply [*msglist*]
Respond [*msglist*]
Schicken einer Antwort an den Verfasser jeder Nachricht in der *msglist* . Die Betrifft-Zeile wird der ersten Nachricht entnommen. Wenn "record" auf einen Dateinamen gesetzt ist, wird die Antwort am Ende dieser Datei gesichert (vgl. UMGEBUNGSVARIABLEN).

reply [*message*]
respond [*message*]
Antwort auf die angegebene Nachricht, auch an alle anderen Empfänger der Nachricht. Wenn "record" auf einen Dateinamen gesetzt ist, wird die Antwort am Ende dieser Datei gesichert (vgl. UMGEBUNGSVARIABLEN).

Save [*msglist*]
Sichern der angegebenen Nachrichten in einer Datei, deren Name vom Verfasser der ersten Nachricht abgeleitet wird. Der Dateiname ist der Name des Verfassers ohne die Netzadressen. Vgl. auch die Kommandos **Copy**, **followup**, und **Followup** und "outfolder" (UMGEBUNGSVARIABLEN).

save [*filename*]
save [*msglist*] *filename*
Sichern der angegebenen Nachrichten in der angegebenen Datei. Falls die Datei nicht vorhanden ist, wird sie erstellt. Die Nachricht wird aus der *mailbox* gelöscht, wenn *mailx* endet, sofern nicht "keepsave" gesetzt ist (vgl. auch UMGEBUNGSVARIABLEN und die Kommandos **exit** und **quit**).

set
set *name*
set *name=string*
set *name=number*
Definieren einer Variablen, die mit *name* bezeichnet wird. Der Variablen kann eine Null, eine Zeichenfolge oder ein numerischer Wert zugewiesen werden. **Set** allein gibt alle definierten Variablen und ihre Werte aus. Vgl. UMGEBUNGSVARIABLEN hinsichtlich der Einzelbeschreibungen der *mailx* Variablen.

shell
Ruft eine interaktive Shell auf (vgl. auch "SHELL" (UMGEBUNGSVARIABLEN)).

size [*msglist*]
Gibt die Größe in Zeichen der angegebenen Nachrichten aus.

source *filename*
Lesen der Kommandos aus der angegebenen Datei und dann Rückkehr in den Kommando-Modus.

top [*msglist*]
Ausgabe der obersten Zeilen der angegebenen Nachrichten. Wenn die Variable "toplines" (obere Zeilen) gesetzt ist, wird diese als die auszugebende Zeilenanzahl genommen (vgl. UMGEBUNGSVARIABLEN). Der Standard ist 5.

touch [*msglist*]
Durchlaufen der angegebenen Nachrichten. Wenn eine Nachricht in *msglist* nicht explizit in eine Datei gesichert wurde, wird sie bei normaler Beendigung nach *mbox* geschrieben oder in die durch die Umgebungsvariable MBOX angegebene Datei. Vgl. **exit** und **quit**.

Type [*msglist*]
Print [*msglist*]
Ausgabe der angegebenen Nachricht am Bildschirm, einschließlich aller Nachrichtenkopf-Felder. Hebt durch das **ignore** Kommando eingestellte Fehlerunterdrückung auf.

type [*msglist*]
print [*msglist*]
Ausgabe der angegebenen Nachrichten. Wenn "terminal" gesetzt ist, werden die Nachrichten, die länger als die von der "terminal"-Variablen angegebenen Zeilenzahl sind, mit dem Kommando seitenweise ausgegeben, das in der "PAGER"-Variablen angegeben ist. Das Standardkommando ist *pg*(1) (vgl. UMGEBUNGSVARIABLEN) .

undelete [*msglist*]
Wiederherstellen der angegebenen gelöschten Nachrichten. Stellt nur Nachrichten wieder her, die in der aktuellen Postsitzung gelöscht wurden. Wenn "autoprint" gesetzt ist, wird die letzte der wiederhergestellten Nachrichten ausgegeben (vgl. UMGEBUNGSVARIABLEN).

unset *name* ...
Löschen der angegebenen Variablen. Wenn die Variable von der Ausführungs-Umgebung importiert wurde (d. h. eine Shell-Variable), dann kann sie nicht gelöscht werden.

version
Ausgabe der aktuellen Version und Versionsdatum.

visual [*msglist*]
Editieren der angegebenen Nachrichten mit einem Bildschirm-Editor. Die Nachrichten werden in eine temporäre Datei geschrieben und die "VISUAL" Variable wird zur Bestimmung des Editornamens verwendet (vgl.UMGEBUNGSVARIABLEN).

write [*msglist*] *filename*
Schreiben der angegebenen Nachrichten in die angegebene Datei, jedoch ohne Nachrichtenkopf und abschließende Leerzeile. Sonst gleichwertig mit dem Kommando save.

xit
exit
mailx beenden, ohne die *mailbox* zu ändern. Nachrichten werden nicht in der *mbox* gesichert (Vgl. auch quit).

z[+ | −]
Die Nachrichtenkopfanzeige eine volle Bildschirmseite vorwärts oder rückwärts rollen. Die Anzahl angezeigter Nachrichtenköpfe wird mit der "screen"- Variablen gesetzt (vgl. UMGEBUNGSVARIABLEN).

TILDE-ESCAPE-FOLGEN
Die folgenden Kommandos können nur im *input mode* eingegeben werden, indem man am Anfang einer Zeile das Tilde-Escape-Zeichen (~) eingibt. Vgl. "escape" (UMGEBUNGSVARIABLEN) zur Änderung dieses Sonderzeichens.

~! *shell-command*
Wechsel in die Shell.

~.
Dateiende simulieren (beendet Nachrichteneingabe).

~: *mail-command*
~_ *mail-command*
Aufforderung zur Kommandoeingabe wird dargestellt. Nur gültig, falls eine Nachricht gesendet werden soll und gleichzeitig Post gelesen wird.

~?
Ausgabe einer Zusammenfassung der Tilde-Escape-Zeichen.

~A
Einfügen der Signier-Zeichenfolge "sign" in die Nachricht (vgl. UMGEBUNGSVARIABLEN).

~a
Einfügen der Signier-Zeichenfolge "sign" in die Nachricht (vgl. UMGEBUNGSVARIABLEN).

~b *name ...*
Hinzufügen von *name*s auf die blinde Durchschlagsliste (blind carbon copy - Bcc).

~c *name ...*
Hinzufügen von *name*s auf die Durchschlagsliste (carbon copy - Cc).

~d
Einlesen der Datei *dead.letter*. Vgl. "DEAD" (UMGEBUNGSVARIABLEN) bezüglich einer Beschreibung dieser Datei.

~e
Aufrufen des Editors für Teilnachricht. Vgl. auch "EDITOR" (UMGEBUNGSVARIABLEN).

~f [*msglist*]
Weiterleiten der angegebenen Nachrichten. Die Nachrichten werden ohne Änderung in die Nachricht eingefügt.

~h
Eingabe-Aufforderung für Betrifft-Zeile und To, Cc, und Bcc-Listen. Wenn das Feld mit einem Anfangswert angezeigt wird, kann es genauso editiert werden, als ob es gerade eingegeben worden wäre.

~i *string*
In den Text der Nachricht den Wert der benannten Variable einfügen. Zum Beispiel ist **~A** gleichwertig zu '**~i** Sign.' Auf gesetzte und exportierte Umgebungsvariablen der Shell kann ebenfalls über '**~i**' zugegriffen werden.

~m [*msglist*]
Einfügen der angegebenen Nachrichten in den Brief, wobei der neue Text nach rechts verschoben wird. Nur gültig, wenn eine Nachricht gesendet und gleichzeitig Post gelesen wird.

~p
Ausgabe der eingegebenen Nachricht.

~q
Verlassen des Eingabemodus durch Simulieren einer Unterbrechung. Wenn der Hauptteil der Nachricht nicht Null ist, wird die Teilnachricht in *dead.letter* gesichert. Vgl. "DEAD" (UMGEBUNGSVARIABLEN) bezüglich einer Beschreibung dieser Datei.

~r *filename*
~~< *filename*
~~ < !*shell-command*
Einlesen der angegebenen Datei. Wenn das Argument mit einem Ausrufezeichen (!) beginnt, wird die übrige Zeichenfolge als ein beliebiges Shell-Kommando angesehen und ausgeführt, wobei die Standardausgabe dieses Kommandos in die Nachricht eingefügt wird.

~s *string* ...
Setzen der Betrifft-Zeile auf *string*.

~t *name* ...
Hinzufügen der angegebenen *name*s zur "To"-Liste (Empfängerliste).

~v
Aufrufen eines bevorzugten Bildschirm-Editors für die Teilnachricht. Vgl. auch "VISUAL" (UMGEBUNGSVARIABLEN).

~w *filename*
Schreiben der Teilnachricht ohne Kopf in die angegebene Datei.

~x
Beenden wie bei ~q , außer daß die Nachricht in *dead.letter* gesichert wird.

~| *shell-command*
Den Hauptteil der Nachricht über eine Pipe an das angegebene *shell-command* schicken. Wenn das *shell-command* einen erfolgreichen Endestatus liefert, wird die Nachricht durch die Ausgabe des Kommandos ersetzt.

UMGEBUNGSVARIABLEN

Die folgenden Umgebungsvariablen sind der Ausführungsumgebung entnommen und können innerhalb von *mailx* nicht geändert werden.

HOME=*directory*
Das Ausgangsverzeichnis des Benutzers

MAILRC=*filename*
Der Name der Startdatei. Standard ist $HOME/.mailrc.

Die folgenden Variablen sind interne *mailx* Variablen. Sie können von der Ausführungsumgebung importiert oder jederzeit über das Kommando **set** gesetzt werden. Das Kommando **unset** kann zum Löschen der Variablen verwendet werden.

addsopt
Standardmäßig eingeschaltet. Wenn */bin/mail* nicht verwendet wird, sollte **noaddsopt** angegeben werden. (Vgl. nachstehende ACHTUNG!)

allnet
Alle Netznamen, deren letzte Komponente (Login-Name) übereinstimmen, werden gleich behandelt. Dadurch wird erreicht, daß die Nachrichten-Angaben in *msglist* ähnliche Wirkung haben. Standard ist **noallnet**. Vgl. auch das Kommando **alt**ernates und die "metoo" Variable.

append
Bei Beendigung, Nachrichten ans Ende der Datei *mbox* anhängen, anstatt sie an den Anfang zu stellen. Standard ist **noappend.**

askcc
Eingabe-Aufforderung für die Cc-Liste, nachdem Nachricht eingegeben wird. Standard ist **noaskcc.**

asksub
Eingabe-Aufforderung für Betrifft, wenn es auf der Kommandozeile nicht mit der Option **-s** angegeben wird. Standardmäßig eingeschaltet.

autoprint
Aktiviert automatische Ausgabe der Nachricht nach **d**elete und **u**ndelete Kommandos. Standard ist **noautoprint.**

bang
Aktiviert die besondere Bedeutung des Ausrufzeichens (!) für Kommandozeilen zum Wechseln in die Shell wie in *vi*(1). Standard ist **nobang.**

cmd=*shell-command*
Setzen des Standardkommandos für das **p**ipe Kommando. Kein Standardwert.

conv=*conversion*
Umwandeln der uucp Adressen in den angegebenen Adressenstil. Die einzige gültige Umwandlung ist jetzt *internet*, das von einem Postzustellungs-Programm fordert, daß die RFC822-Norm für elektronische Post-Adressierung eingehalten wird. Umwandlung wird standardmäßig abgeschaltet. Vgl. auch "sendmail" und die Option **-U** in der Kommandozeile.

crt=*number*
Die Nachrichten, die mehr als *number* Zeilen haben, werden über eine Pipe an das Kommando geschickt, das durch den Wert der "PAGER"-Variablen angegeben ist (standardmäßig *pg*(1)). Standardmäßig abgeschaltet.

DEAD=*filename*
Der Name der Datei, in die bei unvorhergesehener Unterbrechung Teilbriefe gesichert werden. Standard ist $HOME/-dead.letter.

debug
Ausführliche Fehlermeldungen für die Fehlersuche ausgeben. Nachrichten werden nicht gesendet. Standard ist **nodebug**.

dot
Nimmt einen Punkt allein auf einer Zeile bei Eingabe an dem Terminal als Dateiende an. Standard ist **nodot**.

EDITOR=*shell-command*
Das Kommando, das aufgerufen wird, wenn **edit** oder **~e** Kommandos verwendet werden. Standard ist *ed*(1).

escape=*c*
Escape-Zeichen ~ durch *c* ersetzen. Tritt in Kraft bei nächster gesendeter Nachricht.

folder=*directory*
Das Verzeichnis zur Sicherung der Standardpostdatei. Vom Benutzer angegebene Dateinamen, die mit einem (+) beginnen, werden zu einem richtigen Dateinamen, indem dem Dateinamen dieser Verzeichnisname vorangestellt wird. Wenn *directory* nicht mit einem Schrägstrich (/) beginnt, wird $HOME vorangestellt. Die Verwendung der (+)-Konstruktion in einer *mailx* Kommandozeile setzt voraus, daß "folder" eine exportierte *sh* Umgebungsvariable ist. Es gibt keinen Standardwert für die "folder" -Variable. Vgl. auch "outfolder" weiter unter.

header
Beim Start von *mailx* wird die Zusammenfassung der Nachrichtenköpfe ausgegeben. Standardmäßig aktiviert.

hold
Alle eingelesenen Nachrichten in der *mailbox* aufbewahren statt in der Standard-Datei *mbox*. Standard ist **nohold**.

ignore
Ignorieren von Unterbrechungen beim Eingeben von Nachrichten. Praktisch bei störungsanfälligen Wählleitungen. Standard ist **noignore**.

ignoreeof
Ignoriert Dateiende während Nachrichteneingabe. Eingabe muß mit einem Punkt (.) in einer Zeile allein oder durch das Kommando ~. abgeschlossen werden. Standard ist **noignoreeof**. Vgl. auch oben erwähnter "dot".

keep
Wenn die *mailbox* leer ist, wird sie auf Länge Null abgeschnitten statt entfernt. Standardmäßig abgeschaltet.

keepsave
Nachrichten, die in anderen Dateien gesichert wurden, in *mailbox* aufbewahren, anstatt sie zu löschen. Standard ist **nokeepsave**.

MBOX=*filename*
Der Name der Datei zur Sicherung von Nachrichten, die gelesen wurden. Das Kommando **xit** hebt diese Funktion auf, das gleiche gilt für die Sicherung der Nachricht in einer anderen Datei. Standard ist $HOME/mbox.

metoo
Wenn Ihr Login als Empfänger erscheint, wird es nicht von der Liste gestrichen. Standard ist **nometoo**.

LISTER=*shell-command*
Das Kommando (und Optionen), das zur Ausgabe des Inhalts des "folder"-Verzeichnisses verwendet werden soll. Der Standard ist *ls*(1).

onehop
Bei Beantwortung einer Nachricht, die ursprünglich an mehrere Empfänger geschickt wurde, müssen die anderen Empfängeradressen normalerweise relativ zur Station des Verfassers angegeben werden. Mit diesem Schalter muß die Empfängeradresse nicht abgestimmt werden. Das erhöht die Effizienz in einem Netz, wo jede Station direkt zu allen anderen Stationen senden kann (d. h. in einem Schritt).

outfolder
Dateien, in denen abgeschickte Nachrichten aufgezeichnet werden, werden in dem durch die "folder"-Variable angegebenen Verzeichnis angelegt, es sei denn, der Pfadname ist absolut. Standard ist **nooutfolder**. Vgl. oben erwähnten "folder" und die Kommandos **Save**, **Copy**, **followup** und **Followup**.

page
Verwendung mit dem Kommando **pipe**, um einen Formularvorschub nach jeder Nachricht einzufügen, die durch die Pipe geschickt wird. Standard ist **nopage**.

PAGER=*shell-command*
Das Kommando wird als ein Filter zur Ausgabe-Paginierung verwendet. Dieses Kommando kann auch zur Angabe der zu benutzenden Optionen verwendet werden. Standard ist *pg*(1).

prompt=*string*
Setzen der Eingabe-Aufforderung für *command mode* auf *string*. Standard ist "? ".

quiet
Ausgabe der Anfangsnachricht und Version beim Starten von *mailx* wird unterdrückt. Standard ist **noquiet**.

record=*filename*
Aufzeichnen der gesamten abgeschickten Post in *filename*. Standardmäßig abgeschaltet. Vgl. auch oben erwähnten "outfolder".

save
Aktivieren der Sicherung von Nachrichten in *dead.letter* bei Unterbrechung oder Sendefehler. Vgl. "DEAD" hinsichtlich einer Beschreibung dieser Datei. Standardmäßig aktiviert.

screen=*number*
Setzen der Zeilenanzahl für eine volle Bildschirmseite von Kopfzeilen für das Kommando **headers**.

sendmail=*shell-command*
Alternatives Kommando zum Schicken von Nachrichten. Standard ist */bin/rmail*(1).

sendwait
Warten auf die Beendigung eines mail-Prozesses im Hintergrund, ehe zurückgekehrt wird. Standard ist **nosendwait**.

SHELL=*shell-command*
Der Name des gewünschten Kommando-Interpreters. Standard ist *sh*(1).

showto
Wenn die Nachricht von Ihnen ist, wird bei Anzeige der Kopfzeilen der Name des Empfängers statt des Verfassernamens ausgegeben.

sign=*string*
Ist die Variable, die in den Text einer Nachricht eingefügt wird, falls das Kommando **~a** (autograph) angegeben wird. Kein Standard (vgl. auch **~i** (TILDE-ESCAPE-FOLGEN)).

Sign=*string*
Ist die Variable, die in den Text einer Nachricht eingefügt wird, falls das Kommando **~A** (autograph) angegeben wird. Kein Standard (vgl. auch **~i** (TILDE-ESCAPE-FOLGEN)).

toplines=*number*
Die Anzahl der bei dem Kommando **top** auszugebenden Zeilen. Standard ist 5.

VISUAL=*shell-command*
Der Name eines bevorzugten Bildschirm-Editors. Standard ist *vi*(1).

DATEIEN

$HOME/.mailrc	Individuelle Startdatei
$HOME/mbox	Sekundäre Speicherdatei
/usr/mail/*	Post-Verzeichnis
/usr/lib/mailx/mailx.help*	Help-Dateien
/usr/lib/mailx/mailx.rc	Wahlweise globale Startdatei
/tmp/R[emqsx]*	Temporäre Dateien

SIEHE AUCH:

ls(1), mail(1), pg(1).

ACHTUNG!

Die Optionen **-h**, **-r** und **-U** können nur verwendet werden, wenn *mailx* nicht auf */bin/mail* beruht.

FEHLER

Wo *shell-command* zugelassen ist, sind Argumente nicht immer erlaubt. Experimentieren wird empfohlen.

Interne Variablen, die von der Ausführungsumgebung importiert wurden, können nicht mit **unset** gelöscht werden.

Die gesamte Internet-Adressierung wird nicht voll von *mailx* unterstützt. Die neuen Normen bedürfen einer gewissen Einführungszeit.

Versuche, einzeilige Nachrichten zu schicken, die nur aus einem "." bestehen, werden bei *mail*(1) wie Ende von Nachrichten behandelt (das Standard-Postlieferungsprogramm).

BEZEICHNUNG

makekey – Chiffrierschlüssel generieren

ÜBERSICHT

/usr/lib/makekey

BESCHREIBUNG

Makekey erhöht die Sicherheit von Verschlüsselungsverfahren, die auf einem Chiffrierschlüssel beruhen, da der Zeitbedarf zum Ausprobieren der möglichen Schlüssel erheblich ansteigt. Das Programm liest 10 Byte von der Standardeingabe und schreibt 13 Byte auf die Standardausgabe. Die ausgegebene Bytefolge wird in Abhängigkeit von der Eingabe anhand eines Verfahrens berechnet, das die Entschlüsselung erschwert, weil ein nicht unbeträchtlicher Bruchteil einer Sekunde dazu benötigt wird.

Die ersten acht Eingabebyte (der *Eingabeschlüssel*) dürfen aus beliebigen ASCII-Zeichen bestehen. Die beiden letzten Byte (die *Chiffre*) sollten aus Ziffern, Groß- und Kleinbuchstaben sowie den Zeichen . und / gebildet werden. Die Chiffre wird in die ersten beiden Zeichen der ausgegebenen Bytefolge übernommen. Die verbleibenden 11 Ausgabezeichen bilden den *Ausgabeschlüssel*. Sie werden dem gleichen Zeichenvorrat entnommen wie die Chiffre.

Die Umformung des Eingabeschlüssels läuft grundsätzlich folgendermaßen ab: Anhand der Chiffre wird einer von 4.096 Verschlüsselungsalgorithmen ausgewählt. Diese basieren alle auf dem DES-Algorithmus des National Bureau of Standards, der aber in 4.096 verschiedene Verfahren aufgeteilt wird. Mit dem Eingabeschlüssel als Schlüssel wird eine konstante Zeichenfolge gebildet und an den Algorithmus übergeben, der daraufhin mehrfach durchlaufen wird. Die 64 Ausgabebit des Algorithmus werden auf die 66 Bit des *Ausgabeschlüssels* verteilt.

Makekey ist für Programme vorgesehen, die Daten verschlüsseln müssen. Ein- und Ausgabe sind in der Regel Pipes.

SIEHE AUCH

ed(1), crypt(1), vi(1).
passwd(4) im *Programmer's Reference Manual.*

ACHTUNG

Dieses Kommando ist Bestandteil der Dienstprogramme zur Datenschutz-Administration, die nur in den USA erhältlich sind.

BEZEICHNUNG

mesg – Nachrichten erlauben oder verbieten

ÜBERSICHT

mesg [**-n**] [**-y**]

BESCHREIBUNG

mesg mit dem Argument **n** verbietet Nachrichten über *write*(1), indem die Schreibberechtigung auf dem Terminal des Benutzers für alle anderen Benutzer entzogen wird. *mesg* mit dem Argument **y** setzt diese Berechtigung wieder in Kraft. Ohne Argument berichtet *mesg* den aktuellen Zustand, ohne ihn zu ändern.

DATEIEN

/dev/tty*

SIEHE AUCH:

write(1).

DIAGNOSE

Endestatus ist 0, wenn Nachrichten empfangbar sind, 1 wenn nicht, 2 bei Fehlern.

BEZEICHNUNG

mkdir – Verzeichnisse erstellen

ÜBERSICHT

mkdir [**-m** mode] [**-p**] dirname ...

BESCHREIBUNG

mkdir erstellt die angegebenen Verzeichnisse mit Modus 777 (Änderung möglich mit *umask* (1)).

Standardmäßige Einträge in ein Verzeichnis (z.B. die Dateien . für das Verzeichnis selbst und .. für das übergeordnete) werden automatisch durchgeführt. *mkdir* kann diese Einträge nicht mit Namen erstellen. Erstellung eines Verzeichnisses erfordert Schreiberlaubnis im übergeordneten Verzeichnis.

Die Eigentümernummer und Gruppennummer der neuen Verzeichnisse sind jeweils auf die realen Benutzer- und Gruppennummern des Prozesses gesetzt.

Zwei Optionen stehen bei *mkdir* zur Verfügung:

-m Diese Option gestattet Benutzern, den für die neuen Verzeichnisse zu verwendenden Modus anzugeben. Die Möglichkeiten für Modus werden bei *chmod*(1) erklärt.

-p Mit dieser Option erstellt *mkdir* das Verzeichnis *dirname,* indem alle nicht vorhandenen übergeordneten Verzeichnisse zuerst erstellt werden.

BEISPIEL

Zur Erstellung der Unterverzeichnis-Struktur **ltr/jd/jan** geben Sie ein:

```
mkdir -p ltr/jd/jan
```

SIEHE AUCH

sh(1), rm(1), umask(1).
intro(2), mkdir(2) im *Programmer's Reference Manual.*

DIAGNOSE

mkdir liefert den Endecode 0, wenn alle in der Kommandozeile angegebenen Verzeichnisse erfolgreich erstellt sind. Andernfalls wird eine Fehlermeldung angegeben und ungleich Null zurückgegeben. Ein Fehlercode wird in *errno* gespeichert.

BEZEICHNUNG

netbench – Netzwerk-Test

ÜBERSICHT

netbench protocol sendhost sinkhost repetitions messagesize

BESCHREIBUNG

Netbench ist der Client des Servers *netbenchd*(1C). Wenn das Programm *netbench* auf dem Host *sendhost* gestartet wird, stellt es eine Verbindung zu dem *netbenchd*-Server her, der auf dem Host *sinkhost* läuft, und sendet *repetition* Pakete in Länge von *messagesize* Byte in dem angegebenen Protokoll *protocol* (entweder UDP oder TCP).

Der Client *netbench* und der zugehörige Server *netbenchd* sammeln statistische Daten über die Datenübertragung und berechnen unabhängig voneinander Leistungs- und Auslastungskennziffern für eine Sitzung.

Der Client nimmt folgende Messungen vor:

- Anzahl gesendeter Byte
- Anzahl gesendeter Pakete
- für das Senden der Pakete verbrauchte tatsächliche Zeit
- für das Senden der Pakete verbrauchte Systemzeit
- für das Senden der Pakete verbrauchte Benutzerzeit
- Durchsatz in Byte pro Sekunde
- Durchsatz in Bit pro Sekunde
- Durchsatz in Paketen pro Sekunde
- mittlere Verzögerung pro Paket in Sekunden

Der Server nimmt folgende Messungen vor:

- Anzahl empfangener Byte
- Anzahl empfangener Pakete
- für den Empfang der Pakete verbrauchte Gesamtzeit
- für den Empfang der Pakete verbrauchte Systemzeit
- für den Empfang der Pakete verbrauchte Benutzerzeit
- Durchsatz in Byte pro Sekunde
- Durchsatz in Bit pro Sekunde
- Durchsatz in Paketen pro Sekunde
- mittlere Verzögerung pro Paket in Sekunden

Vor der Beendigung einer Sitzung übermittelt der Server seine Ergebnisse an den Client. Er selbst vergißt diese Messungen am Ende der Sitzung. Der Client sammelt sowohl die Testdaten des Servers als auch seine eigenen und schreibt alle verfügbaren Kennziffern in die Standardausgabedatei. Diese Kennziffern werden in verdichteter Form auch zur Fortschreibung der Datei *netbench.log* im aktuellen Arbeitsverzeichnis des Client benutzt. Der Server gibt *keine* Daten aus (abgesehen von Fehlermeldungen auf der Systemkonsole).

Netbench unterhält zwei Sockets: einen Steuersocket zum Austausch von Steuerinformationen mit dem *netbenchd*-Server und einen Datensocket, über den die Daten zum Server gesendet werden. Wenn *netbench* ausgeführt wird, läuft folgendes Protokoll ab:

1) Über den Steuersocket wird eine Verbindung zum Server angefordert, beim TCP-Protokoll außerdem eine Verbindung über den Datensocket.
2) Die Messungen werden gestartet.
3) Über den Datensocket wird die gewünschte Anzahl von Paketen gesendet. Zum Abschluß wird ein "Übertragungsende"-Signal über den Datensocket verschickt.
4) Die Messungen werden beendet.
5) Über den Steuersocket wird die Endemeldung gesendet.
6) Die Leistungs- und Auslastungskennziffern für den Client werden berechnet. Über den Steuersocket werden die vom Server berechneten statistischen Daten eingelesen. Alle Testkennziffern werden in die Standardausgabedatei und in eine Protokolldatei geschrieben.
7) Die Steuerverbindung wird abgebaut, beim TCP-Protokoll auch die Datenverbindung.

In der Datei /etc/hosts müssen Einträge für die Hosts *sendhost* und *sinkhost* vorhanden sein (siehe *hosts(5)*). Außerdem müssen in der Datei /etc/services sowohl TCP- als auch UDP-Einträge für die Dienste *benchdata* (Datenverbindung) und *benchctl* (Steuerverbindung) vorhanden sein. /etc/services könnte z. B. die folgenden Einträge enthalten:

```
benchctl    2048/udp  # wird tatsächlich an einen TCP-Socket gebunden

benchdata   2049/udp
benchctl    2050/tcp
benchdata   2051/tcp
```

Weitere Informationen sind unter *services(5)* zu finden.

Auf dem Host *sinkhost* muß natürlich ein *netbenchd*-Server aktiv sein, damit *netbench* seine Aufgabe erfüllen kann.

BEISPIELE

```
netbench udp ramses cleo 1000 1024
netbench tcp ramses ramses 1000 2048
```

DATEIEN

netbench.log
/etc/hosts
/etc/services

SIEHE AUCH

hosts(5), services(5), netbenchd(1C)

FEHLER

Über UDP kann immer nur ein *netbench*-Client von einem *netbenchd*-Server bedient werden. Dagegen können viele TCP-Clients gleichzeitig von einem Server bedient werden.

BEZEICHNUNG

netstat – anzeigen des Netzwerk-Status

ÜBERSICHT

netstat [**-Aan**] [**-f** *Adreßfamilie*] [*System*] [*Kern*]
netstat [**-imnrs**] [**-f** *Adreßfamilie*] [*System*] [*Kern*]
netstat [**-n**] [**-I** *Schnittstelle*] *Intervall* [*System*] [*Kern*]

BESCHREIBUNG

Das Kommando *netstat* zeigt den Inhalt verschiedener netzwerkbezogener Datenstrukturen symbolisch an. Je nach Optionen für die dargestellten Daten stehen mehrere Ausgabeformate zur Verfügung. Mit der ersten Form des Kommandos wird eine Liste der aktiven Sockets für die einzelnen Protokolle erstellt. Mit der zweiten Form wird, je nach der gewählten Option, der Inhalt einer anderen Netzwerkdatenstrukturen dargestellt. Wird die dritte Form verwendet und ein *Intervall* definiert, zeigt *netstat* fortlaufend die Daten zum Paketverkehr an den konfigurierten Netzwerkschnittstellen an.

Die Optionen haben folgende Bedeutung:

-A Bei der Standardmaske wird die Adresse aller Protokollsteuerblöcke angezeigt, die zu Sockets gehören; wird zur Testhilfe verwendet.

-a Bei der Standardmaske wird der Status aller Sockets angezeigt. Normalerweise erscheinen von Servern benutzte Sockets nicht.

-i Zeigt den Status von Schnittstellen, die automatisch konfiguriert wurden (statisch in eine System konfigurierte Schnittstellen, die beim Laden des Systems jedoch nicht lokalisiert werden, werden nicht gezeigt).

-I *Schnittstelle*
Zeigt nur zu dieser Schnittstelle gehörende bzw. dafür spezifische Daten an; wird, wie unten beschrieben, mit einem *Intervall* verwendet.

-m Zeigt von den Speicherverwaltungsroutinen aufgezeichnete statistische Daten an (das Netzwerk verwaltet einen eigenen Pool von Speicherpuffern).

-n Zeigt die Netzwerkadressen als Zahlen (normalerweise interpretiert *netstat* die Adressen und versucht sie symbolisch anzuzeigen). Diese Option kann mit allen Anzeigeformaten verwendet werden.

-s Zeigt statistische Daten pro Protokoll.

-r Zeigt die Leitwegtabellen. Ist auch **-s** angegeben, werden statt dessen Leitwegstatistiken angezeigt.

-f *Adreßfamilie*
Beschränkt statistische Daten oder Berichte des Adreßsteuerblocks auf die angegebene *Adreßfamilie.* Die folgenden Adreßfamilien werden erkannt: *inet* für **AF_INET**, *ns* für **AF_NS** und *unix* für **AF_UNIX**.

Mit den Argumenten *System* und *Kern* können die Standardwerte "/unix" und "/dev/kmem" ersetzt werden.

Die Standardmaske für aktive Sockets zeigt die lokale und die entfernte Adresse, die Größe für Sende- und Empfangswarteschlangen (in Byte), das Protokoll und den internen Status des Protokolls an. Die Adreßformate haben die Form "host.anschluß" oder "netzwerk.anschluß", wenn die Adresse eines Sockets eine Netzwerkadresse, aber keine bestimmte Host-Adresse angibt. Sind Host- und Netzwerkadresse bekannt, werden sie symbolisch gemäß den Datenbanken */etc/hosts* bzw. */etc/networks* angezeigt. Ist ein symbolischer Name für eine Adresse unbekannt, oder wurde die Option **-n** angegeben, wird die Adresse numerisch, je nach Adreßfamilie, ausgegeben. Nähere Angaben zum Internet-"Punktformat" enthält der Handbucheintrag *inet*(3N). Nicht näher spezifizierte (globale) Adressen und Anschlüsse erscheinen als "*".

Die Schnittstellenmaske zeigt in einer Tabelle kumulierte Daten über die Anzahl der übertragenen Pakete, Fehler und Kollisionen aus. Die Netzwerkadressen der Schnittstelle und die maximale Übertragungseinheit ("mtu") werden ebenfalls angezeigt.

Die Leitweg-Tabellenmaske zeigt die verfügbaren Leitwege und ihren Status an. Jeder Leitweg besteht aus einem Ziel-Host oder -Netzwerk und einem Gateway zum Weiterleiten von Paketen. Im Schalterfeld angezeigt, welchen Status der Leitweg hat ("U" für aktiv ("up")), ob der Leitweg zu einem Gateway ("G") führt und ob der Leitweg dynamisch durch Umleitung auf einen anderen Gateway ("D") angelegt wurde. Direkte Leitwege werden für alle mit dem lokalen Host verbundenen Schnittstellen angelegt; das Gateway-Feld für solche Einträge enthält die Adresse der Ausgangsschnittstelle. Das Feld refcnt gibt die aktuelle Anzahl der aktiven Zugriffe auf den Leitweg an. Verbindungsorientierte Protokolle benutzen für die Dauer einer Verbindung normalerweise nur einen Leitweg, verbindungslosen Protokollen dagegen wird ein Leitweg zugeordnet, während sie an dasselbe Ziel senden. Das Auslastungsfeld enthält einen Zähler für die Anzahl der auf diesem Leitweg gesendeten Pakete. Der Schnittstelleneintrag gibt an, welche Netzwerkschnittstelle für den Leitweg benutzt wird.

Wird *netstat* mit dem Argument *Intervall* aufgerufen, wird ein laufender Zähler mit statistischen Daten zu den Netzwerkschnittstellen angezeigt. Diese Maske besteht aus einer Spalte für die Primärschnittstelle (die er-

ste bei der automatischen Konfiguration gefundene Schnittstelle) und einer Spalte, in der Daten zu allen Schnittstellen zusammengefaßt sind. Die Primärschnittstelle kann mit der Option *-I* durch eine andere Schnittstelle ersetzt werden. Die erste Zeile jedes Informationsbildschirms enthält eine Zusammenfassung der Daten seit dem letzten Laden. Die nachfolgenden Ausgabezeilen enthalten Werte, die während des vorherigen Intervalls gesammelt wurden.

SIEHE AUCH

hosts(5), networks(5), protocols(5), services(5), trpt(8C)

FEHLERQUELLEN

Der Begriff Fehler ist schlecht definiert.

BEZEICHNUNG

newform – Format einer Textdatei ändern

ÜBERSICHT

newform [**-s**] [**-i**tabspec] [**-o**tabspec] [**-b**n] [**-e**n] [**-p**n] [**-a**n] [**-f**] [**-c**char] [**-l**n] [files]

BESCHREIBUNG

newform liest Zeilen der angegebenen Dateien *files* oder die Standardeingabe, wenn keine Eingabedatei genannt wird, und gibt die Zeilen auf der Standardausgabe aus. Zeilen werden gemäß der auf der Kommandozeile angegebenen Optionen umformatiert.

Mit Ausnahme von **-s** können Optionen auf der Kommandozeile in beliebiger Reihenfolge erscheinen, wiederholt und mit den wahlweisen *files* vermischt werden. Kommandozeilen-Optionen werden in der angegebene Reihenfolge verarbeitet. Dies bedeutet, daß Optionsfolgen wie "**-e15 -l60**" andere Ergebnisse als die Folge "**-l60 -e15**" ergeben. Optionen werden auf alle *files* in der Kommandozeile angewendet.

-s Trennt auf jeder Zeile führende Zeichen bis zum ersten Tabulatorzeichen ab und stellt bis zu 8 der abgetrennten Zeichen ans Ende der Zeile. Bei mehr als 8 abgetrennten Zeichen (wobei das erste Tabulatorzeichen nicht gezählt wird) wird das achte Zeichen durch einen * ersetzt und alle rechts davon stehenden Zeichen werden gelöscht. Der erste Tabulator wird immer gelöscht.

Eine Fehlermeldung und Programmabbruch wird eintreten, wenn diese Option für eine Datei ohne Tabulatorzeichen auf jeder Zeile verwendet wird. Die abgetrennten Zeichen werden intern gesichert, bis alle anderen angegebenen Optionen auf der Zeile durchgeführt worden sind. Die Zeichen werden dann am Ende der verarbeiteten Zeile hinzugefügt.

Zum Beispiel, um eine Datei mit führenden Ziffern, einem oder mehreren Tabulatoren und Text auf jeder Zeile, in eine Datei umzuwandeln, die mit dem Text anfängt und bei der alle Tabulatoren nach dem ersten durch Leerzeichen erweitert sind, bis Spalte 72 mit Leerzeichen aufgefüllt wird (oder bei Spalte 72 abgeschnitten wird) und die führenden Ziffern bei Spalte 73 anfangen, muß das folgende Kommando durchgeführt werden:

```
newform -s -i -l -a -e file-name
```

-i*tabspec* Eingabetabulator-Angabe: erweitert Tabulatoren durch Leerzeichen. *Tabspec* erkennt alle in *tabs*(1) beschriebenen Tabulatorformate. Zusätzlich kann *tabspec* -- sein, *new-*

form nimmt dann an, daß die Tabulatorangabe beim Lesen der ersten Zeile der Standardeingabe (vgl. *fspec*(4)) zu finden ist. Wenn kein *tabspec* angegeben ist, ist *tabspec* standardmäßig **-8**. Ein *tabspec* **-0** erwartet keine Tabulatoren; wenn welche gefunden werden, werden sie als **-1** behandelt.

-o*tabspec* Ausgabetabulator-Angabe: ersetzt Leerzeichen durch Tabulatoren gemäß der angegebenen Tabulatorangabe. Die Tabulatorangaben sind die gleichen wie bei **-i***tabspec*. Wenn kein *tabspec* angegeben ist, ist *tabspec* standardmäg **-8**. Ein *tabspec* **-0** bedeutet, daß in der Ausgabe keine Leerzeichen in Tabulatorzeichen umgewandelt werden.

-b*n* Schneidet *n* Zeichen vom Anfang der Zeile ab, wenn die Zeilenlänge größer als die effektive Zeilenlänge ist (vgl. **-l***n*). Standardmäßig werden so viele Zeichen abgetrennt, bis die effektive Zeilenlänge erreicht ist. Der Standardwert wird verwendet, wenn **-b** ohne *n* benutzt wird. Diese Option kann wie folgt zum Löschen der Folge-Nummer in einem COBOL-Programm verwendet werden:
newform -l1 -b7 file-name

-e*n* Wie bei **-b***n*, außer daß Zeichen vom Ende der Zeile abgeschnitten werden.

-p*n* *n* Zeichen (vgl. **-c***k*) am Anfang einer Zeile hinzufügen, wenn die Zeilenlänge kleiner als die effektive Zeilenlänge ist. Standardmäßig werden so viele Zeichen hinzugefügt wie zur Erzielung der effektiven Zeilenlänge erforderlich sind.

-a*n* Wie bei **-p***n*, außer daß Zeichen am Ende einer Zeile angehängt werden.

-f Schreibt die Formatzeile mit der Tabulatorangabe auf die Standardausgabe, bevor andere Zeilen ausgegeben werden. Die ausgegebene Formatzeile entspricht dem in der letzten **-o** Option angegebenen Format. Wenn keine **-o** Option angegeben ist, wird die ausgegebene Zeile die Voreinstellung **-8** enthalten .

-c*k* Das Zeichen, das vorne oder hinten angehängt wird, wird auf *k* gesetzt. Standardzeichen für *k* ist ein Leerzeichen.

-l*n* Setzt die effektive Zeilenlänge auf *n* Zeichen. Wird *n* nicht angegeben, gilt bei **-l** standardmäßig 72. Die Standardzeilenlänge ohne die Option **-l** ist 80 Zei-

chen. Es ist zu beachten, daß Tabulatoren und Backspace-Zeichen als ein Zeichen angesehen werden (Tabulatoren können mit **-i** in Leerzeichen erweitert werden). **-l1** muß verwendet werden, um die effektive Zeilenlänge kürzer als alle in einer Datei vorhandenen Zeilen zu machen, damit die Option **-b** aktiviert wird.

DIAGNOSE

Alle Fehlermeldungen sind schwerwiegend.

usage: ...	*newform* wurde mit einer ungültigen Option aufgerufen.
not -s format	Kein Tabulator auf einer Zeile.
can't open file	Selbsterklärend.
internal line too long	Eine Zeile überschreitet 512 Zeichen nach Erweiterung im internen Arbeitspuffer
tabspec in error	Eine Tabulatorangabe ist falsch formatiert oder angegebene Tabulator-Stopps erscheinen nicht.
tabspec indirection illegal	Eine aus einer Datei (oder Standardeingabe) gelesene *tabspec* darf keine *tabspec* enthalten, die auf eine andere Datei (oder Standardeingabe) verweist.

0 - normale Ausführung
1 - bei Fehlern

SIEHE AUCH

csplit(1), tabs(1).
fspec(4) im *Programmer's Reference Manual.*

FEHLER

newform registriert normalerweise nur physikalische Zeichen; bei den Optionen **-i** und **-o** registriert *newform* auch Backspace-Zeichen zur Ausrichtung der Tabulatoren in die entsprechenden logischen Spalten.

newform fordert den Benutzer nicht explizit zur Eingabe von *tabspec* auf, falls *tabspec* von der Standardeingabe gelesen werden soll (bei Verwendung von **-i--** oder **-o--**).

Wenn die Option **-f** verwendet wird, und die letzte angegebene **-o** Option **-o--** war, der ein **-o--** oder ein **-i--** vorausgingen, ist die Formatzeile mit der Tabulatorangabe falsch.

BEZEICHNUNG

newgrp – Gruppe neu definieren

ÜBERSICHT

newgrp [–] [group]

BESCHREIBUNG

newgrp ändert die Gruppenidentifikation eines Benutzers. Der Benutzer bleibt angemeldet und das aktuelle Verzeichnis ist unverändert, jedoch wird die Zugriffsberichtigung auf Dateien mit Berücksichtigung der neuen realen und effektiven Gruppennummern berechnet. Dem Benutzer wird durch *newgrp* immer eine neue Shell gegeben, die die aktuelle Shell ersetzt, unabhängig davon, ob *newgrp* erfolgreich beendet wurde oder auf Grund einer Fehlerbedingung (d. h. unbekannte Gruppe).

Exportierte Variablen behalten ihre Werte nach Aufruf von *newgrp*; jedoch werden alle nicht exportierten Variablen entweder auf den Standardwert zurückgesetzt oder auf Null gesetzt. Systemvariablen (wie zum Beispiel PS1, PS2, PATH, MAIL und HOME), sofern diese nicht vom System oder ausdrücklich vom Benutzer exportiert wurden, werden auf Standardwerte zurückgesetzt. Hat ein Benutzer zum Beispiel eine primäre Eingabe-Aufforderungsfolge (PS1), die nicht $ (Standard) ist und hat PS1 nicht exportiert, dann ist nach Aufruf von *newgrp,* ob erfolgreich oder nicht, PS1 auf die Standardeingabeaufforderung $ gesetzt. Es ist zu beachten, daß das Shell-Kommando *export (vgl. sh(1))* die Methode zum Exportieren von Variablen ist, damit diese ihren zugewiesenen Wert behalten, wenn neue Shell-Prozeduren aufgerufen werden.

Ohne Argumente setzt *newgrp* die Gruppenidentifikation auf die Gruppe zurück, die in der Paßwort-Datei für den Benutzer angegeben ist. Hiermit kann die Wirkung eines früheren *newgrp* -Aufrufs aufgehoben werden.

Wenn das erste Argument für *newgrp* ein – ist, wird die Umgebung so geändert, als hätte sich der Benutzer tatsächlich als Mitglied der neuen Gruppe neu angemeldet.

Ein Paßwort wird verlangt, wenn die Gruppe ein Paßwort und der Benutzer keins hat, oder wenn die Gruppe ein Paßwort hat und der Benutzer nicht in **/etc/group** als ein Mitglied dieser Gruppe aufgeführt ist.

DATEIEN

/etc/group Gruppendatei des Systems.
/etc/passwd Paßwortdatei des Systems

SIEHE AUCH:

login(1), sh(1) im *User's Reference Manual.*
group(4), passwd(4), environ(5) im *Programmer's Reference Manual.*

FEHLER

Es gibt keine geeignete Möglicheit zur Eingabe eines Paßworts in **/etc/group.** Die Verwendung von Gruppenpaßwörtern wird nicht empfohlen, weil sie keinen effektiven Datenschutz bieten. Gruppenpaßwörter werden in Zukunft kaum noch eingesetzt.

BEZEICHNUNG

news – Nachrichten ausgeben

ÜBERSICHT

news [**–a**] [**–n**] [**–s**] [items]

BESCHREIBUNG

news hält den Benutzer auf dem Laufenden über aktuelle Ereignisse. Allgemein werden diese Ereignisse durch Dateien im Verzeichnis **/usr/news** beschrieben.

Bei Aufruf ohne Argumente gibt *news* den Inhalt aller aktuellen Dateien in **/usr/news** aus, die allerneuesten zuerst. Vor jeder Datei werden entsprechende Kopfzeilen ausgegeben. *news* verwendet als "Aktualitätszeit" ("currency") die Änderungszeit der Datei **.news_time** im Home-Verzeichnis des Benutzers (dieses Verzeichnis wird mit Hilfe der Umgebungsvariablen **$HOME** bestimmt). Nur Dateien, die neuer als diese Aktualitätszeit sind, werden als "aktuell" angesehen.

–a Option veranlaßt *news* zur Angabe aller Nachrichten unabhängig von deren Aktualitätszeit. In diesem Fall wird die gespeicherte Zeit nicht geändert.

–n Option veranlaßt *news* zur Aufzeichnung der Namen der aktuellen Nachrichten, ohne deren Inhalt auszugeben und ohne Änderung der gespeicherten Zeit.

–s Option veranlaßt *news*, die Anzahl der vorhandenen aktuellen Nachrichten zu berichten, ohne deren Namen oder Inhalt anzugeben und ohne die gespeicherte Zeit zu ändern. Es wird empfohlen, einen solchen Aufruf von *news* in die eigene **.profile** Datei oder in die Datei **/etc/profile** des Systems einzufügen.

Alle anderen Argumente werden als spezielle Nachrichten angesehen, die ausgegeben werden sollen.

Wenn ein *delete* bei der Ausgabe einer Nachricht eingegeben wird, wird die Ausgabe gestoppt und es wird mit der nächsten Nachricht begonnen. Ein erneutes *delete* – gleich nach dem ersten – veranlaßt den Abbruch des Programms.

DATEIEN

/etc/profile
/usr/news/ *
$HOME/.news_time

SIEHE AUCH:

profile(4), environ(5) im *Programmer's Reference Manual.*

BEZEICHNUNG

nice – Kommando mit niedriger Priorität ausführen

ÜBERSICHT

nice [–increment] command [arguments]

BESCHREIBUNG

nice führt *command* mit einer niedrigeren CPU-Priorität aus. Falls das Argument *increment* (im Bereich 1-19) vorliegt, wird es benutzt; wenn nicht, wird ein Inkrement von 10 angenommen.

Der Systemverwalter kann Kommandos ausführen mit einer höheren Priorität als normal, indem ein negatives Inkrement verwendet wird wie z.B. **--10**.

SIEHE AUCH:

nohup(1).
nice(2) im *Programmer's Reference Manual*.

DIAGNOSE

nice gibt den Endestatus des entsprechenden Kommandos zurück.

FEHLER

Ein *increment* das größer als 19 ist, ist gleichwertig mit 19.

BEZEICHNUNG

nl - Filter für Zeilennumerierung

ÜBERSICHT

nl [-htype] [-btype] [-ftype] [-vstart#] [-iincr] [-p] [-lnum] [-ssep] [-wwidth] [-nformat] [-ddelim] file

BESCHREIBUNG

nl liest Zeilen aus der angegebenen Datei *file* oder der Standardeingabe, wenn keine *file* angegeben ist, und gibt die Zeilen auf der Standardausgabe aus. Zeilen werden links numeriert gemäß den angegebenen Kommando-Optionen.

nl sieht den gelesenen Text nach logischen Seiten durch. Zeilennumerierung wird am Anfang jeder logischen Seite zurückgesetzt. Eine logische Seite besteht aus einem Kopf, einem Hauptteil und einem Fußtextabschnitt. Leere Abschnitte sind gültig. Unterschiedliche Optionen zur unabhängigen Zeilennumerierung für Kopf, Hauptteil und Fußtext stehen zur Verfügung (z. B. können Kopf- und Fußtext ohne Zeilennumerierung sein, während Leerzeilen nur im Hauptteil numeriert werden).

Der Anfang eines logischen Seitenabschnitts wird durch Eingabezeilen signalisiert, die nur das (die) folgende(n) Begrenzungszeichen enthalten:

Zeileninhalt	*Beginn von*
\:\:\:	Kopf
\:\:	Hauptteil
\:	Fußtext

Wenn nicht anders durch Optionen angegeben, nimmt *nl* an, daß der gelesene Text eine einzige logische Seite ist, die nur aus einem Hauptteil besteht.

Kommando-Optionen können in beliebiger Reihenfolge erscheinen und mit einem wahlweisen Dateinamen vermischt sein. Nur eine Datei kann gewählt werden. Die Optionen lauten:

-b*type* Gibt an, welche Zeilen des Hauptteils einer logischen Seite numeriert werden sollen. Erkannte *types* und deren Bedeutung lauten:

a	Numerierung aller Zeilen
t	Nur Numerierung der Zeilen mit abdruckbarem Text
n	Keine Zeilennumerierung
p*string*	Nur Numerierung der Zeilen, die den in *string* angegebenen regulären Ausdruck enthalten.

-h*type* Wie bei **-b***type* mit Ausnahme des Kopfes. Standard *type* für den Kopf einer logischen Seite ist **n** (keine Zeilennumerierung). Standard *type* für den Hauptteil einer logischen Seite ist **t** (Textzeilennumerierung).

-f*type* Wie bei **-b***type* mit Ausnahme von Fußtext. Standard für Fußtext einer logischen Seite ist **n** (keine Zeilennumerierung).

-v*start#* *start#* ist der Anfangswert, der zur Numerierung der Zeilen auf der logischen Seite verwendet wird. Standard ist **1**.

-i*incr* *incr* ist das Inkrement, das bei der Numerierung der Zeilen auf der logischen Seite verwendet wird. Standard ist **1**.

-p Numerierung nicht bei logischem Seitenwechsel neu beginnen.

-l*num* *num* ist die Anzahl der Leerzeilen, die als eine Zeile angesehen werden. Zum Beispiel wird bei **-l2** die zweite von aufeinanderfolgenden Leerzeilen numeriert (wenn die entsprechende **-ha**, **-ba** und/oder **-fa** Option gesetzt ist). Standard ist **1**.

-s*sep* *sep* ist das (die) Zeichen, das (die) bei der Trennung der Zeilennummer und der entsprechenden Textzeile verwendet wird (werden). Standard *sep* ist ein Tabulator.

-w*width* *width* ist die Anzahl der Zeichen, die für die Zeilennummer verwendet werden. Standard *width* ist **6**.

-n*format* *format* ist das Zeilennumerierungsformat. Erkannte Werte sind: **ln**, linksbündig, führende Nullen werden unterdrückt; **rn**, rechtsbündig, führende Nullen werden unterdrückt; **rz**, rechtsbündig, führende Nullen werden beibehalten. Standard *format* ist **rn** (rechtsbündig).

-d*xx* Die Begrenzungszeichen, die den Anfang eines logischen Seitenabschnitts angeben, können von Standardzeichen (\:) auf zwei vom Benutzer angegebene Zeichen geändert werden. Wenn nur ein Zeichen eingegeben wird, bleibt das zweite Zeichen das Standardzeichen (:). Zwischen dem **-d** und den Begrenzungszeichen darf kein Leerzeichen erscheinen. Ein Backslash wird durch zwei Backslashes eingegeben.

BEISPIEL

Das Kommando:

```
nl -v10 -i10 -d!+ file1
```

numeriert file1 bei Zeilennummer 10 mit einem Inkrement von zehn. Die logischen Seitenbegrenzer sind ! +.

SIEHE AUCH:

pr(1).

BEZEICHNUNG

nohup – ein Kommando ausführen, unabhängig von Hangup- und Unterbrechungs-Signalen

ÜBERSICHT

nohup command [arguments]

BESCHREIBUNG

nohup führt *command* aus, wobei Hangup- und Unterbrechungs-Signale ignoriert werden. Wenn die Ausgabe nicht vom Benutzer umgeleitet ist, werden Standardausgabe und Standardfehlerausgabe an **nohup.out** geschickt. Wenn **nohup.out** nicht im aktuellen Verzeichnis beschreibbar ist, wird die Ausgabe an **$HOME/nohup.out** umgelenkt.

BEISPIEL

Der Einsatz von *nohup* wird bei Pipes oder Kommando-Listen empfohlen. Pipes und Kommando-Listen müssen hierfür in eine Datei geschrieben werden, also als Shell-Prozedur. Folgender Aufruf ist dann richtig:

```
nohup sh file
```

wobei *nohup* sich auf den Gesamtinhalt von *file* bezieht. Wenn die Shell-Prozedur *file* häufig ausgeführt werden muß, kann die Angabe von *sh* umgangen werden, indem *file* Ausführungserlaubnis erteilt wird. Die Hinzufügung eines Und-Zeichens bewirkt, daß der Inhalt von *file* im Hintergrund läuft und Unterbrechungen ignoriert werden (vgl. *sh*(1)):

```
nohup file &
```

Folgendes Beispiel zeigt an, wie der Inhalt von *file* aussehen könnte:

```
sort ofile > nfile
```

SIEHE AUCH:

chmod(1), nice(1), sh(1),
signal(2) im *Programmer's Reference Manual.*

ACHTUNG!

Im Falle des folgenden Kommandos

```
nohup command1; command2
```

gilt *nohup* nur für command1. Das Kommando

```
nohup (command1; command2)
```

ist syntaktisch falsch.

BEZEICHNUNG

od – Oktal-Dump

ÜBERSICHT

od [**–bcdosx**] [file] [[**+**]offset[**.**] [**b**]]

BESCHREIBUNG

od erstellt einen Auszug von *file* in einem oder mehreren Formaten, die mit dem ersten Argument festgelegt werden. Wenn das erste Argument fehlt, ist **–o** standard. Die Bedeutung der Format-Optionen lautet:

–b Oktale Interpretation von Bytes.

–c Interpretation von Bytes in ASCII. Gewisse nicht darstellbare Zeichen erscheinen in der Ersatzdarstellung von C: Null=**\0**, Rückschrittzeichen = **\b**, Formularvorschub = **\f**, New-Line-Zeichen = **\n**, Wagenrücklauf = **\r**, Tabulator = **\t**: andere erscheinen als Oktalzahl mit 3 Ziffern.

–d Dezimale, vorzeichenlose Interpretation von Worten.

–o Oktale Interpretation von Worten.

–s 16-Bit-Worte als Dezimalzahl mit Vorzeichen interpretieren.

–x Hexadezimale Interpretation von Worten.

Das Argument *file* gibt an, von welcher Datei ein Auszug erstellt werden soll. Wenn kein Dateiargument angegeben ist, wird die Standardeingabe verwendet.

Das Offset-Argument gibt den Versatz in der Datei an, von wo aus ein Auszug erstellt werden soll. Dieses Argument wird normalerweise als oktale Bytes interpretiert. Wenn **.** angehängt ist, wird das Offset-Argument dezimal interpretiert. Wenn **b** angehängt ist, wird das Offset-Argument in Blöcken von je 512 Bytes interpretiert. Wenn das Dateiargument ausgelassen ist, muß dem Offset-Argument **+** vorausgehen.

Der Auszug wird bis zum Dateiende erstellt.

BEZEICHNUNG

pack, pcat, unpack – verdichten und Erweitern von Dateien

ÜBERSICHT

pack [–] [**-f**] name ...

pcat name ...

unpack name ...

BESCHREIBUNG

pack versucht, die angegebenen Dateien in einer verdichteten Form zu speichern. Wenn möglich (und nützlich) wird jede Eingabedatei *name* durch eine verdichtete Datei *name*.z ersetzt, wobei diese die gleichen Zugriffsmodi, Zugriff- und Modifikationszeilen sowie Eigentümer wie *name* hat. Die Option **-f** erzwingt die Verdichtung von *name*. Dies ist praktisch beim Verdichten eines gesamten Verzeichnisses, selbst wenn einige der Dateien nicht verdichtet werden. Wenn *pack* erfolgreich ist, wird *name* entfernt. Verdichtete Dateien können auf ihr Originalformat umgestellt werden, indem das Kommando *unpack* oder *pcat* eingegeben wird.

pack verwendet Huffman-Code (Minimum an Redundanz) auf einer Byte-für-Byte-Basis. Wenn das – Argument verwendet wird, wird ein interner Schalter gesetzt, der bewirkt, daß auf der Standardausgabe ausgegeben wird wie oft ein Byte verwendet wird sowie seine relative Häufigkeit und der Code für das Byte. Zusätzliches Auftreten von – anstelle von *name* bewirkt ein Setzen und Rücksetzen des internen Schalters.

Die erreichte Verdichtung hängt von der Eingabedateigröße und der Verteilung der Zeichenfrequenz ab. Da ein Decodier-Baum den ersten Teil jeder .z Datei bildet, ist es gewöhnlich nicht sinnvoll, Dateien, die kleiner als drei Blöcke sind, zu verdichten, es sei denn die Verteilung der Zeichenfrequenz ist sehr verzerrt, was bei Druckergrafiken oder Bildern eintreten kann.

Normalerweise werden Textdateien auf 60-75% ihrer originalen Größe verringert. Lademodule, die einen größeren Zeichensatz verwenden und eine einheitlichere Verteilung der Zeichen haben, zeigen eine geringe Verdichtung an, wobei die verdichtete Version ca. 90% der originalen Größe ist.

pack gibt einen Wert zurück, der die Anzahl der Dateien ist, die nicht verdichtet wurden.

Keine Verdichtung tritt ein, wenn:

die Datei schon verdichtet ist;
der Dateiname mehr als 12 Zeichen hat;
die Datei Verweise hat;

die Datei ein Verzeichnis ist;
die Datei nicht geöffnet werden kann;
keine Plattenspeicherblöcke bei der Verdichtung gesichert werden;
eine Datei *name*.z schon vorhanden ist;
die .z Datei nicht erstellt werden kann;
ein Ein-/Ausgabe-Fehler bei Verarbeitung eintrat.

Der letzte Teil des Dateinamens darf nicht mehr als 12 Zeichen enthalten, damit Platz für die angehängte .z Erweiterung vorhanden ist. Verzeichnisse können nicht verdichtet werden.

pcat führt für verdichtete Dateien das aus, was *cat*(1) für Normaldateien macht, außer daß *pcat* nicht als Filter verwendet werden kann. Die angegebenen Dateien werden erweitert und auf die Standardausgabe geschrieben. Sie können eine verdichtete Datei *name*.z auflisten mit:

pcat name.z

oder nur:

pcat name.

Zur Erstellung einer erweiterten Kopie *nnn* von einer verdichteten Datei *name*.z (ohne *name*.z zu zerstören) verwenden Sie das Kommando:

pcat name > nnn

pcat gibt die Anzahl Dateien zurück, die nicht erweitert werden konnten. *pcat* könnte versagen, wenn:

der Dateiname (.z ausgeschlossen) mehr als 12 Zeichen hat;
die Datei nicht geöffnet werden kann;
die Datei nicht Ausgabe von *pack* ist.

unpack erweitert Dateien, die von *pack* erstellt wurden. Bei jeder Datei *name* die im Kommando angegeben wird, wird eine Datei *name*.z (oder nur *name,* wenn *name* mit .z endet) gesucht. Wenn diese Datei eine verdichtete Datei ist, wird sie durch ihre erweiterte Version ersetzt. Die neue Datei hat kein .z in ihrem Namen und besitzt die gleichen Zugriffsmodi, Zugriffs- und Modifikationsdaten und Eigentümer wie die verdichtete Datei.

Unpack gibt einen Wert zurück, der die Anzahl der Dateien angibt, die nicht erweitert werden konnten. Fehler können aus den gleichen Gründen wie bei *pcat* und auf Grund der folgenden Tatsachen eintreten:

eine Datei mit dem "erweiterten" Namen ist schon vorhanden;
die erweiterte Datei kann nicht erstellt werden.

SIEHE AUCH

cat(1).

BEZEICHNUNG

passwd – das Login-Paßwort ändern

ÜBERSICHT

passwd [name]

BESCHREIBUNG

Dieses Kommando ändert oder installiert das zu dem Login *name* entsprechende Paßwort.

Gewöhnliche Benutzer können nur das Paßwort ändern, das ihrem Login *name* entspricht.

passwd fordert gewöhnliche Benutzer auf, das alte Paßwort einzugeben. Dann wird zweimal das neue Paßwort angefordert. Bei der ersten Eingabe des neuen Paßworts prüft *passwd,* ob das alte Paßwort "alt" genug ist. "Veralterung" des Paßworts ist der Zeitraum (gewöhnlich eine bestimmte Anzahl von Tagen) zwischen Paßwort-Änderungen. Wenn die "Veralterung" unzureichend ist, wird das neue Paßwort abgelehnt und *passwd* wird beendet; vgl. *passwd*(4).

Wenn die "Veralterung" ausreichend ist, wird überprüft, ob das neue Paßwort korrekt ist. Bei der zweiten Eingabe des neuen Paßworts werden die beiden Kopien des neuen Paßworts verglichen. Wenn die beiden Kopien nicht identisch sind, wird die Eingabe-Aufforderung für ein neues Paßwort mindestens zweimal wiederholt.

Paßwörter müssen entsprechend den folgenden Bedingungen konstruiert sein:

Jedes Paßwort muß mindestens sechs Zeichen haben. Nur die ersten acht Zeichen sind signifikant.

Jedes Paßwort muß mindestens zwei alphabetische Zeichen und mindestens ein numerisches oder Sonderzeichen enthalten. In diesem Fall bedeutet "alphabetisch" Groß- und Kleinbuchstaben.

Jedes Paßwort muß sich vom Login *name* des Benutzers unterscheiden bzw. darf nicht der Login *name* in umgekehrter Reihenfolge oder in zirkulärer Schreibweise, d.h. den Anfangsbuchstaben ans Ende stellen, konstruiert sein. Groß-/Kleinschreibung wird bei Vergleichen gleichwertig behandelt.

Neue Paßwörter müssen sich von alten durch mindestens drei Zeichen unterscheiden. Groß-/Kleinschreibung wird bei Vergleichen gleichwertig behandelt.

Der Benutzer mit der effektiven Benutzernummer Null ist der Systemverwalter; vgl. *id*(1) und *su*(1). Systemverwalter können jedes Paßwort ändern; deshalb wird der Systemverwalter nicht zur Eingabe des alten Paßworts von *passwd* aufgefordert. Systemverwalter müssen sich nicht an die Paßwort-Veralterung und Paßwort-Konstruktion halten. Durch Eingabe eines Carrige-Return nach der Eingabe-Aufforderung für ein neues Paßwort, kann ein Systemverwalter ein Paßwort löschen.

DATEIEN

/etc/passwd

SIEHE AUCH:

login(1).
crypt(3C), passwd(4) im *Programmer's Reference Manual.*
id(1M), su(1M) im *Administrator's Reference Manual.*

BEZEICHNUNG

paste – gleiche Zeilen mehrerer Dateien oder aufeinanderfolge Zeilen einer Datei mischen

ÜBERSICHT

paste file1 file2. ..
paste **–d** list file1 file2. ..
paste **–s** [**–d**list] file1 file2. ..

BESCHREIBUNG

Bei den beiden ersten Kommandoformen verkettet *paste* entsprechende Zeilen der angegebenen Eingabedateien *file1, file2,* usw. Jede Datei wird wie eine Spalte oder Spalten einer Tabelle behandelt, und sie werden horizontal zusammengefügt (paralleles Mischen). Man kann dieses Kommando als das Gegenstück von *cat(1)* ansehen, daß senkrecht verkettet, d. h. eine Datei nach der anderen. Bei der letzten, oben erwähnten Kommandoform ersetzt *paste* die Funktion eines älteren Kommandos gleichen Namens durch Verbindung aufeinanderfolgender Zeilen der Eingabedatei (serielles Mischen). Bei allen erwähnten Kommandoformen werden Zeilen mit dem *tab* Zeichen oder mit Zeichen, die in einer wahlweisen Liste *list* angegeben sind, zusammengefügt. Ausgabe erfolgt auf die Standardausgabe, es kann also als Anfang einer Pipe oder als Filter verwendet werden, wenn es – anstelle eines Dateinamens – benutzt wird.

Die Optionen haben folgende Bedeutung:

–d Ohne diese Option werden die New-Line-Zeichen bei allen Dateien, außer der letzten (oder die letzte Zeile bei der Option **–s**) durch ein *tab* Zeichen ersetzt. Diese Option gestattet Ersetzung von *tab* Zeichen durch ein oder mehrere alternative Zeichen (siehe unten).

list Ein oder mehrere Zeichen direkt nach **–d** ersetzen das Standardzeichen *tab* als Zeilenverkettungszeichen. Die Liste wird zirkulär verwendet, d. h. nach Abarbeitung wird sie erneut benutzt. Bei Parallelmischen (d. h. keine **–s** Option) werden die Zeilen von der letzten Datei immer mit einem New-Line-Zeichen beendet, und nicht durch *list.* Die Liste kann folgende Sonder-Escape-Folgen enthalten: **\n** (New-Line-Zeichen), **\t** (Tabulator), **** (Backslash) und **\0** (leere Zeichenfolge, kein Nullzeichen). Anführungszeichen können erforderlich sein, wenn Zeichen in der Shell eine besondere Bedeutung haben (z. B. für einen Backslash ist *–d*"\\\\" zu verwenden).

–s Mischt aufeinanderfolgende Zeilen anstatt einer Zeile aus jeder Eingabedatei. Zur Verkettung wird *tab* benutzt, es sei denn, ein *list* ist bei der Option **–d** angegeben. Unabhängig von *list* muß das allerletzte Zeichen der Datei ein New-Line-Zeichen sein.

- Kann zum Lesen einer Zeile von der Standardeingabe anstatt eines beliebigen Dateinamens verwendet werden (keine Eingabe-Aufforderung).

BEISPIELE

ls | paste -d" " -	Verzeichnis in einer Spalte auflisten
ls | paste - - - -	Verzeichnis in vier Spalten auflisten
paste -s -d"\t\n" file	Zeilenpaare zu Zeilen verbinden

SIEHE AUCH

cut(1), grep(1), pr(1).

DIAGNOSE

line too long	Ausgabezeilen sind auf 511 Zeichen beschränkt.
too many files	Mit Ausnahme von Option -s können nicht mehr als 12 Eingabedateien angegeben werden.

BEZEICHNUNG

pg – Datei-Ausgabefilter für CRT-Terminals

ÜBERSICHT

pg [*-number*] [**-p** *string*] [**-cefns**] [*+linenumber*] [*+/pattern/*] [files...]

BESCHREIBUNG

Das Kommando *pg* ist ein Filter, der für CRT-Terminals die bildschirmweise Ausgabe von *files* ermöglicht. (Der Dateiname – und/oder NULL-Argumente zeigen an, daß *pg* von der Standardeingabe lesen muß). Nach jeder vollen Bildschirmseite folgt eine Eingabe-Aufforderung. Wenn der Benutzer einen Wagenrücklauf eingibt, wird eine weitere Seite angezeigt; andere Möglichkeiten werden nachstehend erläutert.

Dieses Kommando unterscheidet sich von anderen Seitenausgaben, da Sie zurückgehen und schon ausgegebene Seiten wieder ansehen können. Die Methode wird weiter unten genauer erklärt.

Zur Bestimmung von Terminal-Attribute fragt *pg* die Datendatei *terminfo*(4) nach dem durch die Umgebungsvariable **TERM** angegebenen Typ des Terminals. Wenn **TERM** nicht definiert ist, wird der Typ **dumb** angenommen.

Die Optionen der Kommandozeile lauten:

-number
: Eine ganze Zahl, die die Größe (in Zeilen) des Fensters angibt, die *pg* statt der Voreinstellung verwenden soll. (Bei einem Bildschirm mit 24 Zeilen ist die Standard-Fenstergröße 23).

-p *string*
: Veranlaßt *pg*, *string* als Eingabe-Aufforderung zu verwenden. Wenn die Eingabe-Aufforderungsfolge ein "%d" enthält, wird das erste Auftreten von "%d" in der Eingabe-Aufforderung durch die aktuelle Seitennummer ersetzt, wenn das Eingabe-Aufforderungszeichen ausgegeben wird. Die Standardeingabeaufforderungsfolge ist ":".

-c Setzt den Cursor auf die Ausgangsposition zurück und löscht den Bildschirm vor Anzeige einer Seite. Diese Option wird ignoriert, wenn **clear_screen** nicht in der Datendatei *terminfo*(4) für diesen Bildschirmtyp definiert ist.

-e Verhindert, daß *pg* am Ende jeder Datei anhält.

-f Normalerweise trennt *pg* Zeilen, die länger als die Bildschirmbreite sind. Einige Zeichenfolgen im Text (z. B. Escape-Folgen zum Unterstreichen) rufen aber unerwünschte Ergebnisse hervor. Die Option *-f* verhindert das Trennen von Zeilen durch *pg*.

-n Normalerweise müssen Kommandos durch ein *<newline>* Zeichen abgeschlossen werden. Diese Option bewirkt ein automatisches Kommandoende, sobald ein Kommandobuchstabe eingegeben ist.

-s Veranlaßt *pg*, alle Nachrichten und Eingabe-Aufforderungen im Hervorhebe-Modus (gewöhnlich Invers-Modus) auszugeben.

+linenumber
Beginn bei *linenumber.*

+/pattern/
Beginnt an der ersten Zeile, die den regulären Ausdruck enthält.

Die Antworten, die eingegeben werden können, wenn *pg* pausiert, können in drei Kategorien aufgeteilt werden: zur Prüfung, Suche und zur Änderung der Umgebung.

Kommandos, die normalerweise weitere Suche veranlaßen, wird eine Adresse *address* vorausgestellt, d. h. eine wahlweise mit Vorzeichen versehene Zahl, die die Stelle angibt, von wo aus mehr Text angezeigt werden soll. Diese *address* wird je nach Kommando als Seite oder Zeile interpretiert. Eine mit Vorzeichen versehene *address* gibt eine Stelle relativ zu der aktuellen Seite oder Zeile an, eine vorzeichenlose *address* gibt eine Adresse relativ zum Anfang der Datei an. Jedes Kommando hat eine Standardadresse, die verwendet wird, wenn keine andere Adresse bereitgestellt ist.

Die Kommandos und ihre Standards lauten wie folgt:

(+1)*<newline>* oder *<blank>*
Eine Seite wird angezeigt. Die Adresse wird in Seiten angegeben.

(+1)**l** Bei Angabe einer relativen Adresse simuliert *pg* Vorwärts- und Rückwärtsrollen des Bildschirms um die angegebene Zeilenanzahl. Mit einer absoluten Adresse gibt dieses Kommando eine volle Bildschirmseite aus, die an der angegebenen Zeile beginnt

(+1) **d** oder **^D**
Simuliert Vorwärts- und Rückwärtsrollen der halben Bildschirmseite.

Die folgenden Kommandos haben keine *address.*

. oder **^L**
Bei Eingabe eines einzelnen Punkts wird die aktuelle Textseite wieder angezeigt.

$ Zeigt den letzten Teil in der Datei an, der auf einen Bildschirm paßt. Wenn die Eingabe eine Pipe ist, ist diese Option mit Vorsicht zu verwenden.

Die folgenden Kommandos stehen zum Suchen von Textmustern im Text zur Verfügung. Die regulären, in *ed*(1) beschriebenen Ausdrücke können verwendet werden. Sie müssen immer mit einem *<newline>* abgeschlossen werden, selbst wenn die Option *−n* angegeben ist.

i/*pattern*/
: Sucht vorwärts nach dem *i*-ten Auftreten (Standard *i* =1) von *pattern*. Die Suche beginnt direkt nach der aktuellen Seite und geht bis zum Ende der aktuellen Datei (nicht zirkulär).

i^*pattern*^
i?*pattern*?
: Sucht rückwärts nach dem *i*-ten Auftreten (Standard *i* =1) von *pattern*. Die Suche beginnt direkt vor der aktuellen Seite und geht zum Anfang der aktuellen Datei (nicht zirkulär). Die ^ Schreibweise ist nützlich bei Adds 100 Terminals, die das ? nicht richtig bearbeiten.

Nach der Suche zeigt *pg* normalerweise die gefundene Zeile am Bildschirmanfang an. Wenn man **m** oder **b** zum Suchkommando hinzufügt, wird von nun an die gefundene Zeile in der Mitte oder unten am Fenster angezeigt. Der Zusatz **t** kann zum Wiederherstellen der ursprünglichen Situation verwendet werden.

Der Benutzer von *pg* kann die Umgebung mit den folgenden Kommandos modifizieren:

*i***n**
: *i*te nächste Datei in der Kommandozeile untersuchen. Das *i* ist eine vorzeichenlose Zahl, Standardwert ist 1.

*i***p**
: *i*te vorhergehende Datei in der Kommandozeile untersuchen. *i* ist eine vorzeichenlose Zahl, Standardwert ist 1.

*i***w**
: Zeigt ein anderes Textfenster an. Wenn *i* vorhanden ist, wird die Fenstergröße auf *i* gesetzt.

s*filename*
: Die Eingabe in der angegebenen Datei speichern. Nur die aktuelle, momentan untersuchte Datei wird gesichert. Das Leerzeichen zwischen **s** und *filename* ist wahlweise. Dieses Kommando muß immer mit *<newline>* abgeschlossen werden, selbst wenn die Option *−n* angegeben ist.

h
: Help-Kommando. Angabe einer abgekürzten Zusammenfassung verfügbarer Kommandos.

q oder **Q**
: *pg* verlassen.

! *command*

Command wird zu der Shell geleitet, deren Namen aus der Umgebungsvariablen **SHELL** genommen wird. Wenn diese nicht verfügbar ist, wird die Standard-Shell verwendet. Dieses Kommando muß immer mit *<newline>* abgeschlossen werden, selbst wenn die Option *-n* angegeben ist.

Bei Ausgabe auf den Bildschirm kann der Benutzer jederzeit die Abbruchtaste (normalerweise Control-\) oder die Unterbrechungstaste (Break) betätigen. *pg* stoppt dann die Ausgabe und zeigt das Eingabe-Aufforderungszeichen an. Der Benutzer kann dann eines der obigen Kommandos ganz normal eingeben. Leider geht hierbei Ausgabe verloren was auf die Tatsache zurückzuführen ist, daß Zeichen in der Ausgabe-Warteschlange des Terminals entfernt werden, wenn das Abbruchsignal eintritt.

Wenn die Standardausgabe kein Terminal ist, verhält sich *pg* wie *cat*(1), außer daß ein Kopf vor jeder Datei ausgegeben wird (wenn es sich um mehr als eine Datei handelt).

BEISPIEL

Ein Beispiel für den Einsatz von *pg* beim Lesen von Systemnachrichten wäre

```
news | pg -p "(page %d):"
```

HINWEISE

Beim Warten auf Bildschirmeingabe reagiert *pg* auf **BREAK**, **DEL** und ^ mit Beendigung der Ausführung. Zwischen Eingabe-Aufforderungen unterbrechen diese Signale jedoch die aktuelle Aktion von *pg* und setzen den Benutzer in Eingabaufforderungs-Modus. Diese Signale sollten vorsichtig angewendet werden, wenn Eingabe von einer Pipe gelesen wird, da eine Unterbrechung wahrscheinlich die anderen Kommandos in der Pipe beendet.

Benutzer von Berkeley's *more* werden feststellen, daß die Kommandos z und f zur Verfügung stehen und daß der Abschluß /, ^ oder ? bei den Suchkommandos weggelassen werden kann.

DATEIEN

/usr/lib/terminfo/?/ *	Datendatei mit Daten für Terminals
/tmp/pg *	temporäre Datei, wenn Eingabe aus einer Pipe ist

SIEHE AUCH:

ed(1), grep(1).
Programmer's Reference Manual.

FEHLER

Wenn Bildschirmtabulatoren nicht alle acht Positionen gesetzt sind, können unerwünschte Ergebnisse eintreten.

Bei Verwendung von *pg* als Filter mit einem anderen Kommando, das die Ein-/Ausgabeoptionen des Terminals ändert, können die Einstellungen nicht korrekt wiederhergestellt werden.

BEZEICHNUNG

pr – Dateien ausgeben

ÜBERSICHT

pr [[**-column**] [**-w**width] [**-a**]] [**-e**ck] [**-i**ck] [**-drtfp**] [+page] [**-n**ck] [**-o**offset] [**-l**length] [**-s**separator] [**-h**header] [file ...]

pr [[**-m**] [**-w**width]] [**-e**ck] [**-i**ck] [**-drtfp**] [+page] [**-n**ck] [**-o**offset] [**-l**length] [**-s**separator] [**-h**header] file1 file2 ...

BESCHREIBUNG

pr wird zur Formatierung und Ausgabe des Inhalts einer Datei verwendet. Wenn *file* **-** ist oder wenn keine Dateien angegeben sind, nimmt *pr* die Standardeingabe an. *pr* gibt die angegebenen Dateien auf der Standardausgabe aus.

Standardmäßig wird die Ausgabe in Seiten eingeteilt, die als Kopfzeile die Seitennummer, das Datum und die Zeitangabe, zu der die Datei zuletzt geändert wurde und den Dateinamen enthalten. Die Seitenlänge ist 66 Zeilen, die 10 Zeilen für Kopfzeile und Fußnote einschließen. Die Kopfzeile besteht aus zwei Leerzeilen, einer Textzeile (kann mit der Option **-h** geändert werden) und zwei Leerzeilen; die Fußnote sind fünf Leerzeilen. Bei einspaltiger Ausgabe muß die Zeilenbreite nicht gesetzt werden und ist unbegrenzt. Bei mehrspaltiger Ausgabe kann die Zeilenbreite gesetzt werden; standardmäßig sind 72 Spalten vorgesehen. Fehlermeldungen (Optionen die nicht funktionieren) werden ans Ende der Standardausgabe gestellt, die mit einem Terminal verbunden ist, und nicht mitten in die normale Ausgabe. Seiten werden durch eine Reihe von Zeilenvorschüben statt der Formularvorschubszeichen getrennt.

Standardmäßig sind Spalten gleich breit und durch mindestens ein Leerzeichen voneinander getrennt. Nicht passende Zeilen werden getrennt. Falls die Option **-s** verwendet wird, werden die Zeilen nicht getrennt und Spalten werden durch das Trennzeichen *separator* getrennt.

Zur Erstellung einer mehrspaltigen Ausgabe muß *-column* oder die Option **-m** benutzt werden. **-a** sollte nur mit *-column* und nicht mit **-m** verwendet werden.

Die Optionen der Kommandozeile lauten:

+page

Anzeige beginnt bei der mit *page* numerierten Seite (Standard ist 1).

-column

column Spalten ausgeben (Standard ist 1). Die Ausgabe sieht so aus wie bei eingeschalteten Optionen **-e** und **-i** für mehrspaltige Ausgabe. Nicht mit **-m** zu verwenden.

-a Mehrspaltige Ausgabe quer über die Seite, und zwar eine Zeile per Spalte. *columns* muß größer als eins sein. Wenn eine Zeile zu lang für eine Spalte ist, wird sie getrennt.

-m Mischt und gibt alle Dateien gleichzeitig aus, eine Datei pro Spalte. Maximal können acht Dateien angegeben werden. Wenn eine Zeile zu lang für eine Spalte ist, wird sie getrennt. Kann nicht mit *-column* verwendet werden.

-d Doppelter Freiraum bei Ausgabe. Bei doppeltem Freiraum der Ausgabe werden am Seitenanfang auftretende Leerzeilen gelöscht.

-e*ck* Eingabe-Tabulatorzeichen auf die Zeichenpositionen $k+1$, $2*k+1$, $3*k+1$, usw. einstellen. Wenn k oder 0 ausgelassen sind, wird als standardmäßige Tabulatoreinstellung jede achte Position angenommen. Tabulatorzeichen der Eingabe werden auf die entsprechende Anzahl von Leerstellen erweitert. Falls *c* (beliebiges Nichtziffern-Zeichen) angegeben ist, wird es als das Eingabe-Tabulatorzeichen behandelt (Standard für *c* ist das Tabulatorzeichen).

-i*ck* In der Eingabe werden soweit wie möglich Zwischenraumzeichen durch Einfügen von Tabulatorzeichen an den Zeichenpositionen $k+1$, $2*k+1$, $3*k+1$, usw. ersetzt. Falls k oder 0 ausgelassen sind, wird als standardmäßige Tabulatoreinstellung jede achte Position angenommen. Falls *c* (beliebiges Nichtziffern-Zeichen) angegeben ist, wird es als das Ausgabe-Tabulatorzeichen behandelt (Standard für *c* ist das Tabulatorzeichen).

-n*ck* Stellt k -stellige Zeilennumerierung bereit. (Standard für k ist 5). Die Zahl steht auf den ersten $k+1$ Zeichenpositionen jeder Spalte einer einspaltigen Ausgabe oder jeder Zeile einer Ausgabe **-m***c* (beliebiges Nichtziffern-Zeichen) angegeben ist, wird es zur Abtrennung beliebiger nachfolgender Zeichen an die Zeilennummer angehängt; Standard für *c* ist ein Tabulatorzeichen).

-w*width*
Stellt die Breite einer Zeile auf *width* Zeichenpostitionen ein (Standard ist 72). Dies ist nur bei mehrspaltiger Ausgabe wirksam (*-column* und **-m**). Es gibt keine Zeilenbegrenzung bei einspaltiger Ausgabe.

-o*offset*
Versetzt jede Zeile um *offset* Zeichenpositionen (Standard ist 0). Die Anzahl der Zeichenpositionen pro Zeile ist die Summe aus Zeilenbreite und *offset*.

-l*length*
Stellt die Länge einer Seite auf *length* (Standard ist 66) ein. **-l**0 wird auf **-l**66 zurückgesetzt. Wenn der Wert von *length* 10 oder kleiner ist, tritt **-t** in Kraft, da Kopfzeilen und Fußnoten unterdrückt werden. Standardmäßig enthält die Ausgabe einen 5zeiligen Kopf und eine 5zeilige Fußnote, es bleiben also 56 Zeilen für Text übrig. Wenn **-l***length* verwendet wird und *length* größer als 10 ist, stehen dem Benutzer *length*-10 Zeilen für Text zur Verfügung. Wenn *length* 10 oder kleiner ist, wird die Ausgabe von Kopf und Fußnoten unterdrückt, um mehr Platz für Text bereitzustellen.

-h *header* Ausgabe der Textzeile
header im Kopf anstelle des Dateinamens. **-h** wird ignoriert, wenn **-t** oder **-l***length* angegeben ist und der Wert von *length* 10 oder kleiner ist. (**-h** ist die einzige *pr* -Option, die eine Leerstelle zwischen Option und Argument erfordert.)

-p Pausiert vor Beginn jeder Seite, wenn auf einem Terminal ausgegeben wird; (*pr* gibt ein akustisches Signal aus und wartet auf Betätigung der CR-Taste.)

-f Ein einzelnes Formularvorschubzeichens bei neuen Seiten verwenden (Standard ist eine Folge von Zeilenvorschüben). Pausiert vor Beginn der ersten Seite, wenn die Standardausgabe einem Terminal zugeordnet ist.

-r Keine Fehlermeldungen bei Dateien ausgeben, die nicht geöffnet werden können.

-t Der fünfzeilige Kopf und die fünfzeilige Fußnote, die normalerweise bei jeder Seite angegeben sind, werden nicht ausgegeben. Ausgabe nach der letzten Zeile jeder Datei beenden ohne Auffüllen von Leerzeichen bis zum Seitenende. Die Option **-t** setzt die Option **-h** außer Kraft.

-s*separator*
Trennt Spalten mit einem einzelnen Zeichen *separator* anstelle der entsprechenden Anzahl von Leerzeichen (Standard für *separator* ist ein Tabulatorzeichen). Verhindert die Trennung von Zeilen bei mehrspaltiger Ausgabe, es sei denn, **-w** wird angegeben.

BEISPIELE

Ausgabe von **file1** und **file2** in zweizeiligem, dreispaltigem Format mit der Überschrift "file list":

```
pr -3dh "file list" file1 file2
```

Kopieren von **file1** auf **file2**, Erweiterung der Tabulatorzeichen bis Spalte 10, 19, 28, 37, ... :

```
pr -e9 -t <file1 >file2
```

file1 und **file2** gleichzeitig in einer zweispaltigen Liste ohne Kopf oder Fußnote ausgeben, wobei beide Spalten Zeilennummern haben:

```
pr -t -n file1 | pr -t -m -n file2 -
```

DATEIEN

/dev/tty* Falls die Standard-Ausgabe auf eine Gerätedatei **/dev/tty*** geschickt wird, werden andere Ausgaben, die ebenfalls an dieses Terminal gerichtet sind, unterbrochen, bis die Standard-Ausgabe beendet ist. Dadurch werden Fehlermeldungen aufgrund von Unterbrechungen während der Ausgabe verhindert.

SIEHE AUCH:

cat(1), pg(1). §

BEZEICHNUNG

ps – Prozeßstatus angeben

ÜBERSICHT

ps [options]

BESCHREIBUNG

ps gibt bestimmte Informationen über aktive Prozesse aus. Ohne *options* werden Informationen über Prozesse,die mit dem kontrollierenden Terminal im Zusammenhang stehen, ausgegeben. Die Ausgabe besteht aus einer kurzen Liste, die nur die Prozeßnummer, Terminalnamen, Gesamtausführungszeit und den Kommandonamen enthält. Andernfalls wird die Ausgabe der Daten durch Auswahl entsprechender *options* gesteuert.

options akzeptieren Namen oder Listen als Argumente. Argumente können entweder durch Kommata voneinander getrennt sein oder in Anführungszeichen stehen und durch Kommata oder Leerzeichen voneinander getrennt sein. Werte für *proclist* und *grplist* müssen numerisch sein.

Die *options* sind nachstehend in der Reihenfolge ihres Informationsumfangs und Wirkungsbereichs aufgeführt:

-e	Angabe der Daten von every (jedem) Prozeß, der gerade läuft.
-d	Angabe der Daten über alle Prozesse außer dem Führungsprozess der Prozeßgruppe.
-a	Angabe der Daten über alle am häufigsten abgefragten Prozesse: außer Führungsprozess der Prozeßgruppe und Prozesse, die nicht mit einem Terminal verbunden sind.
-f	Erzeugen einer **full** (kompletten) Liste. (Vgl. unten bzgl. Bedeutung der Spalten in einer kompletten Liste.)
-l	Erzeugen einer **long** (langen) Liste (vgl. unten).
-n *name*	Argument ist ein alternativer Systemname *name* anstelle von **/unix**.
-t *termlist*	Nur Prozeßdaten über Prozesse auflisten, die zum in *termlist* angegebenen Terminal gehören. Terminalkennungen können auf zwei verschiedene Weisen angegeben werden: der Gerätedateiname (z. B. **tty04**) oder, wenn der Gerätedateiname mit **tty** anfängt, nur die Ziffernkennung (z. B. **04**).
-p *proclist*	Nur Prozeßdaten über Prozesse auflisten, deren Prozeßnummern in *proclist* angegeben sind.
-u *uidlist*	Nur Prozeßdaten über Prozesse auflisten, deren Benutzernummer oder Login-Name in uidlist aufgeführt sind. In der Auflistung wird die numerische Benutzernummer

angegeben, es sei denn, Sie geben die Option **-f** an, die den Login-Namen ausgibt.

-g *grplist* Nur Prozeßdaten über Prozesse auflisten, deren Führungsprozeß-ID-Nummer(n) in *grplist* erscheint (erscheinen) (Ein Führungsprozeß ist ein Prozeß, dessen Prozeßnummer identisch mit seiner Prozeßgruppennummer ist. Eine Login-Shell ist ein bekanntes Beispiel eines Führungsprozesses.)

Bei der Option **-f** versucht *ps* den Kommandonamen und die Argumente zu bestimmen, die bei Prozeßerstellung angegeben wurden, indem das Kommando den Benutzerblock prüft. Wenn dies nicht gelingt, wird der Kommandoname ausgegeben, und zwar in eckigen Klammern, so wie er ohne die Option **-f** erschienen wäre.

Die Spaltenüberschriften und die Bedeutung der Spalten in einer *ps* Liste sind nachstehend angegeben; die Buchstaben **f** und **l** zeigen die Option an (**full** (komplett) oder **long** (lang)), die bewirken, daß die entsprechende Überschrift erscheint; **all** (immer) bedeutet, daß die Überschrift immer erscheint. Es ist zu beachten, daß diese beiden Optionen nur bestimmen, welche Daten für einen Prozeß bereitgestellt werden; sie bestimmen nicht, welche Prozesse aufgelistet werden.

F (l) Zum Prozeß gehörige (hexadezimale und zusätzliche) Anzeiger

Targon/31

- 00 Prozeß ist beendet: Eintrag in Prozeßtabelle jetzt verfügbar.
- 01 Ein Systemprozeß: immer im Hauptspeicher.
- 02 Vater verfolgt (trace) Prozeßablauf.
- 04 Signal, das der Vater während der Ablaufverfolgung geschickt hat, hat den Prozess gestoppt. Der Vater wartet [*ptrace*(2)].
- 08 Prozeß ist momentan im Hauptspeicher.
- 10 Prozeß momentan im Hauptspeicher: gesperrt bis ein Ereignis beendet ist.

VAX-PROZESSOR

- 00 Prozeß ist beendet: Eintrag in Prozeßtabelle jetzt verfügbar.
- 01 Prozeß momentan im Hauptspeicher.
- 02 Ein Systemprozeß: immer im Hauptspeicher.

04 Prozeß ist momentan im Hauptspeicher: gesperrt bis ein Ereignis beendet ist.
08 Sollte nicht bei diesem System eintreten.
10 Vater verfolgt Prozeßablauf.
20 Signal, das der Vater während der Ablaufverfolgung geschickt hat, hat den Prozess gestoppt. Der Vater wartet [*ptrace*(2)].

S (l) Der Zustand des Prozesses:
S Pausieren: Prozeß wartet auf die Beendigung eines Ereignisses.
R Bereit: Prozeß ist in Bereit-Warteschlange.
I Untätig: Prozeß wird gerade erzeugt.
Z Zombie-Zustand: Prozeß beendet und Vater wartet nicht.
T Ablauf wird verfolgt: Prozeß durch ein Signal gestoppt, weil Vater seinen Ablauf verfolgt.
X SXBRK -Zustand: Prozeß wartet auf mehr Platz im Hauptspeicher.

UID (f,l) Die Benutzernummer des Prozeßeigentümers (der Login-Name wird bei der Option **-f** ausgegeben).

PID (all) Die Prozeßnummer des Prozesses (dieser Wert ist notwendig, um einen Prozeß abzubrechen).

PPID (f,l) Die Prozeßnummer des Vaterprozesses.

C (f,l) Prozessor-Zeit für Scheduling.

PRI (l) Die Priorität des Prozesses (höhere Zahlen bedeuten niedrigere Priorität).

NI (l) Prioritätswert (nice-Wert), der in Prioritäts-Berechnung verwendet wird.

ADDR (l) Die Speicherplatzadresse des Prozesses.

SZ (l) Die Größe (Seiten oder Einheiten) des austauschbaren Prozeßbereichs im Arbeitsspeicher.

WCHAN (l) Die Adresse eines Ereignisses, auf das der Prozeß wartet oder im SXBRK -Status (der Prozeß läuft, wenn CHAN nicht gesetzt).

STIME (f) Die Anfangszeit des Prozesses, angegeben in Stunden, Minuten und Sekunden. (Ein Prozeß, der vor mehr als vierundzwanzig Stunden vor der Ausführung der *ps* -Anfrage begann, wird in Monaten und Tagen angegeben.)

TTY (all) Das kontrollierende Terminal für den Prozeß (die Nachricht **?** wird ausgegeben, wenn es kein kontrollierendes Terminal gibt).

TIME (all) Die bisher aufgelaufene Ausführungszeit für den Prozeß.

COMMAND (all) Der Kommandoname (der volle Kommandoname und seine Argumente werden bei der Option **-f** ausgegeben).

Ein Prozeß, der beendet ist und einen Vater hat, aber auf den der Vater noch nicht gewartet hat, wird mit **<defunct>** bezeichnet.

DATEIEN

/dev
/dev/sxt/*
/dev/tty*
/dev/xt/* Dateien mit Suchvorrichtung für Terminalnamen ("tty")
/dev/kmem virtueller Speicherplatz des Systemkerns
/dev/swap das Standard-Swapgerät
/dev/mem Speicherplatz
/etc/passwd Paßwort-Datei mit Benutzernummern
/etc/ps_daten interne Datenstruktur
/unix Systemnamensliste

SIEHE AUCH:

kill(1), nice(1).
getty(1m) im *Administrator's Reference Manual* .

ACHTUNG!

Beim Ablauf von *ps* können Veränderungen eintreten; die von *ps* angezeigte Ausgabe gilt nur für einen Bruchteil einer Sekunde und kann schon ungenau sein, wenn Sie sie sehen. Einige Daten, die bei "defunct" Prozessen angezeigt werden, sind ungültig.

Wenn keine *termlist*, *proclist*, *uidlist* oder *grplist* angegeben ist, prüft *ps* *stdin*, *stdout* und *stderr* in dieser Reihenfolge und sucht nach dem kontrollierenden Terminal und berichtet über Prozesse, die mit diesem Terminal verbunden sind. Wenn nun in dieser Situation *stdin*, *stdout* und *stderr* umgelenkt werden, findet *ps* kein kontrollierendes Terminal und kein Bericht wird erstellt.

Bei einem schwer belasteten System kann *ps* einen *lseek(2)* -Fehler und Abbruch melden. *ps* sucht eventuell nach einer ungültigen Adresse von prozeßbestimmten Daten im System: nach Erhalt der Adresse eines Prozesses der prozeßbestimmten Daten im System ist es eventuell für *ps* nicht möglich, diese Adresse vor Verlassen des Prozesses zu finden und die Adresse wird ungültig.
ps -ef kann eventuell den aktuellen Beginn einer Terminal-Login-Sitzung nicht melden, sondern gibt einen früheren Zeitpunkt an, als ein 'getty' zuletzt auf der Bildschirmleitung neu erzeugt wurde.

BEZEICHNUNG

pwd – aktueller Verzeichnisname

ÜBERSICHT

pwd

BESCHREIBUNG

pwd gibt den Pfadnamen des (aktuellen) Arbeitsverzeichnisses aus.

SIEHE AUCH

cd(1).

DIAGNOSE

"Cannot open .." und "Read error in .." zeigen mögliche Dateisystemfehler an und sollten dem UNIX Systemverwalter gemeldet werden.

BEZEICHNUNG

rcp – kopieren entfernter Dateien

ÜBERSICHT

rcp [**-p**] Datei1 Datei2
rcp [**-p**] [**-r**] Datei ... Verzeichnis

BESCHREIBUNG

Rcp kopiert Dateien zwischen Rechnern. Jedes der Argumente *Datei* und *Verzeichnis* kann jeweils einen entfernten Dateinamen in der Form "ehost:Pfad"oder einen lokalen Dateinamen (ohne Doppelpunkte (':') bzw. mit einem Schrägstrich ('/') vor dem Doppelpunkt).

Wurde die Option **-r** angegeben und gehören zu den Quelldateien auch Verzeichnisse, kopiert *rcp* jeden Teilbaum, der zu diesem Namen gehört; in diesem Fall muß das Ziel ebenfalls ein Verzeichnis sein.

Besteht *Datei2* bereits, bleiben deren Zugriffsreche und Besitzer unverändert; ansonsten werden die Zugriffsrechte der mit *umask*(2) geänderten Quelldatei auf dem Ziel-Host verwendet. Die Option **-p** bewirkt, daß *rcp* versucht, in den Kopien die Zeit der Änderungen und die Zugriffsrechte der Quelldateien zu erhalten (kopieren) und *umask* zu ignorieren.

Ist *Pfad* kein vollständiger Pfadname, wird dieses Argument auf das Login-Verzeichnis auf *ehost* bezogen interpretiert. Ein *Pfad* auf einem entfernten Host-System kann in Anführungszeichen stehen (\, " oder ´) so daß die Metazeichen entfernt interpretiert werden.

Rcp fordert nicht zur Eingabe eines Paßwörtern auf; der aktuelle lokale Benutzername muß in *ehost* vorhanden sein und die Ausführung entfernter Kommandos über *rsh* (1C) zulassen.

Rcp verwendet auch Fremdkopien, wenn auf dem aktuellen Rechner weder Quell- noch Zieldateien vorhanden sind. Host-Namen können auch die Form "ename@ehost" haben, wenn anstelle des aktuellen Benutzernamens auf dem entfernten Host *ename* verwendet werden soll. Der Name des Ziel-Hosts kann auch die Form "ehost.ename" annehmen, um Ziel-Rechner zu unterstützen, auf denen 4.2BSD-Versionen von *rcp* laufen.

SIEHE AUCH

cp(1), ftp(1C), rsh(1C), rlogin(1C)

FEHLERQUELLEN

Findet nicht alle Stellen, an denen als Ziel eines Kopiervorgangs eine Datei steht, wo nur ein Verzeichnis zulässig wäre.
Läßt sich durch die Ausgaben zu Kommandos in .loginl, .profile- oder .cshrc-Dateien auf dem entfernten Host irritieren.

BEZEICHNUNG

rlogin – Netzwerk–Login

ÜBERSICHT

rlogin [**-ec**] [**-8**] [**-L**] [**-l** *username*] *rhost*
rhost [**-ec**] [**-8**] [**-L**] [**-l** *username*]

BESCHREIBUNG

Rlogin verbindet Ihr Terminal auf dem aktuellen lokalen Host *lhost* mit dem entfernten Host *rhost* (remote host).

Auf jedem Host ist eine Datei */etc/hosts.equiv* vorhanden mit einer Liste entfernter Hosts *rhost*, mit denen er Benutzernamen teilt. (Hostnamen müssen Standard-Hostnamen gemäß *rsh*(1C) sein.) Wenn Sie sich mit *rlogin* unter dem lokalen Benutzernamen auf einem äquivalenten Host anmelden, müssen Sie kein Paßwort eingeben. Außerdem kann jeder Benutzer eine eigene Äquivalenzliste in der Datei .rhosts in seinem Login-Verzeichnis führen. Jede Zeile dieser Datei sollte den Namen eines entfernten Hosts *rhost* enthalten, gefolgt von einer Leerstelle und einem Benutzernamen *username*. Dadurch können weitere Fälle definiert werden, in denen Anmeldungen ohne Paßwort-Eingabe erlaubt sind. Wenn der lokale Benutzer nicht äquivalent zu dem entfernten Benutzer ist, wird er zur Eingabe des Benutzernamens und des Paßworts in dem entfernten Rechner aufgefordert wie in *login*(1). Zur Vermeidung bestimmter Sicherheitsprobleme muß Root oder der Benutzer, der im entfernten Host das Login-Verzeichnis besitzt, Besitzer der Datei .rhosts sein. Die **Ausnahme** bildet hier das Targon-/31-System, das aus Sicherheitsgründen vom entfernten Benutzer immer ein Paßwort verlangt.

Der Terminaltyp im entfernten Rechner ist gleich dem Typ des lokalen Terminals (entsprechend den Angaben in der Umgebungsvariablen TERM). Die Anzeige der Eingabe erfolgt auf dem entfernten Rechner, so daß eine Anmeldung mit *rlogin* transparent für den Benutzer ist (abgesehen von Zeitverzögerungen). Die Flußsteuerung mit ^S und ^Q sowie das Leeren der Ein-/Ausgabepuffer bei Unterbrechungen sind sichergestellt. Das optionale Argument **-8** erlaubt durchgängig einen acht Bit breiten Eingabedatenpfad. Andernfalls werden die Paritätsbit ausgeblendet, es sein denn, auf dem entfernten Rechner werden ^S/^Q als Start-/Stopzeichen verwendet. Wenn das Argument **-L** (litout) angegeben wird, ist während der rlogin-Sitzung der Litout-Modus aktiv. Die Eingabe der Zeile "~." bewirkt, daß die Verbindung zu dem entfernten Host abgebrochen wird ("~" ist das Fluchtzeichen). Mit der Option **-e** (escape) kann das Fluchtzeichen umdefiniert werden. Zwischen der Optionskennung und dem Argumentzeichen darf keine Leerstelle stehen.

SIEHE AUCH
rsh(1C)

DATEIEN
/usr/hosts/* für die *rhost*-Variante des Kommandos

FEHLER
Die Umgebung sollte ausführlicher beschrieben werden.

BEZEICHNUNG

rm, rmdir – Dateien oder Verzeichnisse entfernen

ÜBERSICHT

rm [**-f**] [**-i**] file...

rm -r [**-f**] [**-i**] dirname ... [file ...]

rmdir [**-p**] [**-s**] dirname ..

BESCHREIBUNG

rm entfernt aus einem Verzeichnis die Einträge von einer oder mehreren Dateien. Wenn ein Eintrag der letzte Verweis zur Datei war, wird die Datei entfernt. Das Entfernen einer Datei erfordert Schreibberechtigung in dem Verzeichnis, jedoch keine Lese- oder Schreibberechtigung für die Datei selbst.

Wenn eine Datei keine Schreibberechtigung hat und die Standardeingabe ein Terminal ist, werden alle Schutzbits (oktal) der Datei ausgegeben, gefolgt von einem Fragezeichen. Dies ist ein Eingabe-Aufforderungszeichen zur Bestätigung. Wenn die Antwort mit **y** (für ja) beginnt, wird die Datei gelöscht, andernfalls bleibt die Datei erhalten.

Wenn die Standardeingabe kein Terminal ist, ist zu beachten, daß das Kommando so arbeitet als ob die Option **-f** in Kraft ist.

rmdir entfernt die angegebenen Verzeichnisse, die leer sein müssen.

Drei Optionen sind bei *rm* gültig:

-f Diese Option entfernt alle Dateien (schreib-geschützt oder nicht) aus einem Verzeichnis, ohne Bestätigung des Benutzers. In einem schreib-geschützten Verzeichnis dürfen die Dateien niemals entfernt werden (unabhängig von der Berechtigung), es werden jedoch keine Nachrichten angezeigt. Wenn das Entfernen aus einem geschützten Verzeichnis versucht wurde, kann diese Option eine Fehlermeldung nicht unterdrücken.

-r Diese Option entfernt rekursiv beliebige Verzeichnisse und Unterverzeichnisse in der Argumentenliste. Das Verzeichnis wird geleert und gelöscht. Es ist zu beachten, daß der Benutzer normalerweise beim Entfernen von im Verzeichnis enthaltenen, schreibgeschützten Dateien durch eine Eingabe-Aufforderung zur Bestätigung aufgefordert wird. Die schreib-geschützten Dateien werden jedoch ohne Bestätigung gelöscht, wenn die Option **-f** benutzt wird oder wenn die Standardeingabe kein Terminal ist und die Option **-i** nicht verwendet wird.

Wenn das Entfernen eines nichtleeren, schreib-geschützten Verzeichnis versucht wurde, versagt das Kommando immer (selbst bei der Option **-f**) und eine Fehlermeldung wird ausgegeben.

-i Mit dieser Option wird das Entfernen von schreib-geschützten Dateien interaktiv bestätigt. Die Option **-f** wird übergangen und diese Option bleibt in Kraft, selbst wenn die Standardeingabe kein Terminal ist.

Zwei Optionen sind bei *rmdir* gültig:

-p Diese Option gestattet dem Benutzer, das Verzeichnis *dirname* und leer gewordene übergeordnete Verzeichnisse zu löschen. Eine Meldung wird auf der Standardausgabe ausgegeben, die aussagt, ob der ganze Pfad gelöscht ist oder ein Teil des Pfads aus bestimmten Gründen erhalten bleibt.

-s Diese Option wird zur Unterdrückung der Meldung verwendet, die auf der Standardfehlerausgabe ausgegeben wird, wenn **-p** in Kraft ist.

DIAGNOSE

Alle Meldungen sind gewöhnlich direkt verständlich.
Es ist verboten, die Dateien "." und ".." zu löschen, um die Folgen zu vermeiden, die aus Versehen durch Eingabe folgender Kommandos eintreten können:

```
rm -r .*
```

rm und *rmdir* liefern beide den Endecode 0, wenn alle angegebenen Dateien erfolgreich entfernt wurden. Andernfalls geben sie einen Endecode ungleich Null zurück.

SIEHE AUCH:

unlink(2), rmdir(2) im *Programmer's Reference Manual.*

BEZEICHNUNG

rpcgen – RPC-Kompiler

ÜBERSICHT

rpcgen -h [**-o** *outfile*] [*inputfile*]
rpcgen -c [**-o** *outfile*] [*infile*]
rpcgen *infile*
rpcgen [**-s** *transport*]* [**-o** *outfile*] [*infile*]

BESCHREIBUNG

Rpcgen ist ein Werkzeug, das C-Code zur Implementierung des RPC-Protokolls generiert. Die Eingabe für *rpcgen* wird in einer Sprache formuliert, die eine starke Ähnlichkeit mit C hat. Diese Sprache wird als RPCL (Remote Procedure Call Language = RPC-Sprache) bezeichnet. *Rpcgen* arbeitet in vier Modi. Der erste Modus wird zur Konvertierung von RPCL-Definitionen in C-Definitionen eingesetzt, die als Deklarationsdatei verwendet werden können. Der zweite Modus kompiliert die XDR-Routinen, die zur Serialisierung des durch RPCL beschriebenen Protokolls erforderlich sind. Der dritte Modus kompiliert beides. Die Deklarationen werden in einer Datei mit dem Namen *infile* und der Erweiterung **.h** abgelegt, die XDR-Routinen in einer Datei mit dem gleichen Namen und der Erweiterung **.c**. Der vierte Modus wird zur Kompilierung von Programmskeletten für RPC-Server eingesetzt. Diese müssen lediglich noch um lokale Prozeduren ergänzt werden. Zur Implementierung eines RPC-Servers sind daher keinerlei RPC-Kenntnisse erforderlich.

Die Eingabedatei kann Kommentare (wie in C) und Präprozessor-Anweisungen enthalten. Kommentare werden ignoriert, während die Anweisungen ohne weitere Verarbeitung in die Deklarationsdatei übernommen werden.

XDR-Routinen können durch individuelle Routinen ersetzt werden, wenn Sie die entsprechenden Datentypen undefiniert lassen. Wenn ein Datentyp nicht definiert ist, geht *rpcgen* davon aus, daß es eine Routine mit dem Namen 'xdr_' gibt, gefolgt von dem Namen des undefinierten Datentyps.

OPTIONEN

-c Es werden XDR-Routinen kompiliert.

-h Es werden C-Datendefinitionen kompiliert (es wird eine Deklarationsdatei ausgegeben).

-o *outfile*
Es muß der Name der Ausgabedatei angegeben werden. Wenn keine Ausgabedatei angegeben ist, wird in die Standardausgabedatei geschrieben (nur in den Modi **-c**, **-h** und **-s**).

-s *transport*
Für das angegebene Transportprotokoll *transport* wird ein Server kompiliert. Unterstützt werden die Transportprotokolle **udp** und **tcp**. Diese Option kann mehrfach aufgerufen werden. Dadurch kann ein Server kompiliert werden, der mehrere Transportprotokolle bedient.

ANWENDUNG

Übersicht über die RPCL-Syntax:

Diese Übersicht über die RPCL-Syntax kann zum Aufbau von Eingabedateien für *rpcgen* herangezogen werden. Sie soll als Hilfe zum Verständnis der Sprache dienen und stellt keine exakte Definition der Sprache dar.

Grundlegende Datentypen:

[unsigned] char
[unsigned] short
[unsigned] int
[unsigned] long
unsigned
float
double
void
bool

Abgesehen von dem Booleschen Datentyp **bool**, verwendet RPCL die gleichen grundlegenden Datentypen wie C. *Rpcgen* gibt **bool**-Deklarationen als **int**-Deklarationen in die Deklarationsdatei aus. (Genaugenommen wird eine Deklaration mit dem Typ **bool_t** erzeugt. Dieser Typ wiederum wurde mit einer **#define**-Anweisung dem Typ **int** gleichgesetzt.) Außerdem ist zu beachten, daß **void**-Deklarationen nur innerhalb einer **union**- oder **program**-Deklaration erscheinen dürfen. Wem es lästig ist, den Präfix **unsigned** auszuschreiben, kann statt dessen die Abkürzungen **u_char**, **u_short**, **u_int** und **u_long** verwenden.

Deklarationen:

RPCL unterstützt drei Arten von Deklarationen:

declaration:
simple-declaration
pointer-declaration
vector-declaration

simple-declaration:
type-name object-ident

pointer-declaration:
*type-name *object-ident*

vector-declaration:
type-name object-ident [*size*]

(*size* kann eine Integer- oder eine symbolische Konstante sein)

Im Vergleich zu C wurde der Leistungsumfang der RPCL-Deklarationen teilweise eingeschränkt und teilweise erweitert. Die Einschränkung besteht darin, daß die direkte Deklaration von mehrdimensionalen Arrays und von Zeigern auf Zeiger nicht erlaubt ist (die Deklaration mit **typedef** hingegen ist möglich). In RPCL sind zwei zusätzliche Arten von Deklarationen möglich:

Daten vom Typ "opaque" werden als Vektor definiert:

opaque *object-ident* [*size*]

Im RPCL-Protokoll ergibt dies ein Objekt mit einer Länge von *size* Byte. Beachten Sie, daß dies nicht gleichbedeutend mit der Deklaration von *size* Zeichen ist, da ein XDR-Zeichen aus 32 Bit besteht. Daten vom Typ "opaque" werden in der Deklarationsdatei als Zeichen-Arrays mit einer Länge von *size* Byte dargestellt.

Strings werden in RPCL anders als in C als Vektor deklariert:

string *object-ident* [*max-size*]

Wenn *max-size* nicht angegeben wird, kann ein String theoretisch beliebig groß sein. String-Deklarationen werden folgendermaßen konvertiert:

char **object-ident*

Typ-Definitionen:

XDR-Routinen können nur durch die Definition eines Datentyps erzeugt werden. Für jeden definierten Datentyp *zetype* wird die entsprechende XDR-Routine mit dem Namen *xdr_zetype* erzeugt.

Datentypen können auf sechs verschiedene Arten definiert werden:

type-definition:
typedef
enumeration-def (Aufzählung)
structure-def (Struktur)
variable-length-array-def (Array variabler Länge)
discrimiated-union-def (Markierte Variante)
program-def (Programm)

Die ersten drei Datentyp-Definitionen sind mit den entsprechenden C-Definitionen nahe verwandt. In C gibt es keine Möglichkeit, Arrays variabler Länge zu definieren. XDR-Unions sind in RPCL anders definiert als in C. Programm-Definitionen sind eigentlich keine Typ-Definitionen, sind aber trotzdem sehr nützlich.

XDR-Definitionen dürfen nicht geschachtelt werden. Die folgende Definition z. B. könnte *rpcgen* nicht verarbeiten:

```
struct tusnicht {
        struct habsgetan {
                int oje;
        } tutmirleid;
        enum habswiedergetan { OJE, ACHJE } verzeihung;
};
```

Typedefs

Eine **typedef**-Deklaration hat folgenden Aufbau:

typedef:
typedef *declaration* ;

Der *object-ident*-Teil in *declaration* ist der Name des neuen Typs, während der *type-name*-Teil der Name des Typs ist, von dem der neue Typ abgeleitet wird.

Aufzählungstypen

Aufzählungstypen haben die folgende Syntax:

enumeration-def:
enum *enum-ident* {
enum-list
};

enum-list:
enum-symbol-ident [= *assignment*]
enum-symbol-ident [= *assignment*] , *enum-list*

(*assignment* kann eine Integer- oder eine symbolische Konstante sein)

Wenn keine ausdrückliche Zuweisung erfolgt, wird der um 1 erhöhte Wert des vorherigen Elements der Aufzählung übernommen. Wird dem ersten Element einer Aufzählung nicht ausdrücklich ein Wert zugewiesen, dann erhält es den Wert 0.

Strukturen:

structure-def:

struct *struct-ident* {
declaration-list
};

declaration-list:

declaration ;
declaration ; *declaration-list*

Arrays variabler Länge:

variable-length-array-def:

array *array-ident* {
unsigned *length-identifier* ;
vector-declaration ;
};

Die Definition eines Arrays variabler Länge ähnelt dem Aufbau einer Struktur-Definition. Beispiel:

```
array mp_int {
    unsigned len;
    short val[MAX_MP_LENGTH];
};
```

Daraus werden die folgenden Anweisungen kompiliert:

```
struct mp_int {
    unsigned len;
    short *val;
};
typedef struct mp_int mp_int;
```

Unterschiedliche Unions:

discriminated-union-def:

union *union-ident* **switch** (*discriminant-declaration*) {
case-list
[**default** : *declaration* ;]
};

case-list:

case *case-ident* : *declaration* ;
case *case-ident* : *declaration* ; *case-list*

discriminant-declaration:

declaration

Die Definition von Unions ist gewissermaßen eine Kombination der C-Anweisungen *union* und *switch*. Beispiel:

```
union net_object switch (net_kind kind) {
case MACHINE:
        struct sockaddr_in sin;
case USER:
        int uid;
default:
        string whatisit;
};
```

Daraus werden die folgenden Anweisungen kompiliert:

```
struct net_object {
        net_kind kind;
        union {
                struct sockaddr_in sin;
                int uid;
                char *whatisit;
        } net_object;
};
typedef struct net_object net_object;
```

Beachten Sie, daß die *union*-Komponente der kompilierten Struktur den gleichen Namen hat wie der Typ selbst.

Programmdefinitionen:

program-def:

program *program-ident* {
version-list
} = *program-number* ;

version-list:

version
version version-list

version:

version *version-ident* {
procedure-list
} = *version-number* ;

procedure-list:

procedure-declaration
procedure-declaration procedure-list

procedure-declaration:

type-name procedure-ident (*type-name*) = *procedure-number* ;

Programmdefinitionen weichen so stark von den üblichen Konstrukten einer Programmiersprache ab, daß wir sie am besten anhand eines Beispiels erklären. Nehmen Sie an, daß Sie einen Server erstellen möchten, mit dem das Datum abgefragt oder gesetzt werden kann. Die Programm-Definition könnte z. B. so aussehen:

```
program DATE_PROG {
        version DATE_VERS {
                date DATE_GET(timezone) = 1;
                void DATE_SET(date) = 2;        /* Greenwich-Zeit */
        } = 1;
} = 100;
```

Daraus wird die folgende Deklarationsdatei generiert:

```
#define DATE_PROG 100
#define DATE_VERS 1
#define DATE_GET 1
#define DATE_SET 2
```

Die **define**-Anweisungen sind für das Client-Programm zum Aufruf der Server-Prozeduren vorgesehen.

Wenn Sie *rpcgen* zum Generieren eines Servers einsetzen, müssen Sie über einige wichtige Punkte Bescheid wissen. Die Schnittstelle zwischen dem Server und einer lokalen Prozedur wird durch den Prozedurnamen gewährleistet. Der Server geht davon aus, daß eine C-Funktion mit dem gleichen Namen wie in der Programmdefinition vorhanden ist. Der Name darf aber nur aus Kleinbuchstaben bestehen und muß mit der Versionsnummer enden. Die folgende lokale Prozedur ist eine Implementierung von DATE_GET:

```
date *  /* übergibt den Zeiger auf die Ergebnisse */
date_get_1(tz)
        timezone *tz;   /* für den Zeiger auf die Argumente */
{
        static date d;  /* muß static sein! */

        /*
         * Datum ermitteln
         * und in d speichern
         */
        return(&d);
}
```

Der Name der Routine ist der gleiche wie der Name, der in der **#define**-Anweisung angegeben wurde. Er besteht allerdings nur aus Kleinbuchstaben besteht, und die Versionsnummer wird angehängt. XDR gibt das Argument rekursiv aus, sobald ihm die Ergebnisse von der lokalen Prozedur übergeben wurden. Daher sollten Sie eine Kopie von den Argumentdaten anlegen, die Sie bei späteren Aufrufen wieder benötigen. XDR kümmert sich auch nicht um die Belegung oder Freigabe von Speicher für die Ergebnisse. Sie müssen daher selbst für die Speicherung der Ergebnisse sorgen.

Make-Regeln zur Kompilierung von XDR-Deklarationsdateien

Zum Kompilieren von XDR-Routinen und Deklarationsdateien können an *make*(1) entsprechende Suffix-Transformationsregeln übergeben werden. Die RPCL-Protokolldateien haben üblicherweise die Dateierweiterung **.x**. Dazu können die folgenden *make*-Regeln verwendet werden:

```
.SUFFIXES: .x
.x.c:
        rpcgen -c $< -o $@

.x.h:
        rpcgen -h $< -o $@
```

SIEHE AUCH

Network File System (NFS)

FEHLER

Beim Einsatz von Programmdefinitionen können Namenskollisionen auftreten, weil der theoretische Gültigkeitsbereich in der Praxis nicht zutrifft. Dies kann in den meisten Fällen durch die Vergabe eindeutiger Namen für Programme, Versionen, Prozeduren und Typen vermieden werden.

BEZEICHNUNG

rsh – Netzwerk–Shell

ÜBERSICHT

rsh [**–l** *username*] [**–n**] *host command*
host [**–l** *username*] [**–n**] *command*

BESCHREIBUNG

Rsh stellt eine Verbindung zu dem angegebenen Host *host* her und führt das angegebene Kommando *command* aus. *Rsh* übergibt die Standardeingabe an das entfernte Kommando und übernimmt von dem entfernten Kommando die Standardausgabe und die Standardfehlerdatei. Unterbrechungs-, Abbruch- und Endesignale werden an das entfernte Kommando weitergereicht. Daher hat die Beendigung des entfernten Kommandos in der Regel auch die Beendigung von *rsh* zur Folge.

Der entfernte Benutzername stimmt mit dem lokalen Benutzernamen überein, außer wenn mit der Option **–l** ein unterschiedlicher entfernter Benutzername *username* angegeben wird. Dieser entfernte Benutzername muß (im Sinn von *rlogin*(1C)) äquivalent zu dem lokalen Benutzerkonto sein. Die Angabe eines Paßworts in einem Kommando ist nicht vorgesehen.

Die Angabe des Arguments *command* bewirkt, daß ein einzelnes Kommando ausgeführt wird. Wenn diese Angabe fehlt, wird der Benutzer mit *rlogin*(1C) auf dem entfernten Host angemeldet.

Shell-Metazeichen *ohne* Anführungszeichen werden von dem lokalen Rechner interpretiert, *mit* Anführungszeichen von dem entfernten Rechner. Daher bewirkt das Kommando

```
rsh otherhost cat remotefile >> localfile
```

daß die entfernte Datei *remotefile* an die lokale Datei *localfile* angehängt wird. Hingegen bewirkt das Kommando

```
rsh otherhost cat remotefile ">>" otherremotefile
```

daß die entfernte Datei *remotefile* an die andere entfernte Datei *otherremotefile* angehängt wird.

Hostnamen werden in der Datei /etc/hosts definiert. Jedem Host wird ein längerer und eindeutiger Standardname zugeordnet (der erste Name in einem Eintrag) und optional ein oder mehrere Aliasnamen. Die Hostnamen der lokalen Rechner sind auch als Kommandos in dem Verzeichnis /usr/hosts angelegt. Wenn dieses Verzeichnis in den Suchpfad des Benutzers eingetragen wird, kann die Angabe von **rsh** im Kommandoaufruf entfallen.

DATEIEN

/etc/hosts
/usr/hosts/*

SIEHE AUCH

rlogin(1C)

FEHLER

Wenn ein Benutzer mit *csh*(1) arbeitet und *rsh* im Hintergrund laufen läßt, sollte die Eingabe von dem Terminal auf ein anderes Gerät umgelenkt werden. Andernfalls hängt sich das Terminal auf, selbst dann, wenn das entfernte Kommando keine Lesebefehle absetzt. Wenn keine Eingaben erwünscht sind, sollte die Eingabe von *rsh* mit Hilfe der Option **-n** auf /dev/null umgelenkt werden.

Interaktive Kommandos wie *rogue*(6) oder *vi*(1) sind unter *rsh* nicht lauffähig; in diesen Fällen muß auf *rlogin*(1C) zurückgegriffen werden.

Stopsignale beenden lediglich den lokalen *rsh*-Prozeß. Dies ist ein offenkundiger Fehler, derzeit aber schwer zu beheben (aus Gründen, die hier nicht näher ausgeführt werden können).

BEZEICHNUNG

ruptime – Host-Status lokaler Rechner anzeigen

ÜBERSICHT

ruptime [**-a**] [**-r**] [**-l**] [**-t**] [**-u**]

BESCHREIBUNG

Alle Hosts eines Netzwerks verschicken einmal pro Minute an alle anderen Hosts im Netzwerk eine Statusmeldung. *Ruptime* bildet aus diesen Angaben für jeden Rechner im lokalen Netzwerk eine Statuszeile (wie *uptime*) und gibt sie aus.

Wenn ein Rechner 11 Minuten lang keine Statusmeldung sendet, wird er als inaktiv ("down") angezeigt.

Benutzer, die seit mindestens einer Stunde keine Eingabe gemacht haben, werden nur angezeigt, wenn die Option **-a** (all) angegeben wird.

Standardmäßig wird die ausgegebene Liste nach Host-Namen sortiert. Die Optionen **-l**, **-t** und **-u** bewirken, daß stattdessen nach der durchschnittlichen Auslastung (load), nach der Laufzeit seit dem letzten Systemstart (uptime) bzw. nach der Anzahl der Benutzer (users) sortiert wird. Die Option **-r** (reverse) bewirkt, daß die Sortierfolge umgekehrt wird.

DATEIEN

/usr/spool/rwho/whod.*	Informationen über andere Rechner

SIEHE AUCH

rwho(1C)

BEZEICHNUNG

rusers – angemeldete Benutzer in lokalen Rechnern anzeigen (RPC-Version)

ÜBERSICHT

rusers [**-a**] [**-h**] [**-i**] [**-l**] [**-u**] [*host*]

BESCHREIBUNG

Das Kommando *rusers* gibt eine ähnliche Liste aus wie *users* und *who*, aber für entfernte Rechner. *Rusers* sendet (als Broadcast) eine Anfrage an alle Rechner in einem lokalen Netzwerk und gibt die eingehenden Antworten aus. Die ausgegebene Liste wird standardmäßig nach der Reihenfolge der eingehenden Antworten sortiert. Diese Sortierfolge kann aber durch eine der unten aufgeführten Optionen geändert werden. Wenn mit dem Argument *host* bestimmte Host-Rechner angegeben werden, richtet *rusers* seine Anfrage statt an alle nur an die angegebenen Host-Rechner.

Standardmäßig besteht die Ausgabeliste aus einer Zeile pro Rechner (wie bei *users*(1)). Bei Angabe der Option **-l** wird die Liste nach dem Muster von *rwho*(1) aufgebaut. Zusätzlich wird die Leerlaufzeit angezeigt, wenn ein Benutzer seit mindestens einer Minute keine Eingabe mehr gemacht hat.

Ein entfernter Rechner antwortet auf Anfragen von *rusers* nur dann, wenn auf ihm ein *rusersd*-Dämon läuft. Dieser wird in der Regel von *inetd*(1C) gestartet.

OPTIONEN

-a (all) Auch Rechner ohne angemeldete Benutzer werden angezeigt.

-h (host name) Die Liste wird alphabetisch nach Host-Namen sortiert.

-i (idle time) Die Liste wird nach der Leerlaufzeit sortiert.

-l (long) Es wird eine ausführlichere Liste nach dem Muster von *who(1)* ausgegeben.

-u (user) Die Liste wird nach der Anzahl der Benutzer sortiert.

DATEIEN

/etc/servers

SIEHE AUCH

rwho(1C), inetd(1C), rusersd(1C)

FEHLER

Broadcast-Meldungen (Meldungen an alle) sind über Gateways nicht möglich.

BEZEICHNUNG

rwall – Meldungen an alle Netzwerkbenutzer senden

ÜBERSICHT

rwall *hostname* ...
rwall **-n** *netgroup* ...
rwall **-h** *host* **-n** *netgroup*

BESCHREIBUNG

Rwall liest eine Meldung von der Standardeingabe, bis das Dateiende erreicht ist. Dieser Meldung stellt *rwall* die Zeile "Broadcast Message ..." (Meldung an alle) voran und sendet sie dann an alle Benutzer, die in den angegebenen Rechnern angemeldet sind. Die Option **-n** bewirkt, daß die Meldung an die angegebenen, in *netgroup*(4) definierten Netzwerkgruppen verschickt wird.

Ein Rechner kann eine solche Nachricht nur dann empfangen, wenn *rwalld* auf dem Rechner aktiv ist. *rwalld* wird in der Regel aus /etc/servers von dem Dämon *inetd*(1C) gestartet.

DATEIEN

/etc/servers

SIEHE AUCH

wall(1), netgroup(4), rwalld(1C), shutdown(1M)

FEHLER

Rwall wartet nur kurz auf Bestätigung. Nur so kann eine große Gruppe von Rechnern (von denen manche inaktiv sein können) innerhalb einer akzeptablen Zeit versorgt werden. Daher kann es vorkommen, daß die Meldung zu einem stark ausgelasteten Rechner nicht durchkommt.

BEZEICHNUNG

rwho – angemeldete Benutzer in lokalen Rechnern anzeigen

QUELLE

4.2BSD

ÜBERSICHT

rwho [**–a**]

BESCHREIBUNG

Das Kommando *rwho* gibt eine ähnliche Liste aus wie *who*, aber für alle Rechner in einem lokalen Netzwerk. Wenn ein Rechner 5 Minuten lang keine Statusmeldung sendet, wird er als inaktiv ("down") angesehen. *Rwho* zeigt in diesem Fall auch die zuletzt in diesem Rechner angemeldeten Benutzer nicht an.

Wenn ein Benutzer seit mindestens einer *Minute* keine Eingabe mehr gemacht hat, zeigt *rwho* diese Leerlaufzeit an. Ein Benutzer, der seit mindestens einer *Stunde* keine Eingabe mehr gemacht hat, wird nur angezeigt, wenn die Option **–a** (all) angegeben wird.

HINWEISE

Rwho erfüllt den DOD-Standard RFC 810 des US-Verteidigungsministeriums (DOD = Department of Defense):

"a" - "z"
"0" - "9"
"-"
"."
keine Leerstelle
keine Unterscheidung zwischen Groß- und Kleinschreibung

DATEIEN

/usr/spool/rwho/whod.* Informationen über andere Rechner

SIEHE AUCH

ruptime(1C), rwhod(1C)

FEHLER

Bei lokalen Netzwerken mit einer großen Zahl von Rechnern verschlechtert sich das Zeitverhalten erheblich.

BEZEICHNUNG

sag – graphische Darstellung der Systemaktivität

ÜBERSICHT

sag [options]

BESCHREIBUNG

sag zeigt grafisch die Daten der Systemaktivitäten an, die in einer binären Datendatei durch ein vorheriges *sar(1)* Kommando gespeichert wurden. Alle *sar* Datenelemente können einzeln oder kombiniert gezeichnet werden; als "Cross-Plots" oder zeitbezogene Darstellung. Einfache Arithmetikkombinationen der Daten können angegeben werden. *sag* ruft *sar* auf und findet die gewünschten Daten durch Vergleich der Spalten-Überschriften. (Sie können *sar* aufrufen, um festzustellen, welche Daten zur Verfügung stehen). Diese Optionen *options* werden an *sar* weitergereicht:

-s *time* Daten später als *time* in Form von hh[:mm] auswählen. Standard ist 08:00.

-e *time* Daten bis zu *time* wählen. Standard ist 18:00.

-i *sec* Daten in Abständen von Sekunden *sec* wählen.

-f *file* Datei *file* als die Datenquelle für *sar* wählen. Standard ist die aktuelle Tagesdatendatei **/usr/adm/sa/sa***dd*.

Andere Optionen *options*:

-T *term* Erzeugt Ausgabe, die für das Terminal *term* geeignet ist. Vgl. *tplot*(1G) bzgl. bekannter Terminals. Standard für *term* ist **$TERM**.

-x *spec* x-Achsenbezeichnung *spec* in der Form: "name [op name]... [lo hi]"

-y *spec* y-Achsenbezeichnung *spec* in der gleichen Form wie oben.

name ist entweder eine Zeichenfolge, die mit einer Spaltenüberschrift in einem *sar*- Bericht übereinstimmt, mit einem wahlweisen Gerätenamen in eckigen Klammern z.B. **r+w/s[dsk-1]** oder eine ganze Zahl. *Op* ist + - * oder / umgeben von Leerzeichen. Bis zu fünf Namen können angegeben werden. Runde Klammern werden nicht erkannt. Unüblicherweise haben + und - Priorität vor * und /. Auswertung ist von links nach rechts. Folglich wird A / A + B * 100 als (A/(A+B)) *100 bewertet, und A + B / C + D ist (A+B)/(C+D). *Lo* und *hi* sind wahlweise numerische Skalierungsgrenzen. Wenn sie nicht angegeben sind, werden sie aus den Daten abgeleitet.

Ein einzelnes *spec* ist für die x-Achse zugelassen. Wenn nichts angegeben ist, wird *time* verwendet. Bis zu 5 *spec*'s, die durch ; getrennt sind, können für die y -Achse angegeben werden. Die Argumente bei -x und -y sind in "" einzuschließen, wenn Leerzeichen oder \<CR> vorkommen. Der -y Standard ist:

-y "%usr 0 100; %usr + %sys 0 100; %usr + %sys + %wio 0 100"

BEISPIELE

Zur Anzeige der heutigen CPU-Nutzung:

```
sag
```

Zur Anzeige der Aktivität aller Plattenlaufwerke während 15 Minuten:

```
TS =date +%H:%M
sar -o tempfile 60 15
TE=date +%H:%M
sag -f tempfile -s $TS -e $TE -y "r+w/s[dsk]"
```

DATEIEN

/usr/adm/sa/sa*dd* Tägliche Datendatei für Tag *dd*.

SIEHE AUCH:

sar(1), tplot(1G)

BEZEICHNUNG

sar – Bericht über Systemtätigkeit

ÜBERSICHT

sar [**-ubdycwaqvm**] [**-o** file] t [n]

sar [**-ubdycwaqvm**] [**-s** time] [**-e** time] [**-i** sec] [**-f** file]

BESCHREIBUNG

sar sammelt zunächst die in *n* Intervallen zu *t* Sekunden aufgelaufenen Werte der Aktivitätszähler im System; *t* sollte gleich 5 oder größer sein. Wenn die Option **-o** angegeben ist, werden die Werte in der Datei *file* in Binärformat gesichert. Der Standardwert von *n* ist 1. Im zweiten Fall, ohne Angabe des Prüfintervalls, entnimmt **sar** Daten aus der vorher aufgezeichneten Datei *file* und zwar aus der mit der Option **-f** angegebenen Datei oder standardmäßig aus der Tagesdatendatei für Systemtätigkeit **/usr/adm/sa/sa***dd* für den aktuellen Tag *dd*. Die Start- und Endezeiten des Berichts können über die Argumente **-s** und **-e** *time* in Form von *hh*[*:mm*[*:ss*]] angegeben werden. Die Option **-i** wählt Datensätze in *sec* Sekundenabständen. Andernfalls werden alle in der Datendatei gefundenen Intervalle gemeldet.

In beiden Fällen werden die anzugebenden Teilmengen der Daten mit Hilfe von Optionen angegeben:

-u Bericht über CPU-Auslastung (Standard):
%usr, %sys, %wio, %idle – Laufzeitanteil im Benutzer-Modus, im Systemmodus, untätig – Prozeß wartet auf blockorientierte Ein-/Ausgabe oder ist anderweitig untätig. Bei Angabe von **-D** wird %sys zerlegt in die Zeit, die für Aufträge von Remote-Geräten verbraucht wurde (%sys remote) und die Zeit für alle anderen System-Tätigkeiten (%sys lokal).

-b Bericht über die Pufferauslastung:
bread/s, bwrit/s – Datenübertragungen pro Sekunde zwischen Systempuffer und Festplatte oder anderen blockorientierten Geräten;
lread/s, lwrit/s – Zugriffe auf Systempuffer;
%rcache, %wcache – Cache-Speicher-Treffer-Verhältnisse, d. h. (1–bread/lread);
pread/s, pwrit/s – physikalische (raw) Datenübertragungen.

-d Bericht über die Tätigkeit von jedem blockorientierten Gerät, z. B. Festplatte oder Magnetband. Wenn Daten ausgegeben werden, wird *dsk-* gewöhnlich zur Angabe eines Plattenlaufwerks verwendet. Die Angabe für ein Magnetbandlaufwerk ist maschinenabhängig. Die gemeldeten Tätigkeitsdaten sind:
%busy, avque – Zeit, die das Gerät mit der Bedienung einer Übertragungsanforderung beschäftigt war, durchschnittliche Anzahl der während dieser Zeit angestandenen Aufträge;

r+w/s, blks/s – Anzahl von Datenübertragungen von oder zum Gerät, Anzahl der übertragenen Bytes in 512-Byte-Einheiten;
avwait, avserv – Durchschnittliche Zeit in ms, in der Übertragungsanforderungen untätig in einer Warteschlange stehen, und durchschnittliche Bearbeitungszeit (bei Festplatten beinhaltet dieser Wert die Zeit für Suchen, Positionieren und Datenübertragung).

-y Bericht über TTY-Gerät-Aktivitäten:
rawch/s, canch/s, outch/s – Eingabezeichenrate, mit "canon" verarbeitete Eingabezeichenrate, Ausgabezeichenrate;
rcvin/s, xmtin/s, mdmin/s – Empfang-, Übertragungs- und Modem- Unterbrechungsraten.

-c Bericht über Systemaufrufe:
scall/s – alle Systemaufrufarten;
sread/s, swrit/s, fork/s, exec/s – Spezielle Systemaufrufe;
rchar/s, wchar/s – Zeichen, die durch read und write Systemaufrufe übertragen wurden. Bei Angabe von **-D** werden die Systemaufrufe eingeteilt in empfangene, gesendete und rein lokale Aufrufe.

-w Bericht über Systemeinlagerungs- und Auslagerungstätigkeit:
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 erstmaligem Laden einiger Programme);
pswch/s – Prozeßschalter.

-a Bericht über Anwendung von Systemroutinen für Dateizugriff:
iget/s, namei/s, dirblk/s.

-q Bericht über durchschnittliche Warteschlangenlänge bei Beschäftigung und % der belegten Zeit:
runq-sz, %runocc – Ablauf-Warteschlange mit Prozessen, die bereit sind und im Hauptspeicher warten.
swpq-sz, %swpocc – Austausch-Warteschlange der Prozesse, die bereit und ausgelagert sind.

-v Bericht über Status der Prozesse, I-Knoten-Einträge, Dateitabellen:
text-sz, proc-sz, inod-sz, fil-sz, lock-sz – Einträge/Größe für jede Tabelle, wird einmal pro Intervall ausgewertet;
ov – Überläufe, die zwischen Intervallen bei jeder Tabelle eintreten.

BEISPIELE

Bericht über CPU-Tätigkeit des Tages bis zum Aufrufzeitpunkt

```
sar
```

Beobachtung der CPU-Tätigkeit für 10 Minuten und Abspeichern der Daten:

```
sar -o temp 60 10
```

Spätere Prüfung der Festplatten- und Magnetbandauslastung während dieser Zeit:

```
sar -d -f temp
```

DATEIEN

/usr/adm/sa/sa*dd*

Tägliche Datendatei, wobei *dd* Ziffern sind, die den Tag des Monats darstellen.

SIEHE AUCH:

sag(1g).
sar(1M) im *Administrator's Reference Manual*

BEZEICHNUNG

sdiff – Gegenüberstellung von zwei Dateien

ÜBERSICHT

sdiff [options ...] file1 file2

BESCHREIBUNG

sdiff verwendet die Ausgabe von *diff*(1), um eine Gegenüberstellung von zwei Dateien zu erstellen, in der unterschiedliche Zeilen gekennzeichnet sind. Die Zeilen der beiden Dateien werden gegenübergestellt und zwar mit Leerzeichen zwischen den Zeilen, wenn die Zeilen identisch sind, mit < zwischen den Zeilen, wenn die Zeile nur in *file1* vorhanden ist, mit > zwischen den Zeilen, wenn die Zeile nur in *file2* vorhanden ist, und mit | zwischen den Zeilen für Zeilen, die unterschiedlich sind.

Zum Beispiel:

```
x | y
a a
b <
c <
d d
> c
```

Folgenden Optionen sind möglich:

–w *n* Verwendet das nächste Argument *n* als die Breite der Ausgabezeile. Die Standardzeilenlänge ist 130 Zeichen.

–l Gibt nur die linke Seite bei Zeilen aus, die identisch sind.

–s Gibt identische Zeilen nicht aus.

–o *output* Verwendet das nächste Argument *output* als Namen einer dritten Datei, die gemäß den Angaben des Benutzers eine Mischung aus *file1* und *file2* wird. Identische Zeilen von *file1* und *file2* werden nach *output* kopiert. Gruppen von Unterschieden, die *diff*(1) erzeugt, werden ausgegeben; eine Gruppe von Unterschieden hat das gleiche Zwischenzeichen. Nach der Ausgabe einer Gruppe von Unterschieden, gibt *sdiff* die Eingabe-Aufforderung % aus und wartet auf eines der folgenden, vom Benutzer einzugebende, Kommandos:

- **l** hängt die linke Spalte an die Ausgabedatei an
- **r** hängt die rechte Spalte an die Ausgabedatei an
- **s** schaltet Modus 'nicht anzeigen' ein; identische Zeilen werden nicht ausgegeben

v	schaltet Modus 'nicht anzeigen' aus
e l	ruft den Editor mit der linken Spalte auf
e r	ruft den Editor mit der rechten Spalte auf
e b	ruft den Editor mit der Verkettung von linker und rechter Spalte auf
e	ruft den Editor mit Datei der Länge Null auf
q	beendet das Programm

Beim Verlassen des Editors wird die resultierende Datei ans Ende der Ausgabedatei *output* angehängt.

SIEHE AUCH:
diff(1), ed(1).

BEZEICHNUNG

sed – Stream-Editor

ÜBERSICHT

sed [**-n**] [**-e** script] [**-f** sfile] [files]

BESCHREIBUNG

sed kopiert die angegebenen Dateien *files* (Standardeingabe ist voreingestellt), die entsprechend einem Kommando-Skript editiert wurden, auf die Standardausgabe. Bei der Option **-f** wird das Skript aus der Datei *sfile* genommen; diese Optionen werden zusammengefaßt. Wenn es nur eine Option **-e** gibt und keine **-f** Option, kann die Angabe **-e** weggelassen werden. Die Option **-n** unterdrückt die standardmäßig erzeugte Ausgabe. Ein Skript besteht aus Editierkommandos, ein Kommando je Zeile, der folgenden Form:

[address [, address]] function [arguments]

Normalerweise kopiert *sed* zyklisch eine Zeile der Eingabe in den Musterbereich *pattern space* (sofern nach einem Kommando **D** nicht etwas zurückbleibt), wendet der Reihenfolge nach alle Kommandos an, deren *addresses* diesen Musterbereich auswählen, und kopiert zum Schluß den Musterbereich auf die Standardausgabe (außer bei **-n**) und löscht den Musterbereich.

Für späteres Wiederauffinden verwenden einige Kommandos einen Zeichenpuffer *hold space* zur ganzen oder teilweisen Sicherung des Musterbereichs.

Eine *address* ist entweder eine Dezimalzahl, die Eingabezeilen fortlaufend über Dateien hinweg durchzählt, oder ein **$** , das die letzte Zeile der Eingabe adressiert bzw. eine Kontextadresse d. h. ein */regular expression/* ähnlich wie bei *ed*(1), doch wie nachstehend modifiziert:

In einer Kontextadresse ist die Konstruktion *\?regular expression?*, wobei *?* ein beliebiges Zeichen sein kann, identisch zu */regular expression/*. Es ist zu beachten, daß in der Kontextadresse **\xabc\xdefx** das zweite **x** für sich selbst steht, so daß der reguläre Ausdruck **abcxdef** ist.

Die Folge **\n** bezeichnet ein New-Line-Zeichen im Musterbereich.

Ein Punkt **.** bezeichnet ein beliebiges Zeichen außer dem New-Line-Zeichen im Musterbereich.

Eine Kommandozeile ohne Adressen gilt für jede Zeile im Musterbereich.

Eine Kommandozeile mit einer Adresse gilt für jede Zeile im Musterbereich, die zu der Adresse paßt.

Eine Kommandozeile mit zwei Adressen gilt für den Bereich (inklusive) vom ersten Musterbereich, der mit der ersten Adresse übereinstimmt bis zum nächsten Musterbereich, der mit der zweiten übereinstimmt. (Wenn die zweite Adresse eine Zahl ist, die kleiner oder gleich der zuerst ausgewählten Zeilennummer ist, wird nur eine Zeile ausgewählt.) Danach wird der Vorgang wiederholt, indem wieder nach der ersten Adresse gesucht wird.

Sollen die Zeilen im Musterbereich editiert werden, die nicht durch Adressen ausgewählt werden, muß die Funktion ! (Negation) verwendet werden (siehe unten).

In der folgenden Funktionsliste wird das Maximum an zulässigen Adressen für jede Funktion in runden Klammern angezeigt.

Das *text* Argument besteht aus einer oder mehreren Zeilen, wobei, alle außer der letzten, mit \ enden, um das New-Line-Zeichen zu entwerten. Backslashes im Text werden wie Backslashes in der Ersatzzeichenfolge eines s-Kommandos behandelt und können dazu verwendet werden, ursprüngliche Leerzeichen und Tabulatoren am Zeilenanfang zu bewahren, die ansonsten von jeder Skriptzeile entfernt werden. Das Argument *rfile* oder *wfile* muß am Ende einer Kommandozeile stehen und ihm muß genau ein Leerzeichen vorausgehen. Jede *wfile* wird erstellt, ehe die Verarbeitung beginnt. Höchsten 10 verschiedene *wfile* Argumente dürfen vorkommen.

(1) **a**
text
: Anhängen. Schreibt *text* in die Ausgabe, ehe die nächste Eingabezeile gelesen wird.

(2) **b** *label*
: Verzweigen auf : Kommando, das die Marke *label* hat. Wenn *label* leer ist, wird zum Ende des Skripts verzweigt.

(2) **c**
text
: Ändern. Löscht den Musterbereich. Bei 0 oder 1 Adresse, oder am Ende eines 2-Adressen-Bereichs, wird *text* in die Ausgabe geschrieben. Danach wird mit dem nächsten Zyklus begonnen.

(2) **d**
: Löscht den Musterbereich. Danach wird mit dem nächsten Zyklus begonnen.

(2) **D**
: Löscht das ursprüngliche Segment des Musterbereichs bis zum ersten New-Line-Zeichen. Danach wird mit dem nächsten Zyklus begonnen.

(2) **g**
: Ersetzt den Inhalt des Musterbereichs durch den Inhalt des Zeichenpuffers.

(2) **G**
: Hängt den Inhalt des Zeichenpuffers an den Musterbereich an.

(2) **h** Ersetzt den Inhalt des Zeichenpuffers durch den Inhalt des Musterbereichs.

(2) **H** Hängt den Inhalt des Musterbereichs an den Zeichenpuffer an.

(1) **i**
text Einfügen. Schreibt *text* auf die Standardausgabe.

(2) **l** Gibt den Musterbereich auf der Standardausgabe eindeutig an. Nicht druckbare Zeichen werden durch zwei ASCII-Zeichen ausgegeben und lange Zeilen werden umgebrochen.

(2) **n** Kopiert den Musterbereich auf die Standardausgabe. Ersetzt den Musterbereich durch die nächste Zeile der Eingabe.

(2) **N** Hängt die nächste Zeile der Eingabe an den Musterbereich, mit einem New-Line-Zeichen dazwischen, an. (Die aktuelle Zeilennummer verändert sich.)

(2) **p** Ausgabe. Kopiert den Musterbereich in die Standardausgabe.

(2) **P** Kopiert das ursprüngliche Segment des Musterbereichs bis zum ersten New-Line-Zeichen in die Standardausgabe.

(1) **q** Beenden. Verzweigt zum Ende des Skripts. Keinen neuen Zyklus starten.

(2) **r** *rfile* Liest den Inhalt von *rfile*. Schreibt diesen in die Ausgabe, ehe die nächste Eingabezeile gelesen wird.

(2) **s**/*regular expressions*/*replacement*/*flags*
Ersetzt im Musterbereich Vorkommen des *regular expression* durch die Ersatzzeichenfolge *replacement*. Beliebige Zeichen können anstelle von / verwendet werden. Eine detaillierte Beschreibung finden Sie in *ed*(1). *flags* kann fehlen oder aus folgenden Angaben (auch mehrere) bestehen:

n n = 1 – 512. Nur das n-te Vorkommen von *regular expression* wird ersetzt.

g Global. Ersetzt alle nichtüberlappenden Vorkommen von *regular expression*, nicht nur das erste.

p Ausgabe des Musterbereichs, falls eine Ersetzung durchgeführt wurde.

w *wfile*
Schreiben. Hängt den Musterbereich an *wfile* an, falls eine Ersetzung durchgeführt wurde.

(2) **t** *label* Prüfen. Verzweigt auf das Kommando : , das die Marke *label* hat, falls Substitutionen seit dem letzten Lesen einer Eingabezeile oder Ausführung eines **t** durchgeführt worden sind. Wenn *label* leer ist, wird zum Ende des Skripts verzweigt.

(2) **w** *wfile* Schreiben. Hängt den Musterbereich an *wfile* an.

(2) **x** Tauscht den Inhalt der Muster- und Zeichenpuffere aus.

(2) **y**/*string1*/*string2*/
Transformieren. Ersetzt alle Vorkommnisse der Zeichen in *string1* durch die entsprechenden Zeichen in *string2*. Die Längen von *string1* und *string2* müssen gleich sein.

(2)! *function*
Negation. Wendet die *function* (oder Gruppe, wenn *function* { ist) nur bei Zeilen an, die *nicht* durch Adresse(n) angegeben sind.

(0) : *label* Dieses Kommando hat die Marke *label* als Verzweigungsziel der Kommandos **b** und **t**.

(1) = Gibt die aktuelle Zeilennummer auf die Standardausgabe aus.

(2) { Führt die folgenden Kommandos bis zu einem passenden } nur dann aus, wenn der Musterbereich ausgewählt ist.

(0) Ein leeres Kommando wird ignoriert.

(0) # Wenn ein # als erstes Zeichen auf der ersten Zeile einer Skriptdatei erscheint, wird diese ganze Zeile als ein Kommentar behandelt, es gibt jedoch eine Ausnahme. Wenn das Zeichen nach # ein 'n' ist, wird die standardmäßige Ausgabe unterdrückt. Die restliche Zeile wird nach #n ebenfalls ignoriert. Eine Skriptdatei muß mindestens eine Nicht-Kommentar-Zeile enthalten.

SIEHE AUCH:

awk(1), ed(1), grep(1). §

BEZEICHNUNG

setup – initialisieren des Systems für ersten Benutzer

ÜBERSICHT

setup

BESCHREIBUNG

Mit dem Kommando *setup*, das auch mit dem gleichen Namen als Login benutzt werden kann, wird der erste Benutzer als der "Eigentümer" des Geräts eingetragen.

Der Benutzer darf die ersten Logins ins System eingeben, wobei er gewöhnlich mit dem eigenen Login beginnt.

Der Benutzer kann das System dann vor unbefugter Modifikation der Maschinenkonfiguration und der Software schützen, indem er für die Verwaltungs- und Wartungsfunktionen Paßworte vergibt. Normalerweise gibt der erste Benutzer dieses Kommando im Setup-Login ein, daß anfänglich kein Paßwort hat, und gibt dann den verschiedenen Funktionen im System Paßworte. Funktionen, die vom Benutzer keinen Paßwortschutz erhalten, können von einer beliebigen Person ausgeführt werden.

Der Benutzer kann dann für die Systemlogins wie zum Beispiel "root" , "bin", usw. Paßworte vergeben (*vorausgesetzt sie haben noch keine*). Sobald ein Paßwort für ein Login vergeben wurde, kann es nur von diesem Login oder "root" geändert werden.

Der Benutzer kann dann am Gerät das Datum, die Zeit und Zeitzone setzen.

Der Benutzer kann anschließend den Knotennamen des Geräts setzen.

SIEHE AUCH:

passwd(1).

DIAGNOSE

Das Kommando *passwd*(1) gibt eine Meldung aus, wenn das angegebene Paßwort nicht den Standards entspricht.

ACHTUNG!

Wenn das setup Login kein Paßwort hat, kann eine beliebige Person Paßworte für die anderen Funktionen vergeben.

BEZEICHNUNG

sh, rsh - Shell - der Standard- und eingeschränkte Kommando-Interpreter

ÜBERSICHT

sh [**-acefhiknrstuvx**] [args]
rsh [**-acefhiknrstuvx**] [args]

BESCHREIBUNG

sh ist ein Kommandointerpretierer, der Kommandos ausführt, die von einem Terminal oder aus einer Datei gelesen werden. *rsh* ist eine eingeschränkte Version des Standard-Interpreters *sh; rsh* wird zum Einrichten von Login-Namen und Ausführungsumgebungen verwendet, deren Möglichkeiten gegenüber der Standard-Shell eingeschränkter sind. Vgl. nachstehend "Aufrufe" hinsichtlich der Bedeutung von Argumenten bei der Shell.

Definitionen

Ein Leerzeichen *blank* ist ein Tabulator oder ein Leerzeichen. Ein Name *name* ist eine Folge von Buchstaben, Ziffern oder Unterstrichen, die mit einem Buchstaben oder einem Unterstrich beginnen. Ein Parameter *parameter* ist ein Name, eine Ziffer oder eines der Zeichen *, @, #, ?, -, $ und !.

Kommandos

Ein einfaches Kommando *simple-command* ist eine Folge von Worten *words* die nicht aus Leerzeichen bestehen und durch *blanks* getrennt sind. Das erste Wort gibt den Namen des auszuführenden Kommandos an. Wenn nachstehend nicht anders angegeben, werden die restlichen Worte als Argumente an das aufgerufene Kommando weitergeleitet. Der Kommandoname wird als Argument 0 weitergeleitet (vg. *exec*(2)). Der Wert *value* eines *simple-command* ist der Endestatus, wenn das Kommando normal beendet wird, oder (oktal) 200+*status*, wenn es mit Fehler endet (vgl. *signal*(2) bezüglich einer Liste von Rückkehrwerten).

Eine Pipeline *pipeline* ist eine Folge von einem oder mehreren Kommandos *commands* die durch | getrennt sind. Die Standardausgabe aller Kommandos, außer des letzten, wird mit *pipe*(2) an die Standardeingabe des nächsten Kommandos angeschlossen. Jedes Kommando läuft als ein separater Prozeß; die Shell wartet auf die Beendigung des letzten Kommandos. Der Endestatus einer Pipeline ist der Endestatus des letzten Kommandos.

Eine Kommandoliste *list* ist eine Folge einer oder mehrerer Pipelines, die durch ;, &, && oder || getrennt sind, und wahlweise mit ; oder & abgeschlossen sind. Von diesen vier Symbolen haben ; und & gleiche Priorität, die niedriger als die von && und || ist. Die Symbole && und || haben auch gleiche Priorität. Ein Semikolon (;) bewirkt sequentielle

Ausführung der vorhergehenden Pipeline; ein Und-Zeichen (&) veranlaßt asynchrone Ausführung der vorhergehenden Pipeline (d.h. die Shell wartet *nicht* auf die Beendigung dieser Pipeline). Das Symbol && (||) bewirkt, daß die darauffolgende *list* nur ausgeführt wird, wenn die vorhergehende Pipeline einen Endestatus Null (ungleich Null) zurückgibt. Eine beliebige Anzahl New-Line-Zeichen können in *list* zur Begrenzung von Kommandos anstelle von Semikolons vorkommen.

Ein Kommando *command* ist entweder ein *simple-command* oder eines der folgenden. Wenn nicht anders angegeben, ist der von einem Kommando zurückgegebene Wert der Wert des letzten in diesem Kommando ausgeführten *simple-command*.

for *name* [**in** *word* ...] **do** *list* **done**
Jedes Mal, wenn ein **for**-Kommando ausgeführt wird, wird *name* auf das nächste *word* der **in** *word*-Liste gesetzt. Wenn **in** *word* ... fehlt, dann führt das **for**-Kommando die **do** *list* einmal für jeden gesetzten Stellungsparameter aus (vgl. unten *Parameter-Substitution*). Die Ausführung endet, wenn keine Worte mehr in der Liste sind.

case *word* **in** [*pattern* [| *pattern*] ...) *list* ;;]
Ein **case**-Kommando führt die *list* aus, die zum ersten Muster *pattern* das zu *word* paßt, gehört. Das Format von *pattern* ist das gleiche wie bei der Dateinamenserzeugung (vgl. "Dateinamenserzeugung"), außer daß ein Slash, ein führender Punkt oder ein Punkt direkt hinter einem Slash nicht ausdrücklich übereinstimmen muß.

if *list* **then** *list* [**elif** *list* **then** *list*] ... [**else** *list*] **fi**
Die *list* nach **if** wird ausgeführt und, wenn sie einen Null-Ende-Status zurückgibt, wird *list* nach dem ersten **then** ausgeführt. Andernfalls wird die *list* nach **elif** ausgeführt, und wenn der Wert Null ist, wird die *list* nach dem nächsten **then** ausgeführt. Wenn dies nicht gelingt, wird die **else** *list* ausgeführt. Wenn keine **else** *list* oder **then** *list* ausgeführt wird, gibt das **if**-Kommando einen Null-Ende-Status zurück.

while *list* **do** *list* **done**
Ein **while**-Kommando führt die **while** *list* wiederholt aus, und wenn der Endestatus des letzten Kommandos in *list* Null ist, führt das Kommando die **do** *list* aus; andernfalls endet die Schleife. Wenn keine Kommandos in der **do** *list* ausgeführt werden, gibt das **while**-Kommando einen Null-Ende-Status zurück; **until** kann anstelle von **while** zur Negation des Schleifenabbruchtests verwendet werden.

(*list*)
list ist in einer Sub-Shell auszuführen.

{list;}
list wird in der aktuellen Shell ausgeführt.

name () *{list;}*
Definiert eine Funktion, die mit *name* bezeichnet wird. Der Hauptteil der Funktion ist die *list* der Kommandos zwischen { und }. Ausführung der Funktion wird nachstehend erläutert (vgl. " Ausführung ").

Die folgenden Worte werden nur erkannt, wenn sie das erste Wort in einem Kommando sind und nicht quotiert sind.

if then else elif fi case esac for while until do done { }

Kommentare
Ein Wort, das mit # anfängt, bewirkt, daß dieses Wort und alle folgenden Zeichen bis zu einem New-Line-Zeichen ignoriert werden.

Kommando-Substitution
Die Shell liest Kommandos aus der Zeichenfolge zwischen zwei Akzenten (Gravis) (` `), und die Standardausgabe von diesen Kommandos kann als ganzes oder Teil eines Worts verwendet werden. Nachgestellte New-Line-Zeichen von der Standardausgabe werden entfernt.

Die Zeichenfolge wird nicht interpretiert, bevor sie gelesen wird; es werden nur Backslashes (\), die zur Codierung von Sonderzeichen verwendet werden, entfernt. Backslashes können zur Codierung eines Gravis (`) oder eines anderen Backslashs (\) verwendet werden und werden vor Lesen der Kommandofolge entfernt. Codierung von Gravis ermöglicht verschachtelte Kommando-Substitution. Falls die Kommando-Substitution zwischen Anführungszeichen (" ...` ...` ... ") steht, wird ein Backslash, der ein Anführungszeichen codiert (\"), entfernt; andernfalls bleibt er erhalten.

Wenn ein Backslash zur Codierung eines New-Line-Zeichens verwendet wird (**\new-line**), werden der Backslash und das New-Line-Zeichen entfernt (vgl. nachstehendes Kapitel über "Quotierung"). Außerdem werden Backslashes, die zur Codierung von Dollarzeichen (\$) verwendet wurden, entfernt. Da vor dem Lesen der Kommandofolge keine Interpretation erfolgt, ist das Einfügen eines Backslashs zur Quotierung eines Dollarzeichens nicht wirksam. Backslashes, die vor anderen Zeichen als \, `, **new-line** und $ stehen, bleiben wirksam, wenn die Kommandofolge gelesen wird.

Parameter-Substitution

Das Zeichen $ wird verwendet, um ersetztbare Parameter *(parameters)* einzuführen. Es gibt zwei Parameter-Typen: Stellungsparameter und Kennwortparameter. Wenn *parameter* eine Ziffer ist, ist es ein Stellungsparameter. Mit **set** können Stellungsparametern Werte zugewiesen werden. Kennwortparametern (auch als Variablen bekannt) können Werte wie folgt zugewiesen werden:

name = *value* [*name* = *value*] ...

Mustersuche wird bei *value* nicht durchgeführt. Es darf keine Funktion und Variable mit gleichem *name* geben.

${*parameter*}
Der Wert von *parameter* wird, falls vorhanden, eingesetzt. Die geschweiften Klammern sind nur erforderlich, wenn auf *parameter* ein Buchstabe, eine Ziffer oder ein Unterstrich folgt, der nicht als Teil des Namens interpretiert werden soll. Wenn *parameter* ein * oder @ ist, werden alle Stellungsparameter, angefangen mit **$1**, ersetzt (getrennt durch Leerzeichen). Parameter **$0** wird mit Argument 0 belegt, wenn die Shell aufgerufen wird.

${*parameter*:-*word*}
Wenn *parameter* gesetzt und nicht Null ist, wird sein Wert eingesetzt; andernfalls *word*.

${*parameter*:=*word*}
Wenn *parameter* nicht gesetzt oder Null ist, wird er auf *word* gesetzt; der Wert des Parameters wird eingesetzt. Stellungsparametern können nicht auf diese Weise Werte zugewiesen werden.

${*parameter*:?*word*}
Wenn *parameter* gesetzt und nicht Null ist, wird sein Wert eingesetzt; andernfalls wird *word* ausgegeben und die Shell beendet. Wenn *word* nicht angegeben ist, erfolgt die Nachricht: "parameter null or not set".

${*parameter*:+*word*}
Wenn *parameter* gesetzt und nicht-Null ist, wird *word* eingesetzt; andernfalls wird nichts eingesetzt.

In den oben beschriebenen Fällen wird *word* nur ausgewertet, wenn es als ersetzte Zeichenfolge benutzt wird; im folgenden Beispiel wird **pwd** nur ausgeführt, wenn **d** nicht gesetzt oder Null ist:

```
echo ${d:-`pwd`}
```

Wenn der Doppelpunkt (:) bei den obigen Ausdrücken fehlt, prüft die Shell nur, ob *parameter* gesetzt ist oder nicht.

Die folgenden Parameter werden automatisch von der Shell gesetzt:

Die Anzahl der Stellungsparameter als Dezimalzahl.

- An die Shell bei Aufruf oder durch das Kommando **set** übergebene Schalter.

? Der Dezimalwert, der vom letzten synchron ausgeführten Kommando zurückgegeben wurde.

$ Die Prozeß-Nummer dieser Shell.

! Die Prozeß-Nummer des letzten aufgerufenen Hintergrundkommandos.

Die folgenden Parameter werden von der Shell verwendet:

HOME
Das Standardargument (Home-Verzeichnis) für das Kommando *cd*

PATH
Der Suchpfad für Kommandos (vgl. nachstehende " Ausführung "). Der Benutzer kann **PATH** bei *rsh* nicht ändern.

CDPATH
Der Suchpfad für das Kommando *cd*.

MAIL Wenn dieser Parameter auf den Namen einer Postdatei gesetzt ist und der **MAILPATH** Parameter ist nicht gesetzt, informiert die Shell den Benutzer über die Ankunft von Post in der angegebenen Datei.

MAILCHECK
Dieser Parameter gibt an, wie oft (in Sekunden) die Shell die Ankunft von Post in den von den Parametern **MAILPATH** oder **MAIL** bezeichneten Dateien prüft. Der Standardwert ist 600 Sekunden (10 Minuten). Wenn auf 0 gesetzt, führt die Shell vor jeder Eingabe-Aufforderung eine Prüfung durch.

MAILPATH
Eine durch Doppelpunkt (:) getrennte Dateinamensliste. Wenn dieser Parameter gesetzt ist, informiert die Shell den Benutzer über die Ankunft von Post in jeder der angegebenen Dateien. Nach jedem Dateinamen kann ein % und eine Nachricht stehen, die ausgegeben wird, wenn die Modifikationzeit geändert wird. Die Standardmeldung ist *you have mail*.

PS1 Erste Eingabe-Aufforderungsfolge, standardmäßig "$ ".

PS2 Zweite Eingabe-Aufforderungsfolge, standardmäßig "> ".

IFS Interne Feld-Trennsymbole, normalerweise **space**, **tab** und **new-line**.

SHACCT,
Wenn dieser Parameter auf den Namen einer Datei gesetzt ist, die vom Benutzer beschreibbar ist, trägt die Shell für jede ausgeführte Shell-Prozedur einen Abrechnungssatz in der Datei ein.

SHELL
Wenn die Shell aufgerufen wird, durchsucht sie die Umgebung (vgl. "Umgebung') nach diesem Namen. Wenn der Name gefunden wird und 'rsh' der Dateiname ist, wird die Shell eine eingeschränkte Shell.

Die Shell setzt Standardwerte für **PATH, PS1, PS2, MAILCHECK** und **IFS**. **HOME** und **MAIL** werden vom *login(1)* gesetzt.

Leerzeichen-Interpretation
Nach Parameter- und Kommando-Substitution werden die Ergebnisse der Substitutionen auf interne Feld-Trennsymbole geprüft (die in **IFS**) und in verschiedene Argumente aufgeteilt, wenn solche Zeichen gefunden werden. Ausdrückliche Null-Argumente (**""** oder **''**) werden beibehalten. Implizite Null-Argumente (die von *parameter* ohne Wert stammen) werden entfernt.

Ein-/Ausgabe
Eine Kommandoeingabe und -ausgabe kann mit Hilfe einer besonderen Schreibweise, die von der Shell interpretiert wird, umgelenkt werden. Folgende Schreibweisen können beliebig in einem *simple-command* vorkommen oder einem *command* vorausgehen und werden *nicht* als Argumente an das aufgerufene Kommando weitergeleitet. Es ist zu beachten, daß Parameter- und Kommando-Substitution stattfindet bevor *word* oder *digit* verwendet werden.

<word	Verwenden von Datei *word* als Standardeingabe (Dateikennzahl 0).
>word	Verwenden von Datei *word* als Standardausgabe (Dateikennzahl 1). Wenn die Datei nicht existiert, wird sie erstellt; andernfalls wird sie auf Länge Null abgeschnitten.
>>word	Verwenden von Datei *word* als Standardausgabe. Wenn die Datei existiert, wird Ausgabe angehängt (beim ersten Durchsuchen bis zum Dateiende); andernfalls wird die Datei erstellt.
<<[-]word	Nach Ausführung von Parameter- und Kommando-Substitution bei *word* wird die Shell-Eingabe bis zur ersten Zeile gelesen, die mit dem sich ergebenden *word* übereinstimmt, oder bis zum Dateiende. Bei Angabe von << gilt:

1) führende Tabulatorzeichen von *word* werden entfernt, ehe die Shell-Eingabe gelesen wird (aber nachdem Parameter- und Kommando-Substitution bei *word* ausgeführt ist).
2) führende Tabulatorzeichen werden von der Shell-Eingabe entfernt, wenn diese gelesen wird und bevor jede Zeile mit *word* verglichen wird.
3) die Shell-Eingabe wird bis zur ersten Zeile gelesen, die mit dem sich ergebenden *word* übereinstimmt, oder bis zum Dateiende.

Wenn Zeichen von *word* quotiert sind (vgl. "Quotierung" weiter unten), wird die Shell-Eingabe nicht weiter bearbeitet. Wenn keine Zeichen bei *word* quotiert sind:

1) wird Parameter- und Kommando-Substitution vorgenommen,
2) wird (ein quotiertes) New-Line-Zeichen **\new-line** ignoriert,
3) muß \ zur Quotierung der Zeichen \, $ und ` verwendet werden.

Das entstandene Dokument wird die Standardeingabe.

< &digit Verwendet als Standardeingabe die Datei, zu der die Dateikennzahl *digit* gehört. Das gleiche gilt für die Standardausgabe bei der Verwendung von **> &digit.**

<&– Die Standardeingabe wird geschlossen. Das gleiche gilt für die Standardausgabe bei der Verwendung von >&–.

Wenn vor diesen Zeichen eine Ziffer steht, wird die Dateikennzahl der entsprechenden Datei durch diese Ziffer angegeben (statt durch die Standardeinstellung 0 oder 1). Zum Beispiel wird bei:

... 2>&1

die Dateikennzahl 2 der Datei zugeordnet, die momentan die Dateikennzahl 1 hat.

Die Reihenfolge, in der die Umlenkungen angegeben werden, ist signifikant. Die Shell wertet Umlenkungen von links nach rechts aus. Zum Beispiel wird bei:

... 1>*xxx* 2>&1

die Dateikennzahl 1 zunächst der Datei *xxx* zugeordnet. Die Dateikennzahl 2 wird der Datei zugeteilt, die die Dateikennzahl 1 hat (d. h. *xxx*). Wenn die Reihenfolge der Umlenkung umgekehrt wäre, würde die Dateikennzahl 2 dem Bildschirm zugeordnet (vorausgesetzt, die Dateikennzahl 1 war es) und die Dateikennzahl 1 würde der Datei *xxx* zugeordnet werden.

Wir verwenden den Begriff "Kommandos" wie auf Seite 1. Falls ein Kommando *command* aus mehreren *simple-commands* besteht, wird die Umlenkung für das ganze Kommando *command* vor der Auswertung jedes *simple-command* ausgewertet. Das heißt, die Shell wertet die Umlenkung für die gesamte *list* aus, jede Pipeline innerhalb der *list*, jedes *command* innerhalb jeder Pipeline , jede *list* innerhalb jedes *command*.

Wenn nach einem Kommando ein & steht, ist die voreingestellte Standardeingabe für dieses Kommando die leere Datei **/dev/null**. Andernfalls enthält die Umgebung für die Ausführung eines Kommandos die Dateikennzahlen der aufrufenden Shell, entsprechend den Änderungen der Eingabe-/Ausgabeangaben.

Bei der eingeschränkten Shell ist die Umlenkung der Ausgabe nicht erlaubt.

Erzeugung von Dateinamen

Ehe ein Kommando ausgeführt wird, wird jedes Wort *word* im Kommandoaufruf nach den Zeichen *, ? und [durchsucht. Wenn eines dieser Zeichen vorkommt, wird das Wort als ein Muster *pattern* angesehen. Das Wort wird durch alphabetisch sortierte Dateinamen, die zu dem Muster passen, ersetzt. Wenn kein Dateiname gefunden wird, der dem Muster entspricht, bleibt das Wort unverändert. Für das Zeichen . am Anfang eines Dateinamens oder direkt nach einem / sowie das Zeichen / selbst muß eine genaue Übereinstimmung gefunden werden.

* Paßt zu allen Zeichenfolgen, einschließlich der Null-Zeichenfolge.

? Paßt zu jedem einzelnen Zeichen.

[...] Paßt zu jedem der eingebundenen Zeichen. Ein Zeichenpaar, das durch - getrennt ist, paßt zu jedem Zeichen, das lexikalisch zwischen dem Paar (inklusive) liegt. Falls das erste Zeichen nach der öffnenden Klammer "[" ein "!" ist, paßt jedes nicht eingebundene Zeichen.

Quotierung

Die nachstehenden Zeichen haben bei der Shell eine besondere Bedeutung und beenden ein Wort, wenn sie nicht quotiert sind:

; & ( ) | ^ < > new-line space tab

Ein Zeichen kann quotiert werden (d. h. es soll für sich selbst stehen), indem ein Backslash (\) vorangestellt wird oder durch Einschließen in Anführungszeichen (´´ oder ""). Während der Verarbeitung kann die Shell gewisse Zeichen quotieren, damit diese nicht eine besondere Bedeutung annehmen. Backslashes, die ein Einzelzeichen quotieren, werden vor der Kommandoausführung aus dem Wort entfernt. Das Zei-

chenpaar **\new-line** wird bei Kommando- und Parameter-Substitution aus dem Wort entfernt.

Alle Zeichen, die in Einzelanführungszeichen stehen (''), werden durch die Shell quotiert, außer einem einfachen Anführungszeichen. Der Backslash hat innerhalb von Einzelanführungszeichen keine besondere Bedeutung. Ein Einzelanführungszeichen kann in Anführungszeichen stehen (beispielsweise "'").

Innerhalb von Anführungszeichen ("") werden Parameter- und Kommando-Substitution vorgenommen, und die Shell quotiert diese, um Interpretation von Leerzeichen und Erweiterung von Dateinamen zu vermeiden. Falls **$*** innerhalb von Anführungszeichen steht, werden die Stellungsparameter ersetzt und quotiert und durch quotierte Leerzeichen getrennt. ("**$1 $2** ..."); wenn jedoch $@ innerhalb Anführungzeichen steht, werden die Stellungsparameter ersetzt und quotiert und durch nichtquotierte Leerzeichen getrennt ("**$1**" "**$2**" ...). \ quotiert die Zeichen \, `, " und **$**. Das Zeichenpaar **\new-line** wird vor der Parameter- und Kommando-Substitution ersetzt. Wenn ein Backslash vor anderen Zeichen als \, `, " $ und New-Line-Zeichen steht, wird der Backslash selbst von der Shell quotiert.

Eingabe-Aufforderung

Bei interaktivem Gebrauch gibt die Shell den Wert von **PS1** aus, bevor ein Kommando eingelesen wird. Wenn zu einem Zeitpunkt ein New-Line-Zeichen eingegeben wird und mehr Eingabe zur Vervollständigung des Kommandos erforderlich ist, wird ein sekundäres Eingabe-Aufforderungszeichen ausgegeben (d. h. der Wert von **PS2**).

Umgebung

Die Umgebung *environment* (vgl. *environ*(5)) ist eine Liste mit Paaren (Name=Wert), die genau wie eine normale Argumentliste an das ausführende Programm übergeben wird. Die Shell steht mit der Umgebung auf verschiedene Weise im Dialog. Bei Aufruf fragt die Shell die Umgebung ab und erzeugt einen Parameter für jeden gefundenen Namen, und weist ihm den entsprechenden Wert zu. Wenn der Benutzer den Wert eines dieser Parameter modifiziert oder neue Parameter erstellt, wirkt es sich nicht auf die Umgebung aus, sofern nicht das Kommando **export** zum Einbinden der Shell-Parameter in die Umgebung verwendet wird (vgl. auch **set -a**). Ein Parameter kann mit dem Kommando **unset** aus der Umgebung entfernt werden. Die von jedem ausgeführten Kommando gesehene Umgebung besteht also aus nicht modifizierten Paaren (Name=Wert), die ursprünglich von der Shell stammten, abzüglich der Paare, die durch **unset** entfernt wurden, zuzüglich der Änderungen oder Hinzufügungen, die in **export** Kommandos angegeben wurden.

Die Umgebung für jedes *simple-command* kann erweitert werden, indem eine oder mehrere Zuweisungen an Parameter vorangestellt werden. So sind

```
TERM=450 cmd
und
(export TERM; TERM=450; cmd)
```

gleichwertig (was die Ausführung von *cmd* betrifft).

Wenn der Schalter **-k** gesetzt ist, werden *alle* Kennwortparameter in die Umgebung aufgenommen, selbst wenn sie nach einem Kommandonamen auftreten. Hier wird **a=b c** und **c** ausgegeben:

```
echo a=b c
set -k
echo a=b c
```

Signale

Bei einem aufgerufenen Kommando werden die Signale INTERRUPT und QUIT ignoriert, wenn dem Kommando ein & folgt; andernfalls haben Signale die Werte, die die Shell von ihrem Vater erbt; eine Ausnahme ist Signal 11 (nachstehend ist jedoch auch das Kommando **trap** zu beachten).

Ausführung

Jedesmal, wenn ein Kommando ausgeführt wird, werden die oben erwähnten Substitutionen durchgeführt. Wenn der Kommandoname mit einem der nachstehend aufgeführten Sonderkommandos *Special Commands* übereinstimmt, wird dieses in demselben Shell-Prozeß ausgeführt. Wenn der Kommandoname nicht mit einem *Special Command* übereinstimmt, jedoch mit dem Namen einer definierten Funktion identisch ist, wird die Funktion im Shell-Prozeß ausgeführt (der Unterschied zur Ausführung von Shell-Prozeduren ist hier zu beachten). Die Stellungsparameter **$1, $2,** .... werden als Argumente der Funktion gesetzt. Wenn der Kommandoname weder mit einem *Special Command* noch mit dem Namen einer definierten Funktion übereinstimmt, wird ein neuer Prozeß erzeugt; es wird dann versucht, das Kommando über *exec*(2) auszuführen.

Der Shell-Parameter **PATH** definiert den Suchpfad für das Verzeichnis, das das Kommando enthält. Alternative Verzeichnisnamen werden durch einen Doppelpunkt (:) getrennt. Der Standardpfad ist **:/bin:/usr/bin** (bezeichnet das aktuelle Verzeichnis, **/bin** und **/usr/bin** in der angegebenen Reihenfolge). Es ist zu beachten, daß das aktuelle Verzeichnis durch einen leeren Pfadnamen angegeben wird, der direkt hinter dem Gleichheitszeichen stehen kann, oder mitten in der Liste zwischen zwei Doppelpunkt-Begrenzern oder am Ende der Liste. Wenn der Kommandoname ein / enthält, wird der Suchpfad nicht verwendet;

solche Kommandos werden nicht von der eingeschränkten Shell ausgeführt. Andernfalls wird jedes Verzeichnis in dem Pfad nach einer ausführbaren Datei durchsucht. Wenn die Datei Ausführungsberechtigung hat, aber keine **a.out** -Datei ist, wird angenommen, daß sie Shell-Kommandos enthält. Eine Sub-Shell wird zum Lesen erzeugt. Ein in runden Klammern stehendes Kommando wird auch in einer Sub-Shell ausgeführt.

Die Shell vermerkt die Position im Suchpfad, an der ein Kommando gefunden wurde (dadurch werden unnötige *exec*-Aufrufe zu einem späteren Zeitpunkt verhindert). Wenn das Kommando in einem relativen Verzeichnis gefunden wurde, muß die Position immer wieder neu bestimmt werden, sobald das aktuelle Verzeichnis sich ändert. Die Shell vergißt alle gemerkten Positionen, wenn die Variable **PATH** verändert oder das Kommando **hash −r** ausgeführt wird (siehe unten).

Sonderkommandos

Bei diesen Kommandos ist die Umlenkung von Eingabe/Ausgabe erlaubt. Die Dateikennzahl 1 ist die voreingestellte Standardausgabe.

: Keine Wirkung; ein Null-Endestatus wird geliefert.

. *file* Liest und führt Kommandos von *file* aus und kehrt dann zurück. Der durch **PATH** angegebene Suchpfad wird zum Auffinden des Verzeichnisses, in dem *file* enthalten ist, eingesetzt.

break [*n*]
Ausstieg aus der umschließenden **for** oder **while** Schleife, falls vorhanden. Falls *n* angegeben ist, werden *n* Stufen abgebrochen.

continue [*n*]
Wiederholt die umschließende **for** oder **while** Schleife. Wenn *n* angegeben ist, wird bei der *n*-ten umschließenden Schleife fortgefahren.

cd [*arg*]
Andert das aktuelle Verzeichnis auf *arg*. Der Shell-Parameter **HOME** ist das Standardargument *arg*. Der Shell-Parameter **CDPATH** definiert den Suchpfad für das Verzeichnis, das *arg* enthält. Alternative Verzeichnisnamen werden durch einen Doppelpunkt (:) getrennt. Der Standardpfad ist **<null>** (gibt das aktuelle Verzeichnis an). Es ist zu beachten, daß das aktuelle Verzeichnis durch einen leeren Pfadnamen angegeben ist, der direkt hinter dem Gleichheitszeichen oder zwischen den Doppelpunkt-Begrenzern überall in der Pfadliste stehen kann. Wenn *arg* mit einem / beginnt, wird der Suchpfad nicht verwendet. Andernfalls wird jedes Verzeichnis im Pfad nach *arg* durchsucht. Das Kommando *cd* kann nicht von *rsh* ausgeführt werden.

echo [*arg* ...]
Argumente ausgeben. Vgl. *echo*(1) bezüglich Einsatz und Beschreibung.

eval [*arg* ...]
Die Argumente werden als Eingabe für die Shell gelesen und die resultierenden Kommandos werden ausgeführt.

exec [*arg* ...]
Das durch die Argumente angegebene Kommando wird anstelle dieser Shell ohne Erzeugung eines neuen Prozesses ausgeführt. Ein-/Ausgabe-Argumente können vorkommen und bewirken die Modifizierung der Shell-Ein-/Ausgabe, falls keine anderen Argumente angegeben sind.

exit [*n*]
Bewirkt, daß eine Shell mit dem durch *n* angegebenen Endestatus abbricht. Wenn *n* ausgelassen ist, ist der Endestatus der des letzten ausgeführten Kommandos (ein Dateiende kann auch zur Beendigung der Shell führen.)

export [*name* ...]
Die angegebenen Namen *name*s werden automatisch in die Umgebung *environment* der nachfolgend ausgeführten Kommandos exportiert. Wenn keine Argumente angegeben sind, werden die Variablennamen, die für den Export während der aktuellen Shell vorgemerkt wurden, aufgelistet. (Von einer Vater-Shell exportierte Variablennamen werden nur aufgeführt, wenn sie in der aktuellen Shell wieder exportiert wurden.) Funktionsnamen werden *nicht* exportiert.

getopts
Wird in Shell-Skripts zur Unterstützung der Standard-Kommando-Syntaxregeln (vgl. *intro*(1)) verwendet; es analysiert Stellungsparameter und prüft Optionen auf Korrektheit. Vgl. *getopts* (1) bezüglich Gebrauch und Beschreibung.

hash [**-r**] [*name* ...]
Für jeden *name* wird die Position des mit *name* bezeichneten Kommandos bestimmt und von der Shell gemerkt. Die Option **-r** bewirkt, daß die Shell alle gemerkten Positionen vergißt. Wenn keine Argumente angegeben sind, werden Informationen über die gemerkten Kommandos ausgegeben. *Hits* ist die Anzahl der Aufrufe eines Kommandos durch den Shell-Prozeß. *cost* ist ein Maß für die Arbeit, die zum Auffinden eines Kommandos in dem Suchpfad erforderlich ist. Wenn ein Kommando in einem "relativen" Verzeichnis im Suchpfad gefunden wird, wird nach Wechsel in dieses Verzeichnis die gespeicherte Position dieses Kommandos neu berechnet. Kommandos, bei denen dies durchgeführt

wird, werden mit einem Sternchen (*) neben den Daten für *hits* gekennzeichnet. *Cost* wird nach der Neuberechnung erhöht.

newgrp [*arg* ...]
Gleichwertig mit **exec newgrp** *arg* .... Vgl. *newgrp*(1) bezüglich Gebrauch und Beschreibung.

pwd Gibt das aktuelle Verzeichnis an. Vgl. *pwd*(1) bezüglich Gebrauch und Beschreibung.

read [*name* ...]
Eine Zeile wird von der Standardeingabe gelesen und mit Hilfe der internen Feldtrennung **IFS** (normalerweise Leerzeichen oder Tabulatorzeichen) zur Abgrenzung von Worten benutzt. Das erste Wort wird dem ersten *name* zugewiesen, das zweite Wort dem zweiten *name*, usw., wobei übrig gebliebene Worte dem letzten *name* zugewiesen werden. Zeilen können mit **\new-line** fortgesetzt werden. Die im Abschnitt Quotierung aufgeführten Sonderzeichen können durch einen vorangestellten Backslash quotiert werden. Diese Backslashes werden entfernt, ehe Worte an *names* zugewiesen werden; das Zeichen, das dem Backslash folgt, wird nicht interpretiert. Der Rückgabecode ist 0, sofern nicht ein Dateiende auftritt.

readonly [*name* ...]
Die angegebenen *name*s werden als *readonly* markiert und die Werte dieser *name*s können durch eine nachträgliche Zuweisung nicht geändert werden. Wenn keine Argumente angegeben sind, wird eine Liste aller *readonly* Namen ausgegeben.

return [*n*]
Bewirkt die Beendigung einer Funktion mit dem durch *n* angegebenen Rückgabecode. Wenn *n* ausgelassen wird, ist der Rückgabecode der des zuletzt ausgeführten Kommandos.

set [**– –aefhkntuvx** [*arg* ...]]

–a Kennzeichnet Variable, die modifiziert oder für Export erzeugt werden.

–e Bricht sofort ab, falls ein Kommando mit einem Nicht-Null-Endestatus endet.

–f Schaltet die Dateinamenserzeugung ab.

–h Lokalisiert Funktionen und vermerkt sie als Funktionskommandos, wenn sie definiert werden. (Funktionskommandos werden normalerweise lokalisiert, wenn die Funktion ausgeführt wird).

–k Alle und nicht nur die vor einem Kommandonamen stehenden Kennwortparameter werden in die Umgebung eines Kommandos aufgenommen.

-n Liest Kommandos, führt sie aber nicht aus.

-t Beenden nnach dem Lesen und Ausführen eines Kommandos.

-u Behandelt nicht gesetzte Variablen bei der Substitution als Fehler.

-v Shell-Eingabezeilen ausgeben, wenn sie eingelesen werden.

-x Gibt Kommandos und deren Argumente aus, wenn sie ausgeführt werden.

-- Keine Schalter ändern; nützlich beim Setzen von **$1** auf -.

Die Verwendung von + anstelle von - bewirkt, daß diese Schalter ausgeschaltet werden. Diese Schalter können auch beim Aufruf der Shell verwendet werden. Die aktuell gesetzten Schalter können in **$-** gefunden werden. Die restlichen Argumente sind Stellungsparameter und werden der Reihe nach **$1, $2,** .... zugewiesen. Wenn keine Argumente angegeben sind, werden die Werte aller Namen ausgegeben.

shift [*n*]
Die Stellungsparameter **$n+1** ... werden auf **$1** .... umbenannt. Wenn *n* nicht angegeben ist, wird 1 angenommen.

test
Wertet bedingte Ausdrücke aus. Vgl. *test*(1) hinsichtlich Gebrauch und Beschreibung.

times
Gibt die aufgelaufenen Benutzer- und Systemzeiten bei Prozessen aus, die von der Shell erzeugt werden.

trap [*arg*] [*n*] ...
Das Kommando *arg* muß gelesen und ausgeführt werden, wenn die Shell das (die) Signal(e) *n* empfängt. (Es ist zu beachten, daß *arg* einmal beim Setzen von trap und einmal beim Abfangen des Signals analysiert wird.) trap-Kommandos werden in der Reihenfolge der Signalnummern ausgeführt. Ein Setzen von trap für ein Signal, das bei Aufruf der aktuellen Shell ignoriert wurde, ist wirkungslos. Ein Versuch, trap für Signal 11 (Speicherplatzfehler) zu setzen, erzeugt einen Fehler. Wenn kein *arg* vorhanden ist, werden die traps für alle *n* auf die Originalwerte zurückgestellt. Wenn *arg* die Null-Zeichenfolge ist, wird dieses Signal von der Shell und von den Kommandos, die die Shell aufruft, ignoriert. Wenn *n* 0 ist, wird das Kommando *arg* bei Beendigung der Shell ausgeführt. Das Kommando **trap** ohne Argumente gibt eine Liste der jeder Signalnummer zugeordneten Kommandos aus.

type [*name* ...]
Bei jedem *name* wird angegeben, wie er interpretiert würde, wenn er als ein Kommandoname verwendet würde.

ulimit [*n*]
setzt einen Maximum von *n* Blöcken für Dateien fest, die von der Shell und den Sohnprozessen geschrieben werden (Dateien beliebiger Größe können gelesen werden). Wenn *n* ausgelassen ist, wird der aktuelle Grenzwert ausgegeben. Sie können Ihren eigenen Grenzwert reduzieren, jedoch nur der Systemverwalter darf einen Grenzwert erhöhen (vgl. *su*(1M)).

umask [*nnn*]
Die Dateierzeugungs-Maske des Benutzers wird auf *nnn* gesetzt (vgl. *umask*(1)). Wenn *nnn* ausgelassen wird, wird der aktuelle Wert der Maske ausgegeben.

unset [*name* ...]
Für jeden *name* wird die entsprechende Variable oder Funktion entfernt. Bei den Variablen **PATH, PS1, PS2, MAILCHECK** and **IFS** können die Einstellungen nicht rückgängig gemacht werden.

wait [*n*]
Wartet auf Ihren Hintergrundprozeß, dessen Prozeßnummer *n* ist und gibt den Endestatus an. Wenn *n* ausgelassen wird, wird auf die Beendigung von allen momentan aktiven Hintergrundprozessen von Ihrer Shell gewartet. Der Rückgabecode ist Null.

Aufrufen

Wenn die Shell durch *exec*(2) aufgerufen wird und das erste Zeichen des 0-ten Arguments - ist, werden die Kommandos zunächst von */etc/profile* und von *$HOME/.profile* gelesen, falls diese Dateien vorhanden sind. Danach werden Kommandos entsprechend der nachstehenden Beschreibung gelesen. Dies gilt auch für den Fall, wenn die Shell als */bin/sh* aufgerufen wird. Die nachstehenden Optionen werden von der Shell nur bei Aufruf interpretiert. Es ist zu beachten, daß, sofern nicht die Optionen **-c** oder **-s** angegeben sind, das erste Argument als der Name einer Datei angenommen wird, die Kommandos enthält, und die restlichen Argumente als Stellungsparameter für diese Kommandodatei angesehen werden:

-c *string* Wenn die Option **-c** vorhanden ist, werden die Kommandos aus *string* gelesen.

-s Wenn die Option **-s** vorhanden ist, oder wenn keine Argumente übrigbleiben, werden die Kommandos aus der Standardeingabe gelesen. Übriggebliebene Argumente geben die Stellungsparameter an. Die Shell-Ausgabe (außer bei *Sonderkommandos*) wird auf Dateikennzahl 2 geschrieben.

-i Wenn die Option **-i** vorhanden ist, oder wenn die Shell-Eingabe und -Ausgabe mit einem Terminal verbunden sind, ist diese Shell interaktiv. In diesem Fall wird TERMINATE ignoriert (so daß **kill 0** nicht eine interaktive Shell abbricht) und INTERRUPT wird abgefangen und ignoriert (so, daß **wait** unterbrechbar ist). In beiden Fällen wird QUIT von der Shell ignoriert.

-r Wenn die Option **-r** vorhanden ist, ist die Shell eine eingeschränkte Shell.

Die restlichen Optionen und Argumente sind bei dem Kommando **set** beschrieben.

rsh

rsh wird zum Setzen von Login-Namen und Ausführungsumgebungen verwendet, deren Möglichkeiten eingeschränkter sind als die der Standard-Shell. Die Aktionen von *rsh* sind identisch mit denen von *sh*, folgende Funktionen sind jedoch nicht erlaubt:

Änderung des Verzeichnisses (vgl. *cd*(1)),
Wert von **$PATH** setzen,
Pfad- oder Kommandonamen angeben, die / enthalten,
Ausgabe umlenken (**>** und **>>**).

Die obigen Einschränkungen treten in Kraft, nachdem die Datei *.profile* interpretiert wurde.

Eine eingeschränkte Shell kann wie folgt aufgerufen werden: (1) *rsh* steht im Dateinamen des letzten Eintrags in der Datei */etc/passwd* (vgl. *passwd*(4)); (2) die Umgebungsvariable **SHELL** existiert und *rsh* steht im Dateinamen-Teil des entsprechenden Werts; (3) die Shell ist aufgerufen und *rsh* ist der Dateiname von Argument 0; (4) die Shell wird mit der Option **-r** aufgerufen.

Wenn ein auszuführendes Kommando eine Shell-Prozedur ist, ruft *rsh* *sh* auf, um sie auszuführen. Somit können dem Benutzer Shell-Prozeduren zur Verfügung gestellt werden, die voll auf die Standard-Shell zugreifen können, wobei dem Benutzer nur eine beschränkte Auswahl von Kommandos bereitsteht; hierbei wird vorausgesetzt, daß der Endbenutzer in demselben Verzeichnis keine Schreib- und Ausführberechtigung besitzt.

Der Sinn dieser Regeln ist, daß der Autor von *.profile* (vgl. *profile*(4)) vollständige Kontrolle über die Benutzeraktionen hat, indem er kontrollierte Initialisierungen ausführt und dann den Benutzer in ein geeignetes Verzeichnis setzt (höchstwahrscheinlich *nicht* das Login-Verzeichnis).

Der Systemverwalter richtet häufig ein Verzeichnis mit Kommandos ein (d. h. **/usr/rbin**), das gefahrlos von einer eingeschränkten Shell aufgerufen werden kann. Einige Systeme stellen auch einen eingeschränkten Editor *red* zur Verfügung.

ENDESTATUS

Von der Shell entdeckte Fehler wie zum Beispiel Syntaxfehler bewirken, daß die Shell einen Nicht-Null-Endestatus liefert. Wenn die Shell nicht interaktiv verwendet wird, wird die Ausführung der Shell-Datei abgebrochen. Andernfalls gibt die Shell den Endestatus des zuletzt ausgeführten Kommandos zurück (vgl. auch das oben erwähnte Kommando **exit**).

DATEIEN

/etc/profile
$HOME/.profile
/tmp/sh*
/dev/null

SIEHE AUCH:

cd(1), echo(1), env(1), getopts(1), intro(1), login(1), newgrp(1), pwd(1), test(1), umask(1), wait(1).
dup(2), exec(2), fork(2), pipe(2), profile(4), signal(2), ulimit(2) im *Programmer's Reference Manual.*

EINSCHRÄNKUNGEN

Worte, die als Dateinamen bei der Umlenkung der Ein-/Ausgabe verwendet werden, werden nicht nach den Regeln der Dateinamenserzeugung interpretiert (vgl. oben erwähnte "Dateinamenserzeugung"). Zum Beispiel erzeugt **cat file1 >a*** eine mit **a*** bezeichnete Datei.

Da Kommandos in Pipelines als separate Prozesse laufen, wirken sich in einer Pipeline gesetzte Variablen nicht auf die Vater-Shell aus.

Wenn Sie die Fehlermeldung, *cannot fork, too many processes*, erhalten, sollten Sie mit dem Kommando *wait* (1) Ihre Hintergrundprozesse aufräumen. Wenn dies nicht gelingt, ist die System-Prozeßtabelle wahrscheinlich voll oder Sie haben zu viele interaktive Vordergrundprozesse. (Die Anzahl der mit Ihrem Login verbundenen Prozeßnummern ist beschränkt, ebenso die Anzahl der Prozesse, die das System verwalten kann.)

FEHLER

Wenn ein Kommando ausgeführt wird und ein Kommando mit dem gleichen Namen in einem Verzeichnis im Suchpfad vor dem Verzeichnis, in dem das ursprüngliche Kommando gefunden wurde, installiert wird, führt *(exec)* die Shell das ursprüngliche Kommando weiter aus. Der Einsatz von **hash** korrigiert dies.

Wenn Sie das aktuelle Verzeichnis oder das darüberliegende umbenennen, kann es vorkommen, daß **pwd** nicht die richtige Antwort gibt. Verwendung des Kommandos **cd** mit dem kompletten Pfadnamen korrigiert dies.

Es ist zu beachten, daß alle Prozesse einer 3- oder mehrstufigen Pipeline Söhne der Shell sind, und deswegen nicht auf sie gewarten werden kann.

Bei *wait n*, wird, wenn *n* keine aktive Prozeßnummer ist, auf alle momentan aktiven Hintergrundprozesse Ihrer Shell gewartet, und der Rückgabecode ist Null.

BEZEICHNUNG

shl – Verwalter für Shell-Layer

ÜBERSICHT

shl

BESCHREIBUNG

shl erlaubt einem Benutzer, von einem einzigen Terminal mit mehr als einer Shell im Dialog zu stehen. Der Benutzer steuert diese Shells, die als *layers* (Shell-Fenster) bezeichnet werden mit nachstehend erläuterten Kommandos.

Das *current layer* ist das Layer, das Eingabe von der Tastatur empfangen kann. Andere Layer, die versuchen, von der Tastatur zu lesen, werden blockiert. Ausgabe von mehreren Layern wird auf dem Bildschirm vermischt ausgegeben. Um die Ausgabe eines Layers zu blockieren, wenn es nicht aktuell ist, kann die *stty* Option **loblk** innerhalb dieses Layers gesetzt werden.

Das *stty* Zeichen **swtch** (wenn NUL ist es auf ^Z gesetzt) wird verwendet, um von einem Layer die Kontrolle an *shl* zu übergeben. *shl* hat ein eigenes Eingabe-Aufforderungszeichen, **>>>**.

Ein *layer* ist eine Shell, die an ein virtuelles Terminal gebunden ist (**/dev/sxt???**). Das virtuelle Gerät kann wie ein reales Terminal manipuliert werden mit Hilfe von *stty* (1) und *ioctl* (2). Jedes Layer hat eine eigene Prozeß-Gruppennummer.

Definitionen

Ein Name *name* ist eine durch ein Leerzeichen, Tabulator oder New-Line-Zeichen begrenzte Zeichenfolge. Nur die ersten acht Zeichen sind signifikant. Die *names* (1) bis (7) können beim Erstellen eines Layers nicht verwendet werden. Sie werden von *shl* benutzt, wenn kein Name angegeben ist. Sie können bis zur Ziffer abgekürzt werden.

Kommandos

Die folgenden Kommandos können von der *shl* Eingabe-Aufforderungsebene angegeben werden. Beliebige eindeutige Präfixe werden akzeptiert.

create [*name*]

Erstellen eines mit *name* bezeichneten Layers und dieses zum aktuellen Layer machen. Wenn kein Argument angegeben ist, wird das Layer mit einem Namen der Form *(#)* erstellt, wobei # die letzte Ziffer des virtuellen Geräts ist, das an das Layer gebunden wurde. Die Shell Eingabe-Aufforderungsvariable **PS1** wird auf den Namen des Layers gesetzt, dem ein Leerzeichen folgt. Höchstens 7 Layers können erstellt werden.

block *name* [*name* ...]
Für jeden *name* wird die Ausgabe des entsprechenden Layers blockiert, wenn dies nicht das aktuelle Layer ist. Dies ist gleichwertig mit dem Setzen der *stty*-Option innerhalb des Layers.

delete *name* [*name* ...]
Für jeden *name* wird das entsprechende Layer gelöscht. An alle Prozesse der Prozeßgruppe des Layers wird das Signal SIGHUP geschickt (vgl. *signal*(2)).

help (oder **?**)
Gibt die Syntax der *shl* Kommandos aus.

layers [**-l**] [*name* ...]
Für jeden *name* wird dieser Layer-Name und die entsprechende Prozeßgruppe aufgelistet. Die Option **-l** erzeugt eine Liste, die ähnlich der Ausgabe von *ps*(1) ist. Wenn keine Argumente angegeben sind, werden diese Informationen für alle vorhandenen Layer ausgegeben.

resume [*name*]
Macht das mit *name* bezeichnete Layer zum aktuellen Layer. Wenn kein Argument angegeben ist, wird mit dem letzten aktuellen Layer fortgefahren.

toggle
Macht mit dem Layer weiter, das vor dem zuletzt aktuellen Layer aktuell war.

unblock *name* [*name* ...]
Bei den angegebenen *name*(s) wird die Ausgabe des entsprechenden Layers nicht blockiert, wenn es nicht das aktuelle ist. Dies ist gleichwertig mit dem Setzen der *stty*-Option **-loblk** innerhalb des Layers.

quit Beendet *shl*. An alle Layer wird das Signal SIGHUP geschickt.

name
Macht das mit *name* bezeichnete Layer zum aktuellen Layer.

DATEIEN

/dev/sxt???	Virtuelle Terminals
$SHELL	Variable, die den zu verwendenden Pfadnamen der Shell enthält (standardmäßig /bin/sh).

SIEHE AUCH:
sh(1), stty(1).
ioctl(2), signal(2) im *Programmer's Reference Manual*
sxt(7) im System *Administrator's Reference Manual*

BEZEICHNUNG

sleep – Ausführung vorübergehend anhalten

ÜBERSICHT

sleep time

BESCHREIBUNG

sleep hält die Ausführung für *time* Sekunden an. Es dient dazu, ein Kommando nach Ablauf einer gewissen Zeit auszuführen wie beispielsweise bei:

(sleep 105; *command*)&

oder ein Kommando in regelmäßigen Abständen immer wieder auszuführen wie bei:

```
while true
    do
    command
    sleep 37
    done
```

SIEHE AUCH

alarm(2), sleep(3c) im *Programmer's Reference Manual.*

BEZEICHNUNG

slip – Dämon für serielle Netzwerk-Schnittstellen

QUELLE

USENET

ÜBERSICHT

/etc/slip *localhost remotehost line* [*speed*]

BESCHREIBUNG

Der Dämonprozeß *slip* wird zur Installation der seriellen Schnittstelle (tty) für die TCP/IP-Protokollschicht benötigt.

Die Kommunikation zwischen je zwei Hosts muß durch *einen* ständig aktiven *slip*–Dämon sichergestellt werden. *Slip*–Dämonen werden beim Laden anhand des Netzwerkinitialisierungsskripts /etc/rc.local gestartet, sobald das System in den Multi-Usermodus übergeht.

Die folgenden Parameter müssen für *slip* bereitgestellt werden:

localhost	Host-Name des lokalen Rechners
remotehost	Host-Name des entfernten Rechners
line	Pfadname des RS232-Kanals, über den die beiden Rechner miteinander verbunden sind (z. B. /dev/tttyixx).
speed	Baud-Rate des o. a. Kanals (z. B. 4800 9800 ...). Der *speed*–Parameter kann weggelassen werden. In diesem Fall wird standardmäßig 9600 Baud angenommen.

Die Installation serieller Schnittstellen ist dem Superuser vorbehalten.

BEISPIEL

```
/etc/slip '/etc/netbin/hostname' rhost /dev/tttyi31 19200 &
```

DIAGNOSE

Außer den Standard-Fehlernummern der Systemaufrufe (und den zugehörigen Meldungen), gibt *slip* die folgenden Fehlermeldungen aus:

bad speed (falsche Geschwindigkeit) Die angegebene Baud-Rate ist für den angegebenen Kanal nicht zulässig.

Unknown remote host (unbekannter entfernter Host) Der Host ist in der Host-Konfigurationstabelle */etc/hosts* nicht enthalten.

unknown local host (unbekannter lokaler host) Der lokale Host-Name stimmt nicht mit dem Namen überein, den das Betriebssystem kennt.

ANMERKUNGEN

Diese Implementierung basiert auf der USENET-Public-Domain-Version von Rick Adams und Jim Mckie. Es kann keine Garantie dafür übernommen werden, daß alle Anwendungen, die unter den Standard-Schnittstellen lauffähig sind, auch unter dieser Schnittstelle funktionieren.

FEHLER

Wenn einer der beiden Rechner neu geladen wird, startet der *slip*-Dämon möglicherweise nicht erneut, so daß er von Hand aufgerufen werden muß (nach Abbruch des alten Dämons).

Das Kommando *talk* ist unter dieser Schnittstelle nicht lauffähig.

BEZEICHNUNG

sort – Dateien sortieren und/oder mischen

ÜBERSICHT

sort [**-cmu**] [**-o***output*] [**-y***kmem*] [**-z***recsz*] [**-dfiMnr**] [**-btx**] [**+***pos1* [**-***pos2*]] [*files*]

BESCHREIBUNG

sort liest die Zeilen der angegebenen Dateien und schreibt das Ergebnis auf die Standardausgabe. Die Standardeingabe wird gelesen, falls – als Dateiname verwendet wird oder wenn keine Eingabedateien angegeben sind.

Vergleiche werden aufgrund eines oder mehrerer Sortier-Schlüssel, die jeder Eingabezeile entnommen werden, durchgeführt. Standardmäßig ist die gesamte Zeile der Sortier-Schlüssel, und die Zeilen werden gemäß dem Maschinenzeichencode byte-weise lexikographisch geordnet.

Die folgenden Optionen verändern das Standardverhalten:

-c Kontrolliert, ob die Eingabedatei gemäß den Anordnungswünschen sortiert ist; es erfolgt keine Ausgabe, es sei denn, die Datei ist nicht wie gewünscht sortiert.

-m Mischt nur, die Eingabedateien sind bereits sortiert.

-u Unterdrückt alle Zeilen bis auf eine, wenn es sich um Zeilen mit gleichen Sortier-Schlüsseln handelt.

-o*output*

Das angegebene Argument ist der Name einer Ausgabedatei, die anstelle der Standardausgabe verwendet werden soll. Diese Datei kann dieselbe wie eine der Eingabedateien sein. Zwischen **-o** und *output* können wahlweise Leerzeichen stehen.

-y*kmem*

Für das Leistungsverhalten von *sort* ist entscheidend, wieviel Hauptspeicher von *sort* verwendet wird. Es ist Verschwendung, für das Sortieren einer kleinen Datei viel Hauptspeicher zur Verfügung zu stellen. Wenn diese Option ausgelassen wird, verwendet *sort* eine Standard-Speicherkapazität und, falls erforderlich, benutzt das Kommando mehr Speicherplatz. Wenn diese Option mit dem Wert *kmem* angegeben wird, verwendet *sort* diese Anzahl von Kilobytes an Speicherplatz, es sei denn, die erlaubten Höchst- oder Mindestwerte werden nicht eingehalten, dann werden die entsprechenden Grenzwerte verwendet. Folglich beginnt **-y0** immer mit dem Mindestspeicherplatz. **-y** (ohne Argument) beginnt üblicherweise mit dem Höchstspeicherplatz.

-zrecsz

Die Größe der längsten in der Sortier_Phase gelesenen Zeile wird aufgezeichnet, damit Puffer für die Mischphase zugeteilt werden können. Wenn die Sortier-Phase durch die Optionen **-c** oder **-m** ausgelassen ist, wird eine allgemeine System-Standardgröße verwendet. Zeilen, die länger als die Puffergröße sind, verursachen einen fehlerhaften Abbruch von *sort*. Wenn die tatsächliche Anzahl von Bytes (oder ein größerer Wert) in der längsten zu mischenden Zeile angegeben ist, wird ein fehlerhafter Abbruch verhindert.

Die folgenden Optionen heben die Standardsortierregeln auf.

-d "Wörterbuch" Reihenfolge: in den Vergleichen sind nur Buchstaben, Ziffern und Leerzeichen (Leerzeichen und Tabulatoren) signifikant.

-f Bildet Kleinbuchstaben in Großbuchstaben ab.

-i Ignoriere nicht druckbare Zeichen.

-M Vergleich als Monate. Die ersten drei Zeichen des Felds, die keine Leerzeichen sind, werden in Großbuchstaben dargestellt und verglichen, so daß "JAN" < "FEB" < ... < "DEC" ist. Ungültige Felder zählen als kleiner "JAN". Die Option **-M** schließt die Option **-b** ein (vgl. unten).

-n Eine numerische Zeichenfolge, die aus wahlweisen Leerzeichen, wahlweisem Minusvorzeichen und Null oder mehreren Ziffern mit einer wahlweisen Dezimalstelle besteht, wird nach ihrem arithmetischen Wert sortiert. Die Option **-n** schließt die Option **-b** ein (vgl. unten). Es ist zu beachten, daß die Option **-b** nur effektiv ist, wenn eingeschränkte Sortier-Schlüsselangaben (siehe unten) eingesetzt sind.

-r Umkehren der Reihenfolge.

Wenn Sortier-Optionen vor eingeschränkten Sortier-Schlüsselangaben auftreten, werden die gewünschten Sortier-Regeln global auf alle Sortier-Schlüssel angewendet. In Verbindung mit einem speziellen Sortier-Schlüssel (wird nachstehend erläutert) heben die angegebenen Sortier-Optionen alle globalen Sortier-Optionen für diesen Schlüssel auf.

Die Schreibweise *+pos1 -pos2* beschränkt einen Sortier-Schlüssel auf das Feld, das an der Position *pos1* beginnt und direkt vor der Position *pos2* endet. Die Zeichen an Position *pos1* und direkt vor *pos2* sind im Sortier-Schlüssel enthalten (vorausgesetzt, daß *pos2* nicht vor *pos1* steht). Ein Fehlen von *-pos2* bedeutet Ende der Zeile.

Die Angabe von *pos1* und *pos2* beinhaltet den Begriff eines Felds; das ist eine minimale Zeichenfolge gefolgt von einem Feld-Trennsymbol oder einem New-Line-Zeichen. Standardmäßig wird das erste Leerzeichen (Leerzeichen oder Tabulator) einer Folge von Leerzeichen zum Feld-Trennsymbol. Alle Leerzeichen in einer Folge von Leerzeichen werden als Teil des nächsten Felds angesehen; beispielsweise werden alle Leerzeichen am Anfang einer Zeile als Teil des ersten Felds angesehen. Die Behandlung der Feld-Trennsymbole kann mit den folgenden Optionen verändert werden:

-b Ignoriert vorangehende Leerzeichen bei der Bestimmung der Anfangs- und der Endpositionen eines eingeschränkten Sortier-Schlüssels. Wenn die Option **-b** vor dem ersten angegebenen *+pos1* Argument steht, gilt es für alle *+pos1* Argumente. Andernfalls kann die Option **b** individuell mit jedem *+pos1* oder *-pos2* Argument verbunden werden (vgl. unten).

-t*x* Verwendung von *x* als Feld-Trennsymbol; *x* wird nicht als Teil eines Felds angesehen (obwohl es in einem Sortier-Schlüssel vorkommen kann). Jedes Auftreten von *x* ist signifikant (zum Beispiel begrenzt *xx* ein leeres Feld).

pos1 und *pos2* haben die Form *m. n*, wahlweise gefolgt von einer oder mehreren **bdfinr**-Optionen. Eine durch *+m.n* angegebene Anfangsposition bedeutet *n* +1tes Zeichen im *m* +1ten Feld. Ein fehlendes *.n* bedeutet .0, dies zeigt das erste Zeichen im *m* +1ten Feld an. Wenn die Option **b** gesetzt ist, wird *n* ab dem ersten Zeichen (kein Leerzeichen) im *m* +1ten Feld gezählt; *+m.*0**b** bezieht sich auf das erste Zeichen (kein Leerzeichen) im *m* +1ten Feld.

Eine durch *-m.n* angegebene Endposition bedeutet das *n*te Zeichen (einschließlich Trennsymbole) nach dem letzten Zeichen vom *m ten* Feld. Ein fehlendes *.n* bedeutet .0, dies wird als das letzten Zeichen vom *m*ten Feld interpretiert. Wenn die Option **b** gesetzt ist, wird *n* ab dem letzten vorangestellten Leerzeichen im *m* +1ten Feld gezählt; *-m.*1**b** bezieht sich auf das erste Zeichen (kein Leerzeichen) im *m* +1ten Feld.

Wenn es mehrere Sortier-Schlüssel gibt, werden spätere Schlüssel nur verglichen, nachdem bei allen früheren Schlüsseln Gleichheit festgestellt wurde. Bei Zeilen, die sonst als gleich gelten würden, sind jetzt alle Bytes für das Sortieren signifikant.

BEISPIELE

Sortiert den Inhalt von *infile* mit dem zweiten Feld als Sortier-Schlüssel:

```
sort +1 -2 infile
```

Sortiert in umgekehrter Richtung den Inhalt von *infile1* und *infile2*, schreibt die Ausgabe nach *outfile* und verwendet die ersten Zeichen des zweiten Felds als Sortier-Schlüssel:

```
sort -r -o outfile +1.0 -1.2 infile1 infile2
```

Sortiert in umgekehrter Richtung den Inhalt von *infile1* und *infile2* und verwendet das erste Zeichen (kein Leerzeichen) des zweiten Felds als den Sortier-Schlüssel:

```
sort -r +1.0b -1.1b infile1 infile2
```

Gibt die Paßwortdatei (*passwd*(4)) aus, sortiert nach den numerischen Benutzernummern (das dritte, durch Doppelpunkt getrennte Feld):

```
sort -t: +2n -3 /etc/password
```

Gibt die Zeilen der schon sortierten Datei *infile* aus, wobei bis auf die erste Zeile alle Zeilen, die das gleiche dritte Feld haben, unterdrückt werden. (Wenn nur eine Eingabedatei verwendet wird, kann bei den Optionen **-um** der ausgewählte Vertreter für eine Menge gleicher Zeilen vorausgesagt werden.):

```
sort -um +2 -3 infile
```

DATEIEN

/usr/tmp/stm???

SIEHE AUCH:

comm(1), join(1), uniq(1).

ACHTUNG!

Verschiedenen Fehlerbedingungen werden kommentiert und führen zum Abbruch mit einem Endestatus ungleich Null (zum Beispiel wenn Eingabezeilen zu lang sind und wenn bei der Option **-c** die Dateien nicht sortiert sind). Wenn bei der letzten Zeile einer Eingabedatei ein **new-line** Zeichen fehlt, hängt *sort* ein New-Line-Zeichen an, gibt eine Warnung aus und macht weiter.

sort garantiert nicht die Einhaltung der relativen Zeilenanordnung bei gleichen Schlüsseln.

BEZEICHNUNG

spell, hashmake, spellin, hashcheck – Buchstabierfehler suchen

ÜBERSICHT

spell [**-v**] [**-b**] [**-x**] [**-l**] [**+**local_file] [files]

/usr/lib/spell/hashmake

/usr/lib/spell/spellin n

/usr/lib/spell/hashcheck spelling_list

BESCHREIBUNG

spell sammelt Worte aus den mit *files* bezeichneten Dateien und überprüft sie anhand eines Rechtschreibverzeichnisses. Worte, die weder aufgeführt noch von den Worten des Rechtschreibverzeichnisses ableitbar sind (durch Anwendung bestimmter Beugungen, Vor- und/oder Nachsilben) werden auf der Standardausgabe ausgegeben. Falls keine *files* angegeben sind, werden die Worte aus der Standardeingabe gesammelt.

spell ignoriert die meisten Konstruktionen von *troff*(1), *tbl*(1) und *eqn*(1).

Mit der Option **-v** werden alle Worte, die nicht wörtlich im Rechtschreibverzeichnis enthalten sind, ausgegeben, und es werden mögliche Ableitungen von Worten aus dem Rechtschreibverzeichnis angezeigt.

Mit der Option **-b** wird die britische Rechtschreibung überprüft. Neben der Bevorzugung von *centre*, *colour*, *programme*, *speciality*, *travelled*, usw., besteht diese Option auf der Wortendung *-ise* wie *standardise,* im Gegensatz zu Fowler und OED.

Bei der Option **-x** wird für alle Worte der mögliche Wortstamm mit einem Gleichheitszeichen **=** gedruckt.

Standardmäßig berücksichtigt *spell* (wie *deroff*(1)) Ketten von eingebundenen Dateien (**.so** und **.nx** *troff*(1) Anweisungen), es sei denn, die Namen dieser eingebundenen Dateien beginnen mit **/usr/lib.** Bei der Option **-l** berücksichtigt *spell* Ketten aller eingebundenen Dateien.

Bei der Option **+***lokal_file* werden Worte, die in *local_file* gefunden wurden, von der *spell* Ausgabe entfernt. *local_file* ist der Name einer Datei, die der Benutzer zur Verfügung stellt und die ein sortiertes Wortverzeichnis mit einem Wort pro Zeile enthält. Mit dieser Option kann der Benutzer mehrere Wörter angeben, die für jede Anwendung die richtige Rechtschreibung haben (zusätzlich zum Rechtschreibverzeichnis von *spell*) benutzen.

Das Rechtschreibverzeichnis basiert auf vielen Quellen, und obgleich die Zusammenstellung willkürlicher als bei einem gewöhnlichen Wörterbuch ist, ist es bei Eigennamen und weitverbreiteten technische Worten effektiver. Spezial-Vokabular der Biologie, Medizin und Chemie ist kaum abgedeckt.

Passende Hilfsdateien können über Namensargumente angegeben werden, die nachstehend mit den Standardeinstellungen (vgl. *DATEIEN)* angeführt sind. Kopien aller Ausgaben werden in der History-Datei gesammelt. Die Stopp-Liste filtert falsche Rechtschreibung aus (z. B. thier=thy −y+ier), die andernfalls übersehen würde.

Drei Routinen verwalten und prüfen die bei *spell* verwendeten Kontroll-Listen:

hashmake Liest eine Worteliste aus der Standardeingabe und schreibt den entsprechenden neunstelligen Hash-Code auf die Standardausgabe.

spellin Liest *n* Hash-Codes von der Standardeingabe und schreibt eine verdichtete Rechtschreibungsliste auf die Standardausgabe.

hashcheck
Liest ein verdichtetes Rechtschreibverzeichnis und stellt die neunstelligen Hash-Codes für alle Worte wieder her; es schreibt diese Codes auf die Standardausgabe.

DATEIEN

D_SPELL=/usr/lib/spell/hlist[ab]	Hash-Liste für amerikanische und britische Rechtschreibung,
S_SPELL=/usr/lib/spell/hstop	Hash-Stopp-Liste
H_SPELL=/usr/lib/spell/spellhist	History-Datei
/usr/lib/spell/spellprog	Programm

SIEHE AUCH:
deroff(1), sed(1), sort(1), tee(1).
eqn(1), tbl(1), troff(1) im Handbuch Dokumentations-Tools.

FEHLER
Das Rechtschreibverzeichnis deckt Gebiete uneinheitlich ab; bei neuen Installationen wird empfohlen, die Ausgabe einige Monate lang zur Ansammlung häufig verwendeter Worte zu überwachen; diese können in einer getrennten Datei aufbewahrt und mit Hilfe von *spellin* dem Rechtschreibverzeichnis *spellin_list* hinzugefügt werden.

BEZEICHNUNG

spline – glatte Kurve interpolieren

UBERSICHT

spline [options]

BESCHREIBUNG

spline interpretiert Zahlenpaare aus der Standardeingabe als Abszissen und Ordinate einer Funktion. Es stellt ein ähnliches Zahlenpaar, das ungefähr gleiche Abstände hat und das Eingabepaar enthält, auf der Standardausgabe her. Die kubische Spline-Ausgabe hat zwei stetige Funktionsableitungen und genügend viele Punkte, damit diese bei Aufzeichnung durch z.B. *graph(1G)* glatt aussieht.

Die folgenden Optionen *options* werden erkannt, und zwar jede als ein getrenntes Argument:

-a Liefert Abszissen automatisch (sie fehlen in der Eingabe); Abstände werden im nächsten Argument angegeben oder werden als 1 angenommen, wenn das nächste Argument keine Zahl ist.

-k Die Konstante *k* die in der Grenzwertberechnung:

tdefine prime2 'sup down 25 ´´' ndefine prime2 'sup ´´' y sub 0 prime2 ~=~ ky sub 1 prime2 , ~~~~ y sub n prime2 ~=~ ky sub n-1 prime2

verwendet wird, wird durch das nächste Argument angegeben (standardmäßig ist $k = 0$).

-n Ausgabepunkte so anordnen, daß ca. *n* Intervalle zwischen der unteren und oberen *x*Grenze entstehen (standardmäßig ist $n = 100$).

-p Periodische Ausgabe erzeugen, d. h. die Ableitungen stimmen an den Enden überein. Erste und letzte Eingabewerte sollten normalerweise übereinstimmen.

-x Nächste 1 (oder 2) Argumente sind untere (und obere) *x*Grenzen. Normalerweise werden diese Grenzen aus den Daten berechnet. Automatische Abszissen beginnen an der unteren Grenze (standardmäßig 0).

SIEHE AUCH:

graph(1G).

DIAGNOSE

Wenn die *x*-Werte nicht streng monoton sind, gibt *spline* die Eingabe ohne Interpolierung zusätzlicher Punkte wieder.

FEHLER

Maximal 1.000 Eingabepunkte sind möglich.

BEZEICHNUNG

split – eine Datei in Teile spalten

ÜBERSICHT

split [*–n*] [file [name]]

BESCHREIBUNG

split liest die Datei *file* und schreibt sie in Stücken zu *n*-Zeilen (standardmäßig 1000 Zeilen) in eine Reihe von Ausgabedateien. Der Name der ersten Ausgabedatei ist *name* mit angehängtem **aa**, dann geht es alphabetisch weiter bis zu **zz** (höchstens 676 Dateien). *name* darf nicht länger als 12 Zeichen sein. Wenn kein Ausgabename angegeben ist, wird **x** Standard.

Wenn keine Eingabedatei angegeben ist, oder, wenn – anstelle der Eingabedatei angegeben ist, wird die Standardeingabedatei verwendet.

SIEHE AUCH:

bfs(1), csplit(1).

BEZEICHNUNG

stat – für Grafikkommandos empfohlenes statistisches Netz

ÜBERSICHT

node-name [options] [files]

BESCHREIBUNG

stat ist eine Ansammlung von Funktionen auf Kommandoebene (Knoten), die unter Verwendung von *sh*(1) zur Bildung eines statistischen Netzes verbunden werden können. Die Knoten befinden sich in **/usr/bin/graf** (vgl. *graphics*(1G)). Daten durchlaufen das Netz als Zahlenfolgen (Vektoren), wobei eine Zahl folgende Form hat:

[sign](digits)(.digits)[e[sign]digits]

und wie üblich bewertet wird. Eckige Klammern und runde Klammern umgeben Felder. Alle Felder sind wählbar, aber wenigstens eines der Felder in runden Klammern muß vorhanden sein. Wird bei einem Knoten ein Zeichen eingegeben, das nicht Teil einer Zahl ist, wird es als Begrenzer angesehen.

stat Knoten werden in vier Klassen aufgeteilt.

Transformers,	die Eingabevektorelemente in Ausgabevektorelemente abbilden;
Summarizers,	die die Statistik eines Vektors kalkulieren;
Translators,	die Formate umwandeln; und
Generators,	die Quellen für definierbare Vektoren sind.

Nachstehend wird eine Übersichtsliste für *stat* Knoten aufgeführt. Die meisten Knoten nehmen Optionen an, die mit einem führenden Minuszeichen (-) angezeigt sind. Im allgemeinen wird eine Option durch ein Zeichen angegeben, dem ein Wert folgt, wie beispielsweise **c5**. Dies wird als **c** := 5 (**c** wird 5 zugewiesen) interpretiert. Die folgenden Schlüssel werden zur Bestimmung des erwarteten Werttyps verwendet:

c	Zeichen,
i	Ganze Zahl,
f	Gleitkomma oder ganze Zahl,
file	Dateiname und
string	Zeichenfolge von Zeichen in Anführungszeichen, um *shell*-Argumentbegrenzer einschließen zu können.

Optionen ohne Schlüssel sind Schalter. Alle Knoten außer *generators* akzeptieren Dateien als Eingabe, obwohl dies nicht in der Übersicht angegeben wird.

Da die folgenden Schlüssel sich auf englische Wörter beziehen, bleiben die 4 Klassen unübersetzt.

Transformers:

abs [**-c***i*] – absolute value
columns (similarly for -c options that follow)

af [**-c***i* **t v**] – arithmetic function
titled output, **v**erbose

ceil [**-c***i*] – round up to next integer

cusum [**-c***i*] – cumulative sum

exp [**-c***i*] – exponential

floor [**-c***i*] – round down to next integer

gamma [**-c***i*] – gamma

list [**-c***i* **d***string*] – list vector elements
delimiter(s)

log [**-c***i* **b***f*] – logarithm
base

mod [**-c***i* **m***f*] – modulus
modulus

pair [**-c***i* **F***file* **x***i*] – pair elements
File containing base vector, **x** group size

power [**-c***i* **p***f*] – raise to a power
power

root [**-c***i* **r***f*] – take a root
root

round [**-c***i* **p***i* **s***i*] – round to nearest integer, .5 rounds to 1
places after decimal point, **s**ignificant digits

siline [**-c***i* **i***f* **n***i* **s***f*] – generate a line given slope and intercept
intercept, **n**umber of positive integers, **s**lope

sin [**-c***i*] – sine

subset [**-a***f* **b***f* **c***i* **F***file* **i***i* **l***f* **n**l **n**p **p***f* **s***i* **t***i*] – generate a subset
above, **b**elow, **F**ile with master vector, **i**nterval, **l**eave, master contains element **n**umbers to leave, master contains element **n**umbers to **p**ick, **p**ick, **s**tart, **t**erminate

Summarizers:

bucket [**-a***i* **c** **F***file* **h***f* **l***i* **l***f* **n***i*] – break into buckets
average size, File containing bucket boundaries, high, interval, low, number
Input data should be sorted

cor [**-F***file*] – correlation coefficient
File containing base vector

hilo [**- h l o ox oy**]– find high and low values
high only, low only, option form, option form with **x** prepended, option form with **y** prepended

lreg [**-F***file* **i o s**] – linear regression
File containing base vector, intercept only, option form for *siline*, slope only

mean [**-f***f* **n***i* **p***f*] – (trimmed) arithmetic mean
fraction, number, percent

point [**-f***f* **n***i* **p***f* **s**] – point from empirical cumulative density function
fraction, number, percent, sorted input

prod – internal product

qsort [**-c***i*] – quick sort

rank – vector rank

total – sum total

var – variance

Translators:

bar [**-a b f g r***i* **w***i* **x***f* **xa y***f* **ya yl***f* **yh***f*] – build a bar chart
suppress axes, bold, suppress frame, suppress grid, region, width in percent, x origin, suppress x-axis label, y origin, suppress y-axis label, y-axis lower bound, y-axis high bound
Data is rounded off to integers.

hist [**-a b f g r***i* **x***f* **xa y***f* **ya yl***f* **yh***f*] – build a histogram
suppress axes, bold, suppress frame, suppress grid, region, x origin, suppress x-axis label, y origin, suppress y-axis label, y-axis lower bound, y-axis high bound

label [**-b c F***file* **h p r***i* **x xu y yr**] – label the axis of a GPS file
bar chart input, retain case, label File, histogram input, plot input, rotation, x-axis, upper x-axis, y-axis, right y-axis

pie [**-b o p pn***i* **pp***i* **r***i* **v x***i* **y***i*] – build a pie chart
bold, values **o**utside pie, value as **p**ercentage(:=100), value as **p**ercentage(:=i), draw **p**ercent of **pie**, **r**egion, no values, **x** origin, **y** origin
Unlike other nodes, input is lines of the form
[< **i e f** *cc* >] value [label]
ignore (do not draw) slice, **e**xplode slice, **f**ill slice, color slice *c* =(**b**lack, **r**ed, **g**reen, **b**lue)

plot [**-a b c***string* **d f F***file* **g m r***i* **x***f* **xa xl***f* **xh***f* **xl***f* **xn***i* **xt y***f* **ya yi***f* **yh***f* **yl***f* **yn***i* **yt**] – plot a graph
suppress **a**xes, **b**old, plotting **c**haracters, **d**isconnected, **s**uppress **f**rame, **F**ile containing x vector, suppress **g**rid, **m**ark points, **r**egion, **x** origin, suppress **x**-axis label, **x** **i**nterval, **x** **h**igh bound, **x** **l**ow bound, number of ticks on **x**-axis, suppress **x**-axis title, **y** origin, suppress **y**-axis label, **y** **i**nterval, **y** **h**igh bound, **y** **l**ow bound, number of ticks on **y**-axis, suppress **y**-axis **t**itle

title [**-b c l***string* **v***string* **u***string*] – title a vector or a GPS
title **b**old, **r**etain **c**ase, **l**ower title, **u**pper title, **v**ector title

Generators:

gas [**-c***i* **i***f* **n***i* **s***f* **t***f*] – generate additive sequence
interval, **n**umber, **s**tart, **t**erminate

prime [**-c***i* **h***i* **l***i* **n***i*] – generate prime numbers
high, **l**ow, **n**umber

rand [**-c***i* **h***f* **l***f* **m***f* **n***i* **s***i*] – generate random sequence
high, **l**ow, **m**ultiplier, **n**umber, **s**eed

EINSCHRÄNKUNGEN

Einige Knoten haben einen Grenzwert für die Größe des Eingabevektors.

SIEHE AUCH

graphics(1G).
gps(4) im *Programmer's Reference Manual.*

BEZEICHNUNG

stty – die Optionen für ein Terminal setzen

ÜBERSICHT

stty [**-a**] [**-g**] [options]

BESCHREIBUNG

stty setzt gewisse Ein-/Ausgabe-Merkmale für das Terminal, das die aktuelle Standardeingabe ist. Ohne Argumente wird nur die Einstellung von bestimmten Optionen gemeldet.

Wenn dabei einem Zeichen ein (^) vorangeht, ist der Wert dieser Option das entsprechende CTRL-Zeichen (z. B., "^**h**" ist **CTRL-h** ; **CTRL-h** ist identisch mit der "Backspace-Taste.) Die Folge "^´" bedeutet, daß eine Option einen Wert Null hat. Zum Beispiel meldet **stty -a** normalerweise, daß der Wert von **swtch** "^´" ist; wenn jedoch **shl** (1) oder **layers** (1) aufgerufen wurden, hat **stty -a** den Wert "^**z**".

-a Bericht über alle Optionseinstellungen;

-g Bericht über die aktuellen Einstellungen in einer Form, die als Eingabe für ein anderes *stty* Kommando verwendet werden kann.

Optionen in der letzten Gruppe (siehe unten) werden implementiert, indem Optionen der vorhergehenden Gruppen verwendet werden. Es ist zu beachten, daß viele Options-Kombinationen nicht sinnvoll sind, hierfür gibt es jedoch keine internen Prüfungen. Es gibt folgende Optionen:

Kontrollmodi

parenb (**-parenb**)	Einschalten (Abschalten) der Paritätserzeugung und Erkennung.
parodd (**-parodd**)	Ungerade (gerade) Parität wählen.
cs5 cs6 cs7 cs8	Zeichengröße (vgl. *termio*(7)) wählen.
0	Telefonleitung sofort auflegen
110 300 600 1200 1800 2400 4800 9600 19200 38400	Setzen der Terminal-Baudrate auf die angegebene Nummer, wenn vorhanden. (Alle Geschwindigkeiten werden nicht von allen Hardware-Schnittstellen unterstützt.)
hupcl (**-hupcl**)	Unterbrechen (nicht unterbrechen) der Telefon-Verbindung nach Beendigung.
hup (**-hup**)	identisch mit **hupcl** (**-hupcl**).
cstopb (**-cstopb**)	Verwenden von zwei (ein) Stoppbits pro Zeichen.
cread (**-cread**)	Einschalten (abschalten) des Empfängers.
clocal (**-clocal**)	Leitung ohne (mit) Modemsteuerung.
loblk (**-loblk**)	Blockieren der Ausgabe (nicht blockieren) eines nicht-aktuellen Layers.

Eingabemodi

ignbrk (-ignbrk)	Ignorieren (nicht ignorieren) von Break bei Eingabe.
brkint (-brkint)	Signalisieren (nicht signalisieren) von INTR bei Break.
ignpar (-ignpar)	Ignorieren (nicht ignorieren) der Paritätsfehler.
parmrk (-parmrk)	Markieren (nicht markieren) der Paritätsfehler (vgl. *termio*(7)).
inpck (-inpck)	Einschalten (abschalten) der Eingabeparitätsprüfung.
istrip (-istrip)	Verkürzen (nicht verkürzen) der Eingabezeichen bis zu sieben Bits.
inlcr (-inlcr)	Abbilden (nicht abbilden) von NL auf CR bei Eingabe.
igncr (-igncr)	Ignorieren (nicht ignorieren) von CR bei Eingabe.
icrnl (-icrnl)	Abbilden (nicht abbilden) von CR auf NL bei Eingabe.
iuclc (-iuclc)	Abbilden (nicht abbilden) von Großbuchstaben in Kleinbuchstaben bei Eingabe.
ixon (-ixon)	Einschalten (abschalten) der START/STOPP-Ausgabesteuerung. Ausgabe wird durch ASCII DC3 gestoppt und durch ASCII DC1 gestartet.
ixany (-ixany)	Beliebiges Zeichen (nur DC1), kann Ausgabe neu starten.
ixoff (-ixoff)	System soll START-/STOPP- Zeichen senden (nicht senden), wenn die Eingabewarteschlange fast leer/voll ist.

Ausgabemodi

opost (-opost)	Ausgabe nachbearbeiten (Ausgabe nicht nachbearbeiten; alle anderen Ausgabemodi ignorieren).
olcuc (-olcuc)	Abbilden (nicht abbilden) der Kleinbuchstaben in Großbuchstaben bei Ausgabe.
onlcr (-onlcr)	Abbilden (nicht abbilden) von NL auf CR-NL bei Ausgabe.
ocrnl (-ocrnl)	Abbilden (nicht abbilden) von CR auf NL bei Ausgabe.
onocr (-onocr)	Ausgabe (keine Ausgabe) von CRs auf Spalte Null.
onlret (-onlret)	NL führt (führt nicht) die CR-Funktion am Terminal aus.
ofill (-ofill)	Verwendung von Füllzeichen (zeitliche Abstimmung benutzen) für Verzögerungen.

ofdel (-ofdel) Füllzeichen sind DELs (NULs).

cr0 cr1 cr2 cr3 Verzögerungsart für Wagenrücklauf wählen (vgl. *termio*(7)).

nl0 nl1 Verzögerungsart für Zeilenvorschübe wählen (vgl. *termio*(7)).

tab0 tab1 tab2 tab3 Verzögerungsart für Horizontal-Tabulatorzeichen wählen (vgl. *termio*(7)).

bs0 bs1 Verzögerungsart für Backspace-Zeichen wählen (vgl. *termio*(7)).

ff0 ff1 Verzögerungsart für Formularvorschübe wählen (vgl. *termio*(7)).

vt0 vt1 Verzögerungsart für Vertikaltabulatoren wählen (vgl. *termio*(7)).

Lokale Modi

isig (-isig) Einschalten (abschalten) der Zeichenprüfung hinsichtlich der besonderen Kontrollzeichen INTR, QUIT und SWTCH.

icanon (-icanon) Einschalten (abschalten) der Sonderverarbeitung (ERASE und KILL-Verarbeitung).

xcase (-xcase) Ungeänderte Darstellung der Groß-/Kleinschreibung.

echo (-echo) Wiedergabe mit Echo (keine Wiedergabe mit Echo) jedes eingegebenen Zeichens.

echoe (-echoe) Echo (kein Echo) des ERASE-Zeichens als eine Zeichenfolge von Backslash-Leerzeichen-Backslash.
Anmerkung: dieser Modus löscht die ERASed-Zeichen bei vielen Terminals; jedoch wird die Spaltenposition *nicht* gemerkt, so daß als Ergebnis bei Begrenzungszeichen, Tabulatoren und Backspace-Zeichen Verwirrung eintreten kann.

echok (-echok) Echo (kein Echo) von NL nach KILL-Zeichen.

lfkc (-lfkc) Wie bei **echok (-echok)**; veraltert.

echonl (-echonl) Echo (kein Echo) von NL.

noflsh (-noflsh) Abschalten (einschalten) von Puffer ausgeben nach INTR, QUIT oder SWTCH.

Zuweisungen an Steuerzeichen

control-character c — *control-character* auf *c* setzen, mit *control-character* **erase, kill, intr, quit, switch, eof, ctab, min,** oder **time** (**ctab** wird mit **-stappl** verwendet ; **min** und **time** werden mit **-icanon** verwendet; vgl. *termio*(7)). Wenn *c* ein (^) vorausgeht, ist der verwendete Wert das entsprechende Kontrollzeichen (z.B., "**^d**" ist ein **CTRL-d**); "^?" wird als DEL interpretiert und "^-" wird als undefiniert interpretiert.

line *i* — Setzen des Leitungsprotokolls auf *i* (0 < *i* < 127).

Kombinationsmodi

evenp or **parity** — Einschalten von **parenb** und **cs7**.

oddp — Einschalten von **parenb, cs7,** und **parodd.**

-parity, -evenp oder **-oddp** — Abschalten von **parenb** und **cs8** setzen.

raw (**-raw** or **cooked**) — Einschalten (abschalten) von "raw" Eingabe und Ausgabe: (kein ERASE, KILL, INTR, QUIT, SWTCH, EOT oder Nachbearbeitung der Ausgabe).

nl (**-nl**) — Setzen (aufheben) **icrnl, onlcr.** Zusätzlich hebt **-nl inlcr, igncr, ocrnl** und **onlret** auf.

lcase (**-lcase**) — Setzen (aufheben) von **xcase, iuclc** und **olcuc.**

LCASE (**-LCASE**) — wie bei **lcase** (**-lcase**).

tabs (**-tabs** oder **tab3**) — Bei der Ausgabe Tabulatoren beibehalten (auf Zwischenräume erweitern).

ek — Zurücksetzen der ERASE and KILL-Zeichen auf # und @.

sane — Alle Modi auf geeignete Werte zurücksetzen.

term — Alle Modi, die für den Terminal-Typ *term* geeignet sind, zurücksetzen, wobei *term* **tty33, tty37, vt05, tn300, ti700,** oder **teksein**

SIEHE AUCH

tabs(1).
ioctl(2) im *Programmer's Reference Manual.*
termio(7) im System *Administrator's Reference Manual.*

BEZEICHNUNG

su – Systemverwalter oder anderer Benutzer werden

ÜBERSICHT

su [–] [name [arg ...]]

BESCHREIBUNG

Mit Hilfe von *su* kann man ohne Abmeldung ein anderer Benutzer werden. Der Standardbenutzername *name* ist **root** (d. h. Systemverwalter).

Für *su* muß das entsprechende Paßwort angegeben werden (es sei denn Sie sind selbst **root**). Wenn das Paßwort richtig ist, führt *su* mit der für den angegebenen Benutzer realen und effektiven Benutzernummer eine neue Shell aus. Die neue Shell ist das wählbare Programm, das im shell-Feld des speziellen Benutzereintrags in der Paßwortdatei angegeben ist (vgl. *passwd(4)*) oder **/bin/sh,** wenn kein Programm angegeben ist (vgl. *sh(1)*). Zur Wiederherstellung der vorherigen Benutzerrechte geben Sie der neuen Shell **EOF** (cntrl-D) ein.

Zusätzlichen Argumente auf der Kommandozeile werden zu dem als Shell aufgerufenen Programm weitergeleitet. Wenn Programme wie *sh*(1) verwendet werden, bewirkt das Argument **-c** *string,* daß *string* durch die Shell ausgeführt wird, und das Argument **-r,** daß der Benutzer eine eingeschränkte Shell erhält.

Die folgenden Anweisungen sind nur wahr, wenn das wählbare Programm, das im shell-Feld der Paßwort-Datei für den Benutzer angegeben ist, ähnlich wie *sh*(1) ist. Wenn das erste Argument für *su* ein – ist, wird die Umgebung so geändert, wie man es erwartet hätte, wenn der Benutzer sich tatsächlich als der angegebene Benutzer angemeldt hätte. Dies erfolgt durch Aufruf des Programms, das als Shell verwendet wird, mit einem *arg0* Wert, dessen erstes Zeichen – ist, so daß zuerst das Systemprofile (**/etc/profile**) und dann das Profile des angegebenen Benutzers (**.profile** im neuen HOME-Verzeichnis) ausgeführt wird. Andernfalls wird die Umgebung weitergereicht, eventuell mit Ausnahme von **$PATH**, der für **root** auf **/bin:/etc:/usr/bin** gesetzt wird. Wenn das wählbare Programm, das als Shell verwendet wird, **/bin/sh** ist, kann **.profile** des Benutzers *arg0* auf **-sh** oder **-su** prüfen, um zu bestimmen, ob es von *login*(1) oder *su*(1) aufgerufen wurde. Wenn das Benutzerprogramm nicht **/bin/sh** ist, wird **.profile** sowohl von *login*(1) als auch von *su*(1) mit dem *arg0 -program* aufgerufen.

Alle Versuche, mittels *su* ein anderer Benutzer zu werden, werden in der Protokolldatei **/usr/adm/sulog** eingetragen.

BEISPIELE

Um Benutzer **bin** zu werden, wobei Sie Ihre vorher exportierte Umgebung beibehalten, führen Sie aus:

```
su bin
```

Um Benutzer **bin** zu werden, aber die Umgebung so zu ändern, als ob mit **bin** angemeldet worden wäre, ist folgendes auszuführen:

```
su - bin
```

Zur Ausführung von *command* mit der vorübergehend eingestellten Umgebung und Berechtigung von Benutzer **bin**, ist einzugeben:

```
su - bin -c "command args"
```

DATEIEN

/etc/passwd	Systempaßwortdatei
/etc/profile	Systemprofile
$HOME/.profile	Benutzerprofile
/usr/adm/sulog	Protokolldatei

SIEHE AUCH:

env(1), login(1), sh(1) im *User's Reference Manual.*
passwd(4), profile(4), environ(5) im *Programmer's Reference Manual.*

BEZEICHNUNG

sum – Prüfsumme und Blockanzahl einer Datei angeben

ÜBERSICHT

sum [**-r**] file

BESCHREIBUNG

sum berechnet und gibt eine 16-Bit-Prüfsumme aus für die angegebene Datei und gibt auch die Anzahl der Blöcke in der Datei an. Das Kommando wird normalerweise zum Suchen von ungültigen Stellen oder zur Validierung einer Datei verwendet, die über eine Datenübertragungsleitung gesendet wurde. Die Option **-r** bewirkt den Einsatz eines alternativen Algorithmus bei der Berechnung der Prüfsumme.

SIEHE AUCH

wc(1).

DIAGNOSE

(Lesefehler) kann bei den meisten Geräten nicht von Dateiende unterschieden werden; die Blockanzahl ist zu überprüfen.

BEZEICHNUNG

sync – den Super-Block aktualisieren

ÜBERSICHT

sync

BESCHREIBUNG

sync führt die System-Basisprozedur *sync* aus. Wenn das System gestoppt werden soll, muß *sync* zur Sicherstellung der Dateisystemintegrität aufgerufen werden. Alle vorher nicht geschriebenen Systempuffer werden auf die Festplatte geschrieben, so daß alle Modifikationen bis zu diesem Zeitpunkt gesichert werden. Vgl. *sync(2)* hinsichtlich weiterer Information.

ANMERKUNG

Wenn Sie in einer Remote File Sharing Umgebung eine Datei auf einem Remote-Rechner beschreiben, können Sie *sync* nicht verwenden, um das Herausschreiben der Systempuffer zu erzwingen. *sync* schreibt nur lokale Puffer auf lokale Festplatten.

SIEHE AUCH:

sync(2) im *Programmer's Reference Manual.*

BEZEICHNUNG

tabs – Tabulatoren auf einem Terminal setzen

ÜBERSICHT

tabs [tabspec] [**-Ttype**] [**+mn**]

BESCHREIBUNG

tabs setzt Tabulator-Stopps auf dem Terminal des Benutzers gemäß der Tabulatorangaben von *tabspec* nach Löschen vorangegangener Einstellungen. Das Terminal des Benutzers muß über entfernt-einstellbare Hardware-Tabulatoren verfügen.

tabspec Vier Arten von Tabulatorangaben werden von *tabspec* akzeptiert. Sie werden nachstehend erläutert: kompakt (*-code*), wiederholend (*-n*), beliebig (*n1,n2,...*) und dateibezogen (*--file*). Wenn kein *tabspec* angegeben ist, ist der Standardwert **-8**, d. h. "Standard"-Tabulatoren des UNIX-Systems. Die niedrigste Spaltennummer ist 1. Es ist zu beachten, daß bei *tabs* Spalte 1 immer die ganz linke Spalte auf einem Bildschirm bedeutet. Dies gilt auch für Geräte, deren Spaltenmarkierer mit 0 beginnen wie z. B. bei den Geräten DASI 300, DASI 300s und DASI 450.

-code Zur Auswahl einer Folge von Tabulatoren für eine Zeile ist einer der nachstehend aufgeführten Codes zu benutzen. Die zulässigen Codes und ihre Bedeutung lauten wie folgt:

-a 1,10,16,36,72
Assembler, IBM S/370, erstes Format

-a2 1,10,16,40,72
Assembler, IBM S/370, zweites Format

-c 1,8,12,16,20,55
COBOL, übliches Format

-c2 1,6,10,14,49
COBOL Kompaktes Format (Spalten 1–6 sind ausgelassen). Bei Benutzung dieses Codes entspricht das erste eingegebene Zeichen der Spalte 7, durch Angabe eines Leerzeichens wird auf Spalte 8 positioniert, mit Tabulator auf Spalte 12. Dateien, die diese Tabulatoreinstellung verwenden, sollten folgende Formatangabe enthalten (vgl. *fspec*(4)):

: <t -c2 m6 s66 d: >

-c3 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
COBOL Kompaktes Format (Spalten 1–6 sind ausgelassen), mit mehr Tabulatoren als **-c2**. Dies ist das empfohlene Format für COBOL. Die entsprechenden Formatangaben lauten (vgl. *fspec*(4)):

: t < -c3 m6 s66 d: >

-f 1,7,11,15,19,23
FORTRAN
-p 1,5,9,13,17,21,25,29,33,37,41,45,49,53,57,61
PL/I
-s 1,10,55
SNOBOL
-u 1,12,20,44
UNIVAC 1100 Assembler

-n Eine *repetitive* (wiederholende) Angabe setzt Tabulatoren bei Spalten 1+*n*, 1+2**n*, usw. Der Wert **8** hat eine besondere Bedeutung: er stellt die "Standard"-Tabulatoreinstellung des UNIX Systems dar und ist die am häufigsten gefundene Tabulatoreinstellung an einem Terminal. Ein anderer Sonderfall ist der Wert **0**, der bedeutet, daß keine Tabulatoren gesetzt sind.

n1,n2,... Das *arbitrary* (beliebiges) Format erlaubt es dem Benutzer, beliebig viele ausgewählte Zahlen in aufsteigender Reihenfolge einzugeben, die durch Kommata getrennt sind. Bis zu 40 Zahlen sind erlaubt. Wenn Zahlen (außer der ersten) ein Plusvorzeichen vorausgeht, wird es als ein Inkrement angesehen und zum vorhergehenden Wert hinzugefügt. Folglich werden die Formate **1,10,20,30** und **1,10,+10,+10** als identisch angesehen.

--file Wenn der Name einer Datei *file* angegeben ist, liest *tabs* die erste Zeile der Datei und sucht nach einer Formatangabe (vgl. *fspec*(4)). Wenn eine Angabe gefunden wird, setzt das Kommando die Tabulator-Stopps gemäß dieser Angabe, anderfalls werden sie auf **-8** gesetzt. Diese Art der Formatangabe kann verwendet werden, wenn sichergestellt werden soll, daß eine mit Tabulatoren versehene Datei mit den richtigen Tabulatoreinstellungen ausgegeben wird, und kann dann zusammen mit dem Kommando *pr*(1) benutzt werden:

tabs -- file; pr file

Folgende Optionen können ebenfalls verwendet werden; wenn eine bestimmte Option mehr als einmal vorkommt, tritt der zuletzt angegebene Wert in Kraft:

-T*type* *tabs* muß normalerweise zum Setzen der Tabulatoren den Terminal-Typ kennen; der Typ muß immer zur Randeinstellung bekannt sein. *type* ist ein in *term*(5) angegebener Name. Wenn keine **-T**-Option vorliegt, verwendet *tabs* den Wert der Umgebungsvariablen **TERM**. Wenn **TERM** nicht in der Umgebung *environment* (vgl. *environ*(5)) definiert ist, verwendet *tabs* eine Folge, die bei vielen Terminals funktioniert.

+m*n* Das Rand-Argument kann für einige Terminals verwendet werden. Es bewirkt, daß alle Tabulatoren um *n* Spalten nach rechts rücken, da Spalte *n+1* der linke Rand wird. Wenn **+m** ohne einen Wert von *n* angegeben ist, wird der Wert **10** angenommen. Bei TermiNet-Terminals sollte der erste Wert in der Tabulator-Liste **1** sein, oder der Rand wird sogar noch weiter nach rechts verlagert. Der normale (ganz linke) Rand wird bei den meisten Terminals mit **+m0** gesetzt. Der Rand wird bei den meisten Terminals nur nach rechts verrückt, wenn die Option **+m** ausdrücklich angegeben ist.

Tabulator- und Rand-Einstellung erfolgt über die Standardausgabe.

BEISPIELE

tabs -a Beispiel für die Verwendung von *canned* (kompakter) Formatangabe zur Einstellung von Tabulatoren entsprechend dem IBM Assembler: Spalten 1, 10, 16, 36, 72.

tabs -8 Beispiel zur Verwendung von *-n* (*repetitive* (wiederholende) Formatangabe), wobei *n* gleich **8** das Setzen von Tabulatoren auf jede achte Position bewirkt: 1+(1*8), 1+(2*8), ... setzt Tabulatoren auf die Spalten 9, 17 ...

tabs 1,8,36 Beispiel für die Verwendung von *n1,n2,...* (*arbitrary* (beliebige) Formatangabe) zum Setzen von Tabulatoren auf die Spalten 1, 8 und 36.

tabs --$HOME/fspec.list/att4425
Beispiel für die Verwendung von *--file* (*file* (dateibezogene) Formatangabe) zum Setzen der Tabulatoreinstellung gemäß der ersten Zeile von *$HOME/fspec.list/att4425* (vgl. *fspec*(4)).

DIAGNOSE

illegal tabs	wenn beliebige Tabulatoren falsch angeordnet sind
illegal increment	wenn eine Null oder ein fehlendes Inkrement in einer beliebigen Formatangabe gefunden wird
unknown tab code	wenn ein Code nicht gefunden werden kann
can't open	wenn *--file*-Option verwendet wird und die Datei nicht geöffnet werden kann
file indirection	wenn *--file*-Option verwendet wird und die Formatangabe in dieser Datei noch auf eine andere Datei zeigt. Verweise dieser Art sind nicht zugelassen.

SIEHE AUCH:

newform(1), pr(1), tput(1).
fspec(4), terminfo(4), environ(5), term(5) im *Programmer's Reference Manual.*

ANMERKUNG

Das Löschen von Tabulatoren und die Einstellung des linken Randes ist bei verschiedenen Terminals nicht einheitlich.

tabs löscht nur 20 Tabulatoren (bei Terminals, die eine lange Sequence erfordern), aber es kann 64 Tabulatoren setzen.

ACHTUNG!

Die *tabspec*, die bei dem Kommando *tabs* verwendet wird, ist unterschiedlich zu der, die bei dem Kommando *newform* (1) verwendet wird. Zum Beispiel setzt **tabs -8** Tabulatoren auf jede achte Position; während **newform -i-8** anzeigt, daß Tabulatoren an jeder achten Position gesetzt sind.

BEZEICHNUNG

tail – den letzten Teil einer Datei anzeigen

ÜBERSICHT

tail [±[number][**lbc**[**f**]]] [File]

BESCHREIBUNG

tail kopiert die angegebene Datei auf die Standardausgabe, wobei an einer festgelegten Stelle begonnen wird. Wenn kein Dateiname angegeben ist, wird die Standardeingabe verwendet.

Kopieren beginnt mit einem Abstand *+number* vom Anfang oder *–number* vom Ende der Eingabedatei (wenn *number* Null ist, wird der Wert 10 angenommen). Die Angabe *number* wird in Einheiten von Blöcken, Zeilen oder Zeichen gemäß der angehängten Option **l**, **b** oder **c** berechnet. Wenn keine Einheiten angegeben sind, erfolgt eine Berechnung in Zeilen.

Bei der Option **–f** ("follow") wird das Programm, wenn die Eingabedatei keine Pipe ist, nicht beendet, nachdem die Zeilen der Eingabedatei kopiert worden sind, sondern geht in eine Schleife, in der es eine Sekunde untätig bleibt und dann versucht, weitere Datensätze von der Eingabedatei zu lesen und zu kopieren. Folglich kann es verwendet werden, um das Anwachsen einer Datei zu überwachen, die von einem anderen Prozeß geschrieben wird. Zum Beispiel werden mit dem Kommando:

```
tail –f fred
```

die letzen zehn Zeilen der Datei **fred** angegeben, gefolgt von Zeilen, die an die Datei **fred** angehängt wurden zwischen Aufruf und Abbruch von *tail*. Ein anderes Beispiel ist das Kommando:

```
tail –15cf fred
```

Dieses gibt die letzten 15 Zeichen der Datei **fred** aus, gefolgt von Zeilen, die an die Datei **fred** angehängt wurden zwischen Aufruf und Abbruch von *tail*.

SIEHE AUCH:

dd(1M).

FEHLER

"Tails", die sich auf das Dateiende beziehen, werden in einem Puffer gespeichert, und sind folglich in ihrer Länge begrenzt. Abweichendes Verhalten in verschiedener Form kann bei zeichenorientierten Dateien auftreten.

ACHTUNG!

Unabhängig von der Zeilenanzahl kopiert das Kommando *tail* nur die letzten 4096 Bytes einer Datei.

BEZEICHNUNG

talk – Dialog zwischen Netzwerk-Benutzern

ÜBERSICHT

talk person [ttyname]

BESCHREIBUNG

Talk ist ein visuelles Kommunikationsprogramm, das Eingabezeilen von Ihrem Terminal auf das Terminal eines anderen Benutzers kopiert.

Wenn Sie auf Ihrem eigenen Rechner mit einem anderen Benutzer einen Dialog führen wollen, müssen Sie als *person* den Benutzernamen Ihres Partners angeben. Wenn Sie mit einem anderen Benutzer auf einem anderen Rechner einen Dialog führen wollen, müssen Sie *person* in dem folgenden Format angeben:

host!user oder
host.user oder
host:user oder
user@host , obwohl auch häufig *host@user* benutzt wird.

Wenn Sie mit einem mehrfach angemeldeten Benutzer sprechen möchten, können Sie das Argument *ttyname* zur Angabe des gewünschten Terminalnamens benutzen.

Beim ersten Aufruf sendet *talk* die folgende Nachricht an den Benutzer, mit dem Sie einen Dialog führen wollen:

Message from TalkDaemon@his_machine...
talk: connection requested by your_name@your_machine.
talk: respond with: talk your_name@your_machine

Meldung von TalkDämon@sein_rechner...
talk: Verbindung angefordert von ihr_name@ihr_rechner.
talk: antworten Sie mit: talk ihr_name@ihr_rechner

Der Empfänger der Nachricht sollte jetzt seine Antwort eingeben:

talk ihr_name@ihr_rechner

Es spielt keine Rolle, von welchem Rechner aus der Empfänger antwortet, vorausgesetzt, daß der Benutzername gleich ist. Sobald der Dialog in Gang gekommen ist, können beide Seiten gleichzeitig eingeben. Die Antworten des Partners werden in einem getrennten Bildschirmfenster angezeigt. Die Eingabe von Control-L bewirkt, daß der Bildschirm neu aufgebaut wird. Die Tasten zum Löschen eines Zeichens, einer Zeile oder eines Wortes funktionieren wie gewohnt. Zum Beenden des Dialogs genügt die Betätigung der Unterbrechungstaste. *Talk* setzt dann den Cursor an den unteren Bildschirmrand und setzt den Bildschirm zurück.

Mit dem Kommando *mesg* kann das Führen von Dialogen gestattet oder abgelehnt werden. Standardmäßig ist das Führen von Dialogen erlaubt. Bestimmte Kommandos, besonders *nroff* und *pr*(1), nehmen grundsätzlich keine Meldungen entgegen, damit das Terminalbild nicht durcheinandergebracht wird.

DATEIEN

/etc/hosts zur Ermittlung des Partnerrechners
/etc/utmp zur Ermittlung des Partnerterminals

SIEHE AUCH

mesg(1), who(1), mail(1), write(1)

BEZEICHNUNG

tar – archivieren auf Band

ÜBERSICHT

tar [*key*] [*files*]

BESCHREIBUNG

tar speichert Dateien auf und holt Dateien von Magnetbändern. Seine Aktionen werden durch den Schlüssel *key* gesteuert. Der Schlüssel *key* ist eine Zeichenfolge, die einen Funktionsbuchstaben und möglicherweise einen oder mehrere Funktionsmodifikatoren. Andere Argumente für das Kommando sind *files* (oder Namen von Verzeichnissen), die angeben, welche Dateien herausgeschrieben oder wiedereingespielt werden sollen. Ein Verzeichnisname steht immer für die Dateien und (rekursiv) die Unterdateiverzeichnisse dieses Verzeichnisses.

Der Funktionsteil des Schlüssels wird durch einen der folgenden Buchstaben angegeben:

r Ersetzen. Die benannten *files* werden ans Ende des Bandes geschrieben. Die Funktion **c** bezieht diese Funktion ein.

x Holen. Die benannten *files* werden vom Band geholt. Wenn eine angegebene Datei ein Verzeichnis ist, dessen Inhalt auf das Band geschrieben wurde, wird dieses Verzeichnis (rekursiv) vom Band geholt. Sie müssen, wenn angebracht, den relativen Pfadnamen angeben, damit *tar* die Übereinstimmung der Namen feststellen kann. Der Eigentümer, die Modifikationszeit und der Modus werden, sofern möglich, wiederhergestellt. Wenn kein *files* Argument angegeben ist, wird der gesamte Inhalt des Bands geholt. Es ist zu beachten, daß der letzte Eintrag alle früheren überschreibt, wenn mehrere Dateien mit gleichen Namen auf dem Band sind.

t Verzeichnis. Die Namen und andere Informationen über die angegebenen Dateien werden jedesmal, wenn sie auf dem Band erscheinen, ausgegeben. Die Liste ist im Format einer mit dem Kommando *ls -l* erstellten Liste ähnlich. Wenn kein Argument *files* vorliegt, werden alle Namen auf dem Band aufgelistet.

u Aktualisierung. Die angegebenen Dateien *files* werden auf das Band geschrieben, wenn sie nicht schon dort sind oder seit dem letzten Schreiben auf dieses Band modifiziert wurden.

c Erstellt ein neues Band; das Schreiben beginnt am Anfang vom Band anstatt nach der letzten Datei. Dieser Schlüssel schließt den Schlüssel **r** mit ein.

Die folgenden Zeichen können zusätzlich zu dem Buchstaben verwendet werden, der die gewünschte Funktion auswählt.

0,...,7 Dieser Modifikator bestimmt das Laufwerk, auf dem das Band installiert ist. Standardmäßig wird das Laufwerk 1 benutzt. Es ist zu beachten, daß Nixdorf nicht die AT&T-Laufwerkbezeichnungen unterstützt. Falls die Schlüssel l, m oder h nach diesem Modifikator benutzt werden (um eine AT&T Band-Bezeichnung anzugeben), werden l, m oder h als eine der unten geführten Funktionsmodifikatoren interpretiert. Der Schlüssel wird *nicht* als Teil einer Laufwerkbezeichnung ausgewertet.

v Wortreich. Normalerweise führt *tar* seine Arbeit stillschweigend aus. Die Option **v** (wortreich) veranlaßt, daß der Namen jeder behandelten Datei mit dem vorangestellten Funktionsbuchstaben ausgegeben wird. Bei der Funktion **t** gibt **v** mehr Informationen über die Bandeinträge als nur den Namen an.

w Was. Dies veranlaßt *tar* zur Ausgabe der durchzuführenden Aktion, gefolgt von dem Namen der Datei. Danach wird auf die Bestätigung des Benutzers gewartet. Wenn ein Wort, das mit **y** anfängt, angegeben wird, wird die Aktion durchgeführt. Andere Eingaben bedeuten no (keine Aktion).

f Datei. *tar* verwendet *device* als Name des Archivs anstelle von **/dev/rmt?** . Wenn der Name der Datei – ist, schreibt *tar* entsprechend auf die Standardausgabe oder liest von der Standardeingabe. Auf diese Weise kann *tar* als Beginn oder Ende einer Pipe verwendet werden. *tar* kann auch zur Verlagerung von Verzeichnis-Hierarchien verwendet werden durch die Aufruffolge:

```
cd fromdir; tar cf - . | (cd todir; tar xf -)
```

b Blockungsfaktor. Hierdurch verwendet *tar* das Argument *block* als Blockungsfaktor für die Bandaufzeichnung. Der Standard ist 1, der Höchstwert ist 64. Diese Funktion sollte nur benutzt werden, wenn mit blockorientierten Magnetbandarchiven gearbeitet wird. Sie ist jedoch obligatorisch zum Lesen von Archiven von zeichenorientierten Bandgeräten (vgl. **f** oben). Die Blockgröße wird automatisch bestimmt beim Lesen von Bändern (Schlüsselbuchstaben **x** und **t**).

l Verweis. Dies veranlaßt *tar* zur Reklamation, wenn nicht alle Verweise auf die geschriebenen Dateien aufgelöst werden können. Falls **l** nicht angegeben ist, werden keine Fehlermeldungen ausgegeben.

m Modifiziert. Dies veranlaßt *tar* die Modifikationszeiten nicht wiederherzustellen. Die Modifikationszeit der Datei ist die Zeit, zu der sie vom Band geholt wurde.

DATEIEN

/dev/rmt?
/tmp/tar*

SIEHE AUCH:

ar(1), cpio(1), ls(1).

DIAGNOSE

Reklamationen über ungültige Schlüsselzeichen und Band-Schreib-/Lese-Fehler.
Reklamationen, wenn nicht genügend Speicherplatz für die Verweis-Tabellen zur Verfügung steht.

FEHLER

Es gibt keine Möglichkeit, das *n*te Auftreten einer Datei anzufordern.
Bandfehler werden ungeschickt behandelt.
Die Option **u** kann langsam sein.
Die Option **b** sollte nicht bei Archiven verwendet werden, die aktualisiert werden sollen. Das aktuelle Magnetbandlaufwerk kann ein "raw" Magnetband nicht zurücksetzen. Wenn das Archiv auf einer Festplattendatei ist, sollte die Option **b** überhaupt nicht verwendet werden, weil die Aktualisierung eines auf Festplatte gespeicherten Archivs es zerstören kann.
Die aktuelle Grenze der Dateinamenslänge ist 100 Zeichen.

BEZEICHNUNG

tee – Standardeingabe kopieren

ÜBERSICHT

tee [**–i**] [**–a**] [file] ..

BESCHREIBUNG

tee kopiert die Standardeingabe auf die Standardausgabe und erstellt Kopien in den angegebenen Dateien.

–i Ignoriert Unterbrechungen;

–a Bewirkt, daß die Ausgabe in den Dateien ans Ende angehängt wird, anstatt die Dateien zu überschreiben.

BEZEICHNUNG

telnet – Benutzerschnittstelle für das TELNET–Protokoll

ÜBERSICHT

telnet[*host*[*port*]]

BESCHREIBUNG

Telnet ermöglicht die Kommunikation mit einem anderen Host mit Hilfe des **TELNET**-Protokolls. Wenn *telnet* ohne Argumente aufgerufen wird, begibt es sich in den Kommando-Modus, der durch die Eingabe-Aufforderung *telnet*> angezeigt wird. In diesem Modus können die unten aufgeführten Kommandos eingegeben und ausgeführt werden. Wenn *telnet* mit Argumenten aufgerufen wird, führt es das *open*–Kommando mit diesen Argumenten aus (siehe unten).

Sobald eine Verbindung eröffnet ist, begibt sich *telnet* in den Eingabe-Modus. Es hängt von dem entfernten System ab, ob im Eingabe-Modus die zeichenweise oder die zeilenweise Eingabe unterstützt wird.

Bei zeichenweiser Eingabe wird der eingegebene Text meist sofort zur Verarbeitung an den entfernten Host übergeben.

Bei zeilenweiser Eingabe werden alle Eingaben lokal angezeigt und (normalerweise) werden nur vollständig eingegebene Zeilen an den entfernten Host weitergeleitet. Mit dem lokalen Echo-Zeichen (Anfangswert: ^E) kann die lokale Anzeige der Eingabe aus- bzw. eingeschaltet werden (wird meist zur Eingabe von Passwörtern ohne Anzeige der Eingabe benutzt).

Wenn der Schalter *localchars* auf TRUE (EIN) steht, werden sowohl bei zeichenweiser als auch bei zeilenweiser Eingabe die Abbruch-, Unterbrechungs- und Ausgabe-Abbruchzeichen lokal interpetiert und als **TELNET**-Protokollsequenzen an den entfernten Rechner gesendet. Die beiden Optionen **toggle** *autoflush* und **toggle** *autosynch* (siehe unten) bewirken, daß alle nachfolgenden Ausgaben für das Terminal ignoriert werden (bis der entfernte Host die **TELNET**–Sequenz anerkennt), desgleichen alle vorhergehenden Eingaben für das Terminal (bei einem Abbruch oder einer Unterbrechung).

Solange die Verbindung zu dem entfernten Rechner besteht, kann der *telnet*–-Kommando-Modus durch Eingabe des *telnet*–Fluchtzeichens eingeschaltet werden (Anfangswert: ^]). Im Kommando-Modus gelten die üblichen Regeln für das Editieren eingegebener Kommandos.

KOMMANDOS

In der folgenden Liste sind die zulässigen Kommandos aufgeführt. Es müssen nur soviele Zeichen eines Kommandos eingegeben werden, daß es eindeutig identifiziert werden kann. Dies gilt auch für die Argumente der Kommandos **mode, set, toggle** und **display**.

open *host* [*port*]
Es wird eine Verbindung zu dem angegebenen Host eröffnet. Wenn keine Anschlußnummer angegeben ist, versucht *telnet*, über den Standard-Anschluß einen **TELNET**-Server zu ereichen. Als *host* kann entweder ein Hostname (siehe *hosts*(5)) angegeben werden oder eine Internet-Adresse in Punktschreibweise (siehe *inet*(3N)).

close
Die **TELNET**-Sitzung wird beendet. Das Programm kehrt in den Kommando-Modus zurück.

quit
Die **TELNET**-Sitzung wird beendet, desgleichen *telnet* selbst. Ein Dateiendezeichen im Kommando-Modus bewirkt das gleiche.

mode *type*
Für *type* muß *line* (zeilenweise Eingabe) oder *character* (zeichenweise Eingabe) eingegeben werden. Der entfernte Host wird gefragt, ob er mit dem gewünschten Modus einverstanden ist. Wenn der entfernte Host den Modus unterstützt, wird dieser eingeschaltet.

status
Telnet zeigt seinen momentanen Status an, einschließlich des aktuellen Modus und des Partners, mit dem es verbunden ist.

display [*argument...*]
Es werden einige oder alle **set**- und **toggle**-Werte angezeigt (siehe unten).

? [*command*]
Es werden Hilfeinformationen angezeigt. Wenn keine Argumente angegeben werden, gibt *telnet* eine Übersicht der verfügbaren Hilfeinformationen aus. Wenn ein Kommando *command* angegeben ist, gibt *telnet* Hilfeinformation für dieses Kommando aus.

send *arguments*
Es werden ein oder mehrere festgelegte Zeichenfolgen an den entfernten Host geschickt. Die folgenden Argumente können angegeben werden (auch mehrere):

escape
(Anfangswert: ^]) Das momentan gültige *telnet*-Fluchtzeichen wird gesendet.

synch
Die **TELNET-SYNCH**-Sequenz wird gesendet. Der entfernte Rechner ignoriert daraufhin alle bisherigen Eingaben, soweit sie nicht schon eingelesen wurden. Die Se-

quenz wird mit der Eildaten-Funktion (urgent data) des TCP verschickt. (Diese Funktion kann nicht ausgeführt werden, wenn auf dem entfernten Rechner ein 4.2BSD-System läuft. In diesem Fall wird u.U. ein kleines r am Terminal angezeigt.)

brk

(Break = Abbruch) Die **TELNET-BRK**-Sequenz wird gesendet, die für den entfernten Rechner von Bedeutung sein kann.

ip

(Interrupt Process = Prozeßunterbrechung) Die **TELNET-IP**-Sequenz wird gesendet. Sie sollte den entfernten Rechner zum Abbruch des gerade laufenden Prozesses veranlassen.

ao

(Abort Output = Ausgabe-Abbruch) Die **TELNET-AO**-Sequenz wird gesendet. Sie sollte den entfernten Rechner dazu veranlassen, die Ausgabe **von** dem entfernten Rechner **an** das Benutzerterminal abzubrechen.

ayt

(Are You There = Sind Sie da) Die **TELNET-AYT**-Sequenz wird gesendet. Der entfernte Rechner kann darauf antworten, muß es aber nicht.

ec

(Erase Character = Zeichen löschen) Die **TELNET-EC**-Sequenz wird gesendet. Sie sollte den entfernten Rechner dazu veranlassen, das letzte eingegebene Zeichen zu löschen.

el

(Erase Line = Zeile löschen) Die **TELNET-EL**-Sequenz wird gesendet. Sie sollte den entfernten Rechner dazu veranlassen, die aktuelle Eingabezeile zu löschen.

ga

(Go Ahead = Gepufferte Eingabe) Die **TELNET-GA**-Sequenz wird gesendet. Sie ist in aller Regel bedeutungslos für den entfernten Rechner.

nop

(No OPeration = Nulloperation) Die **TELNET-NOP**-Sequenz wird gesendet.

?

Es werden Hilfeinformationen für das **send**-Kommando ausgegeben.

set *argument value*

Mit diesem Kommando können verschiedene *telnet*-Variablen auf einen bestimmten Wert gesetzt werden. Der Argumentwert **off** bewirkt, daß die angegebene Funktion ausgeschaltet wird. Mit dem **display**-Kommando kann der Wert der Variablen angezeigt werden. Die folgenden Variablen sind möglich:

echo

(Anfangswert: ^E) Die angegebene Tastenfolge bewirkt bei zeilenweiser Eingabe, daß die eingegebenen Zeichen lokal angezeigt werden (Normalfall) bzw. daß die Anzeige der Eingabe unterdrückt wird (z. B. bei der Paßworteingabe).

escape

(Anfangswert: ^[) Die angegebene Tastenfolge definiert das *telnet*-Fluchtzeichen und bewirkt den Übergang in den Kommando-Modus (wenn eine Verbindung zu einem entfernten Rechner besteht).

interrupt

Wenn *telnet* sich im Modus *localchars* befindet (siehe **toggle** *localchars* weiter unten) und das Unterbrechungszeichen eingegeben wird, sendet es eine **TELNET-IP**-Sequenz an den entfernten Host (siehe **send** *ip* weiter oben). Der Anfangswert des Unterbrechungszeichens wird auch als **intr**-Zeichen für das Terminal verwendet.

quit

Wenn *telnet* sich im Modus *localchars* befindet (siehe **toggle** *localchars* weiter unten) und das Abbruchzeichen eingegeben wird, sendet es eine **TELNET-BRK**-Sequenz an den entfernten Host (siehe **send** *brk* weiter oben). Der Anfangswert des Abbruchzeichens wird auch als **quit**-Zeichen für das Terminal verwendet.

flushoutput

Wenn *telnet* sich im Modus *localchars* befindet (siehe **toggle** *localchars* weiter unten) und das Ausgabe-Abbruchzeichen eingegeben wird, sendet es eine **TELNET-AO**-Sequenz an den entfernten Host (siehe **send** *ao* weiter oben). Der Anfangswert des Ausgabe-Abbruchzeichens wird auch als **flush**-Zeichen für das Terminal verwendet.

erase

Wenn *telnet* sich im Modus *localchars* befindet (siehe **toggle** *localchars* weiter unten), und wenn bei zeichenweiser Eingabe dieses Zeichen eingegeben wird, sendet *telnet* eine **TELNET-EC**-Sequenz an den entfernten Host (siehe **send** *ec* weiter oben). Der Anfangswert des Zeichenlösch-Zeichens wird auch als **erase**-Zeichen für das Terminal verwendet.

kill

Wenn *telnet* sich im Modus *localchars* befindet (siehe **toggle** *localchars* weiter unten), und wenn bei zeichenweiser Eingabe dieses Zeichen eingegeben wird, sendet *telnet* eine **TELNET-EL**-Sequenz an den entfernten Host (siehe **send** *el* weiter oben). Der Anfangswert des Zeilenlösch-Zeichens wird auch als **kill**-Zeichen für das Terminal verwendet.

eof

Wenn bei zeilenweiser Eingabe dieses Zeichen als erstes in einer Zeile eingegeben wird, sendet *telnet* das Zeichen an den entfernten Host. Der Anfangswert des Dateiende-Zeichens wird auch als **eof**-Zeichen für das Terminal verwendet.

toggle *arguments...*

Mit dem **toggle**-Kommando können verschiedene Schalter ein- oder ausgeschaltet werden (mit TRUE bzw. FALSE). Diese Schalter legen fest, wie *telnet* auf bestimmte Ereignisse reagiert. Die Angabe mehrerer Argumente ist zulässig. Mit dem **display**-Kommando kann der momentane Zustand der Schalter angezeigt werden. Die folgenden Argumente sind erlaubt:

localchars

Wenn dieser Schalter auf TRUE gesetzt ist, werden *flush*-, *interrupt*-, *quit*-, *erase*- und *kill*-Zeichen (siehe **set** weiter oben) lokal akzeptiert und (in aller Regel) in die entsprechende **TELNET**-Steuersequenz umgewandelt (*ao*, *ip*, *brk*, *ec* bzw. *el*; siehe **send** weiter oben). Bei zeilenweiser Eingabe ist dieser Schalter zu Beginn auf TRUE gesetzt, bei zeichenweiser Eingabe auf FALSE.

autoflush

Wenn die Schalter *autoflush* und *localchars* beide auf TRUE stehen, stoppt *telnet* die Ausgabe von Daten auf dem Benutzerterminal, wenn ein *ao*–, *intr*– oder *quit*–Zeichen erkannt und in **TELNET**-Steuersequenzen umgewandelt wird (siehe **set** weiter oben). Der Ausgabestop wird erst wieder aufgehoben, wenn der entfernte Rechner (durch die **TELNET**-Option *Timing Mark*) bestätigt, daß er die **TELNET**-Sequenzen verarbeitet hat. Dieser Schalter steht zu Beginn auf TRUE, falls der Terminalbenutzer nicht das Kommando "stty noflsh" abgesetzt hat, sonst auf FALSE (siehe *stty*(1)).

autosynch

Wenn die Schalter *autosynch* und *localchars* beide auf TRUE stehen, und wenn ein *intr*– oder *quit*–Zeichen eingegeben wird (siehe **set** weiter oben), sendet *telnet* zusätzlich zu der entsprechenden **TELNET**-Sequenz noch die **TELNET–SYNCH**-Sequenz. Dies **sollte** dazu führen, daß der entfernte Rechner ab dann alle vorhergehenden Eingaben ignoriert, solange bis beide **TELNET**-Sequenzen gelesen und bearbeitet wurden. Dieser Schalter steht zu Beginn auf FALSE.

crmod

Mit diesem Schalter kann der Carriage-Return-Modus ein- bzw. ausgeschaltet werden. Wenn dieser Modus eingeschaltet ist, werden die meisten Carriage-Return-Zeichen, die von dem entfernten Host gesendet werden, um ein New-Line-Zeile-Zeichen ergänzt. Dieser Modus hat keine Auswirkung auf die Zeichen, die der Benutzer eingibt, sondern nur auf Zeichen, die von dem entfernten Host gesendet werden. Der Modus kann nur dann sinnvoll eingesetzt werden, wenn der entfernte Host nur Carriage-Return-Zeichen, aber nie New-Line-Zeile-Zeichen sendet. Dieser Schalter steht zu Beginn auf FALSE.

debug

Mit diesem Schalter wird der Testhilfemodus auf Socket-Ebene ein- oder ausgeschaltet (nur sinnvoll für den *Superuser*). Dieser Schalter steht zu Beginn auf FALSE.

options

Mit diesem Schalter wird die Protokollierung einiger interner Verarbeitungsvorgänge des **telnet**-Protokolls ein- bzw. ausgeschaltet, die sich auf **TELNET**-Optionen beziehen. Dieser Schalter steht zu Beginn auf FALSE.

netdata

Mit diesem Schalter wird die Protokollierung aller Netzwerkdaten (im hexadezimalen Format) ein- bzw. ausgeschaltet. Dieser Schalter steht zu Beginn auf FALSE.

?

Es wird eine Übersicht der zulässigen **toggle**-Kommandos ausgegeben.

FEHLER

Zur Unterstützung der Flußkontrolle gibt es kein geeignetes Hilfsmittel.

Auf manchen Host-Systemen muß die Anzeige der Eingabe bei zeilenweiser Eingabe von Hand ausgeschaltet werden.

Die Anzahl der möglichen Voreinstellungen würde die Einrichtung einer *.telnetrc*-Datei rechtfertigen. Eine solche Datei ist aber nicht vorgesehen.

Bei zeilenweiser Eingabe wird das Dateiendezeichen *eof* des Terminals nur dann erkannt (und an das entfernte System weitergesendet), wenn es das erste Zeichen in einer Zeile ist.

BEZEICHNUNG

test – Kommando zur Auswertung von Bedingungen

ÜBERSICHT

test expr
[expr]

BESCHREIBUNG

test bewertet den Ausdruck *expr* und falls sein Wert wahr ist, wird der Endestatus auf Null (wahr) gesetzt; andernfalls wird der Endestatus auf ungleich Null (unwahr) gesetzt; *test* gibt einen Nicht-Null-Endestatus zurück, wenn keine Argumente vorliegen. Wenn Berechtigungen geprüft werden, wird die effektive Benutzernummer des Prozesses verwendet.

Alle Operatoren, Optionen und Klammern (Klammern wie auf der zweiten Zeile der ÜBERSICHT angegeben) müssen separate Argumente für das Kommando *test* sein; normalerweise werden die Argumente durch Leerzeichen getrennt.

Die folgenden Basisprozeduren werden zur Konstruktion von *expr* verwendet:

-r *file*	wahr, wenn *file* existiert und lesbar ist.
-w *file*	wahr, wenn *file* existiert und beschreibbar ist.
-x *file*	wahr, wenn *file* existiert und ablauffähig ist.
-f *file*	wahr, wenn *file* existiert und eine normale Datei ist.
-d *file*	wahr, wenn *file* existiert und ein Dateiverzeichnis ist.
-c *file*	wahr, wenn *file* existiert und eine zeichenorientierte Datei ist.
-b *file*	wahr, wenn *file* existiert und eine blockorientierte Datei ist.
-p *file*	wahr, wenn *file* existiert und eine benannte Pipe (FIFO) ist.
-u *file*	wahr, wenn *file* existiert und ihr s-Bit (Benutzer) gesetzt ist.
-g *file*	wahr, wenn *file* existiert und ihr s-Bit (Gruppe) gesetzt ist.
-k *file*	wahr, wenn *file* existiert und ihr t-Bit gesetzt ist.
-s *file*	wahr, wenn *file* existiert und ihre Größe größer als Null ist.
-t [*fildes*]	wahr, wenn die offene Datei, mit Dateikennzahl *fildes* (standardmäßig 1) einem Terminal zugeordnet ist.

-z *s1*	wahr, wenn die Länge der Zeichenfolge *s1* Null ist.
-n *s1*	wahr, wenn die Länge der Zeichenfolge *s1* ungleich Null ist.
s1 **=** *s2*	wahr, wenn Zeichenfolgen *s1* und *s2* identisch sind .
s1 **!=** *s2*	wahr, wenn Zeichenfolgen *s1* und *s2 nicht* identisch sind.
s1	wahr, wenn *s1 nicht* die Nullzeichenfolge ist.
n1 **-eq** *n2*	wahr, wenn die ganzen Zahlen *n1* und *n2* algebraisch gleich sind. Jeder der Vergleiche **-ne**, **-gt**, **-ge**, **-lt**, und **-le** kann anstelle von **-eq** verwendet werden .

Diese Basisprozeduren können mit den folgenden Operatoren kombiniert werden:

!	einstelliger Negationsoperator.
-a	binärer *und* Operator.
-o	binärer *oder* Operator (**-a** hat höhere Priorität als **-o**).
(expr)	runde Klammern zum Zusammenfassen. Es ist zu beachten, daß runde Klammern in der Shell eine besondere Bedeutung haben und daher quotiert werden müssen.

SIEHE AUCH

find(1), sh(1).

ACHTUNG!

Wenn Sie eine Datei prüfen, die Ihr Eigentum ist, (mit den *-r, -w,* oder *-x* Prüfungen), aber für den Eigentümer die geprüfte Berechtigung nicht gesetzt ist, wird ein Endestatus ungleich Null (unwahr) zurückgegeben, obwohl bei dieser Datei die Bits für *group* oder *other* auf diese Berechtigung gesetzt sind. Der richtige Endestatus wird gesetzt, wenn Sie ein Systemverwalter sind.

Die Operatoren **=** und **!=** haben eine höhere Priorität als die Operatoren **-r** bis **-n** und **=** und **!=** erwarten immer Argumente; daher können **=** und **!=** nicht mit den Operatoren **-r** bis **-n** verwendet werden.

Wenn mehr als ein Argument nach den Operatoren **-r** bis **-n** folgt, wird nur das erste Argument geprüft; die anderen werden ignoriert, es sei denn ein **-a** oder ein **-o** ist das zweite Argument.

BEZEICHNUNG

time – bei einem Kommando die Zeit messen

ÜBERSICHT

time command

BESCHREIBUNG

Das Kommando *command* wird ausgeführt; bei seiner Beendigung gibt *time* die für das Kommando verbrauchte Zeit aus, die im System verbrauchte Zeit und die Zeit zur Ausführung des Kommandos. Zeiten werden in Sekunden angegeben.

Die Zeiten werden auf der Standardfehlerausgabe ausgegeben.

SIEHE AUCH:

times(2) im *Programmer's Reference Manual.*

BEZEICHNUNG

timex – Zeit für Kommando messen, Bericht über Prozeßdaten und Systemaktivität

ÜBERSICHT

timex[options] command

BESCHREIBUNG

Das angegebene Kommando *command* wird ausgeführt; die verbrauchte Zeit, die Benutzerzeit und die im System zur Ausführung verbrauchte Zeit werden in Sekunden angegeben. Wahlweise können Prozeßstatistikdaten für das Kommando *command* und alle seine Söhne aufgelistet oder zusammengefaßt werden, und die gesamte Systemaktivität während der Ausführungszeit kann berichtet werden.

Die Ausgabe von *timex* wird auf die Standardfehlerausgabe geschrieben.

Optionen lauten:

-p Auflisten der Prozeßstatistikdaten für *command* und alle seine Söhne. Die Unteroptionen **f, h, k, m, r** und **t** modifizieren die berichteten Informationen. Die Optionen sind folgende:

- **-f** Angabe der *fork/exec* Anzeige und Spalten mit Systemendestatusangaben in der Ausgabe.
- **-h** Anstelle der mittleren Speicherkapazität wird die gesamt verfügbare CPU-Zeit, die vom Prozeß während seiner Ausführung verbraucht wurde, in Form eines Bruchs angezeigt. Dieser "hog Faktor" wird wie folgt berechnet: (gesamte CPU-Zeit)/(verbrauchte Zeit).
- **-k** Anstelle der Speicherkapazität werden die Gesamt-"kcore"-Minuten angezeigt.
- **-m** Anzeige der mittleren Speicherabzugsgröße (die Standardgröße).
- **-r** Anzeige des CPU-Faktors: (Benutzerzeit/(Systemzeit + Benutzerzeit).
- **-t** Benutzer- und Systemzeiten getrennt anzeigen. Die Anzahl der gelesenen oder geschriebenen Blöcke und die Anzahl der übertragenen Zeichen werden immer angegeben.

-o Bericht über die gesamte Anzahl gelesener oder geschriebener Blöcke und die Gesamtzahl der Zeichen, die von *command* und allen seinen Söhnen übertragen wurden.

-s Bericht der gesamten Systemaktivität (nicht nur auf Grund von *command),* die während der Ausführungszeit von *command* angefallen ist. Alle in *sar*(1) aufgelisteten Datenelemente werden verzeichnet.

SIEHE AUCH

sar(1).

ACHTUNG!

Prozeßdaten, die *command* zugeordnet sind, werden von der Prozeßstatistik-Datei **/usr/adm/pacct** implizit ausgewählt, da der Prozeßstammbaum nicht verfügbar ist. Hintergrundprozesse, die die gleiche Benutzernummer, Terminal-Nummer und Ausführungszeit-Window haben, werden fälschlicherweise einbezogen.

BEISPIELE

Ein einfaches Beispiel:

```
timex -ops sleep 60
```

Eine Terminal-Sitzung beliebiger Komplexität kann durch Zeitmessung einer Sub-Shell gemessen werden:

```
timex -opskmt sh
      session commands
EOT
```

BEZEICHNUNG

toc: dtoc, ttoc, vtoc – Grafikroutinen für Inhaltsverzeichnisse

ÜBERSICHT

dtoc [directory]
ttoc mm-file
vtoc [**-cdhnlmsvn**] [TTOC file]

BESCHREIBUNG

Alle nachstehend aufgeführten Kommandos befinden sich in **/usr/bin/graf** (vgl. *graphics*(1G)).

dtoc Dtoc erstellt ein Text-Inhaltsverzeichnis, TTOC aller Unterdateiverzeichnisse, die in *directory* beginnen (*directory* ist standardmäßig). Die Liste hat einen Eintrag pro Verzeichnis. Die Eintragsfelder von links nach rechts sind Stufennummern, Name des Verzeichnisses und die Anzahl der normalen lesbaren Dateien im Verzeichnis. *Dtoc* ist nützlich zur Ausgabe einer vollen oder teilweisen visuellen Anzeige eines Dateisystems. Bei folgendem Befehl erhalten Sie eine visuelle Anzeige aller lesbaren Verzeichnisse unter /:
dtoc / | vtoc | td

ttoc Ausgabe ist das Inhaltsverzeichnis, das vom Makro .TC *mm*(1) erzeugt wird und in TTOC-Format übersetzt wird. Die Eingabe wird als eine *mm*-Datei angesehen, die die .H -Makrofamilie für Abschnittsköpfe verwendet (vgl. die Software der DOCUMENTER'S WORKBENCH). Wenn kein *file* angegeben ist, wird die Standardeingabe angenommen.

vtoc *Vtoc* erzeugt ein GPS, das einen Hierarchie-Baum eines TTOC beschreibt. Die Ausgabezeichnung besteht aus Kästen, die Text enthalten, und die in einer Baumstruktur verbunden sind. Wenn kein *file* angegeben ist, wird die Standardeingabe angenommen. Jeder TTOC-Eintrag beschreibt einen Kasten und hat die Form:
id [*line-weight*,*line-style*] "*text* " [*mark*]
wobei:

id eine wechselnde Folge von Zahlen und Punkten ist. *id* gibt die Position des Eintrags in der Hierarchie an. *id* **0** ist die Baumwurzel.

line-weight (Linienstärke) ist entweder:
n, normal oder
m, mittel oder
b, fett

line-style (Linienform) ist entweder:
so, durchgezogene Linie
do, punktierte Linie;
dd, punktierte/gestrichelte Linie
da, gestrichelte Linie oder
ld, breit gestrichelte Linie

text ist eine Zeichenreihe in Anführungszeichen. Die Zeichen zwischen den Anführungszeichen werden der Inhalt des Kastens. Um ein Anführungszeichen in einen Kasten zu stellen, muß es codiert werden (\").

mark ist eine Zeichenreihe (Anführungszeichen, wenn Leerzeichen vorhanden sind), mit Punkten, die codiert sind. Die Zeichenfolge wird über die oberste rechte Ecke des Kastens gestellt. Zur Einbeziehung eines Anführungszeichens oder eines Punkts in eine Markierung *mark* müssen diese codiert werden.

Eingabe Beispiel: 1.1 b, da "ABC" DEF
Eingaben können über eine Zeile hinausgehen, indem das Neue-Zeile-Zeichen codiert wird (**\new-line**).

Kommentare werden in **/*,*/** eingeschlossen. Sie können an beliebiger Stelle im TTOC erscheinen.

Optionen:

c Verwendet den Text, wie er eingegeben wurde (Standard ist Großschreibung).

d Verbindet die Kästen mit diagonalen Linien.

h*n* Der horizontale Abstand zwischen Kästen ist *n*% der Kastenbreite.

l Unterdrückt die Kastennummer *id*.

m Unterdrückt die Kastenmarkierung *mark*.

s Kästen dürfen nicht horizontal zusammengefügt werden.

v*n* Der senkrechte Abstand zwischen Kästen ist *n*% der Kastenhöhe.

SIEHE AUCH

graphics(1G).
gps(4) im *Programmer's Reference Manual.*
mm(1) im *Handbuch Dokumentations-Tools.*

BEZEICHNUNG

touch – Zugriffs- und Modifikationszeiten einer Datei aktualisieren

ÜBERSICHT

touch [**-amc**] [mmddhhmm[yy]] files

BESCHREIBUNG

touch bewirkt die Aktualisierung der Zugriffs- und Modifikationszeiten jedes Arguments *files* . Die Datei wird erstellt, wenn sie nicht existiert. Wenn keine Zeit angegeben wird, (vgl. *date*(1)) wird die aktuelle Zeit verwendet. Die Optionen **-a** und **-m** veranlassen, daß das Kommando nur die Zugriffs- oder die Modifikationszeit aktualisiert (Standard ist **-am**). Die Option **-c** verhindert, daß *touch* ohne Angabe die Datei erstellt, falls sie vorher noch nicht existierte.

Der Rückkehrcode von *touch* ist die Anzahl der Dateien, bei denen die Zeit nicht erfolgreich modifiziert werden konnte (einschließlich der Dateien, die nicht existierten und nicht erstellt wurden).

SIEHE AUCH:

date(1).
utime(2) im *Programmer's Reference Manual.*

BEZEICHNUNG

tplot – Grafikfilter

ÜBERSICHT

tplot [–Tterminal [–e raster]]

BESCHREIBUNG

Dieses Kommando liest Plott-Befehle (vgl. *plot*(4)) von der Standardeingabe und erzeugen im allgemeinen auf der Standardausgabe Plott-Befehle, die für ein besonderes Terminal *terminal* geeignet sind. Wenn kein *terminal* angegeben ist, wird die Umgebungsvariable **$TERM** (vgl. *environ*(5)) verwendet. Bekannte *terminals* sind:

300 DASI 300.
300S DASI 300s.
450 DASI 450.
4014 Tektronix 4014.
ver Versatec D1200A. Diese Version von *plot* schreibt ein analysiertes und umgewandeltes Bild in **/usr/tmp/raster$$** und schickt das Ergebnis direkt zum Plotter anstatt zur Standardausgabe. Die Option **–e** verursacht das Schicken einer vorher umgewandelten Datei *raster* zum Plotter.

DATEIEN

/usr/lib/t300
/usr/lib/t300s
/usr/lib/t450
/usr/lib/t4014
/usr/lib/vplot
/usr/tmp/raster$$

SIEHE AUCH:

plot(3X), plot(4), term(5) im *Programmer's Reference Manual.*

BEZEICHNUNG

tput – ein Terminal initialisieren oder die Datenbasis 'Terminfo' abfragen

ÜBERSICHT"

tput [–Ttype] capname [parms ...]

tput [–Ttype] **init**

tput [–Ttype] **reset**

tput [–Ttype] **longname**

BESCHREIBUNG

tput verwendet die Datenbasis *terminfo*(4), um der Shell (vgl. *sh*(1)) die Eigenschaften und Daten von Terminals zur Verfügung zu stellen; weiterhin wird dieses Kommando zum Initialisieren oder Zurücksetzen eines Terminals eingesetzt oder um den ausführlichen Namen des angesprochenen Terminal-Typs abzufragen. *tput* gibt eine Zeichenfolge aus, wenn der Attributwert (*cap*ability *name*) ein Zeichenfolgetyp ist, oder eine ganze Zahl, wenn das Attribut eine ganze Zahl ist. Wenn das Attribut ein Boolescher Typ ist, gibt *tput* nur den Endecode zurück (**0** für TRUE (wahr), wenn das Terminal das Merkmal besitzt, **1** für FALSE (nicht wahr), wenn nicht vorhanden) und erzeugt keine Ausgabe. Vor Verwendung eines auf der Standardausgabe gelieferten Werts sollte der Benutzer den Endecode prüfen ($?, vgl. *sh*(1)), um sicherzustellen, daß der Wert wirklich **0** ist. (Vgl. nachstehend **ENDECODES** und **DIAGNOSE** . (Vgl. *terminfo*(4) bezüglich einer vollständigen Liste von Eigenschaften und dem jeweils zugeordneten *capname*.)

–T*type* Gibt den Typ *type* des Terminals an. Normalerweise ist diese Option nicht erforderlich, weil der Standard der Umgebungsvariablen **TERM** entnommen wird. Wenn **–T** angegeben ist, werden die Shell-Variablen **LINES** (Zeilen) und **COLUMNS** (Spalten) und die Fenstergröße (vgl. *layers*(1)) nicht benutzt.

capname zeigt das Attribut aus der Datenbasis *terminfo*(4) an.

parms Wenn das Attribut eine Zeichenfolge ist, die Parameter hat, werden die Argumente *parms* in die Zeichenfolge sofort eingesetzt. Ein rein numerisches Argument wird dem Attribut als Zahl übergeben.

init Wenn die Datenbasis *terminfo*(4) vorhanden ist und ein Eintrag für das Terminal des Benutzers existiert (vgl. –T*type* oben), wird folgendes eintreten: (1) wenn vorhanden, wird die Initialisierungszeichenfolge des Terminals ausgegeben (**is1**, **is2**, **is3**, **if**, **iprog**), (2) im Eintrag eingegebene Verzögerungen (z.B. New-Line-Zeichen) werden im

Terminal-Treiber gesetzt, (3) Tabulatorenerweiterung wird entsprechend der Eintragsdaten ein- oder ausgeschaltet, und (4) wenn Tabulatoren nicht erweitert sind, werden die standardmäßigen Tabulatoren gesetzt (bei jedem achten Leerzeichen). Wenn ein Eintrag nicht die Daten für eine der vier erwähnten Funktionen enthält, wird diese Funktion ohne weitere Angabe übersprungen.

reset Statt der Ausgabe von Initialisierungszeichenfolgen werden die Rücksetzzeichenfolgen des Terminals ausgegeben, wenn sie vorhanden sind (**rs1**, **rs2**, **rs3**, **rf**). Wenn die Rücksetzzeichenfolgen nicht vorhanden sind, die Initialisierungszeichenfolgen jedoch vorhanden sind, werden diese ausgegeben. Andernsfalls funktioniert **reset** genau wie **init**.

longname Wenn die Datenbasis *terminfo*(4) vorhanden ist und ein Eintrag für das Terminal des Benutzers existiert (vgl. **-T***type* oben), wird der ausführliche Name des Terminals ausgegeben. Der ausführliche Name ist der letzte Name in der ersten Zeile der Terminal-Beschreibung in der Datenbasis *terminfo*(4) (vgl. *term*(5)).

BEISPIELE

tput init Initialisiert das Terminal in der Umgebungsvariable **TERM** gemäß dem Terminal-Typ. Dieses Kommando sollte in der .profile-Datei jedes Benutzers enthalten sein, nachdem die Umgebungsvariable **TERM** exportiert wurde, wie es unter *profile*(4) im Handbuch erläutert wird.

tput -T5620 reset Rücksetzen eines AT&T-Terminals 5620, wobei der Terminal-Typ in der Umgebungsvariable **TERM** überschrieben wird.

tput cup 0 0 Schickt die Folge zur Positionierung des Cursors auf Zeile **0**, Spalte **0** (die obere linke Ecke des Bildschirms, gewöhnlich als Ausgangsposition ("Home") des Cursors bekannt).

tput clear Echo der clear-screen-Zeichenfolge für das aktuelle Terminal (Bildschirm löschen).

tput cols Gibt die Anzahl der Spalten für das aktuelle Terminal an.

tput -T450 cols Gibt die Anzahl der Spalten für das Terminal 450 an.

bold='tput smso'

offbold='tput rmso' Setzen der Shell-Variablen **bold** zum Einschalten des Halbfett-Modus und **offbold** zum Ausschalten diese Modus für das aktuelle Terminal. Darauf kann eine Eingabe-Aufforderung folgen:
echo "${bold}Please type in your name: ${offbold}\c"

tput hc Endecode zeigt an, ob das aktuelle Terminal ein Hardcopy-Gerät ist.

tput cup 23 4 Positioniert den Cursor auf Zeile 23, Spalte 4.

tput longname Zeigt den ausführlichen Namen aus der Datenbasis *terminfo*(4) für den Terminal-Typ an, der in der Umgebungsvariable **TERM** angegeben ist.

DATEIEN

/usr/lib/terminfo/?/*	übersetzte Datenbasis für die Terminalbeschreibung
/usr/include/curses.h	include-Datei von *curses*(3X)
/usr/include/term.h	include-Datei von *terminfo*(4)
/usr/lib/tabset/*	Tabulatoreinstellungen für einige Terminals in einem für die Ausgabe zum Terminal geeigneten Format. (Umschaltfolgen, die Ränder und Tabulatoren setzen); weitere Einzelheiten finden Sie im Abschnitt "Tabulatoren und Initialisierung" von *terminfo*(4)

SIEHE AUCH

stty (1), tabs (1).
profile(4), terminfo(4) im *Programmer's Reference Manual.*
Kapitel 10 im *Programmer's Guide.*

ENDECODES

Wenn *capname* vom Booleschen Typ ist, wird der Wert **0** für TRUE und **1** für FALSE gesetzt.

Wenn *capname* eine Zeichenfolge ist, wird der Wert **0** zurückgegeben, wenn das Merkmal für diesen Terminal-Typ *type* definiert ist (der Wert von *capname* wird auf die Standardausgabe ausgegeben); der Wert **1** wird zurückgegeben, wenn das Merkmal für diesen Terminal-Typ *type* nicht definiert ist (auf die Standardausgabe wird Null ausgegeben).

Wenn *capname* eine ganze Zahl ist, wird immer der Wert 0 zurückgegeben, unabhängig davon, ob *capname* für diesen Bildschirmtyp *type* definiert ist. Zur Feststellung, ob *capname* für diesen Bildschirmtyp *type* definiert ist, muß der Benutzer den Wert auf der Standardausgabe überprüfen. Der Wert **-1** bedeutet, daß *capname* nicht für diesen Bildschirmtyp *type* definiert ist.

Andere Endecodes zeigen einen Fehler an; vgl. **DIAGNOSE** unten.

DIAGNOSE

tput gibt die folgenden Fehlermeldungen aus und setzt die entsprechenden Endecodes.

Endecode	Fehlermeldung
0	-1!(*capname* ist eine numerische Variable, die nicht in der Datenbasis *terminfo*(4) für diesen Terminal-Typ verzeichnet ist z. B. **tput -T450 lines** und **tput -T2621 xmc**)
1	Keine Fehlermeldung wird ausgegeben, vgl. **ENDECODE** oben.
2	Anwendungsfehler
3	Terminal-Typ *type* unbekannt oder keine Datenbasis *terminfo*(4)
4	Merkmal *capname* bei *terminfo*(4) nicht bekannt

BEZEICHNUNG

tr – Zeichen übersetzen

ÜBERSICHT

tr [**-cds**] [string1 [string2]]

BESCHREIBUNG

tr kopiert die Standardeingabe auf die Standardausgabe wobei ausgewählte Zeichen ersetzt oder gelöscht werden. In der Zeichenfolge *string1* gefundene Eingabezeichen werden in die entsprechenden Zeichen von *string2* abgebildet. Beliebige Kombinationen der Optionen **-cds** können verwendet werden:

-c Komplement der Zeichenmenge in *string1* in Hinblick auf die Gesamtheit der ASCII-Zeichen, deren Code zwischen von 001 und 377 oktal liegt.

-d Löscht alle Eingabezeichen in *string1.*

-s komprimiert alle Zeichenfolgen sich wiederholender Ausgabezeichen in *string2* zu Einzelzeichen.

Die folgenden Abkürzungskonventionen können zur Einführung von Zeichenbereichen oder sich wiederholender Zeichen in den Zeichenfolgen verwendet werden:

[a-z] Steht für die Zeichenfolge, deren ASCII-Codes von Zeichen **a** bis **z** inklusiv reichen.

[a*n] Steht für *n* Wiederholungen von **a**. Wenn die erste Ziffer von *n* **0** ist, wird *n* als oktal angesehen; andernfalls wird angenommen, daß *n* dezimal ist. Eine Null oder ein fehlendes *n* wird als groß angesehen; diese Möglichkeit ist zum Auffüllen von *string2* nützlich.

Das Escape-Zeichen \ kann wie in der Shell verwendet werden, um beliebigen Zeichen in einer Zeichenfolge ihre besondere Bedeutung zu nehmen. Außerdem steht \, gefolgt von 1, 2 oder 3 oktalen Ziffern für das Zeichen, dessen ASCII-Code durch diese Ziffern angegeben ist.

BEISPIEL

Das folgende Beispiel erstellt eine Liste aller Worte in *file1* in Form von einem Wort pro Zeile in *file2*, wobei ein Wort maximal aus allen Buchstaben des Alphabets bestehen kann. Zum Schutz gegen Interpretation durch die Shell werden die Zeichenfolgen quotiert. 012 ist der ASCII-Code für das New-Line-Zeile-Zeichen.

```
tr -cs "[A-Z][a-z]" "[\012*]" <file1 >file2
```

SIEHE AUCH

ed(1), sh(1).
ascii(5) im *Programmer's Reference Manual.*

FEHLER

Funktioniert nicht bei ASCII NUL in *string1* oder *string2*; löscht NUL immer in der Eingabe.

BEZEICHNUNG

true, false – Wahrheitswert liefern

ÜBERSICHT

true

false

BESCHREIBUNG

Der Aufruf *true* bewirkt nichts – *true* steht für erfolgreich. Der Aufruf *false* bewirkt nichts – *false* steht für nicht erfolgreich. Sie werden normalerweise in Shell-Prozeduren *sh*(1) verwendet wie zum Beispiel:

```
while true
do
        command
done
```

SIEHE AUCH:

sh(1).

DIAGNOSE

true hat den Endestatus Null, *false* Nicht-Null.

BEZEICHNUNG

tty – den Namen des Terminals abfragen

ÜBERSICHT

tty [-l] [-s]

BESCHREIBUNG

tty gibt den Pfadnamen des Terminals des Benutzers aus.

-l Gibt die synchrone Leitungsnummer aus, an die das Terminal des Benutzers angeschlossen ist, falls es sich um eine aktive synchrone Leitung handelt.

-s Unterdrückt die Ausgabe des Terminal-Pfadnamens, es ist jedoch möglich, den Endecode allein zu überprüfen.

ENDECODES

2 falls ungültige Optionen angegeben wurden,
0 falls die Standardeingabe ein Terminal ist,
1 andernfalls.

DIAGNOSE

"not on an active synchronous line" (nicht auf einer aktiven synchronen Leitung), wenn die Standardeingabe kein synchrones Terminal ist und **-l** angegeben wird.
"not a tty" (kein Terminal), wenn die Standardeingabe kein Terminal ist und **-s** nicht angegeben wurde.

BEZEICHNUNG

umask – Dateierstellungsmaske setzen

ÜBERSICHT

umask [000]

BESCHREIBUNG

Die Dateierstellungsmaske des Benutzers wird auf *000* gesetzt. Die drei oktalen Ziffern beziehen sich auf Lese-/Schreib/Ausführungsberechtigung für Eigentümer *(owner)*, Gruppe *(group)*, und andere *(others)*, (vgl. *chmod*(2) und *umask*(2)). Der Wert jeder angegebenen Ziffer wird von der entsprechenden Ziffer, die bei der Erstellung einer Datei angegeben wird, vom System subtrahiert (vgl. *creat*(2)). Zum Beispiel entfernt **umask 022** die Schreibberechtigung für *group* und *others* (Dateien, die normalerweise mit Modus **777** erstellt sind, erhalten Modus **755**; Dateien, die mit dem Modus **666** erstellt sind, erhalten Modus **644**).

Wenn *000* ausgelassen wird, wird der aktuelle Wert der Maske ausgegeben.

umask wird von der Shell erkannt und ausgeführt.

umask kann in der Benutzerdatei **.profile** (vgl. *profile*(4)) enthalten sein und bei Login aufgerufen werden, um die Berechtigungen des Benutzers beim Erstellen von Dateien und Verzeichnissen automatisch zu setzen.

SIEHE AUCH:

chmod(1), sh(1).
chmod(2), creat(2), umask(2), profile(4) im *Programmer's Reference Manual*.

BEZEICHNUNG

uname – Namen des aktuellen UNIX-Systems ausgeben

ÜBERSICHT

uname [**-snrvma**]

BESCHREIBUNG

uname gibt den aktuellen Systemnamen des UNIX Systems auf die Standardausgabe aus. Es wird meistens zur Bestimmung des Systems benutzt, das man gerade benutzt. Die Optionen veranlassen die Ausgabe von ausgewählten Daten, die von *uname*(2) geliefert werden:

-s Gibt den Systemnamen (Standard) aus.

-n Gibt den Knotennamen aus (der Knotenname ist der Name, unter dem das System in einem Kommunikationsnetz bekannt ist).

-r Gibt das Betriebssystem-Release aus.

-v Gibt die Betriebssystem-Version aus.

-m Gibt den Prozessor-Namen aus.

-a Gibt alle obigen Daten aus.

SIEHE AUCH

uname(2) im *Programmer's Reference Manual.*

BEZEICHNUNG

uniq – sich wiederholende Zeilen in einer Datei melden

ÜBERSICHT

uniq [**-udc** [**+n**] [**-n**]] [input [output]]

BESCHREIBUNG

uniq liest die Eingabedatei und vergleicht nebeneinanderliegende Zeilen. Normalerweise werden die zweiten und folgenden Kopien von sich wiederholenden Zeilen entfernt; der Rest wird in die Ausgabedatei geschrieben. *Input* (Eingabe) und *output* (Ausgabe) müssen immer unterschiedlich sein. Es ist zu beachten, daß sich wiederholende Zeilen nebeneinanderliegen müssen, um gefunden zu werden; vgl. *sort*(1). Wenn die Option **-u** verwendet wird, werden nur die Zeilen, die in der Originaldatei sich nicht wiederholen, ausgegeben. Die Option **-d** gibt an, daß nur eine Kopie von sich wiederholenden Zeilen geschrieben werden soll. Der normale Ausgabemodus ist die Vereinigung der Optionen **-u** und **-d** zur Ausgabe.

Die Option **-c** ersetzt **-u** und **-d** und erzeugt eine Ausgabemeldung im Standardstil, wobei jeder Zeile eine Zahl vorausgestellt ist, die die Häufigkeit des Auftretens angibt.

Die *n* Argumente bestimmen, daß ein am Anfang stehender Teil von jeder Zeile beim Vergleich übersprungen wird:

-n Die ersten *n* Felder zusammen mit allen davor stehenden Leerzeichen werden ignoriert. Ein Feld wird als eine Zeichenfolge definiert, die keine Leerzeichen und Tabulatoren enthält und durch Tabulatoren und Leerzeichen von den angrenzenden Feldern getrennt ist.

+n Die ersten *n* Zeichen werden ignoriert. Felder vor Zeichen werden übersprungen.

SIEHE AUCH

comm(1), sort(1).

BEZEICHNUNG

units – Umrechnungsprogramm

ÜBERSICHT

units

BESCHREIBUNG

units wandelt die in verschiedenen standardisierten Maßen ausgedrückten Mengen in ihre Gegenwerte in anderen Maßen um. Es arbeitet wie folgt interaktiv:

```
you have: inch
you want: cm
        * 2.540000e+00
        / 3.937008e-01
```

Eine Menge wird als eine multiplikative Kombination aus Einheiten angegeben, denen wahlweise ein numerischer Multiplikator vorausgeht. Potenzen werden durch angehängte positive ganze Zahlen angezeigt, die Division durch das übliche Zeichen:

```
you have: 15 lbs force/in2
you want: atm
        * 1.020689e+00
        / 9.797299e-01
```

units führt nur multiplikative Umwandlungen durch; folglich kann es Kelvin in Rankine umwandeln, jedoch nicht Celsius in Fahrenheit. Die meisten geläufigen Einheiten, Abkürzungen und metrischen Präfixe werden erkannt sowie einige wenig übliche Abkürzungen und naturwissenschaftliche Konstanten:

pi	Verhältnis vom Kreisumfang zum Durchmesser,
c	Lichtgeschwindigkeit,
e	Ladung eines Elektrons,
g	Beschleunigung der Schwerkraft,
force	genau wie **g**,
mole	Zahl von Avogadro,
water	Druckhöhe pro Einheit der Wasserhöhe,
au	astronomische Einheit.

Pound (Pfund) wird nicht als eine Maßeinheit erkannt; **lb** wird erkannt. Zusammengesetzte Namen werden zusammen bearbeitet, (z. B. **lightyear**). Britischen Einheiten, die sich von den entsprechenden amerikanischen Einheiten unterscheiden, geht ein br voraus: **brgallon.** Eine vollständige Liste der Einheiten ist mit dem Kommando:

cat /usr/lib/unittab erhältlich.

DATEIEN

/usr/lib/unittab

BEZEICHNUNG

uucp, uulog, uuname - UNIX-nach-UNIX -Systemverbindung

ÜBERSICHT

uucp [options] source-files destination-file (Quelldateien Zieldatei)
uulog [options] **-s**system
uulog [options] system
uulog [options] **-f**system
uuname [**-l**] [**-c**]

BESCHREIBUNG

uucp

uucp kopiert die in *source-files* angegebenen Dateien auf die Datei *destination-file.* Ein Dateiname kann ein Pfadname auf Ihrem Gerät sein oder die Form:

Systemname!Pfadname

aufweisen, wobei *Systemname* einer Liste mit Systemnamen entnommen ist, von der *uucp* Kenntnis hat. *Systemname* kann auch eine Liste von Namen sein wie z. B.

Systemname!Systemname!...!Systemname!Pfadname

In diesem Fall wird versucht, die Datei über den angegebenen Weg zum Ziel zu schicken. In den Abschnitten ACHTUNG! und FEHLER wird auf Einschränkungen hingewiesen. Es ist zu prüfen, ob die Zwischennetzknoten auf der Strecke zum Weiterreichen der Daten bereit sind (siehe Einschränkungen im Abschnitt ACHTUNG!).

Die in *Pfadname* auftretenden Shell-Metazeichen **?**, ***** und **[...]** werden auf dem entsprechenden System erweitert.

Ein Pfadname kann sein:

(1) ein vollständiger Pfadname;

(2) ein Pfadname, dem ein *~user* vorangestellt ist, wobei es sich bei *user* um einen Login-Namen im angegebenen System handelt, der Pfadname wird dann durch das Login-Verzeichnis dieses Benutzers ersetzt;

(3) ein Pfadname, dem *~/destination* vorgestellt ist, wobei *destination* an **/usr/spool/uucppublic** angehängt wird; (ANMERKUNG: Dieses Ziel (destination) wird als Dateiname behandelt, sofern nicht mehr als eine Datei mit diesem Auftrag übertragen werden soll oder das Ziel nicht schon ein Verzeichnis ist. Um sicherzustellen, daß es ein Verzeichnis ist, stellt man dem Ziel ein '/' nach. Beispielsweise würde mit ~/dan/ als Ziel das Verzeichnis /usr/spool/-uucppublic/dan erstellt, wenn dieses noch nicht existiert, und die gewünschte(n) Datei(en) in das betreffende Verzeichnis geschrieben).

(4) allen sonstigen Angaben wird das aktuelle Verzeichnis vorangestellt.

Ergibt sich als Resultat ein fehlerhafter Pfadname für das dezentrale System, ist die Kopie erfolglos. Wenn die *Zieldatei* ein Verzeichnis ist, wird der letzte Teil des Namens der *Quelldatei* verwendet.

uucp bewahrt die jeweilige Ausführungserlaubnis bei der Übertragung und vergibt 0666 Lese- und Schreiberlaubnisse (vgl. *chmod*(2)).

Die folgenden Optionen werden von *uucp* interpretiert:

-c Die lokale Datei nicht in das Spool-Verzeichnis zur Übertragung auf das Remote-Gerät kopieren. (Standard).

-C Die Kopie der lokalen Dateien unbedingt in das Spool- Verzeichnis zur Übertragung schreiben.

-d Alle erforderlichen Verzeichnisse für die Dateikopie erstellen (Standard).

-f Keine Zwischen-Verzeichnisse für die Dateikopie erstellen.

-g*grade* *Grade* ist ein Einzelbuchstabe/zahl; niedrigere ASCII-Zeichen bewirken früheres Schicken des Auftrags während einer bestimmten Verbindung.

-j Die ASCII- Zeichenfolge für die Auftragskennung auf der Standardausgabe ausgeben. Diese Auftragskennung kann von *uustat* verwendet werden, um den Status eines Auftrags abzufragen oder einen Auftrag zu beenden.

-m Nach Beenden des Kopierens dem Auftraggeber eine entsprechende Nachricht schicken.

-n*user* Den *user* am entfernten System davon benachrichtigen, daß eine Datei abschickt wurde.

-r Nicht mit der Dateiübertragung anfangen, sondern Auftrag lediglich in eine Warteschlange setzen.

-s*file* Status der Übertragung in *file* notieren. Es ist zu beachten, daß *file* ein vollständiger Pfadname sein muß.

-x*debug_level*
Fehlersuch-Ausgabe auf der Standardausgabe erzeugen. Der *debug_level* ist eine Zahl zwischen 0 und 9; höhere Zahlen liefern genauere Daten. (Fehlersuche steht nicht zur Verfügung, wenn **uucp** mit-DSMALL übersetzt wurde.)

uulog

uulog fragt eine Protokolldatei von *uucp* oder *uuxqt*-Transaktionen in einer Datei **/usr/spool/uucp/.Log/uucico/***system* oder **/usr/spool/uucp/.Log/uuxqt/***system* ab.

Die Optionen veranlassen *uulog*, Protokolldaten auszugeben:

-s*sys*	Daten über die Datei-Übertragungsarbeiten in Zusammenhang mit dem System *sys* ausgeben.
-f*system*	Ruft "tail-f" für das Dateiübertragungsprotokoll von *system* auf. (Zum Verlassen dieser Funktion die Taste BREAK betätigen.) Andere in Verbindung mit obigen eingesetzten Optionen:
-x	In der Protokolldatei *uuxqt* für das angegebene System nachsehen.
-*number*	Zeigt an, daß ein "tail"-Kommando mit *number* Zeilen ausgeführt werden soll.

uuname

uuname listet die Namen von *uucp* bekannten Systemen auf. Die Option -c liefert die Namen von Systemen, die *cu* bekannt sind. (Wenn Ihr Gerät nicht verschiedene *System*-Dateien für *cu* und *uucp* verwendet, sind beide Listen gleich. Vgl. Datei *Sysfiles*). Die Option -l liefert den lokalen Systemnamen.

DATEIEN

/usr/spool/uucp	Spool-Verzeichnisse
/usr/spool/uucppublic/*	Öffentliches Verzeichnis zum Empfangen und Senden (**/usr/spool/uucppublic**)
/usr/lib/uucp/*	Andere Daten und Programmdateien

SIEHE AUCH:

mail(1), uustat(1C), uux(1C), uuxqt(1M).
chmod (2) im *Programmer's Reference Manual.*

ACHTUNG!

Der Bereich von dezentral zugreifbaren Dateien kann und (sollte normalerweise aus offensichtlichen Sicherheitsgründen) stark eingeschränkt werden. Es wird Ihnen höchstwahrscheinlich nicht möglich sein, Dateien mit Pfadnamen zu holen. Bitten Sie eine verantwortliche Person im Remote-System, Ihnen diese Dateien zu schicken. Aus denselben Gründen können Sie wahrscheinlich auch keine Dateien an willkürliche Pfadnamen schicken. Entsprechend ihrer Verteilung sind die Dateien, auf die dezentral zugegriffen werden kann, die Dateien, deren Name mit **/usr/spool/uucppublic** (gleichwertig mit ~/) beginnt.

Alle von *uucp* empfangenen Dateien sind im Besitz von *uucp*.
Die Option **-m** ist nur bei Verwendung einer einzelnen Datei geeignet. Bei mehreren Dateien, die mit Hilfe der Shell-Sonderzeichen **? * [...]** bezeichnet sind, wird die Option **-m** nicht aktiviert.

uucp ist möglicherweise nicht zu der vorherigen Version von *uucp* kompatibel. Wird diese Funktion eingesetzt, müssen alle Systeme auf dem Weg dieselbe Version von *uucp* aufweisen.

FEHLER

Geschützte Dateien und Dateien in geschützten Verzeichnissen, die dem Auftraggeber gehören, können mit *uucp* geschickt werden. Ist der Auftraggeber jedoch Root und das Verzeichnis nicht von "Anderen" durchsuchbar oder ist die Datei nicht von "Anderen" lesbar, bleibt die Anfrage erfolglos.

BEZEICHNUNG

uustat – uucp- Statusabfrage und Auftragssteuerung

ÜBERSICHT

uustat [**-a**]
uustat [**-m**]
uustat [**-p**]
uustat [**-q**]
uustat [**-k***jobid*]
uustat [**-r***jobid*]
uustat [**-s***system*] [**-u***user*]

BESCHREIBUNG

uustat zeigt den Status vorher angegebener *uucp* Kommandos an bzw. hebt diese auf oder informiert über den allgemeinen Status der *uucp*-Anschlüsse anderer Systeme. Es kann jeweils nur eine der folgenden Optionen pro Kommandoaufruf angegeben werden:

-a Alle Aufträge in der Warteschlange ausgeben.

-m Status der Zugriffsmöglichkeit auf allen Geräten melden.

-p Ein "ps -flp" für alle Prozeß-Nummern ausführen, die sich in den Sperrdateien befinden.

-q Für jedes Gerät die Liste der Aufträge in der Warteschlange ausgeben. Wenn eine Statusdatei für das Gerät vorhanden ist, werden Datum, Uhrzeit und Status der Datei gemeldet. Wenn außerdem neben der Dateinummer von C- oder X-Dateien eine Zahl in Klammern () erscheint, ist dies das Alter der ältesten Datei C./X. (in Tagen) für dieses System. Das Wiederholungsfeld gibt die Anzahl der Stunden bis zum nächsten möglichen Aufruf an. Das Anzahlfeld stellt die Anzahl erfolgloser Versuche dar.
ANMERKUNG: Bei Systemen mit einer geringeren Anzahl noch auszuführender Aufträge könnte dies 30 Sekunden Echtzeit oder länger für die Ausführung brauchen. Ein Beispiel für die durch die Option **-q** erzeugte Ausgabe:

eagle 3C 04/07-11:07 NO DEVICES AVAILABLE
mh3bs3 2C 07/07-10:42 SUCCESSFUL

Mit der obigen Ausgabe wird mitgeteilt, wieviele Kommandodateien für jedes System warten. Jede Kommandodatei kann Null oder mehrere zu schickende Dateien enthalten (Null bedeutet: System aufrufen und prüfen, ob Arbeit auszuführen ist). Datum und Uhrzeit beziehen sich auf die vorhergehende interaktive Verbindung mit dem System, und anschließend wird der Status der interaktiven Verbindung angegeben.

-k*jobid* Den *uucp* Auftrag mit Auftragsnummer *jobid* abbrechen. Der abgebrochene *uucp* Auftrag muß von der Person stammen, die das Kommando *uustat* ausgibt, sofern nicht eine der Personen der Systemverwalter ist.

-r*jobid* *jobid*. aktualisieren. Bei den zu *jobid* gehörenden Dateien werden die Dateizeiten aktualisiert, so daß die Zeit ihrer Änderung auf die aktuelle Uhrzeit umgestellt wird. Dadurch wird der Löschdämon daran gehindert, den Auftrag vor Ablauf der vom Dämon auferlegten Zeitgrenze zu löschen.

Bei *uustat* kann jede oder beide der nachstehenden Optionen angegeben werden:

-s*sys* Den Status aller *uucp* Aufträge für das dezentrale System *sys* melden.

-u*user* Den Status aller vom Benutzer *user* ausgegebenen *uucp* Aufträgen melden.

Die Ausgabe für die Optionen **-s** und **-u** hat folgendes Format:

eaglen0000 4/07-11:01:03 (POLL)
eagleN1bd7 4/07-11:07 Seagledan522 /usr/dan/A
eagleC1bd8 4/07-11:07 Seagledan59 D.3b2al2ce4924
4/07-11:07 Seagledanrmail mike

Bei den beiden obigen Optionen ist das erste Feld das *jobid* des Auftrags. Darauf folgen Datum/Uhrzeit. Das nächste Feld ist entweder ein 'S' oder ein 'R', je nachdem, ob der Auftrag Schicken oder Anfordern einer Datei beinhaltet. Darauf folgt die Nummer des Benutzers, der den Auftrag in die Warteschlange gegeben hat. Das nächste Feld enthält die Größe der Datei, oder im Falle einer dezentralen Ausführung (das Kommando für Remote-Post lautet *rmail*) den Namen des Kommandos. Wenn die Größe in diesem Feld erscheint, wird gleichzeitig auch der Dateiname angegeben. Hierbei kann es sich entweder um einen vom Benutzer angegebenen Namen oder um einen internen Namen handeln (z. B. D.3b2alce4924), der für Dateien in Verbindung mit dezentralen Ausführungen erstellt wurde. (*rmail* in diesem Beispiel).

Wenn keine Optionen angegeben werden, gibt *uustat* den Status aller vom aktuellen Benutzer ausgegebenen *uucp* Anfragen aus.

DATEIEN

/usr/spool/uucp/* Spool-Verzeichnisse

SIEHE AUCH:

uucp(1C).

BEZEICHNUNG

uuto, uupick – public UNIX-to-UNIX -Systemdatei-Kopie

ÜBERSICHT

uuto [options] source-files destination
uupick [**-s** system]

BESCHREIBUNG

uuto schickt *source-files* (Quelldateien) an *destination* (das Ziel). *uuto* verwendet das *uucp*(1C) Kommando zum Senden von Dateien, während es dem lokalen System die Steuerung des Dateizugriffs gestattet. Ein Quelldateiname ist ein Pfadname in Ihrem Gerät. Das Ziel hat die Form:

system!*user*

Hierbei wird *system* einer Liste mit Systemnamen entnommen, die *uucp* kennt (vgl. *uuname*). *User* ist der Login-Name einer Person im angegebenen System.

Zwei *options* stehen zur Verfügung:

-p Vor der Übertragung die Quelldatei in das Spool-Verzeichnis kopieren.

-m Nach Fertigstellung der Kopie dem Absender Post zusenden.

Die Dateien (oder Sub-Bäume bei Angabe von Verzeichnissen) werden nach PUBDIR im *system* geschickt, wo PUBDIR ein in der *uucp* -Quelle definiertes öffentliches Verzeichnis ist. Dieses Verzeichnis ist standardmäßig /usr/spool/uucppublic. Die Dateien werden nach

PUBDIR/receive/*user*/*mysystem*/files gesendet.

Der vorgesehene Empfänger wird mit *mail*(1) von der Ankunft der Dateien benachrichtigt.

Uupick nimmt die an den Benutzer gesendeten Dateien entweder an oder weist sie zurück. *Uupick* sucht insbesondere PUBDIR auf Dateien ab, die für den Benutzer bestimmt sind. Für jeden gefundenen Eintrag (Datei oder Verzeichnis) wird folgende Nachricht auf der Standardausgabe ausgegeben:

from *system*: [file *file-name*] [dir *dirname*] ?

Uupick liest dann eine Zeile der die Datei behandelt werden soll.

<new-line> Zum nächsten Eintrag weitergehen.

d Löschen.

m [*dir*]	In das angegebene Verzeichnis *dir* verlegen. Ist *dir* nicht als vollständiger Pfadname angegeben (in dem $HOME zulässig ist), wird das Ziel relativ zum aktuellen Verzeichnis angenommen. Wird kein Ziel angegeben, ist das aktuelle Verzeichnis der Standard.
a [*dir*]	Wie **m**, jedoch Verlegen aller vom *system* geschickten Dateien.
p	Den Inhalt der Datei ausgeben.
q	Stopp.
EOT (control-d)	Wie **q**.
!*command*	Zum Ausführen von *command* in die Shell wechseln.
*	Zusammenfassung der Kommandos ausgeben.

Wenn *unpick* mit der Option **-s***system* aufgerufen wird, sucht es lediglich PUBDIR nach von *system* geschickten Dateien ab.

DATEIEN

PUBDIR /usr/spool/uucppublic öffentliches Verzeichnis

SIEHE AUCH:

mail(1), uucp(1C), uustat(1C), uux(1C).
uucleanup(1M) im *System Administrator's Reference Manual.*

ACHTUNG!

Zum Senden von Dateien, die mit einem Punkt beginnen (z.B. .profile), muß der Punkt explicit angegeben werden. Zum Beispiel sind: .profile, .prof*, .profil? richtig, jedoch *prof*, ?profile falsch.

BEZEICHNUNG

uux – UNIX-to-UNIX Kommando-Ausführung

ÜBERSICHT

uux [options] command-string

BESCHREIBUNG

uux sammelt null oder mehrere Dateien von verschiedenen Systemen, führt ein Kommando in einem angegebenen System aus und schickt dann die Standardausgabe an eine Datei in einem angegebenen System.

ANMERKUNG: Aus Sicherheitsgründen begrenzen die meisten Installationen die Liste ausführbarer Kommandos unter *uux*. Die meisten lassen nur den Empfang von Post zu (vgl. *mail*(1)). (Zulassungen für die dezentrale Ausführung sind in **/usr/lib/uucp/Permissions** definiert.)

Die Kommandofolge *command-string* besteht aus einem oder mehreren Argumenten, die wie eine Shell-Kommandozeile aussehen, wobei den Kommando- und Dateinamen hier jedoch *system-name*!. vorangestellt sein kann. Ein Null-*system-name* wird als das lokale System angesehen.

Ein Dateiname kann sein:

(1) ein vollständiger Pfadname;

(2) ein Pfadname mit vorangestelltem *~xxx* , wobei *xxx* ein Login-Name im angegebenen System ist und durch das Login-Verzeichnis des betreffenden Benutzers ersetzt wird;

(3) allen anderen Angaben wird das aktuelle Verzeichnis vorangestellt.

Zum Beispiel holt das Kommando

uux "!diff usg!/usr/dan/file1 pwba!/a4/dan/file2 > !~/dan/file.diff"

die Dateien *file1* und *file2* von den Rechnern "usg" und "pwba", führt ein Kommando *diff*(1) aus und schreibt dann die Ergebnisse in *file.diff* im lokalen Verzeichnis PUBDIR/dan/.

Besondere Shell-Zeichen wie z. B. **< >;|** müssen quotiert werden, entweder durch Quotierung des gesamten *command-string* oder der Sonderzeichen als individuelle Argumente.

uux versucht, alle Dateien zum ausführenden System zu bringen. Bei Dateien, die Ausgabedateien sind, muß der Dateiname in runden Klammern eingeschlossen werden. So erhält beispielsweise das Kommando

uux a!cut-f1 b!/usr/file \(c!/usr/file\)

die Datei /usr/file von System "b" und schickt sie an System "a", führt ein Kommando *cut* auf diese Datei aus und schickt das Ergebnis des Kommandos *cut* an das System "c".

uux benachrichtigt Sie, wenn das angeforderte Kommando im dezentralen System nicht zugelassen wurde. Diese Nachricht kann mit der Option **-n** abgeschaltet werden. Die Antwort kommt über Remote-Post vom dezentralen Gerät.

Die folgenden *options* werden von *uux* interpretiert:

- Die Standardeingabe von *uux* wird zur Standardeingabe für den *command-string.*

-a*name* *name* als Benutzer-Kennung verwenden und damit die Benutzer- Nummer des Aufrufers ersetzen. (Eine dementsprechende Nachricht wird an den Benutzer ausgegeben.)

-b Die jeweils gelieferte Standardeingabe an das Kommando *uux* zurückgeben, wenn der Ausgangsstatus ungleich Null ist.

-c Die lokale Datei nicht in das Spool-Verzeichnis zur Übertragung kopieren (Standard).

-C Die Kopie der lokalen Dateien unbedingt in das Spool-Verzeichnis zur Übertragung schreiben.

-g*grade* *Grade* ist ein Einzelbuchstabe/Zahl; niedrigere ASCII-Zeichen bewirkten früheres Schicken des Auftrags während einer bestimmten Verbindung.

-j Die ASCII-Zeichenfolge für jobid, die die Auftragskennung ist, auf der Standardausgabe ausgeben. Diese Auftragskennung kann von *uustat* verwendet werden, um den Status eines Auftrags abzufragen oder einen Auftrag zu beenden.

-n Den Benutzer nicht benachrichtigen, wenn das Kommando erfolglos ist.

-p Wie -: Die Standardeingabe für *uux* wird zur Standardeingabe für den *command-string* gemacht.

-r Noch nicht mit der Dateiübertragung beginnen, den Auftrag lediglich in die Warteschlange setzen.

-s*file* Status der Übertragung in *file* melden.

-x*debug_level*
Eine Fehlersuch-Ausgabe auf der Standardausgabe erzeugen. Der *debug_level* ist eine Zahl zwischen 0 und 9; höhere Zahlen liefern genauere Daten.

-z Eine Erfolgsmeldung an den Benutzer absetzen.

DATEIEN

/usr/lib/uucp/spool	Spool-Verzeichnisse
/usr/lib/uucp/Permissions	Zulassungen für dezentrale Ausführung
/usr/lib/uucp/*	Andere Daten und Programme

SIEHE AUCH:

cut(1), mail(1), uucp(1C), uustat(1C).

ACHTUNG!

Nur das erste Kommando einer Shell-Pipeline kann einen *system-name*! aufweisen. Alle anderen Kommandos werden im System des ersten Kommandos ausgeführt.
Die Anwendung des Shell-Metazeichens * führt wahrscheinlich nicht zum gewünschten Ergebnis. Die Shell-Umlenkzeichen << und >> werden nicht unterstützt.

Die Ausführung von Kommandos in Remote-Systemen findet in einem Ausführungs-Verzeichnis statt, das als das *uucp* System bezeichnet wird. In dieses Verzeichnis werden alle für die Ausführung benötigten Dateien gebracht, wenn sie nicht bereits auf dem betreffenden Rechner sind. Daher muß der einfache Dateiname (ohne Pfad- oder Gerätverweis) innerhalb der *uux* Anfrage eindeutig sein! Das folgende Kommando wird NICHT ausgeführt:

```
uux "a!diff b!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

Dagegen wird aber das Kommando

```
uux "a!diff a!/usr/dan/xyz c!/usr/dan/xyz > !xyz.diff"
```

(Sofern *diff* ein zulässiges Kommando ist.)

FEHLER

Geschützte Dateien und Dateien in geschützten Verzeichnissen, die dem Auftraggeber gehören, können unter Verwendung von *uux* in Kommandos geschickt werden. Ist der Auftraggeber root und das Verzeichnis nicht von "Anderen" durchsuchbar, bleibt der Auftrag allerdings erfolglos.

BEZEICHNUNG

vi – bildschirm-orientierter Editor auf Grundlage von ex

ÜBERSICHT

vi [**-t** tag] [**-r** file] [**-L**] [**-w***n*] [**-R**] [**-x**] [**-C**] [**-c** command] file ...
view [**-t** tag] [**-r** file] [**-L**] [**-w***n*] [**-R**] [**-x**] [**-C**] [**-c** command] file ...
vedit [**-t** tag] [**-r** file] [**-L**] [**-w***n*] [**-R**] [**-x**] [**-C**] [**-c** command] file ...

BESCHREIBUNG

vi (visuell) ist ein bildschirm-orientierter Texteditor auf Grundlage des Zeileneditors *ex*(1). Es ist möglich, den Kommando-Modus von *ex* von *vi* aus und umgekehrt zu verwenden.

Die visuellen Kommandos werden auf dieser Handbuchseite beschrieben; das Setzen von Optionen (wie automatische Zeilennumerierung und automatisches Anlegen einer neuen Ausgabezeile) und alle ex(1)-Zeileneditor-Kommandos werden auf der ex(1)-Handbuchseite beschrieben.
Bei der Anwendung von *vi* werden die an der Datei vorgenommenen Änderungen jeweils am Bildschirm angezeigt. Die Position der Cursor auf dem Bildschirm zeigt die Position innerhalb der Datei an.

AUFRUF-OPTIONEN

Die folgenden Aufruf-Optionen werden von *vi* interpretiert(bzgl. der Optionen, die früher dokumentiert wurden und hier nicht auftauchen, siehe unter NOTES am Ende dieser Handbuchseite):

-t *tag* — Die Datei editieren, die das *tag* (Identifizierungskennzeichen) enthält und den Editor auf die Definitionsstelle des *tag* positionieren.

-r*file* — Die Datei *file* nach einem Editor- oder Systemabsturz editieren. Stellt die Version des *file* wieder her, die sich beim Systemabsturz im Puffer befand.

-L — Zeigt eine Liste aller Dateien, die aufgrund eines System- oder Editorabsturzes gesichert wurden.

-w*n* — Die Standard-Fenstergröße auf den Wert *n* setzen. Dies ist nützlich bei einer langsamen Übertragungsgeschwindigkeit.

-R — Nur-Lese-Modus; die Option **readonly** wird gesetzt, wodurch unbeabsichtigtes Überschreiben der Datei verhindert wird.

-x — Verschlüsselungsoption; wird diese Option benutzt, dann simuliert *vi* das **X**-Kommando des *ex(1)* und verlangt vom Benutzer die Eingabe eines Schlüssels. Dieser Schlüssel wird benutzt um Text mit dem Algorith-

mus *crypt(1)* zu ver- oder entschlüsseln. Das **X** Kommando entscheidet, ob der eingelesene Text verschlüsselt ist oder nicht. Die temporäre Puffer-Datei wird ebenfalls über eine transformierte Form des eingegebenen Schlüssels in der **-x** -Option verschlüsselt. Beachten Sie auch den Abschnitt **ACHTUNG** am Ende dieser Handbuchseite.

-C Verschlüsselungsoption; ähnlich der **-x** -Option, außer, daß der *vi* das **C**-Kommando von *ex(1)* simuliert. Das **C**-Kommando ähnelt dem **X**-Kommando des *ex(1),* außer, daß davon ausgegangen wird, daß der eingelesene Text verschlüsselt ist.

-c *command* Editor-Sitzung mit dem angegebenen Editor- *Commando* beginnen (meist ein Such- oder Positionierkommando).

Das Argument *file* zeigt eine oder mehrere Dateien an, die editiert werden sollen.

Der Aufruf von *view* hat dieselbe Wirkung wie *vi,* außer daß die **readonly** -Option gesetzt ist.

Der Aufruf von *vedit* ist für Einsteiger gedacht. Das **report**-Bit wird auf 1 gesetzt, die **showmode** und **novice** Optionen werden gesetzt und **magic** wird ausgeschaltet. Mit diesen Standardeinstellungen wird das Erlernen des Editors erleichtert.

VI-MODI

Command Normaler und Anfangsmodus. Bei Beendigung anderer Modi wird wieder in diesen Kommando-Modus zurückgegangen. Mit der Taste ESC wird ein Teilkommando abgebrochen.

Input wird angeschaltet durch die Optionen **a i A I o O c C s S R**. Beliebiger Text kann dann eingegeben werden. Im Normalfall wird der Eingabemodus mit dem ESC-Zeichen, im Ausnahmefall mit Unterbrechung (interrupt) beendet.

Last line Lesen der Eingabe für **:** **/**, **?** oder **!**; mit CR wird die Eingabe beendet; durch einen Interrupt wird die Beendigung unterbrochen.

ZUSAMMENFASSUNG DER KOMMANDOS

Beispiele für Kommandos

← ↓ ↑ →	Pfeiltasten steuern den Cursor
h j k l	Gleicher Effekt wie die Pfeiltasten
i*abc*ESC	Text *abc* einfügen
cw*new*ESC	Wort in *new* ändern
ea*s*ESC	Wort in Plural setzen (am Ende des Wortes s anhängen; Eingabe-Modus verlassen)
x	Ein Zeichen löschen
dw	Ein Wort löschen
dd	Eine Zeile löschen
3dd	3 Zeilen löschen
u	Letzte Änderung rückgängig machen
ZZ	vi verlassen, dabei Änderungen speichern
:q!CR	Abbrechen ohne Speichern der Änderungen
/*text***CR**	*text* suchen
^U ^D	Bildschirmanzeige vor- bzw. zurückrollen
:*cmd***CR**	Jedes beliebige ex- oder ed-Kommando

Zahlenwerte vor vi-Kommandos

Einigen Kommandos können Zahlenwerte vorangestellt werden. Sie werden auf eine der folgenden Arten und Weisen interpretiert:

Zeilen-/Spalten-Nummer	**z G |**
Scroll-Größe	**^D ^U**
Wiederholungseffekt	die meisten der restlichen Angaben

Unterbrechen, Löschen

ESC	Ein Einfüge- oder unvollständiges Kommando beenden
DEL	Unterbrechungen (Löschen oder "Ausradieren")

Dateibearbeitung

ZZ	Falls die Datei geändert wurde, schreiben und beenden, sonst beenden
:wCR	Änderungen zurückschreiben
:w!CR	Erzwungenes schreiben, falls Erlaubnis ursprünglich nicht gültig
:qCR	Beenden
:q!CR	Beenden ohne Speichern der Änderungen
:e*name***CR**	Datei*name* editieren
:e!CR	Neueditieren, Änderungen löschen

:e + *name***CR**	Editieren am Datei-Ende beginnen
:e +*n***CR**	Editieren auf Zeile *n* beginnen
:e #CR	Alternative Datei editieren
:e! #CR	Alternative Datei editieren, Änderungen verwerfen
:w *name***CR**	Datei *name* schreiben
:w! *name***CR**	Datei *name* überschreiben
:sh CR	Shell ausführen, dann zurückkehren
:!*cmd***CR**	*cmd* ausführen, dann zurückkehren
:n CR	Nächste Datei aus der Argumentenliste arglist editieren
:n *args***CR**	Neue arglist angeben
^G	Aktuelle Datei und Zeile anzeigen
:ta *tag***CR**	Cursor auf *tag* positionieren

Im allgemeinen kann jedes *ex* oder *ed* Kommando (wie zum Beispiel *substitute* oder *global*) angegeben werden, wenn ihm ein Doppelpunkt vorausgestellt wird und ein **CR** folgt.

Positionieren innerhalb einer Datei

^F	Einen Bildschirm vorwärtsgehen
^B	Einen Bildschirm zurückgehen
^D	Einen halben Bildschirm vorwärtsrollen
^U	Einen halben Bildschirm zurückrollen
nG	Anfang der Zeile (*n*) (standardmäßig an das Ende)
/pat	Nächste zum Muster *pat* passende Zeile
?*pat*	Vorherige zum Muster *pat* passende Zeile
n	Letztes /- oder ?-Kommando
N	Letztes /- oder ?-Kommando umkehren
/pat/+n	*nte* Zeile nach *pat*
?*pat***?−***n*	*nte* Zeile vor *pat*
]]	Nächster Abschnitt/nächste Funktion
[[	Vorheriger Abschnitt/vorherige Funktion
(	Satzanfang
)	Satzende
{	Absatzanfang
}	Absatzende
%	Passenden (, {,) oder } suchen

Einstellen des Bildschirms

^L	Löschen und neuschreiben des Fensters
^R	Löschen und neuschreiben, falls **^L** die → -Taste ist
zCR	Neuschreiben, aktuelle Zeile oben im Fenster
z−CR	Neuschreiben, aktuelle Zeile unten im Fenster
z.CR	Neuschreiben, aktuelle Zeile in Fenstermitte

/pat/z-CR	Zeile *pat* unten ins Fenster
zn.CR	*n* -zeiliges Fenster verwenden
^E	Fenster um 1 Zeile vorwärtsrollen
^Y	Fenster um 1 Zeile zurückrollen

Markieren und Rückkehr

`` `` ```	Cursor auf vorherigen Kontext positionieren
` ' `	Cursor auf erstes Nicht-Leerzeichen in der Zeile
m*x*	Aktuelle Position mit Buchstaben *x* markieren
`*x*	Cursor auf Markierung x positionieren *x*
´*x*	Cursor auf erstes Nicht-Leerzeichen in der Zeile

Zeilen-Positionieren

H	Auf oberste Bildschirmzeile
L	Auf letzte Bildschirmzeile
M	Auf mittlere Bildschirmzeile
+	Auf nächste Zeile, erstes Zeichen
-	Auf vorherige Zeile, erstes Zeichen
CR	Return (Wagenrücklauf), das gleiche wie +
↓ or j	Nächste Zeile, dieselbe Spalte
↑ or k	Vorherige Zeile, dieselbe Spalte

Zeichen-Positionieren

^	Auf erstes Nicht-Leerzeichen
0	An Zeilenanfang
$	Zeilenende
h	oder →
l	oder ←
^H	das gleiche wie ← (Backspace)
space	das gleiche wie → (Leerzeichen)
f*x*	Vorwärtssuche nach *x*
F*x*	Rückwärtssuche nach *x*
t*x*	Vorwärts bis *x*
T*x*	Rückwärts bis *x*
;	Letztes **f, F, t** oder **T** wiederholen
,	Umkehrung von ;
n|	Auf Spalte *n*
%	Passende ( ) { oder } suchen
T}	

Wörte, Sätze, Abschnitte

w	ein Wort vorwärts
b	ein Wort zurück

e	Wortende
)	Zum nächsten Satz
}	Zum nächsten Absatz
(	Einen Satz zurück
{	Einen Absatz zurück
W	Auf nächstes durch ein Leerzeichen abgegrenztes Wort
B	Um ein W zurück
E	Ans Ende von W

Berichtigungen beim Einfügen

^H	Letztes Zeichen löschen (Backspace)
^W	letztes Wort löschen
erase	Ihr Löschzeichen wie **^H**
kill	Ihr Abbruch-Zeichen, Eingabe auf dieser Zeile löschen
****	**^H**, Ihr Lösch- und Abbruchzeichen quotieren
ESC	Einfügen beenden, zurück zum Kommando
DEL	Unterbrechung, beendet Einfügen
^D	Rückwärtstabulator über *autoindent* (automatisches Einrücken)
^^D	Rückwärtstabulator zum Anfang der Zeile; linke Grenze von *autoindent* nicht zurücksetzen
0^D	jedoch auch noch nächsten Rand linke Grenze von *autoindent* zurücksetzen
^V	nicht abdruckbares Zeichen quotieren

Einfügen und ersetzen

a	Hinter Cursor anhängen
i	Vor Cursor einfügen
A	Am Zeilenende anhängen
I	Vor erstem Zeichen einfügen
o	Leerzeile unter der aktuellen Zeile einfügen
O	Leerzeile über der aktuellen Zeile einfügen
r*x*	Einzelzeichen durch *x* ersetzen
R*text***ESC**	Zeichen ersetzen

Operatoren

Nach Operatoren folgt eine Bewegung des Cursors, und sie betreffen alle Texte, die durch diese Bewegung durchlaufen würden. Da beispielsweise **w** ein Wort weitergeht, löscht **dw** das Wort, auf das gegangen worden wäre. Mit einer Verdoppelung des Operationssymbols, z.B. **dd**, werden ganze Zeilen erfaßt.

d	**Löschen**	
c	**Ändern**	
y		Zeilen in den Puffer schreiben
>		Verlagerung nach links
<		Verlagerung nach rechts
!		Durch Kommando filtern

Verschiedene Operationen

C	Den Rest der Zeile ändern (**c$**)
D	Den Rest der Zeile löschen (**d$**)
s	Zeichen ersetzen (**cl**)
S	Zeilen ersetzen (**cc**)
J	Zeilen miteinander verbinden
x	Zeichen löschen (**dl**)
X	Zeichen vor dem Cursor (**dh**) löschen
Y	Zeilen zusammenziehen (**yy**)

Yank und Put (Zusammenziehen und Einfügen)

Mit Put wird der Text eingefügt, der als letzter gelöscht oder zusammengezogen wurde. Wird ein Puffer angegeben (ASCII-Kleinbuchstaben **a** - **z**), wird statt dessen der Text aus diesem Puffer eingefügt.

3yy	3 Zeilen puffern
3yl	3 Zeichen puffern
p	Text hinter Cursor einfügen
P	Text vor Cursor einfügen
"*x*p	von Puffer *x* einfügen
"*x*y	in den Puffer *x* schreiben
"*x*d	löschen und in den Puffer *x* schreiben

Rückgängig machen, Wiederausführen, Zurückgewinnen

u	Letzte Änderung rückgängig machen
U	Aktuelle Zeile wiederherstellen

.	Letzte Änderung wiederholen
"*d* p	*d*te Löschung zurückgewinnen

AUTOREN

vi und *ex* wurden von der University of California, Berkeley California, Computer Science Division, Department of Electrical Engineering and Computer Science entwickelt.

DATEIEN

/tmp	Standardverzeichnis für temporäre Dateien; dies kann über die **directory**-Option geändert werden (siehe *ex(1)*-**set**-Kommando.
/usr/lib/terminfo/?/*	Übersetzte Datenbasis für Terminal-Beschreibung
/usr/lib/.COREterm/?/*	Teilmenge der übersetzten Datenbasis zur Terminal-Beschreibung

NOTES

Zwei Optionen wurden, obwohl sie weiterhin unterstützt werden, ersetzt durch Optionen, die den Kommando-Standard-Syntaxregeln entsprechen (siehe Einleitung). Die **-r**-Option ohne ein Optionsargument wurde durch **-L** und *+Kommando* wurde durch das **-c**-Kommando ersetzt.

SIEHE AUCH

ed(1), edit(1), ex(1).
User's Guide.
Einführung in den Editor vi.

ACHTUNG!

Unbefugte Eingriffe in Einträge von */usr/lib/.COREterm/?/** oder */usr/lib/terminfo/?/** (z. B. Ändern oder Entfernen eines Eintrags) können sich negativ auf Programme wie *vi*(1) auswirken, die erwarten, daß die Eingabe vorhanden und richtig ist! So kann insbesondere das Entfernen des "dumb" Terminals unerwartete Probleme verursachen!

FEHLER

Software-Tabulatoren, die **^T** verwenden, funktionieren nur unmittelbar nach dem *autoindent* (automatisches Einrücken).

Bei den Verlagerungen nach links und rechts auf intelligenten Terminals werden die Operationen zum Einfügen und Löschen von Zeichen nicht verwendet.

BEZEICHNUNG

wait – auf Beendigung des Prozesses warten

ÜBERSICHT

wait [*n* :]

BESCHREIBUNG

Auf den Hintergrundprozeß mit der Prozeßnummer *n* warten und den Status seiner Beendigung melden. Wurde *n* nicht angegeben, wird auf alle jeweils aktiven Hintergrundprozesse Ihrer Shell gewartet, und der Rückgabecode ist Null.

Die Shell führt *wait* selbst aus, ohne einen neuen Prozeß zu erzeugen.

SIEHE AUCH

sh(1).

EINSCHRÄNKUNG

Wenn Sie die Fehlermeldung *cannot fork, too many processes* (Kann keinen neuen Prozeß erzeugen, zu viele Prozesse) erhalten, versuchen Sie, Ihre Hintergrundprozesse mit dem Kommando *wait* (1) zu säubern. Wenn dies nicht hilft, ist die Systemprozeß-Tabelle wahrscheinlich voll oder Sie haben zu viele aktive Vordergrundprozesse. (Die Anzahl der Prozesse, die unter Ihrer Login-Kennung laufen können und die vom System verwaltet werden können, ist begrenzt.)

FEHLER

Nicht alle Prozesse einer drei- oder mehrstufigen Pipe sind Söhne der Shell und deshalb kann auch nicht auf alle Prozesse gewartet werden.

Wenn *n* keine aktive Prozeßnummer ist, wird auf alle jeweils aktiven Hintergrundprozesse Ihrer Shell gewartet, und der Rückgabecode ist Null.

BEZEICHNUNG

wall – an alle Benutzer schreiben

ÜBERSICHT

/etc/wall

BESCHREIBUNG

wall liest seine Standardeingabe bis zum Dateiende. Es sendet diese Nachricht dann an alle zum gegebenen Zeitpunkt angemeldeten Benutzer, wobei der Nachricht

Broadcast Message from ... (Nachricht an alle)

vorangestellt wird.

Es wird zur Warnung aller Benutzer, beispielsweise vor einem Abschalten der Anlage, verwendet.

Der Absender muß der Systemverwalter sein, der jeden von den Benutzern möglicherweise vorgesehenen Schutz überschreiben kann (vgl. *mesg*(1)).

DATEIEN

/dev/tty*

SIEHE AUCH

mesg(1), write(1).

DIAGNOSE

"Cannot send to ..." (Kann nicht an ...schicken), wenn die Eröffnung einer Bildschirmdatei des Benutzers erfolglos ist.

BEZEICHNUNG

wc – Worte zählen

ÜBERSICHT

wc [**-lwc**] [names]

BESCHREIBUNG

wc zählt Zeilen, Worte und Zeichen in den angegebenen Dateien oder, wenn keine *names (Namen)* angegeben sind, in der Standardeingabe. Außerdem wird die Summe aller angegebenen Dateien berechnet. Ein Wort ist eine maximale Zeichenkette, die durch Leer-, Tabulator- oder New-Line-Zeichen begrenzt ist.

Die Optionen **l**, **w**, und **c** können in beliebiger Kombination verwendet werden, um anzugeben, daß nur jeweils die Anzahl an Zeilen, Worten bzw. Zeichen ausgegeben werden soll. Der Standardwert ist **-lwc**.

Wenn in der Kommandozeile *names* angegeben sind, werden diese zusammen mit der Anzahl ausgegeben.

BEZEICHNUNG

who – wer ist im System

ÜBERSICHT

who [**-uTlHqpdbrtas**] [file]

who am i

who am I

BESCHREIBUNG

who kann den Benutzernamen, die Terminal-Leitung, die Login-Uhrzeit, die seit der letzten Aktivität auf der Leitung abgelaufene Zeit sowie die Prozeßnummer des Kommandointerpretierers (Shell) für jeden aktuellen UNIX-Benutzer auflisten. Um diese Informationen zu erhalten, prüft es die Datei **/etc/utmp** bei Login. Wenn die Datei *file* angegeben ist, wird diese Datei (die im *utmp*[4] Format sein muß) geprüft. Im allgemeinen ist *file* **/etc/wtmp**, die eine Liste aller Logins seit der letzten Erstellung der Datei enthält.

who mit der Option **am i** oder **am I** bezeichnet den aufrufenden Benutzer.

Das allgemeine Ausgabe-Format ist:

name [state] line time [idle] [pid] [comment] [exit]

Die Angaben zu Name *name*, Leitung *line*, und Zeit *time* werden von allen Optionen mit Ausnahme von **-q** geliefert; die Angabe zu Status *state* wird nur von **-T** geliefert; die Angaben zu Ruhezeit *idle* und Prozeßnummer *pid* werden nur von **-u** und **-l** geliefert und die Angaben für Kommentar *comment* und Beenden *exit* kommen nur von **-a**. Die für **-p**, **-d**, und **-r** gelieferten Angaben werden in der nachstehenden Erläuterung der einzelnen Optionen näher erklärt.

In Verbindung mit Optionen kann *who* Login, Logoff, Neulade-Vorgänge und Änderungen der Systemuhr sowie andere Prozesse auflisten, die vom *init*-Prozeß erzeugt werden. Diese Optionen sind:

-u Diese Option listet nur die jeweils zum gegebenen Zeitpunkt angemeldeten Benutzer auf. *name* ist der Login-Name des Benutzers. *line* ist die Bezeichnung der Leitung, die im Verzeichnis **/dev** zu finden ist. *time* ist die Uhrzeit, zu der sich der Benutzer angemeldet hat (Login). Die Spalte *idle* enthält die Anzahl an Stunden und Minuten seit der letzten Aktivität in der jeweiligen Leitung. Ein Punkt (.) zeigt an, daß während der letzten Minute eine Aktivität auf dem Terminal stattgefunden hat, die daher als "aktuell" zu bezeichnen ist.Wenn mehr als 24 Stunden abgelaufen sind oder die Leitung seit dem letzten Urladen nicht benutzt wurde, wird die Eingabe als **old** (alt) bezeichnet.

Dieses Feld ist dann nützlich, wenn ermittelt werden soll, ob jemand an dem Terminal arbeitet. *pid* ist die Prozeßnummer der Shell des Benutzers. *comment* ist das zu dieser Leitung gehörende Kommentarfeld, wie in **/etc/inittab** angegeben (vgl. *inittab*[4]). Dieses kann Angaben über die jeweilige Lage des Terminals, die Telefonnummer, den Bildschirmtyp, sofern festverdrahtet, usw. enthalten.

-T Diese Option ist wie die Option **-s** , außer daß der *state* der Terminal-Leitung ausgegeben wird. Der *state* gibt an, ob eine andere Person an dieses Terminal schreiben darf. Ein + erscheint, wenn das Terminal für jemanden beschreibbar ist; ein - zeigt an, daß dieses nicht möglich ist. **root** kann auf alle Leitungen schreiben, die ein + oder ein - im *state* -Feld aufweisen. Wenn eine unbrauchbare Leitung angetroffen wird, erfolgt die Ausgabe von **?** .

-l Diese Option listet nur die Leitungen auf, bei denen das System darauf wartet, daß sich jemand anmeldet. In diesen Fällen lautet das *name* -Feld **LOGIN** . Andere Felder sind wie Benutzer-Einträge, wobei jedoch das Feld *state* nicht vorhanden ist.

-H Bei dieser Option werden Spaltentitel über die normale Ausgabe gesetzt.

-q Hierbei handelt es sich um ein schnelles *who* , das nur den Namen und die Zahl der zum gegebenen Zeitpunkt angemeldeten Benutzer anzeigt. Bei Benutzung dieser Option werden alle anderen Optionen ignoriert.

-p Bei dieser Option wird jeder andere Prozeß aufgelistet, der zum gegebenen Zeitpunkt aktiv ist und vorher von *init* erzeugt wurde. Das *name* -Feld ist die Bezeichnung für das von *init* ausgeführte Programm, das in **/etc/inittab** gefunden wird. Die Felder *state*, *line*, und *idle* haben keine Bedeutung. Das Feld *comment* zeigt das *id* -Feld aus **/etc/inittab** für die Leitung, das diesen Prozeß erzeugt hat. Vgl. *inittab*(4).

-d Diese Option zeigt alle Prozesse, die abgelaufen sind und nicht wieder von *init* erzeugt wurden. Für beendete Prozesse erscheint das *exit* (Ende)-Feld; es enthält die Endecodes des beendeten Prozesses (diese Werte werden von *wait*[2] zurückgegeben). Dies kann nützlich sein, um die Ursache der Beendigung des Prozesses zu ermitteln.

-b Diese Option zeigt Uhrzeit und Datum des letzten Neuladens an.

-r Diese Option zeigt den aktuellen *run-level* (Betriebsart) des *init*-Prozesses an. Zusätzlich liefert sie den Prozeßbeendigungsgrund, die Prozeßnummer und den Prozeß-Endestatus (vgl. *utmp*(4)) in den Spalten *idle*, *pid*, und *comment*.

-t Diese Option zeigt die letzte Änderung der Systemuhr (über den Befehl *date*[1]) durch **root** an. Vgl. *su*(1).

-a Mit dieser Option werden **/etc/utmp** oder das angegebene Datei *file* mit allen eingeschalteten Optionen verarbeitet.

-s Dies ist die Standardoption, die nur Felder *name*, *line*, und *time* auflistet.

Hinweis für den Systemverwalter: Nach einem Herunterfahren auf Single-User-Betrieb gibt *who* ein Eingabe-Aufforderungszeichen zurück; der Grund hierfür liegt darin, daß *who* keine genaue Nachricht über diese Betriebsart abgeben kann, weil **/etc/utmp** bei Login aktualisiert wird und es im Single-User-Betrieb kein Login gibt. *who am i* gibt jedoch die korrekten Angaben zurück.

DATEIEN

/etc/utmp
/etc/wtmp
/etc/inittab

SIEHE AUCH

date(1), login(1), mesg(1), su(1M).
init(1M) im *Administrator's Reference Manual.*
wait(2), inittab(4), utmp(4) im *Programmer's Reference Manual.*

BEZEICHNUNG

whois – DARPA-Internet-Benutzernamen-Auskunft

ÜBERSICHT

whois name

BESCHREIBUNG

whois help

erzeugt eine hilfreiche Ausgabe ähnlich der folgenden:

Please enter a name or a handle ("ident"), such as "Smith" or "SRI-NIC". Starting with a period forces a name-only search; starting with exclamation point forces handle-only. Examples:

```
Smith           [looks for name or handle SMITH ]
!SRI-NIC            [looks for handle SRI-NIC only ]
.Smith, John    [looks for name JOHN SMITH only ]
```

Adding "..." to the argument will match anything from that point, e.g. "ZU..." will match ZUL, ZUM, etc.

To have the ENTIRE membership list of a group or organization, if you are asking about a group or org, shown with the record, use an asterisk character '*' directly preceding the given argument. [CAUTION: If there are a lot of members this will take a long time!] You may of course use exclamation point and asterisk, or a period and asterisk together.

Die Ausgabe lautet übersetzt:
Bitte geben Sie einen Namen oder einen Handle ein ("ident"), z. B. "Smith" oder "SRI-NIC". Wird als erstes Zeichen ein Punkt angegeben, dann wird nur nach dem Namen gesucht. Wird als erstes Zeichen ein Ausrufungszeichen angegeben, dann wird nur nach dem Handle gesucht. Beispiele:

```
Smith       [sucht nach dem Namen oder dem Handle SMITH]
!SRI-NIC    [sucht nur nach dem Handle SRI-NIC]
.Smith, John [sucht nur nach dem Namen JOHN SMITH]
```

Folgt dem Argument "...", wird nach Namen/Handles gesucht, die mit dem Argument beginnen. "ZU..." findet z. B. ZUL, ZUM, usw.

Wenn Sie Auskunft über eine Gruppe oder Organisation haben möchten und eine Liste ALLER Mitglieder sehen wollen, müssen Sie dem Argument einen Stern '*' unmittelbar voranstellen. [VORSICHT: Bei einer großen Gruppe oder Organisation kann das sehr lange dauern!] Natürlich sind auch Ausrufungszeichen und Stern oder Punkt und Stern gleichzeitig erlaubt.

BEZEICHNUNG

write – an einen anderen Benutzer schreiben

ÜBERSICHT

write user [line]

BESCHREIBUNG

write kopiert Zeilen Ihres Terminals auf das Terminal eines anderen Benutzers. Beim ersten Aufruf wird die Nachricht

Message from *yourname* **(tty??)** [*date*]...

(Nachricht von ... Ihr Name, (Terminal) Datum) an die Person gesendet, mit der Sie Verbindung aufnehmen wollen. Bei erfolgreicher Herstellung der Verbindung werden zwei Tonsignale an Ihr Terminal ausgegeben, die anzeigen, daß die von Ihnen geschriebene Nachricht gesendet wird.

An dieser Stelle sollte der Empfänger der Nachricht zurückschreiben. Die Verbindung bleibt bestehen, bis ein Dateiende vom Terminal gelesen, eine Unterbrechung geschickt oder vom Empfänger ein "mesg n" ausgeführt wurde. An diesem Punkt schreibt *write* dann **EOT** auf das andere Terminal und beendet den Vorgang.

Falls Sie einem Benutzer schreiben wollen, der mehr als einmal angemeldet ist, können Sie das Argument *line* zur Anzeige der Leitung oder des Terminals verwenden, an die die Sendung gehen soll (z. B. **tty00**); andernfalls wird der erste in **/etc/utmp** gefundene beschreibbare Anschluß für den Benutzer angenommen und folgende Nachricht abgesezt:

user is logged on more than one place.
(*user* ist an mehreren Stellen angemeldet)
You are connected to "*terminal*".
(Sie sind mit *terminal* verbunden)
Other locations are:
terminal
(Andere Anschlußstellen sind: *terminal*)

Die Schreiberlaubnis kann mit dem Kommando *mesg(1)* verweigert oder gewährt werden. Normalerweise wird das Schreiben an andere standardmäßig zugelassen. Bestimmte Kommandos, wie z. B. *pr*(1) lassen Nachrichten jedoch nicht zu, um dadurch eine Störung der eigenen Ausgabe zu verhindern. Hat der Benutzer jedoch die Erlaubnis des Systemverwalters, können Nachrichten auch an ein schreibgesperrtes Terminal geschickt werden.

Steht das Zeichen ! am Anfang einer Zeile, ruft *write* die Shell auf, um den Rest der Zeile als Kommando auszuführen.

Für die Verwendung von *write* wird folgendes Protokoll vorgeschlagen: Wenn Sie zum ersten Mal mit *write* an einen anderen Benutzer schreiben wollen, warten Sie zunächst, bis dieser Ihnen mit *write* zurückschreibt, bevor Sie mit dem Senden der Nachricht beginnen. Jeder Teilnehmer sollte eine Nachricht mit einem deutlichen Signal beenden, (d. h. mit (o) für "over"), damit die andere Person weiß, wann sie antworten kann. Das Signal (oo) (für "over and out") wird empfohlen, wenn die Verbindung beendet werden soll.

DATEIEN

/etc/utmp zum Suchen des Benutzers
/bin/sh zum Ausführen des Kommandos !

SIEHE AUCH:

mail(1), mesg(1), pr(1), sh(1), who(1).

DIAGNOSE

"user is not logged on" wenn die Person, an die Sie mit *write* schreiben wollen, nicht angemeldet ist.

"Permission denied" wenn die Person, an die Sie mit *write* schreiben wollen, keine Erlaubnis hierzu gibt (mit *mesg*).

"Warning: cannot respond, set mesg -y" wenn Ihr Terminal auf *mesg n* gesetzt ist und der Empfänger Ihnen nicht antworten kann.

"Can no longer write to user" wenn der Empfänger keine Schreiberlaubnis erteilt hat (*mesg n*), nachdem Sie mit dem Schreiben begonnen haben.

BEZEICHNUNG

xargs – Argumentliste(n) aufbauen und Kommando ausführen

ÜBERSICHT

xargs [flags] [command [initial-arguments]]

BESCHREIBUNG

xargs kombiniert die angegebenen *initial-arguments* (Anfangsargumente) mit den von der Standardeingabe gelesenen Argumenten, um das angegebene Kommando *command* einmal oder mehrmals auszuführen. Die Anzahl der für jeden *command*-Aufruf gelesenen Argumente und die Art, in der sie kombiniert sind, werden durch die angegebenen Optionen bestimmt.

command, das eine Shell-Datei sein kann, wird unter Verwendung von **$PATH** gesucht. Falls *command* fehlt, wird **/bin/echo** aufgerufen.

Von der Standardeingabe eingelesene Argumente werden als fortlaufende Zeichenfolgen definiert, die durch ein oder mehrere Leer-, Tabulator- oder New-Line-Zeichen begrenzt sind; Leerzeilen werden stets gelöscht. Leerzeichen und Tabulatorzeichen können in einem Argument vorkommen, wenn sie mit Escape- oder Anführungszeichen quotiert werden. Zeichen in (einzelnen oder doppelten) Anführungszeichen werden wörtlich verstanden, und die Begrenzungsanführungszeichen werden entfernt. Außerhalb von Anführungszeichen dient ein Backslash (\) zur Quotierung (escape) des nächsten Zeichens.

Beim Aufbau jeder Argumentliste wird mit den Anfangsargumenten *initial-arguments* begonnen, auf die dann einige Argumente folgen, die von der Standardeingabe eingelesen werden. (Ausnahme: vgl. Option (flag) **-i**). Die Optionen **-l**, **-l**, und **-n** bestimmen, wie die Argumente für jeden Kommandoaufruf ausgewählt werden. Ist keiner dieser Schalter angegeben, folgen den *initial-arguments* Argumente, die kontinuierlich von der Standardeingabe eingelesen werden, bis ein interner Puffer voll ist. Dann wird *command* mit den gesammelten Argumenten (args) ausgeführt. Dieser Prozeß wird wiederholt, bis keine Argumente mehr vorhanden sind. Bei Konflikten zwischen den Optionen (z. B. **-l** und **-n**) hat die letzte Option (flag) Vorrang. Die Werte von *flag* sind:

-l*number* *command* wird ausgeführt für jede nicht-leere Anzahl *number* von Argumentzeilen aus der Standardeingabe. Der letzte Aufruf von *command* erfolgt mit weniger Argumentzeilen, wenn weniger als *number* Zeilen übrigbleiben. Es wird davon ausgegangen, daß eine Zeile mit dem ersten New-Line-Zeichen endet, *wenn* das letzte Zeichen der Zeile *kein* Leerzeichen oder Tabulatorzeichen ist; ein nachgestelltes Leer-

Tabulatorzeichen zeigt die Fortsetzung über die nächste nicht leere Zeile an. Wurde *number* ausgelassen, wird 1 gesetzt. Option **-x** wird gesetzt.

-i*replstr*
: Einfügemodus: *command* wird für jede Zeile aus der Standardeingabe ausgeführt, wobei die gesamte Zeile als Einzelargument angesehen wird, das in *initial-arguments* eingesetzt wird für jedes Vorkommen von *replstr* in *initial-arguments*. Maximal können 5 Argumente in *initial-arguments* jeweils ein oder mehrere Vorkommen von *replstr* enthalten. Leer- und Tabulatorzeichen am Anfang jeder Zeile werden entfernt. Aufgebaute Argumente dürfen nicht größer als 255 Zeichen werden, und die Option **-x** wird ebenfalls gesetzt. Wenn keine anderen Angaben erfolgen, wird { } für *replstr* angenommen.

-n*number*
: *command* unter Verwendung möglichst vieler Standard-Eingabeargumente verwenden, wobei *number* Argumente das Maximum darstellt. Es werden weniger Argumente verwendet, wenn ihre Gesamtgröße *size* Zeichen übersteigt, und beim letzten Aufruf, wenn weniger als *number* Argumente übrigbleiben. Wenn die Option **-x** auch angegeben ist, muß jede *number* von Argumenten in die *size* -Begrenzung passen, weil *xargs* sonst die Ausführung abbricht.

-t
: Ablaufverfolgungsmodus: Das *command* und jede aufgebaute Argumentliste werden vor ihrer Ausführung auf die Ausgabe mit der Dateikennzahl 2 ausgegeben.

-p
: Eingabe-Aufforderungs-Modus: Der Benutzer wird gefragt, ob *command* bei jedem Aufruf ausgeführt werden soll. Der Ablaufverfolgungsmodus (**-t**) wird eingeschaltet, um den auszuführenden Kommandoaufruf auszugeben; anschließend folgt die Eingabe-Aufforderung ?... . Mit der Antwort y (der wahlweise beliebige Eingaben folgen können) wird das Kommando ausgeführt; mit allem anderen, d. h. auch mit einem einfachen Carriage-Return, wird der jeweilige Aufruf von *command* übersprungen.

-x Bewirkt, daß *xargs* die Ausführung abbricht, wenn eine Argumentliste größer als *size* Zeichen sein würde; **-x** wird automatisch von den Optionen **-i** und **-l** gesetzt. Wenn keine der Optionen **-i**, **-l** oder **-n** angegeben ist, muß die Gesamtlänge aller Argumente innerhalb der mit *size* festgesetzten Grenzen liegen.

-s*size* Die maximale Gesamtgröße jeder Argumentliste wird auf *size* Zeichen gesetzt; *size* muß eine positive ganze Zahl sein, die kleiner als oder gleich 470 ist. Wenn **-s** nicht angegeben ist, wird 470 als Standardwert angenommen. Es ist zu beachten, daß die Zeichenanzahl für *size* jeweils ein zusätzliches Zeichen für jedes Argument und die Anzahl der Zeichen in der Bezeichnung des Kommandos einschließt.

-e*eofstr* *eofstr* wird als die logische Zeichenfolge für das Dateiende (EOF) angesehen. Der Unterstrich (_) wird als logisches **EOF**-Zeichen angenommen, wenn **-e** nicht angegeben ist. **-e** ohne *eofstr*-Codierung schaltet die Angabemöglichkeit für logische **EOF**-Zeichenfolge ab (der Unterstrich wird wörtlich genommen). *xargs* liest die Standardeingabe, bis entweder die Dateiende- oder die logische **EOF**-Zeichenfolge angetroffen wird.

xargs bricht ab, wenn es entweder einen Rückgabecode **-1** vom *command* erhält oder *command* nicht ausführen kann. Wenn es sich bei *command* um ein Shell-Programm handelt, sollte ausdrücklich *exit* mit einem geeigneten Wert (vgl. *sh*(1)) zum Beenden aufgerufen werden, damit eine unbeabsichtigte Rückkehr mit **-1** vermieden wird.

BEISPIELE

Das folgende Beispiel verlagert alle Dateien vom Verzeichnis $1 ins Verzeichnis $2. Die einzelnen *mv* -Kommandos werden ausgegeben bevor sie ausgeführt werden.

```
ls $1 | xargs -i -t mv $1/{} $2/{}
```

Im folgenden Beispiel wird die Ausgabe der in Klammern eingeschlossenen Kommandos auf einer Zeile zusammenfaßt, das Ergebnis wird dann an das Ende der Datei *log* angehängt.

```
(logname; date; echo $0 $*) | xargs >>log
```

Der Benutzer wird gefragt, welche Dateien im aktuellen Verzeichnis archiviert werden sollen und diese werden dann (1.) jeweils einzeln oder (2.) mehrere auf ein Mal in *arch* archiviert.

```
1. ls | xargs -p -l ar r arch
2. ls | xargs -p -l | xargs ar r arch
```

Als nächstes soll *diff*(1) mit aufeinanderfolgenden Argumentpaaren aufgerufen werden, die ursprünglich als Shell-Argumente eingegeben wurden:

```
echo $* | xargs -n2 diff
```

SIEHE AUCH

sh(1).