NIXDORF
COMPUTER

# TARGON® /35

# Benutzerkommandos

Ausgabe 1. 1. 1986 Bestell-Nr. 10135.00.4.93

Ihre Aufnahme in die Verteilerliste für den Änderungsdienst erfolgt nur, wenn Sie diese Karte einschicken. Änderungen erhalten Sie durch die betreuende Nixdorf-Niederlassung.

Bitte abtrennen und in die Tasche im Handbuchrücken einstecken.

Systemliteratur

TARGON® /35

Bitte senden Sie Änderungen und Ergänzungen zu diesem Literaturteil an die folgende Anschrift.

Betreuende Nixdorf-Niederlassung (unbedingt angeben):

Name: _____

in Firma: _____

oder Bereich NCAG: _____

Verkehrsnummer:
10135.00.4.93

Ausgabedatum der letzten Änderung
lt. Organisationsblatt (unbedingt angeben):

_____

Postkarte

Nixdorf Computer AG
Abt. ZSI
Fürstenallee 7

D-4790 Paderborn
West-Germany

## Organisationsblatt

### 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: | 01.01.86 | Rel. 1 |

## Anregungen/Korrekturen

### Anregungen/Korrekturen

Sollten Ihnen bei der Benutzung dieses Teils der Systemliteratur Fehler aufgefallen sein oder haben Sie Anregungen zur Verbesserung des Handbuchs, so bitten wir Sie, diese schriftlich zu formulieren und an folgende Anschrift zu schicken:

Nixdorf Computer AG
Abt. ZSI
Fürstenallee 7

D-4790 Paderborn

# Einleitung

Dieses Reference Manual beschreibt die Ihnen auf dem System TARGON /35 zur Verfügung stehenden Benutzerkommandos.

Die Befehle sind in alphabetischer Reihenfolge geordnet. Jede Kommandobeschreibung ist ein eigenständiges Modul dieses Nachschlagewerks. Die Befehlsnamen erscheinen jeweils in der Kopf- und Fußzeile; die Seiten sind entsprechend numeriert.

Die Module sind in folgende Teile gegliedert:

1. Syntax des Befehlsaufrufs
2. Funktionsbeschreibung der Kommandos einschließlich ihrer Optionen
3. Beispiele
4. Hinweise auf Dateien, die von den Befehlen angesprochen oder angelegt werden
5. Hinweise auf mögliche Fehlermeldungen
6. Verweise auf verbundene Befehle und ergänzende Systemliteratur

Die Syntax der Befehle ist folgendermaßen aufgebaut:

*kursiv* gedruckte Worte stehen als Symbole, für die konkrete Werte oder Namen eingesetzt werden müssen,

[...] in eckigen Klammern eingeschlossene Teile können fehlen,

... drei Punkte bedeuten, daß das unmittelbar vorangehende Element beliebig oft wiederholt werden kann.

Alle übrigen Zeichen und Worte in Normalschrift sind Endsymbole und müssen stets geschrieben werden.

Zur besseren Übersicht folgt dieser Einleitung eine Aufzählung aller beschriebenen Kommandos mit einem kurzen Hinweis auf ihre Funktion.

## Einleitung

Zusätzlich stehen Ihnen auf dem UNIX-Supermini TARGON /35 die Kommandos der UNIX-Erweiterung der University of California, Berkeley zur Verfügung. Die original englischsprachige Literatur können Sie beziehen bei:

Computer Science Division
Department of Electrical Engineering and Computer Science
University of California
Berkeley, California 94720/USA

## Kommandoübersicht

| | |
|---|---|
| **acctcom** | Suchen und Ausgeben von Benutzerprozeßverwaltungsdateien |
| **adb** | Vollständiger Debugger |
| **admin** | Anlegen und Verwalten von SCCS-Dateien |
| **ar** | Archivdatei erstellen und aktualisieren |
| **asa** | Interpretation von ASA-Vorschubsteuerzeichen |
| **awk** | Mustererkennungs- und Verarbeitungssprache |
| **banner** | Vergrößerte Ausgabe der angegebenen Argumente |
| **basename, dirname** | Liefern von Pfadnamenteilen |
| **bc** | Sprache für Arithmetik mit beliebiger Genauigkeit |
| **bdiff** | Vergleichen von großen Dateien |
| **bfs** | Programm zum Durchsuchen großer Dateien |
| **bs** | Compiler/Interpreter für relativ kleine Programme |
| **cal** | Kalenderanzeige |
| **calendar** | Erinnerungs-Service |
| **cancel** | Druckauftrag an Drucker-Spooler löschen |
| **cat** | Verketten und Anzeigen |
| **cb** | Formatieren von C-Programmen |
| **cc** | C-Compiler |
| **cd** | Wechseln des aktuellen Verzeichnisses |
| **cdc** | Ändern des Delta-Kommentars eines SCCS-Deltas |
| **cflow** | Kontrollflußdiagramme von C-Programmen |
| **chmod** | Ändern der Zugriffsrechte |
| **chown, chgrp** | Ändern der Eigentümer oder der Gruppe |

## Kommandoübersicht

| | |
|---|---|
| cmp | Vergleichen zweier Dateien |
| col | Herausfiltern von Zeilenvorschüben |
| comb | Zusammenfassung von SCCS-Deltas |
| comm | Suchen gleicher Zeilen in zwei Dateien |
| cp | Kopieren |
| cpio | Kopieren von Dateien |
| cpp | C Preprozessor |
| csplit | Aufspalten von Zusammenhängen |
| cu | Anrufen und Anmelden in ein anderes System |
| cut | Ausgabe vorgegebener Felder aus Zeilen einer Datei |
| cxref | Crossreferenzliste von C-Programmen |
| | |
| date | Anzeigen und Einstellen des Datums und der Uhrzeit |
| dc | Tischrechner |
| dd | Konvertieren und Kopieren von Dateien |
| delta | Einbringen eines Deltas (Verändern) in SCCS-Dateien |
| deroff | Entfernen von nroff-, tbl- und eqn-Konstrukten |
| df | Ermittlung des freien Plattenspeicherplatzes |
| diff | Ermittlung von Dateiunterschieden |
| diff3 | Vergleich von drei Dateiversionen |
| diffmk | Markieren von Differenzen zwischen Dateien für nroff/troff |
| dircmp | Vergleich zweier Verzeichnisse |
| du | Ermittlung der Speicherbelegung |
| | |
| echo | Anzeige von Argumenten |
| ed, red | Standard-Editor |

## Kommandoübersicht

**edit** Texteditor

**enable, disable**
Drucker-Status auf betriebsbereit bzw. nicht betriebsbereit setzen

**env** Änderung der Umgebung bei Ausführung von Kommandos

**eqn, neqn, checkeq**
Preprozessor für nroff/troff

**ex** Texteditor

**expr** Berechnung von Argumenten

**f77** Fortran 77 Compiler

**factor** Zerlegung einer Zahl in ihre Faktoren

**false** Lieferung des Rückgabe-Codes 1

**file** Ermittlung von Dateitypen

**find** Suchen von Dateien

**get** Erzeugen von Versionen einer SCCS-Datei

**getopt** Analyse von Kommando-Optionen

**greek** Auswahl eines Terminalfilters

**grep, egrep, fgrep**
Durchsuchen von Dateien nach Suchmustern

**help** Hilfe

**hyphen** Überprüfen von Trennstellen

**id** Anzeige von Benutzer- und Gruppenkennung

**ipcrm** IDs für Nachrichtenwarteschlange, Semaphorgruppe oder gemeinsam benutzten Speicher löschen

**ipcs** Melden des Status von Interprozeß-Kommunikationsfunktionen

## Kommandoübersicht

| | |
|---|---|
| join | Relationaler Datenbank-Operator |
| kill | Interrupt senden |
| ld | Binder (Link-Editor) |
| lex | Generierung von Programmen für eine einfache lexikalische Analyse |
| line | Lesen einer Zeile |
| lint | Prüfung der Syntax und Semantik von C-Programmen |
| ln | Erstellen einer Verknüpfung (Link) |
| login | Anmelden in das System |
| logname | Anzeige des Login-Namens |
| lorder | Ausgabe von Referenzlisten für Objekt- oder Archivdateien |
| lp | Druckauftrag an Drucker-Spooler senden |
| lpstat | Spoolersystem-Statusinformationen anfordern |
| ls | Auflisten des Inhalts von Verzeichnissen |
| m4 | Makro-Prozessor |
| machid | Ermittlung des Prozessortyps |
| mail, rmail | Elektronische Post |
| make | Pflege, Aktualisierung und Regenerierung von Programmsystemen |
| makekey | Generierung eines Verschlüsselungs-Codes |
| man, manprog | Ausgabe von Manual-Teilen |
| mesg | Erlaubt oder Verbietet das Senden von Nachrichten an ein Terminal |
| mkdir | Anlegen eines Verzeichnisses |
| mv | Verschieben oder Umbenennen von Dateien oder Verzeichnissen |

## Kommandoübersicht

| | |
|---|---|
| **newform** | Ändern des Formats einer Textdatei |
| **newgrp** | Anmelden in eine neue Gruppe |
| **news** | Anzeige des Inhalts der Dateien aus /usr/news |
| **nice** | Ausführung eines Kommandos mit niedriger Priorität |
| **nl** | Numerierung von Textzeilen |
| **nm** | Anzeige der Namenlisten von Objektdateien |
| **nohup** | Ausführung eines Kommandos ohne Rücksicht auf Unterbrechungssignale |
| **nroff** | Textformatierer |
| **od** | Ausdrucken von Dateiinhalten (Dump) |
| **pack** | Komprimieren von Dateien |
| **passwd** | Eingeben oder Ändern des Login-Kennworts |
| **paste** | Mischen von gleichen Zeilen mehrerer Dateien oder zusammengehörenden Zeilen einer Datei |
| **pcat** | Dekomprimieren von Dateien |
| **pr** | Dateien ausgeben |
| **prof** | Anzeigen von Profildateien |
| **prs** | Ausdrucken von SCCS-Dateien |
| **ps** | Anzeige Prozeß-Status |
| **pwd** | Anzeige aktuelles Verzeichnis |
| **ratfor** | Fortran Preprozessor |
| **regcmp** | Kompilieren regulärer Ausdrücke |
| **rm, rmdir** | Löschen von Dateien; Löschen von Verzeichnissen |
| **rmdel** | Entfernen eines Deltas aus einer SCCS-Datei |

## Kommandoübersicht

| | |
|---|---|
| **sact** | Anzeigen der Editieraktivitäten der aktuellen SCCS-Datei |
| **sar** | Auswertung der Systemaktivitäten |
| **sccsdiff** | Vergleich zweier Versionen einer SCCS-Datei |
| **sdb** | Symbolischer Debugger |
| **sdiff** | Gleichzeitige Anzeige zweier Dateien |
| **sed** | Stream-Editor |
| **sh, rsh** | Shell, restricted Shell, die normale/eingeschränkte Kommandosprache |
| **size** | Ermitteln des Speicherplatzes einer Objektdatei |
| **sleep** | Verzögerung der Ausführung eines Prozesses |
| **sno** | SNOBOL-Interpreter |
| **sort** | Sortieren oder Mischen von Dateien |
| **spell, hashmake, spellin, hashcheck** | Feststellen von Rechtschreibfehlern |
| **spline** | Glätten von Kurven durch Interpolation |
| **split** | Aufteilen einer Datei auf mehrere Ausgabedateien |
| **strip** | Löschen von Symboltabellen und Zeilennummer-Informationen in Objektdateien |
| **stty** | Einrichtung der Terminaloptionen |
| **su** | Zeitweilige Änderung der Benutzer-ID |
| **sum** | Anzeige der Blockanzahl einer Datei sowie Berechnung einer Prüfsumme |
| **sync** | Synchronisation der Systempuffer mit der Platte |
| **tabs** | Setzen von Tabs an einem Terminal |
| **tail** | Anzeigen des letzten Teils einer Datei |
| **tar** | Dateienarchivierung auf Magnetband |
| **tbl** | Preprozessor für nroff |

## Kommandoübersicht

| | |
|---|---|
| **tee** | Duplizieren der Standardeingabe |
| **test** | Überprüfen von Datei-Stati und Parameterwerten |
| **time** | Angabe über die Laufzeit eines Programms |
| **timex** | Generierung eines Berichts über die Systemaktivitäten |
| **touch** | Aktualisierung der Zugriffs- und Änderungsdaten von Dateien |
| **tr** | Umwandlung von Zeichen |
| **true** | Lieferung des Rückgabe-Codes 0 |
| **tsort** | Sortieren partiell sortierter Dateien |
| **tty** | Anzeigen Terminalname |
| | |
| **umask** | Einstellen der Standardwerte für Zugriffsrechte |
| **uname** | Anzeige aktueller Systemdaten |
| **unget** | Rückgängigmachen eines vorausgegangenen get-Befehls |
| **uniq** | Ermittlung doppelter Zeilen einer Datei |
| **units** | Umwandeln von Maßeinheiten |
| **universe, att, ucb** | Setzen bzw. Ändern der Umgebung |
| **unpack** | Dekomprimieren von Dateien |
| **uucp, uulog, uuname** | Kopieren von UNIX- zu UNIX-System |
| **uustat** | Statusabfrage und Job Control für uucp |
| **uuto, uupick** | UNIX-zu-UNIX Datei-Transfer mit vereinfachter Dateizugriffsstruktur |
| **uux** | Ausführung von Kommandos auf Fremdsystemen |

## Kommandoübersicht

| | |
|---|---|
| val | Ermitteln von SCCS-Dateien |
| vc | Versionskontrolle |
| vi | Bildschirmorientierter Editor |
| wait | Warten auf Beendigung aller Hintergrundprozesse |
| wc | Durchzählen von Dateiinhalten |
| what | Anzeige der Versionen von SCCS-Dateien |
| who | Wer arbeitet mit dem System? |
| write | Senden von Nachrichten an einen anderen Benutzer |
| xargs | Erstellen von Argumentlisten und Ausführen von Befehlen |
| yacc | Programm für Syntaxanalyse |

## acctcom

acctcom – Suchen und Ausgeben von Benutzerprozeßverwaltungsdateien

SYNTAX:

acctcom [[*Optionen*] [*Datei*]] ...

BESCHREIBUNG:

Acctcom liest aus *Datei*, der Standardeingabe oder der Datei /usr/adm/pacct und gibt ausgewählte Sätze auf der Standardausgabe aus. Jeder dieser Sätze beschreibt die Ausführung eines Prozesses.

Ausgegeben werden:

- Kommandoname
- Benutzer
- Terminalname
- Startzeit
- Endezeit
- real-gebrauchte Zeit in Sekunden
- CPU-Zeit in Sekunden
- mittlere Größe in K

Durch Angabe der unten beschriebenen Optionen kann die Ausgabe modifiziert werden.

Der Kommandoname ist mit einem # gekennzeichnet, wenn das Kommando mit Superuser-Privilegien ausgeführt wurde. Ist ein Prozeß mit einem nicht-bekannten Terminal verbunden, erscheint in der Ausgabespalte für den Terminalnamen ein Fragezeichen.

Geben Sie keine Eingabedateien an oder ist die Standardeingabe einem Terminal oder der Datei /dev/null zugeordnet, wird /usr/adm/pacct gelesen. In allen anderen Fällen wird die Standardeingabe benutzt.

## acctcom

Geben Sie *Dateien* an, werden diese in der angegebenen Reihenfolge gelesen.

Optionen:

| | |
|---|---|
| -b | Die zuletzt ausgeführten Kommandos werden zuerst ausgegeben. |
| -f | Ausgabe der fork/exec-Flags sowie des System-Rückgabe-Codes. |
| -h | Statt der mittleren Speicherbelegung wird die CPU-Zeit, die der Prozeß während seiner Ausführung benötigte, angezeigt. |
| -i | Ausgabe der E/A-Zähler. |
| -k | Statt der Speichergröße wird die genaue Speicherbelegungszeit in Minuten ausgegeben. |
| -m | Ausgabe der mittleren Speichergröße (Standard). |
| -r | Ausgabe des CPU-Faktors. |
| -t | Getrennte Ausgabe von Systemzeit und CPU-Zeit. |
| -v | Keine Ausgabe der Spaltenüberschriften. |
| -l*Leitung* | Ausgabe von Daten über die Prozesse, die das Terminal /dev/*Leitung* betreffen. |
| -u*Benutzer* | Ausgabe von Daten über die Prozesse, die dem angegebenen *Benutzer* gehören. Sie können *Benutzer* als Benutzer-ID oder als Login-Namen angeben. Ein # spezifiziert die Prozesse, die unter Superuser-Privilegien ausgeführt werden, ein ? diejenigen, die zu einer unbekannten Benutzer-ID gehören. |
| -g*Gruppe* | Ausgabe von Daten über die Prozesse, die der angegebenen *Gruppe* gehören. Sie können *Gruppe* als Gruppen-ID oder als Gruppenname angeben. |
| -d*MM/TT* | Jede Zeitangabe, die Sie nach dieser Option angeben, wird – statt den letzten 24 Stunden – dem entsprechenden Monat und Tag zugeordnet. Dies ist nötig, wenn Sie ältere Dateien bearbeiten wollen. |

## acctcom

-s*Zeit* Ausgabe von Daten über die Prozesse, die zum oder nach dem angegebenen Zeitpunkt aktiv waren. *Zeit* geben Sie in folgendem Format an:

*Stunde*[:*Minute*[:*Sekunde*]].

-e*Zeit* Ausgabe von Daten über die Prozesse, die zum oder vor dem angegebenen Zeitpunkt aktiv waren.

-S*Zeit* Ausgabe von Daten über die Prozesse, die zum oder nach dem angegebenen Zeitpunkt gestartet wurden.

-E*Zeit* Ausgabe von Daten über die Prozesse, die zum oder vor dem angegebenen Zeitpunkt beendet waren. Geben Sie -S und -E zusammen mit der gleichen *Zeit* an, erhalten Sie Daten über die Prozesse, die zum angegebenen Zeitpunkt aktiv waren.

-n*Muster* *Muster* ist ein regulärer Ausdruck im Sinne von ed. Ausgabe erfolgt nur für solche Kommandos, deren Namen durch das angegebene *Muster* angesprochen werden.

-q Es werden nur die statistischen Werte – die Sie mit der -a-Option zusätzlich erhalten – ausgegeben.

-o*Ausgabedatei*
Die Ausgabe erfolgt in die angegebene Datei.

-H*Faktor* Ausgabe von Daten über die Prozesse, die den angegebenen *Faktor* übersteigen. Unter *Faktor* wird der HOG-Faktor – wie unter der -h-Option beschrieben – verstanden.

-O*Sekunden* Ausgabe von Daten über die Prozesse, deren CPU-Systemzeit die angegebene Anzahl Sekunden übersteigt.

-C*Sekunden* Ausgabe von Daten über die Prozesse, deren gesamte CPU-Zeit (System und Benutzer) die angegebene Anzahl Sekunden übersteigt.

-I*Zeichen* Ausgabe von Daten über die Prozesse, die mehr als die angegebene Anzahl von Zeichen transferiert haben.

## acctcom

DATEIEN:

/etc/passwd

/etc/group

/usr/adm/pacct

HINWEIS:

Acctcom berichtet nur über Prozesse, die bereits beendet sind. Angaben über aktive Prozesse erhalten Sie mit dem ps-Kommando.

SIEHE AUCH:

ps, su
Systemliteratur TARGON: „Systemschnittstellen und Programmierung"
Systemliteratur TARGON /35: „System Administrator's Guide"

## adb

### adb – Vollständiger Debugger

SYNTAX:

adb [-w] [-l *Verzeichnis*] [*Objektdatei*] [*Core-Datei*]

BESCHREIBUNG:

Adb ist ein universeller Debugger, mit dem Sie Dateien untersuchen und eine kontrollierte Umgebung für die Ausführung von UNIX-Systemprogrammen bereitstellen können.

*Objektdatei* ist in der Regel eine ausführbare Programmdatei, die vorzugsweise eine Symboltabelle enthalten sollte. Enthält sie keine Symboltabelle, können die symbolischen Leistungsmöglichkeiten von adb nicht ausgenutzt werden, obwohl es trotzdem möglich ist, die Datei zu untersuchen. Der Standardwert für *Objektdatei* ist a.out. *Core-Datei* ist eine Core-Image-Datei, die nach Ausführung von *Objektdatei* entsteht; der Standardwert für *Core-Datei* ist core.

Anforderungen an adb werden von der Standardeingabe gelesen, und die Antworten werden auf der Standardausgabe ausgegeben. Geben Sie die Option -w an, werden sowohl *Objektdatei* als auch *Core-Datei* nötigenfalls angelegt und zum Lesen und Schreiben eröffnet, so daß Dateien mit Hilfe von adb geändert werden können.

Die Option -l gibt ein Verzeichnis an, in dem Dateien, die mit $< oder $<< (siehe unten) ausgelesen werden sollen, gesucht werden; der Standardwert ist /usr/lib/adb.

Adb ignoriert QUIT; bei INTERRUPT erfolgt ein Sprung zum nächsten adb-Kommando.

Anforderungen an adb haben im allgemeinen die folgende Form:

[*Adresse*] [,*Anzahl*] [*Kommando*] [;]

## adb

Wenn *Adresse* vorhanden ist, wird *Arbeitszeiger* auf *Adresse* gesetzt. Anfangs ist *Arbeitszeiger* auf 0 gesetzt. Bei den meisten Kommandos gibt *Anzahl* an, wie oft das Kommando ausgeführt wird. Der Standardwert für *Anzahl* ist 1. *Adresse* und *Anzahl* sind Ausdrücke.

Die Interpretation einer Adresse hängt von dem Kontext ab, in dem sie steht. Wird ein Unterprozeß ausgetestet, werden Adressen in der üblichen Weise im Adreßraum des Unterprozesses interpretiert. Weitere Einzelheiten über die Adreßabbildung finden Sie im Abschnitt „Adressen".

### Ausdrücke

| | |
|---|---|
| . | Der Wert von *Arbeitszeiger*. |
| + | Der Wert von *Arbeitszeiger*, erhöht um das aktuelle Inkrement. |
| ^ | Der Wert von *Arbeitszeiger*, vermindert um das aktuelle Inkrement. |
| " | Die letzte eingegebene Adresse. |
| *Ganzzahl* | Eine ganze Zahl. Die Präfixe 0o und 0O bewirken, daß die Zahl als Oktalzahl interpretiert wird; die Präfixe 0t und 0T bewirken, daß die Zahl als Dezimalzahl interpretiert wird; die Präfixe 0x und 0X bewirken, daß die Zahl als Hexadezimalzahl interpretiert wird. Somit gilt: 0o20 = 0t16 = 0x10 = sechzehn. Wenn kein Präfix angegeben ist, wird die standardmäßige Basis verwendet; siehe das Kommando $d. Die Standard-Basis ist anfangs hexadezimal. Die Hexadezimalziffern sind 123456789abcdefABCDEF mit den bekannten Werten. Hier ist zu beachten, daß einer Hexadezimalzahl, deren höchstwertige Stelle normalerweise ein Buchstabe ist, ein Präfix 0x (oder 0X) (bzw. eine führende Null, wenn die Standard-Basis hexadezimal ist) vorangestellt sein muß. |
| *Ganzzahl.Nachkommateil* | Eine 32-Bit-Gleitkommazahl. |
| 'cccc' | Der ASCII-Wert von bis zu vier Zeichen. Mit einem \ kann die Spezialbedeutung von ' aufgehoben werden. |

## adb

*<Name* Der Wert von *Name*, bei dem es sich entweder um einen Variablennamen oder einen Registernamen handelt. Adb führt eine Reihe von Variablen (siehe unter „Variablen"), die mit einzelnen Buchstaben oder Ziffern benannt sind. Wenn *Name* ein Registername ist, wird der Wert des Registers aus dem System-Header in *Core-Datei* entnommen. Bei den Registernamen handelt es sich um diejenigen, die vom Kommando $r ausgegeben werden.

*Symbol* *Symbol* ist eine Folge von Groß- oder Kleinbuchstaben, Unterstrichen oder Ziffern, die nicht mit einer Ziffer beginnt. Mit dem Backslash (\) kann die Sonderbedeutung anderer Zeichen aufgehoben werden. Der Wert von *Symbol* wird aus der Symboltabelle in *Objektdatei* entnommen. Dem Symbol wird – wenn nötig – ein _ vorangestellt.

*_Symbol* In C beginnt der „wahre Name" eines externen Symbols mit einem Unterstrich. Unter Umständen ist es notwendig, diesen Namen explizit anzugeben, um ihn von internen oder verborgenen Variablen eines Programms zu unterscheiden.

*Routine*.[*Name*]
Die Adresse der Variable *Name* in der angegebenen C-Routine. Sowohl *Routine* als auch *Name* sind Symbole. Wird *Name* weggelassen, ist der Wert die Adresse des zuletzt aktivierten C-Stack-Frames, der sich auf *Routine* bezieht.

(*Ausdruck*) Der Wert von *Ausdruck*.

## adb

### Einstellige Operatoren

**Ausdruck* Der Inhalt der Speicheradresse, die von *Ausdruck* in *Core-Datei* angesprochen wird.

*@Ausdruck* Der Inhalt der Speicheradresse, die von *Ausdruck* in *Objektdatei* angesprochen wird.

*-Ausdruck* Ganzzahlige Negation.

*~Ausdruck* Bitweises Komplement.

*#Ausdruck* Logische Negation.

### Zweistellige Operatoren

Zweistellige Operatoren sind linksseitig assoziativ und binden schwächer als einstellige Operatoren.

*e1+e2* Ganzzahlige Addition.

*e1-e2* Ganzzahlige Subtraktion.

*e1*e2* Ganzzahlige Multiplikation.

*e1%e2* Ganzzahlige Division.

*e1&e2* Bitweise Konjunktion.

*e1|e2* Bitweise Disjunktion.

*e1#e2* Aufgerundet zum nächsten Vielfachen von *e2*.

### Kommandos

Die meisten Kommandos bestehen aus einem Verb, an das sich ein Modifikator oder eine Liste von Modifikatoren anschließt. Die folgenden Kommandos stehen zur Verfügung: (An die Kommandos ? und / kann sich ein * anschließen; weitere Einzelheiten siehe Abschnitt „Adressen".).

*?f* Speicherstellen, die bei *Adresse* in *Objektdatei* beginnen, werden entsprechend dem Format *f* ausgegeben. *Arbeitszeiger* wird um die Summe der Inkremente für jeden Formatbuchstaben erhöht.

## adb

*/f* Speicherstellen, die bei *Adresse* in *Core-Datei* beginnen, werden entsprechend dem Format *f* ausgegeben; *Arbeitszeiger* wird wie bei ? erhöht.

*=f* Der Wert von *Adresse* selbst wird in der vom Format *f* angegebenen Form ausgegeben. (Bei Format i wird ? für die Teile der Instruktion ausgegeben, die später folgende Worte ansprechen.)

Ein Format besteht aus einem oder mehreren Zeichen, die eine Ausgabeform kennzeichnen. Vor jedem Formatzeichen kann eine Dezimalzahl stehen, die einen Wiederholungsfaktor für das Formatzeichen darstellt. Während ein Format schrittweise durchlaufen wird, wird *Arbeitszeiger* jeweils um den bei den einzelnen Formatbuchstaben angegebenen Wert erhöht. Geben Sie kein Format an, wird das letzte Format verwendet. Es gibt folgende Formatbuchstaben:

- o 2 2 Bytes im Oktalformat ausgeben. Allen Oktalzahlen, die von adb ausgegeben werden, wird ein O vorangestellt.
- O 4 4 Bytes im Oktalformat ausgeben.
- q 2 Im Oktalformat mit Vorzeichen ausgeben.
- Q 4 Long-Oktalwert mit Vorzeichen ausgeben.
- d 2 Im Dezimalformat ausgeben.
- D 4 Im Long-Dezimalformat ausgeben.
- x 2 2 Bytes im Hexadezimalformat ausgeben.
- X 4 4 Bytes im Hexadezimalformat ausgeben.
- u 2 Als vorzeichenlose Dezimalzahl ausgeben.
- U 4 Als long vorzeichenlose Dezimalzahl ausgeben.
- f 4 Den 32-Bit-Wert als Gleitkommazahl ausgeben.
- F 8 Als doppeltgenaue Gleitkommazahl ausgeben.
- b 1 Das angesprochene Byte im Oktalformat ausgeben.
- c 1 Das angesprochene Zeichen ausgeben.

## adb

C 1 Das angesprochene Zeichen mit der standardmäßigen Escape-Konvention ausgeben, wobei Steuerzeichen als ^X und das Löschzeichen als ^? ausgegeben wird.

s *n* Die angesprochenen Zeichen ausgeben, bis ein Null-Zeichen erreicht wird.

S *n* Eine Zeichenkette mit der Escape-Konvention ^X ausgeben (siehe C oben). Der Wert *n* ist die Länge der Zeichenkette einschließlich des Endezeichens Null.

Y 4 4 Bytes im Datumformat ausgeben.

i *n* Als Maschineninstruktionen ausgeben. Der Wert *n* gibt die Anzahl der von der Instruktion belegten Bytes an. Bei diesem Ausgabeformat werden die Variablen 1 und 2 auf den Teil disp1 bzw. disp2 der ausgegebenen Instruktion gesetzt. Enthält eine Instruktion kein disp1- oder disp2-Feld, so wird die entsprechende Variable auf Null gesetzt.

a 0 Den Wert von *Arbeitszeiger* in symbolischer Form ausgeben. Bei Symbolen wird überprüft, ob sie den richtigen Typ haben:

- / Lokales oder globales Datensymbol
- ? Lokales oder globales Textsymbol
- = Lokales oder globales absolutes Symbol

p 4 Den angesprochenen Wert in symbolischer Form ausgeben und dabei dieselben Regeln für den Symbolaufruf wie bei a anwenden.

t 0 Wenn eine Ganzzahl vorausgeht, erfolgt ein Tabsprung zum nächsten entsprechenden Tabstopp. Bei 8t wird beispielsweise zum nächsten Tabstopp mit Abstand 8 gesprungen.

r 0 Ein Leerzeichen ausgeben.

n 0 Ein Neue-Zeile-Zeichen ausgeben.

"..." 0 Die in " eingeschlossene Zeichenkette ausgeben.

^ *Arbeitszeiger* wird um das aktuelle Inkrement vermindert. Es wird nichts ausgegeben.

## adb

+ *Arbeitszeiger* wird um eins erhöht. Es wird nichts ausgegeben.

- *Arbeitszeiger* wird um eins vermindert. Es wird nichts ausgegeben.

Neue-Zeile
Das vorangehende Kommando mit Wiederholungsfaktor 1 wiederholen.

[?/]| *Wert* [*Maske*]
Wörter, die bei *Arbeitszeiger* beginnen, werden mit *Maske* maskiert und mit *Wert* verglichen, bis eine Übereinstimmung gefunden wird. Wird L angegeben, so muß die Übereinstimmung sich auf 4 Bytes und nicht nur 2 Bytes erstrecken. Wird keine Übereinstimmung gefunden, so bleibt *Arbeitszeiger* unverändert. Anderenfalls wird *Arbeitszeiger* auf die gefundene Position gesetzt. Geben Sie *Maske* nicht an, wird -1 verwendet.

[?/]w *Wert*
Den 2 Byte langen Wert auf die adressierte Stelle schreiben. Lautet das Kommando W, so werden 4 Bytes geschrieben. Ungerade Adressen sind unzulässig, wenn in den Adreßraum des Unterprozesses geschrieben wird.

[?/]m *b1 e1 f1*[?/]
Es werden neue Werte für (*b1*, *e1*, *f1*) gespeichert. Sind weniger als drei Ausdrücke angegeben, bleiben die übrigen Parameter der Tabelle unverändert. Schließt sich an das ? oder den / ein * an, wird das zweite Segment (*b2*, *e2*, *f2*) der tabellarischen Zuordnung geändert. Wird die Liste mit einem ? oder / beendet, wird die Datei (*Objektdatei* bzw. *Core-Datei*) bei weiteren Anforderungen verwendet (so daß /m? beispielsweise bewirkt, daß / sich auf *Objektdatei* bezieht).

>*Name*
*Arbeitszeiger* wird der angegebenen Variablen bzw. dem Register zugewiesen.

! Es wird eine Shell aufgerufen, die den Rest der Zeile hinter dem ! liest.

## adb

*$Modifikator*

Diverse Kommandos. Folgende Modifikatoren stehen zur Verfügung:

<[*f*] Kommandos aus der Datei *f* lesen. Wird dieses Kommando in einer Datei ausgeführt, bleiben die weiteren Kommandos in der Datei unberücksichtigt. Lassen Sie *f* weg, endet der aktuelle Eingabestrom hier. Geben Sie einen Wiederholungsfaktor an und dieser ist gleich Null, wird das Kommando ignoriert. Der Wert des Wiederholungsfaktors wird in der Variablen 9 abgestellt, bevor das erste Kommando in *f* ausgeführt wird.

<<[*f*] Ähnlich wie <, allerdings kann dieses Kommando in einer Kommandodatei verwendet werden, ohne daß die Datei geschlossen wird. Während der Ausführung dieses Kommandos wird die Variable 9 gerettet und bei Beendigung des Kommandos wiederhergestellt. Die Anzahl der <<-Dateien, die gleichzeitig eröffnet sein können, ist relativ gering.

>[*f*] Die Ausgabe an die Datei *f* anfügen. Die Datei wird angelegt, wenn sie noch nicht existiert. Lassen Sie *f* weg, werden die Ausgabedaten an das Terminal gegeben.

? Ausgabe der Prozeßnummer, des Signals, das zum Halt oder zur Beendigung führte sowie der Register. Dies ist der Standard, wenn Sie keinen Modifikator angeben.

r Die allgemeinen Register und die von pc adressierte Instruktion ausgeben. *Arbeitszeiger* wird auf pc gesetzt.

b Alle Prüfpunkte (Breakpoints), die zugehörigen Wiederholungsfaktoren und Kommandos ausgeben.

c Backtrace im C-Stack. Wenn *Adresse* angegeben ist, wird sie als die Adresse des aktuellen Frames und nicht als Inhalt des Frame-Zeigerregisters interpretiert. Benutzen Sie C, werden die Namen und (32-Bit-) Werte aller automatischen und statischen Variablen für jede aktive Funktion ausgegeben. Ist *Anzahl* angegeben, werden nur die ersten Zähler-Frames ausgegeben.

d Die Standard-Basis auf *Adresse* setzen und den neuen Wert zurückmelden. *Adresse* wird entsprechend der (alten) aktuellen Basis interpretiert. Durch 10$d wird die Standard-Basis also nie geändert. Soll die Standard-Basis 10 sein, müssen Sie 0t10$d angeben.

e Die Namen und Werte von externen Variablen werden ausgegeben.

w Die Seitenbreite für die Ausgabe auf *Adresse* setzen (Standard = 80).

s Die Obergrenze für Symbolübereinstimmungen auf *Adresse* setzen (Standard = 255).

o Alle eingegebenen Ganzzahlen werden als oktal betrachtet.

q Adb verlassen.

v Alle Variablen ungleich Null im Oktalformat ausgeben.

m Die Adreßtabelle (address map) ausgeben.

:*Modifikator*

Verwaltung eines Unterprozesses. Die folgenden Modifikatoren stehen zur Verfügung:

b*c* Einen Prüfpunkt (Breakpoint) auf *Adresse* setzen. Der Prüfpunkt wird Anzahl -1 mal ausgeführt, bevor das Programm gestoppt wird. Jedesmal, wenn das Programm auf den Prüfpunkt läuft, wird das Kommando *c* ausgeführt. Setzt dieses Kommando *Arbeitszeiger* auf Null, bewirkt der Prüfpunkt einen Programmstopp.

d Den Prüfpunkt auf *Adresse* löschen.

r *Objektdatei* als Unterprozeß ausführen. Ist *Adresse* explizit angegeben, erfolgt der Einsprung in das Programm an dieser Stelle; anderenfalls wird das Programm an seiner Standard-Einsprungstelle aufgerufen. Der Wert *Anzahl* gibt an, wieviele Prüfpunkte ignoriert werden sollen, bevor das Programm gestoppt wird. Argumente für den Unterprozeß können in derselben Zeile wie das Kommando angegeben werden. Ein Argument, das mit < oder > beginnt, richtet die Standardeingabe bzw. die Standardausgabe für das Kommando ein.

## adb

cs Der Unterprozeß wird mit dem Signal *s* fortgesetzt. Ist *Adresse* angegeben, wird der Unterprozeß auf dieser Adresse fortgesetzt. Ist kein Signal spezifiziert, wird das Signal gesendet, das den Unterprozeß gestoppt hat. Für das Überspringen von Prüfpunkten gilt dasselbe wie bei r.

ss Wie bei c, jedoch wird der Unterprozeß *Anzahl* Male im Einzelschritt ausgeführt. Ist kein aktueller Unterprozeß vorhanden, wird *Objektdatei* wie bei r als Unterprozeß ausgeführt. In diesem Fall kann kein Signal gesendet werden. Der Rest der Zeile wird als Argument für den Unterprozeß interpretiert.

k Der aktuelle Unterprozeß, falls ein solcher vorhanden ist, wird beendet.

### Variablen

Adb bietet eine Reihe von Variablen. Benannte Variablen werden anfangs von adb gesetzt, anschließend aber nicht mehr benutzt. Numerierte Variablen sind wie folgt für die Verständigung reserviert:

0 Der letzte ausgegebene Wert.

1 Der letzte Offset-Teil einer Instruktions-Quelle.

2 Der vorangehende Wert der Variablen 1.

9 Der Wiederholungsfaktor beim letzten Kommando $< oder $<<.

Beim Einsprung werden die folgenden Werte aus dem System-Header in der Core-Datei gesetzt. Ist die Core-Datei keine Core-Image-Datei, werden diese Werte aus *Objektdatei* übernommen und gesetzt.

b Die Basisadresse des Datensegments.

d Die Länge des Datensegments.

e Die Einsprungstelle.

m Die „magische" Zahl (0407, 0410, 0413, 0414, 0415).

s Die Länge des Stack-Segments.

t Die Länge des Text-Segments.

## adb

c Die Basisadresse des Kontroll-Stack (wird auch gesetzt, wenn ein Programm ausgeführt wird).

### Adressen

Die Adresse in einer Datei, die zu einer angegebenen Adresse gehört, wird durch eine zu der Datei gehörende Zuordnung (map) bestimmt. Jede Zuordnung wird durch zwei Tripels (*b1*, *e1*, *f1*) und (*b2*, *e2*, *f2*) dargestellt, und die einer angegebenen Adresse entsprechende Dateiadresse wird wie folgt errechnet:

$$b1 < Adresse < e1 \Rightarrow \text{Dateiadresse} = Adresse + f1 - b1$$

oder

$$b2 < Adresse < e2 \Rightarrow \text{Dateiadresse} = Adresse + f2 - b2$$

Anderenfalls ist die angeforderte Adresse unzulässig. In manchen Fällen (z. B. bei Programmen mit getrenntem I- und D-Bereich) können sich die beiden Segmente für eine Datei überlappen. Wenn sich an ein ? oder / ein * anschließt, wird nur das zweite Tripel verwendet. Die anfängliche Einstellung beider Zuordnungen eignet sich für normale a.out-Dateien und Core-Dateien. Entspricht eine Datei nicht der erwarteten Art, wird für diese Datei *b1* auf 0, *e1* auf die maximale Dateigröße und *f1* auf 0 gesetzt; auf diese Weise kann die gesamte Datei ohne Adreßumsetzung untersucht werden.

DATEIEN:

a.out

core

## adb

DIAGNOSE:

„Adb" wird ausgegeben, wenn kein aktuelles Kommando oder Format vorhanden ist.
Hinweise bei nicht-ansprechbaren Dateien, bei Syntaxfehlern, beim Abbruch von Kommandos usw. Der Rückgabecode ist 0, außer wenn das letzte Kommando erfolglos beendet wurde oder einen Rückgabecode ungleich Null geliefert hat.

HINWEISE:

Da zum Interpretieren der Argumente des Kommandos :r keine Shell aufgerufen wird, werden die üblichen Metazeichen und die Expandierung von Variablen nicht unterstützt.

Wird $c mit einer expliziten Adresse verwendet, ist der Name der Routine in der ersten Zeile des Trace möglicherweise falsch.

Derzeit gibt es keine Möglichkeit, die Anzahl der an eine Funktion übergebenen Argumente auf dem System TARGON /35 festzustellen. Deshalb gibt $c bei allen Funktionen 12 Argumente aus.

SIEHE AUCH:

cc, sdb
Systemliteratur TARGON: „Systemschnittstellen und Programmierung"

## admin

admin – Anlegen und Verwalten von SCCS-Dateien

SYNTAX:

admin [*Optionen*] [*Datei* ...]

BESCHREIBUNG:

Admin wird zum Anlegen neuer SCCS-Dateien und zum Ändern von Parametern bereits angelegter SCCS-Dateien benutzt. Schlüsselbuchstaben (nachfolgend Optionen genannt), die durch ein Minuszeichen (-) eingeleitet werden und Dateinamen (SCCS-Dateinamen müssen mit der Zeichenfolge 's.' beginnen) sind die Arten von Argumenten, die in Verbindung mit dem admin-Kommando zulässig sind. Ist die angegebene Datei noch nicht angelegt, so wird sie unter Berücksichtigung der angegebenen Optionen angelegt. Parameter, denen kein Anfangswert zugewiesen wird, erhalten einen Standardwert. Wird ein admin-Kommando zu einer bereits angelegten Datei ausgeführt, so wird der durch die Optionen spezifizierte Dateiteil verändert oder gelöscht.

Wird ein Verzeichnisname angegeben, so verhält sich das admin-Kommando so, als wären alle Dateien dieses Verzeichnisses angegeben. Nicht-SCCS-Dateien und nichtlesbare Dateien werden stillschweigend ignoriert. Wird ein '-' als Datei angegeben, so wird die Standardeingabe gelesen. Jede von der Standardeingabe eingegebene Zeile wird als Dateiname verwendet. Auch hier werden Nicht-SCCS-Dateien und nichtlesbare Dateien stillschweigend ignoriert.

Die Reihenfolge der Optionen hat keinen Einfluß auf die Bearbeitung der angegebenen Dateien. Folgende Optionen sind zulässig:

-n Diese Option signalisiert, daß eine neue SCCS-Datei angelegt werden soll.

-i[*Name*] Name der Datei, aus der der Text für die neue SCCS-Datei entnommen wird. Der Text bildet das erste Delta der Datei (siehe -r-Option bzgl. des Delta-Numerierungsschemas). Geben Sie die -i-Option ohne einen Namen an, wird der Text bis zur Erkennung des Dateiendes aus der Standardeingabe gelesen. Fehlt die -i-Option, so wird die Datei ohne Textin-

halt angelegt. Durch ein admin-Kommando mit der -i-Option kann nur jeweils eine SCCS-Datei angelegt werden. Wird ein admin-Kommando zum Anlegen mehrerer SCCS-Dateien verwendet, so werden diese Dateien ohne Textinhalt angelegt (-i-Option darf nicht verwendet werden). Die -i-Option schließt die -n-Option ein.

-r*REL* Das angegebene *Release*, in dem das Anfangsdelta eingefügt wird. Diese Option kann nur in Verbindung mit der -i-Option verwendet werden. Wird die -r-Option nicht benutzt, so wird das Anfangsdelta in das Release 1 eingefügt. Die Stufe des anlegenden Deltas ist grundsätzlich 1 (standardmäßig wird das erste Delta 1.1 genannt).

-t[*Name*] Name der Datei, der der beschreibende Text entnommen wird. Wird die -t-Option in Verbindung mit der -n oder der -i-Option verwendet, so ist der Dateiname anzugeben. Ist die SCCS-Datei bereits vorhanden, so wird der beschreibende Text bei Verwendung der -t-Option ohne Dateinamen gelöscht. Wird beim gleichen Fall *Name* angegeben, so wird der beschreibende Text ausgetauscht.

-f*Flag*[*Wert*]
Diese Option definiert ein Flag und weist ihm gegebenenfalls einen Wert zu. Mehrere -f-Optionen können in einem admin-Kommando angegeben werden. Die erlaubten *Flags* und deren Werte sind:

b ermöglicht den Gebrauch der -b-Option beim get-Kommando, um Zweigdeltas anzulegen.

c[*ceil*] Eine Zahl kleiner oder gleich 9999. Das höchste Release („Decke"), das durch das get-Kommando für Editierungen wiederhergestellt werden kann. Der Standardwert ist 9999.

f[*floor*] Eine Zahl größer als 0, aber kleiner als 9999. Das kleinste Release („Boden"), das durch das get-Kommando zur Editierung wiederhergestellt werden kann. Der Standardwert ist 1.

d[*SID*] Die SCCS-Identifikationsnummer (SID), die bei einem get-Kommando verwendet wird.

## admin

i Bewirkt, daß die 'No id keywords (ge6)'-Meldung, falls sie beim get- oder delta-Kommando auftritt, als ein schwerwiegender Fehler interpretiert wird. Ist dieses Flag nicht gesetzt, gilt diese Meldung lediglich als Warnung. Die Meldung wird ausgegeben, wenn kein SCCS-Identifikations-Schlüssel (siehe get) in dem wiederhergestellten Text gefunden wird oder in der SCCS-Datei abgespeichert ist.

j Erlaubt konkurrierende get-Kommandos zur Editierung des gleichen SID einer SCCS-Datei. Somit werden verschiedene konkurrierende Änderungen der SCCS-Datei ermöglicht.

l[*Liste*] Liste von Releases, zu denen keine Deltas mehr angelegt werden können (wird dennoch ein get -e-Kommando mit einer solchen Version aufgerufen, erfolgt eine entsprechende Fehlermeldung).

Die Liste hat folgende Syntax:

<*Liste*> ::= <*Umfang*> | <*Liste*>,<*Umfang*>
<*Umfang*> ::= <*Releasenummer*> | a

Der Buchstabe a in der Liste ist gleichbedeutend der Spezifizierung aller Releases der genannten SCCS-Datei.

n Veranlaßt das delta-Kommando 'wertlose' Deltas für übersprungene Releases anzulegen (z. B. Delta 5.1 wird nach Delta 2.7 angelegt, Release 3 und 4 werden übersprungen). Geben Sie dieses Flag an, dienen die 'wertlosen' Deltas als Verankerungspunkte für spätere Zweigdeltas. Wird das 'n'-Flag nicht angegeben, werden die Verankerungspunkte nicht angelegt und folglich können keine Zweigdeltas in den übersprungenen Releases angebracht werden.

q[*Text*] Vom Benutzer definierbarer Text, der alle Vorkommen des %Q%-Schlüssels bei der Wiederherstellung durch get ersetzt.

admin

m[*Mod*] Modulname der SCCS-Datei, der alle Vorkommen des %M%-Schlüssels im SCCS-Dateitext bei der Wiederherstellung mittels get-Kommando ersetzt. Ist das 'm'-Flag nicht angegeben, wird der %M%-Schlüssel durch den SCCS-Dateinamen – ohne die Zeichenfolge 's.' – ersetzt.

t[*Typ*] Bei der Wiederherstellung eines Deltas durch das get-Kommando werden alle abgespeicherten %Y%-Schlüssel durch den Typ des Moduls ersetzt.

v[*Pgm*] Bei der Ausführung des delta-Kommandos verlangt delta Modification-Request-Nummern, die die Änderung begründen. Der optionale Inhalt gibt den Namen eines 'MR-Nummern-Prüfprogramms' an. Geben Sie das 'v'-Flag beim Anlegen der Datei an, muß auch die -m-Option angegeben werden, auch wenn diese keinen Inhalt hat.

-d*Flag* Veranlaßt das Löschen des angegebenen Flags aus der SCCS-Datei. Die -d-Option kann nur dann angegeben werden, wenn bereits angelegte SCCS-Dateien bearbeitet werden. Mehrere -d-Optionen können an ein einzelnes admin-Kommando angefügt werden. Alle erlaubten Flags sind unter der Beschreibung der -f-Option zu finden.

-l[*Liste*] Liste von Releases, die nicht gesperrt sein sollen. Unter der Beschreibung der -f-Option ist die Beschreibung des 'l'-Flags und der Syntax von *Liste* zu finden.

-a*Name* Login-Name oder Gruppen-ID der bzw. die an die Liste der Benutzer mit der Erlaubnis SCCS-Dateien anzulegen, angefügt wird. Wird eine Gruppen-ID angegeben, so sind darin alle Mitglieder der Gruppe enthalten. Mehrere -a-Optionen dürfen in einem admin-Kommando angegeben werden. Ist die Benutzerliste leer, kann jeder Benutzer Deltas anfügen.

-e*Name* Login-Name oder Gruppen-ID der bzw. die aus der Benutzerliste gelöscht werden soll. Um eine gesamte Benutzergruppe aus der Benutzerliste zu löschen, kann man die Gruppen-ID bzw. die Login-Namen aller Gruppenmitglieder angeben. Die -e-Option können Sie mehrfach in einem admin-Kommando angeben.

## admin

-y[*Kommentar*]

Der angegebene Kommentar wird in der SCCS-Datei für das erste Delta abgelegt. Die Einfügung erfolgt in der vom delta-Kommando bekannten Weise. Geben Sie diese Option nicht an, wird eine Standardkommentarzeile folgenden Formats eingefügt:

date and time created JJ/MM/TT HH:MM:SS by <Login-Name>

Die Ausgabe der -y-Option ist nur in Verbindung mit der -i und/oder -n-Option möglich (Anlegen einer SCCS-Datei).

-m[*MR-Liste*]

Die angegebene Liste der Modification Request (*MR*) Nummern wird in die SCCS-Datei eingefügt. Das 'v'-Flag muß gesetzt sein und die *MR*-Nummern werden bestätigt, wenn das 'v'-Flag einen Inhalt hat (den Namen des *MR*-Nummern-Inkraftsetzungsprogramms). Fehlermeldungen erfolgen, wenn Sie das 'v'-Flag nicht gesetzt haben oder die *MR*-Inkraftsetzung fehlschlägt.

-h Veranlaßt admin zur Prüfung der SCCS-Datei. Die Prüfung erfolgt durch den Vergleich einer neu errechneten Prüfsumme (die Summe aller Zeichen der SCCS-Datei, außer die der ersten Zeile) mit der Prüfsumme, die in der ersten Zeile der SCCS-Datei abgelegt ist. Es werden entsprechende Meldungen ausgegeben.

Diese Option bewirkt einen Schreibschutz für die angegebene Datei und hebt die Wirkung aller nachfolgenden Optionen auf. Diese Option ist nur sinnvoll, wenn die zu bearbeitende Datei bereits angelegt ist.

-z Die SCCS-Datei-Prüfsumme wird neu berechnet und in der SCCS-Datei in der ersten Zeile abgestellt (siehe -b).

## admin

DATEIEN:

Die letzte Komponente aller SCCS-Dateinamen muß die Form s.*Dateiname* haben. Neue SCCS-Dateien bekommen den Modus 444 (r--r--r--). Um eine Datei anlegen zu können, muß der Benutzer natürlich Schreiberlaubnis in dem entsprechenden Verzeichnis haben. Admin führt alle Schreibvorgänge in einer temporären Datei, der X-Datei, aus (X.Dateiname siehe get). Die X-Datei wird mit dem Modus 444 angelegt, wenn admin eine neue SCCS-Datei anlegt. Ist die entsprechende SCCS-Datei bereits vorhanden, wird die X-Datei mit dem gleichen Modus angelegt. Nach erfolgreicher Ausführung von admin wird die SCCS-Datei (falls sie angelegt ist) gelöscht und die X-Datei mit dem Namen der SCCS-Datei versehen. Diese Vorgehensweise gewährleistet, daß SCCS-Dateien nur bei fehlerfreiem Durchlauf geändert werden.

Verzeichnisse, die SCCS-Dateien enthalten, sollten den Modus 755 (rwxr-xr-x), SCCS-Dateien selbst den Modus 444 erhalten. Durch diese Zugriffsrechte kann nur der Besitzer des Verzeichnisses Änderungen vornehmen; SCCS-Dateien können nur durch SCCS-Kommandos geändert werden.

Ist es aus irgendeinem Grund erforderlich auf eine SCCS-Datei schreibend zuzugreifen, sollten Sie den Erlaubnisstatus auf 644 (rw-r--r--) setzen, so daß Sie Änderungen der Datei mit Hilfe des Standard-Editors ed vornehmen können.

Bei einer Änderung sollten Sie vorsichtig vorgehen. Mit dem admin -h-Kommando können Sie prüfen, ob eine Datei fehlerhaft ist. Haben Sie die Korrektur beendet, muß ein admin -z-Kommando abgesetzt werden, damit die Prüfsumme neu berechnet wird und die SCCS-Datei wieder als fehlerfrei gilt. Um die Richtigkeit zu prüfen, sollten Sie anschließend noch ein admin -h-Kommando absetzen.

Admin benutzt temporäre Lock-Dateien (Z.*Dateiname*), um gleichzeitige Änderungen von SCCS-Dateien durch verschiedene Benutzer auszuschließen. Weitere Informationen sind der Beschreibung des get-Kommandos zu entnehmen.

## admin

SIEHE AUCH:

delta, ed, get, help, prs, what
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# ar

ar – Archivdatei erstellen und aktualisieren

SYNTAX:

ar *Funktion* [*Position*] *Archivdatei Datei* ...

BESCHREIBUNG:

Ar verwaltet gleichartige Dateien, die in einer Bibliothek (Archivdatei) zusammengefaßt sind. Der Parameter *Funktion* gibt an, welche Bearbeitung erfolgen soll.

Im folgenden sind die möglichen Funktionen erläutert:

d delete.
Die angegebenen Dateien werden aus der angegebenen Archivdatei gelöscht.

r replace.
Die angegebenen Dateien werden in der Archivdatei ersetzt. Wird zusätzlich die Option 'u' verwendet, werden nur die Dateien ersetzt, deren Daten später verändert wurden, als die Archivdateien. Wenn Sie eines der Positionszeichen 'abi' anwenden, muß das Argument *Position* vorhanden sein und angeben, ob neue Dateien hinter (a) oder vor (b oder i) *Position* stehen sollen. Anderenfalls werden neue Dateien an das Ende der Archivdatei angefügt.

q quickly.
Die angegebenen Dateien werden in der vorgegebenen Reihenfolge an das Ende der Archivdatei angefügt. Das Programm prüft nicht, ob die Dateien bereits in der Bibliothek vorhanden sind.

t table.
Es wird eine Tabelle ausgedruckt, in der der Inhalt der Archivdatei aufgeführt ist. Geben Sie keine Dateinamen vor, werden sämtliche in der Bibliothek enthaltenen Dateien ausgedruckt.

p print.
Die angegebenen Dateien in der Bibliothek werden ausgegeben.

ar

m move.
Die angegebenen Dateien werden an das Ende der Bibliothek verlegt. Wenn Sie ein Positionszeichen verwenden, muß das Argument *Position* vorhanden sein und – wie bei 'r' – angeben, wohin die Dateien verlegt werden sollen.

x extract.
Die angegebenen Dateien werden aus der Bibliothek kopiert. Geben Sie keine Dateinamen an, werden sämtliche in der Archivdatei enthaltenen Dateien ausgegeben. Durch die Funktion 'x' wird die Archivdatei nicht verändert.

Die genannten Funktionen können mit Hilfe folgender Optionen modifiziert werden:

v verbose.
Diese Option liefert Angaben zu den bearbeiteten oder in der Bibliothek enthaltenen Dateien.
Verwenden Sie 'v' zusammen mit 't', erhalten Sie eine ausführliche Liste aller Dateiinformationen.
Wenden Sie zusätzlich mit 'v' 'x' an, wird jeder Datei eine Bezeichnung vorangestellt.

c create.
Unterdrückt die standardmäßige Nachricht über die Kreierung einer neuen Archivdatei.

l local.
Temporäre Dateien, die normalerweise im Verzeichnis /tmp angelegt werden, werden im aktuellen Verzeichnis kreiert.

DATEIEN:

/tmp/v* temporär

HINWEIS:

Das convert-Kommando können Sie benutzen, um ältere Archivdateien so umzuwandeln, daß sie mit ar bearbeitet werden können.

SIEHE AUCH:

ld, lorder

## asa

asa – Interpretation von ASA-Vorschubsteuerzeichen

SYNTAX:

asa [*Dateien*]

BESCHREIBUNG:

Asa interpretiert die Ausgabedaten von Fortran-Programmen, in denen ASA-Vorschubsteuerzeichen vorkommen. Das Kommando verarbeitet entweder die Dateien, die als Argumente angegeben sind, oder bei fehlenden Dateinamen die Standardeingabe. Es wird grundsätzlich davon ausgegangen, daß das erste Zeichen einer Zeile ein Steuerzeichen ist. Die Steuerzeichen haben folgende Bedeutung:

' ' (Blank) einfacher Zeilenvorschub vor dem Drucken

0 Doppelter Zeilenvorschub vor dem Drucken

1 Seitenvorschub vor dem Drucken

\+ Vorangehende Zeile überdrucken

Zeilen, die mit keinem der obengenannten Zeichen beginnen, werden so behandelt, als ob sie mit ' ' beginnen. Das erste Zeichen einer Zeile wird nicht gedruckt. Wenn solche Zeilen vorkommen, wird eine entsprechende Diagnosemeldung auf der Standardfehlerausgabe ausgegeben. Dieses Programm bewirkt, daß die erste Zeile jeder Eingabedatei auf einer neuen Seite beginnt.

Um die Ausgabedaten von Fortran-Programmen, die ASA-Vorschubsteuerzeichen enthalten, in korrekter Form zu betrachten, kann man asa in folgender Weise als Filter benutzen:

a.out | asa | lp

Die Ausgabedaten können dann entsprechend formatiert und in Seiten eingeteilt dem Drucker zugeleitet werden. Fortran-Ausgabedaten, die in eine Datei geschickt wurden, können wie folgt angezeigt werden:

asa *Datei*

SIEHE AUCH:

f77, fsplit, ratfor

# awk

## awk – Mustererkennungs- und Verarbeitungssprache

SYNTAX:

awk [-Fc] [*Programm*] [*Parameter*] [*Datei* ...]

BESCHREIBUNG:

Awk durchsucht jede Eingabedatei nach Zeilen, auf die ein oder mehrere Muster aus *Programm* passen. Mit jedem Muster in *Programm* kann eine Aktion verbunden sein, die ausgeführt wird, wenn das Muster auf eine Textzeile paßt. Die Muster können entweder im Klartext auf der Kommandozeile als *Programm* erscheinen oder mit -f *Programm* angegeben werden.

Sie können Parameter in der Form x=..., y=... an awk übergeben.

Die Eingabedateien werden der Reihe nach gelesen; sind keine angegeben, so wird aus der Standardeingabe gelesen; der Dateiname '-' bezeichnet dabei die Standardeingabe. Jede Zeile wird mit dem Musterteil jeder Programmzeile verglichen; die zugehörige Aktion wird für jede gefundene Übereinstimmung ausgeführt.

Eine Eingabezeile besteht aus Feldern, die durch Tabulatorzeichen oder Leerzeichen (white space) getrennt sind. (Diese Standardeinstellung kann durch FS geändert werden, s. u.) Die Felder werden mit $1, $2, ..., $n bezeichnet; $0 bezieht sich auf die gesamte Zeile.

Eine Programmzeile hat die Form:

*Muster* {*Aktion*}

Eine fehlende {*Aktion*} bedeutet, daß die Eingabezeile ausgegeben wird. Ein fehlendes *Muster* paßt auf alle Zeilen.

## awk

Eine *Aktion* ist eine Reihe von Anweisungen der folgenden Form:

if (*Bedingung*) *Anweisung* [else *Anweisung*]

while (*Bedingung*) *Anweisung*

for (*Ausdruck*; *Bedingung*; *Ausdruck*) *Anweisung*

break

continue

{[*Anweisung*] ...}

*Variable* = *Ausdruck*

print [*Liste von Ausdrücken*] [>*Ausdruck*]

printf *Format* [, *Liste von Ausdrücken*] [>*Ausdruck*]

next # Überspringe restliche Muster in der Eingabezeile

exit # Überspringe den Rest der Eingabe

Eine Anweisung wird durch ein Semikolon, ein Zeilenendezeichen oder die geschlossene geschweifte Klammer beendet. Eine leere Liste von Ausdrücken steht für die gesamte Zeile. Ein Ausdruck nimmt je nach Kontext entweder einen Textwert oder einen numerischen Wert an. Er wird mit Hilfe der Operatoren '+', '-', '*', '/', '%' und ' ' (Leerzeichen für Verkettung) gebildet.

Die C-Operatoren '++', '--', '+=', '==', '*=', '/=' und '%=' können Sie ebenfalls in Ausdrücken verwenden. Variable können Skalare, Arrayelemente (mit x[i] bezeichnet) oder Felder sein. Sie werden mit der leeren Zeichenkette initialisiert. Array-Indizes müssen nicht unbedingt numerisch sein; dies erlaubt eine Art von assoziativer Speicherung. Zeichenkettenkonstanten werden mit doppelten Anführungsstrichen ("...") markiert.

Die print-Anweisung gibt ihre Argumente auf die Standardausgabe aus (oder in eine Datei, wenn >*Datei* angegeben ist), getrennt durch das aktuelle Ausgabefeldtrennzeichen und durch das Ausgabesatztrennzeichen. Printf formatiert die Liste von Ausdrücken gemäß dem angegebenen Format.

## awk

Die eingebaute Funktion length gibt die Länge ihrer Argumente, als Zeichenkette aufgefaßt, zurück. Falls keine Argumente vorhanden sind, ist die Länge der Gesamtzeile gemeint. Außerdem gibt es die eingebauten Funktionen exp, log, sqrt und int. Letztere schneidet ihr Argument auf einen ganzzahligen Wert ab.

Substr(*s*, *m*[, *n*]) gibt eine Teilkette von *s* der Länge *n* zurück, die bei Position *m* beginnt. Die Funktion sprintf(*Fmt*, *Ausdruck*) formatiert die angegebenen Ausdrücke gemäß dem in *Fmt* angegebenen Format und gibt die resultierende Zeichenkette zurück.

Ein *Muster* ist eine beliebige logische Verknüpfung ('!', '||', '&&') von regulären Ausdrücken und Vergleichsausdrücken. Reguläre Ausdrücke müssen von Schrägstrichen eingeschlossen sein und sind die gleichen wie in egrep. Einzelne reguläre Ausdrücke in einem Muster beziehen sich auf die gesamte Zeile. Sie können auch in Vergleichsausdrücken vorkommen.

Ein Muster kann aus zwei Mustern bestehen, die durch ein Komma getrennt sind. In diesem Fall wird die Aktion für alle Zeilen zwischen dem Vorkommen des ersten und dem Vorkommen des zweiten Musters ausgeführt.

Ein *Vergleichsausdruck* ist einer der folgenden:

*Ausdruck Erkennungsausdruck regulärer Ausdruck*

*Ausdruck Vergleichsoperator Ausdruck*

wobei *Vergleichsoperator* irgendeiner der 6 Vergleichsoperatoren in C sein kann und *Erkennungsoperator* entweder '~' (enthält) oder '!~' (enthält nicht).

Eine *Bedingung* ist ein arithmetischer Ausdruck, ein Vergleichsausdruck oder eine logische Kombination von beiden.

Die speziellen Muster BEGIN und END können benutzt werden, um die Steuerung vor der ersten und nach der letzten Zeile zu übernehmen. BEGIN muß dabei das erste Muster sein und END das letzte.

## awk

Ein einzelnes Zeichen *c* können Sie als Feldtrennzeichen definieren, indem Sie das Programm mit

BEGIN {FS=*c*}

beginnen oder beim Programmaufruf die '-F*c*'-Option benutzen.

Andere Variablennamen mit spezieller Bedeutung sind NF (Anzahl der Felder in der laufenden Eingabezeile); NR (die Ordnungsnummer der laufenden Eingabezeile); FILENAME (Name der momentanen Eingabedatei); OFS (Ausgabefeldtrennzeichen, standardmäßig das Leerzeichen); ORS (Ausgabesatztrennzeichen, standardmäßig das Zeilenendezeichen) und OFMT (Ausgabeformat für Zahlen, standardmäßig %.6g ).

Beispiele:

Gebe alle Zeilen aus, die länger als 72 Zeichen sind:

```
length > 72
```

Gebe die beiden ersten Felder in umgekehrter Reihenfolge aus:

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

Addiere jeweils das erste Feld, gebe die Summe und den Durchschnitt aus:

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

Gebe die Felder in umgekehrter Reihenfolge aus:

```
{for (i = NF; i > 0; --i) print $i}
```

Gebe alle Zeilen zwischen dem Vorkommen von /start/ und /stop/ aus (diese eingeschlossen):

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

## awk

Gebe alle Zeilen aus, deren erstes Feld vom ersten Feld der vorherigen Zeilen verschieden ist:

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

HINWEIS:

Es gibt keine explizite Unterscheidung zwischen Zahlen und Zeichenketten. Damit ein Ausdruck als Zahl aufgefaßt wird, addiere man 0; damit er als Zeichenkette aufgefaßt wird, hänge man die leere Zeichenkette an (' ').

SIEHE AUCH:

grep, lex, sed
Systemliteratur TARGON: „Textmustererkennung und Verarbeitung"

# banner

banner – Vergrößerte Ausgabe der angegebenen Argumente

SYNTAX:

banner *Argument* ...

BESCHREIBUNG:

Banner gibt die angegebenen Argumente (jedes bis zu 10 Zeichen lang) auf der Standardausgabe aus. Geben Sie Kleinbuchstaben ein, werden diese in Großbuchstaben umgewandelt.

SIEHE AUCH:

echo

## basename, dirname

basename, dirname – Liefern von Pfadnamenteilen

SYNTAX:

basename *Zeichenkette* [*Endung*]

dirname *Zeichenkette*

BESCHREIBUNG:

Basename löscht den in *Zeichenkette* angegebenen Pfadnamen – bis auf die letzte Komponente – und falls vorhanden – die angegebene Endung. Das Resultat wird in die Standardausgabe geschrieben.
Dieser Befehl wird normalerweise innerhalb von Shell-Prozeduren angewendet.

Dirname liefert den in *Zeichenkette* angegebenen Pfadnamen, mit Ausnahme der letzten Komponente.

Die folgende Shell-Prozedur, mit dem Argument '/usr/src/cmd/cat.c' aufgerufen, kompiliert die genannte Datei und speichert das Ergebnis im aktuellen Verzeichnis in der Datei cat:

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

Im zweiten Beispiel wird der Shell-Variablen NAME der Wert '/usr/src/cmd' zugewiesen:

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

HINWEIS:

Der Basisname '/' ist gleich Null und verursacht einen Fehler.

SIEHE AUCH:

sh

## bc

bc – Sprache für Arithmetik mit beliebiger Genauigkeit

SYNTAX:

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

BESCHREIBUNG:

Bc ist ein interaktiver Prozessor für eine Sprache, die C ähnelt, aber eine Arithmetik mit unbegrenzter Genauigkeit bietet. Bc übernimmt Eingabedaten von beliebigen Dateien und liest sodann die Standardeingabe. Das Argument -l steht für den Namen einer mathematischen Bibliothek mit beliebiger Genauigkeit. Bc-Programme haben folgende Syntax: L bedeutet Buchstabe (letter) von a-z, E bedeutet Ausdruck (expression), S bedeutet Anweisung (statement).

Kommentare
werden in /* und */ eingeschlossen.

Namen
einfache Variablen: L
Array-Elemente: L[E]
Die Worte „ibase", „obase" und „scale"

Weitere Operanden
beliebig lange Zahlen wahlweise mit Vorzeichen und Dezimalpunkt.
(E)
sqrt(E)
length(E) Anzahl signifikanter Dezimalziffern
scale(E) Anzahl Nachkommastellen
L(E,...,E)

Operatoren
+ - * / % ^ (% bedeutet Rest, ^ bedeutet Potenzierung)
++ -- (Präfix und Postfix; gilt für Namen)
== <= >= != < >
= =+ =- =* =/ =% =^

## bc

Anweisungen

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

Funktionsdefinitionen

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

Funktionen in mathematischer Bibliothek mit -l

| | |
|---|---|
| s(x) | Sinus |
| c(x) | Cosinus |
| e(x) | Exponentialfunktion |
| l(x) | Logarithmus |
| a(x) | Arcustangens |
| j(n,x) | Bessel-Funktion |

Alle Funktionsargumente werden mit ihrem Wert übergeben („call by value").

Der Wert einer Anweisung wird, wenn er ein Ausdruck ist, ausgegeben, es sei denn der Hauptoperator ist eine Zuweisung. Anweisungen können durch Semikolon oder Neue-Zeile-Zeichen getrennt werden. Eine Zuweisung zu scale beeinflußt die Anzahl der Stellen, die bei arithmetischen Operationen beibehalten werden, in der Form wie bei dc. Durch eine Zuweisung zu ibase oder obase wird die Basis der Eingabezahl bzw. der Ausgabezahl gesetzt.

## bc

Ein und derselbe Buchstabe kann gleichzeitig als Array, als Funktion und als einfache Variable verwendet werden. Alle Variablen haben globale Bedeutung für das Programm. Variablen mit „auto" werden bei Funktionsaufrufen weiter nach unten geschoben. Wenn Arrays als Funktionsargumente verwendet oder als automatische Variablen definiert werden, müssen hinter dem Array-Namen leere eckige Klammern stehen.

Bc ist effektiv ein Preprozessor für dc, und bc ruft automatisch dc auf, außer wenn die Option -c (nur kompilieren) angegeben ist. In diesem Fall werden die Eingabedaten für dc stattdessen an die Standardausgabe übergeben.

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 == 0) return(s)
                s = s+c
        }
}
```

definiert eine Funktion zur Errechnung des annähernden Wertes der Exponentialfunktion, und

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

gibt die annähernden Werte der Exponentialfunktion der ersten zehn Ganzzahlen aus.

## bc

DATEIEN:

| | |
|---|---|
| usr/lib/lib.b | Mathematische Bibliothek |
| /usr/bin/dc | Eigentliche Tischrechnerfunktion |

HINWEISE:

&&, || werden noch nicht unterstützt.
In for-Anweisungen müssen alle drei E's codiert sein.
Quit wird beim Lesen und nicht beim Ausführen interpretiert.

SIEHE AUCH:

dc

## bdiff

bdiff – Vergleichen von großen Dateien

SYNTAX:

bdiff *Datei1 Datei2* [*n*] [-s]

BESCHREIBUNG:

Bdiff wird in der gleichen Weise angewandt wie der diff-Befehl, um festzustellen, welche Zeilen in den angegebenen Dateien geändert werden müssen, damit sie identisch werden. Bdiff wird benutzt, um Dateien zu vergleichen, die für die Anwendung von diff zu lang sind. Bdiff ignoriert identische Zeilen am Anfang der Dateien, teilt den verbleibenden Rest in Abschnitte zu *n* Zeilen auf und veranlaßt den Vergleich der einzelnen Abschnitte durch diff. Default-Wert von *n* ist 3500. Die Angabe eines Wertes für *n* ist sinnvoll, wenn Abschnitte von 3500 Zeilen zu groß zum Vergleichen für diff sind. Geben Sie *Datei1* oder *Datei2* mit '-' an, wird die Standardeingabe gelesen.

Die Eingabe der -s-Option bewirkt, daß keine Fehlermeldung erfolgt.

Die Ausgabe von bdiff ist exakt die gleiche wie die Ausgabe von diff. Es werden Zeilennummern angegeben, die sich auf die einzelnen Abschnitte beziehen.

DATEI:

/tmp/bd?????

SIEHE AUCH:

diff

## bfs

### bfs – Programm zum Durchsuchen großer Dateien (big file scanner)

SYNTAX:

bfs [-] *Name*

BESCHREIBUNG:

Bfs hat Ähnlichkeit mit ed, allerdings kann es nur lesen und verarbeitet wesentlich größere Dateien. Die Dateien können bis zu 1024 KByte groß sein und bis zu 32 K Zeilen mit bis zu 255 Zeichen pro Zeile umfassen. Bfs ist beim Absuchen einer Datei in der Regel leistungsfähiger als ed, da die Datei nicht in einen Puffer kopiert wird. Die Routine ist besonders gut brauchbar zum Erkennen von Abschnitten einer großen Datei, die mit csplit zum Editieren in kleinere, besser handhabbare Teile unterteilt werden kann.

In der Regel wird die Größe der abgesuchten Datei in derselben Form angezeigt, wie die Größe einer Datei, die mit dem Kommando w weggeschrieben wird. Wird das wahlfreie - angegeben, so wird die Größe nicht angezeigt. Eingaben werden mit * angefordert, wenn P und eine Zeilenschaltung eingegeben werden (wie bei ed). Die Aufforderung (Prompt) zur Eingabe läßt sich durch erneute Eingabe von P und Zeilenschaltung wieder ausschalten. Ist Prompt-Modus aktiv, werden bei Fehlern auch Fehlermeldungen ausgegeben.

Es werden alle bei ed beschriebenen Adreßausdrücke unterstützt. Außerdem können reguläre Ausdrücke neben / und ? mit zwei weiteren Symbolen umgeben werden: > bedeutet Suchen in Vorwärtsrichtung ohne zyklische Fortsetzung bei Dateiende, und < bedeutet Suchen in Rückwärtsrichtung ohne zyklische Fortsetzung am Dateiende. Bei den Markierungsnamen gibt es einen geringfügigen Unterschied: Es können nur die Buchstaben a bis z verwendet werden, und alle 26 Markierungen werden gespeichert.

Die Kommandos e, g, v, k, p, q, w, =, ! und das Leerkommando haben dieselbe Wirkung wie bei ed. Es können auch Kommandos in der Form ---, +++-, +++=, -12 und +4p gebildet werden. Die Kommandos 1,10p und 1,10 bewirken beide die Ausgabe der ersten zehn Zeilen. Das

## bfs

Kommando f gibt nur den Namen der zu bearbeitenden Datei aus; das System merkt sich keinen Dateinamen. Das Kommando w ist unabhängig von der Umleitung, der Verkürzung und der Komprimierung der Ausgabe (siehe die Kommandos xo, xt und xc unten). Es stehen die folgenden zusätzlichen Kommandos zur Verfügung:

xf*Datei* Weitere Kommandos werden aus *Datei* übernommen. Ist das Dateiende erreicht, trifft ein Unterbrechungssignal ein oder tritt ein Fehler auf, liest das System in der Datei weiter, die das xf enthält. Die Kommandos xf können bis zu zehn Stufen geschachtelt werden.

xn Die derzeit verwendeten Markierungen listen. (Markierungen werden mit dem Kommando k gesetzt.)

xo[*Datei*] Weitere Eingaben vom Kommando p und vom Leerkommando werden in *Datei* umgeleitet, die nötigenfalls mit Modus 666 angelegt wird. Geben Sie keine Datei an, werden die Ausgabedaten zur Standardausgabe umgeleitet. Jedes Umleiten bewirkt ein Verkürzen oder Erstellen der Datei.

:*Marke* Hierdurch wird eine Marke (Label) in einer Kommandodatei positioniert. Die Marke wird durch das Neue-Zeile-Zeichen beendet, und Leerzeichen zwischen : und dem Markenanfang werden ignoriert. Mit diesem Kommando können auch Kommentare in eine Kommandodatei eingefügt werden, da Marken (Labels) nicht unbedingt angesprochen werden müssen.

(...)xb/*regulärer Ausdruck/Marke*
Ist der Ausführungsverlauf des Kommandos positiv, erfolgt ein Sprung (vorwärts oder rückwärts) zu der angegebenen Marke (Label). Unter folgenden Bedingungen verläuft das Kommando negativ:

1. Eine der Adressen liegt nicht zwischen 1 und $.
2. Die zweite Adresse ist kleiner als die erste.
3. Der reguläre Ausdruck stimmt mit keiner Zeile in dem angegebenen Bereich einschließlich der ersten und letzten Zeile überein.

bfs

Bei positivem Verlauf wird ein Punkt auf die gefundene Zeile gesetzt und ein Sprung zur angegebenen Marke durchgeführt. Dies ist das einzige Kommando, das bei fehlerhaften Adressen keine Fehlermeldung ausgibt. Deshalb kann damit getestet werden, ob Adressen fehlerhaft sind, bevor andere Kommandos ausgeführt werden. Es wird darauf hingewiesen, daß das Kommando

xb/~/*Marke*

ein unbedingter Sprung ist.
Das Kommando xb ist nur dann zulässig, wenn es nicht von einem Terminal gelesen wird. Wird es aus einer Pipe gelesen, ist nur ein Sprung in Vorwärtsrichtung möglich.

xt*Anzahl* Die Ausgabedaten vom Kommando p und vom Leerkommando werden auf maximal *Anzahl* Zeichen verkürzt. Die anfängliche Anzahl beträgt 255.

xv[*Ziffer*][*Leerzeichen*][*Wert*]
Der Variablenname ist die direkt hinter xv angegebene Ziffer. Die Kommandos xv5100 und xv5 100 sind gleichbedeutend und weisen der Variablen 5 den Wert 100 zu. Das Kommando Xv61,100p weist der Variablen 6 den Wert 1,100p zu. Soll eine Variable angesprochen werden, müssen Sie ein % vor den Variablennamen stellen. Werden die vorgenannten Zuweisungen für die Variablen 5 und 6 zugrundegelegt, geben

1,%5p
1,%5
%6

die ersten 100 Zeilen aus.

g/%5/p

sucht global nach den Zeichen 100 und gibt alle Zeilen aus, in denen diese Zeichen gefunden werden. Um die spezielle Bedeutung von % zu deaktivieren, muß dem Zeichen ein \ vorangestellt werden.

bfs

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

wäre eine Möglichkeit, Zeilen zu finden und zu listen, die printf für Zeichen, dezimale Ganzzahlen oder Strings enthalten.
Eine weitere Möglichkeit, die das Kommando xv bietet, ist die, daß die erste Ausgabezeile eines UNIX-Systemkommandos in einer Variablen abgelegt werden kann. Voraussetzung dafür ist lediglich, daß das erste Zeichen Wertzuweisung ein ! ist. Beispiel:

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

stellt die aktuelle Zeile in die Variable 5, gibt sie aus und erhöht die Variable 6 um eins. Soll die spezielle Bedeutung von ! als erstes Zeichen der Wertzuweisung deaktiviert werden, so muß ein \ vorangestellt werden.

```
xv7\!date
```

legt den Wert !date in Variable 7 ab.

xbz*Marke*
xbn*Marke* Diese beiden Kommandos fragen den letzten unter der angegebenen Marke gespeicherten Rückgabecode von der Ausführung eines UNIX-Systemkommandos (!Kommando) bzw. einen Wert ungleich Null ab. In beiden folgenden Beispielen wird nach den nächsten fünf Zeilen gesucht, die die Zeichenkette size enthalten.

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

## bfs

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

xc[*Schalter*]

Hat *Schalter* den Wert 1, werden die Ausgabedaten vom Kommando p und vom Leerkommando komprimiert. Bei Schalter gleich 0 erfolgt keine Komprimierung. Das Kommando xc ohne Argument wechselt den Wert von *Schalter*. Anfangs ist *Schalter* so gesetzt, daß keine Komprimierung erfolgt. Bei der komprimierten Ausgabe werden Folgen von Tabs und Blanks auf ein Blank reduziert, und leere Zeilen werden unterdrückt.

DIAGNOSE:

„?" bei fehlerhaften Kommandos, wenn Sie sich nicht im Kommandomodus befinden.
Sind Sie im Kommandomodus, werden selbsterklärende Fehlermeldungen ausgegeben.

SIEHE AUCH:

csplit, ed

## bs

bs – Compiler/Interpreter für relativ kleine Programme

SYNTAX:

bs [*Datei* [*Argumente*]]

BESCHREIBUNG:

Bs wurde aus Basic and Snobol4 entwickelt und hat einige zusätzliche Merkmale der Programmiersprache C. Bs ist für solche Programmieraufgaben vorgesehen, bei denen die Programmentwicklungszeit genauso wichtig ist wie die Ausführungsgeschwindigkeit des Programms. Die Formalitäten der Datendeklaration und Datei-/Prozeßmanipulation sind auf ein Minimum reduziert. Das zeilenweise Debugging, die Trace- und Dump-Anweisungen sowie aussagekräftige Fehlermeldungen zur Laufzeit vereinfachen das Testen von Programmen. Außerdem bietet bs die Möglichkeit, unvollständige Programme zu testen. Innere Funktionen können getestet werden, bevor die äußeren Funktionen geschrieben werden und umgekehrt.

Geben Sie in der Kommandozeile eine Datei an, werden Eingaben aus dieser Datei genommen, bevor die Standardeingabe gelesen wird. Standardmäßig werden Anweisungen, die aus der Datei gelesen werden, für eine spätere Ausführung kompiliert. Anweisungen, die über die Standardeingabe eingegeben werden, werden normalerweise sofort ausgeführt (siehe Kompilieren und Ausführen weiter unten). Handelt es sich bei der abschließenden Operation nicht um eine Zuweisung, wird das Ergebnis des Ausdrucks direkt berechnet und ausgegeben.

Bs-Programme bestehen aus Eingabezeilen. Ist das letzte Zeichen in einer Zeile ein \, wird die Zeile fortgesetzt. Bs läßt Zeilen der folgenden Form zu:

*Anweisung*
*Marke Anweisung*

Eine Marke (Label) ist ein Name (siehe unten), an den sich ein Doppelpunkt anschließt. Eine Marke und eine Variable können denselben Namen haben.

## bs

Eine bs-Anweisung ist entweder ein Ausdruck oder ein Schlüsselwort, an das sich null oder mehrere Ausdrücke anschließen. Einige Schlüsselwörter (clear, compile, !, execute, include, ibase, obase und run) werden beim Kompilieren grundsätzlich ausgeführt.

### Anweisungssyntax

*Ausdruck* Der angegebene Ausdruck wird ausgeführt, um seine Nebenwirkungen festzustellen (Wert, Zuweisung oder Funktionsaufruf). Die Einzelheiten der Ausdrücke entsprechen der Beschreibung der Anweisungstypen weiter unten.

break Springt aus der innersten for/while-Schleife.

clear Löscht die Symboltabelle und die kompilierten Anweisungen. Clear wird sofort ausgeführt.

compile[*Ausdruck*]
Nachfolgende Anweisungen werden kompiliert (setzt die standardmäßig geltende sofortige Ausführung außer Kraft). *Ausdruck* wird ausgewertet und als Dateiname für die weitere Eingabe verwendet. Mit dem letzteren Fall ist ein clear (Loschen) verknüpft. Compile wird sofort ausgeführt.

continue Überspringt die restlichen Befehle der aktuellen For/while-Schleife und testet erneut die Endebedingung.

dump[*Name*]
Der Name und der aktuelle Wert aller nicht-lokalen Variablen wird ausgegeben. Wahlweise werden nur die benannten Variablen ausgewertet. Nach einem Fehler oder einer Unterbrechung werden die Nummer der letzten Anweisung und (möglicherweise) die Benutzerfunktion trace ausgegeben.

exit[*Ausdruck*]
Rücksprung zur Systemebene. *Ausdruck* wird als Prozeßstatus zurückgeliefert.

## bs

execute Zum sofortigen Ausführungsmodus zurückkehren (ein Interrupt hat die gleiche Wirkung). Diese Anweisung bewirkt nicht, daß gespeicherte Anweisungen ausgeführt werden (siehe run unten).

for *Name* = *Ausdruck Ausdruck Anweisung*
for *Name* = *Ausdruck Ausdruck*
...
next

for *Ausdruck*, *Ausdruck*, *Ausdruck Anweisung*
for *Ausdruck*, *Ausdruck*, *Ausdruck*
...
next Die for-Anweisung führt eine Anweisung (erste Form) oder eine Gruppe von Anweisungen (zweite Form) wiederholt aus, wobei der Vorgang von einer benannten Variablen gesteuert wird. Die Variable nimmt den Wert des ersten Ausdrucks an, wird dann bei jedem Schleifendurchlauf so lange um eins erhöht, bis der Wert des zweiten Ausdrucks überschritten wird. Bei der dritten und vierten Form sind drei durch Kommata getrennte Ausdrücke erforderlich. Der erste Ausdruck ist der Initialisierungswert, der zweite ist die Abfrage (bei wahr wird fortgesetzt), und der dritte ist die Aktion bei Schleifenfortsetzung (normalerweise eine Erhöhung).

fun f([a,...])[v,...]
...
nuf Fun definiert den Funktionsnamen, die Argumente und lokalen Variablen für eine benutzereigene Funktion. Bis zu zehn Argumente und lokale Variablen sind zulässig. Bei den Namen darf es sich nicht um Felder (Arrays) handeln, und sie dürfen nicht mit E/A-Vorgängen verknüpft sein. Funktionsdefinitionen dürfen nicht geschachtelt werden.

freturn Eine Möglichkeit, den negativen Ausgang einer benutzereigenen Funktion zu signalisieren. Siehe auch den Abfrageoperator (?) weiter unten. Ist keine Abfrage vorhanden, so liefert freturn lediglich 0 zurück. Ist eine Abfrage aktiv, verzweigt freturn zu diesem Ausdruck (möglicherweise unter Umgehung dazwischenliegender Funktionsrücksprünge).

## bs

goto*Name* Verzweigung zu der intern gespeicherten Anweisung mit der entsprechenden Marke.

ibase*N* Ibase setzt die Basis (Zahlensystem-Grundzahl) der Eingabe auf *N*. *N* muß einen der Werte 8, 10 (Standard) oder 16 haben. Die Hexadezimalwerte 10-15 werden als a-f eingegeben. Die erste Stelle muß eine Ziffer sein (d. h. f0a muß als 0f0a eingegeben werden). Ibase (und obase, siehe unten) werden sofort ausgeführt.

if *Ausdruck Anweisung*
if *Ausdruck*
...
[else
...]
fi

Die Anweisung (erste Form) bzw. Gruppe von Anweisungen (zweite Form) wird ausgeführt, wenn das Ergebnis des Ausdrucks ungleich Null ist. Die Zeichenfolgen 0 und "" (NULL) haben den Wert Null. In der zweiten Form gibt das optionale else die Möglichkeit, eine Gruppe von Anweisungen auszuführen, wenn die erste Gruppe nicht ausgeführt wird. In einer Zeile mit else darf keine andere Anweisung als if stehen; in einer Zeile mit fi dürfen nur weitere fi's stehen. Die Verkürzung von else und if zu einem elif ist möglich. Zum Abschließen einer Sequenz mit if...elif...[else...] ist nur ein einziges fi erforderlich.

include*Ausdruck*

Der Ausdruck muß einen Dateinamen ergeben. Die Datei muß bs-Quellanweisungen enthalten. Diese Anweisungen werden Bestandteil des zu kompilierenden Programms. Include-Anweisungen dürfen nicht geschachtelt werden.

obase*N* Obase setzt die Basis (Zahlensystem-Grundzahl) für die Ausgabe auf *N* (siehe ibase oben).

## bs

onintr*Marke*

onintr Das Kommando onintr ermöglicht die programmgesteuerte Fortsetzung nach Unterbrechungen (Interrupts). In der ersten Form wird zu der angegebenen Marke verzweigt, so als ob statt des onintr ein goto ausgeführt worden wäre. Die Wirkung der Anweisung wird nach jeder Unterbrechung gelöscht. In der zweiten Form bewirkt onintr, daß bs bei einer Unterbrechung abbricht.

return[*Ausdruck*]

Der Ausdruck wird ausgewertet und das Ergebnis als Wert eines Funktionsaufrufes zurückgegeben. Haben Sie keinen Ausdruck angegeben, wird Null zurückgeliefert.

run Der Zufallszahlengenerator wird zurückgesetzt. Es wird zur ersten internen Anweisung verzweigt. Wenn die run-Anweisung in einer Datei enthalten ist, sollte sie die letzte Anweisung sein.

stop Die Ausführung interner Anweisungen wird gestoppt. Bs kehrt zum sofortigen Ausführungsmodus zurück.

trace[*Ausdruck*]

Die trace-Anweisung steuert die Überwachung von Funktionen. Ist der Ausdruck leer (oder ergibt den Wert Null), wird die Überwachung ausgeschaltet. Anderenfalls wird ein Protokoll der Aufrufe/Rücksprünge von Benutzerfunktionen ausgegeben. Jeder Rücksprung vermindert den Wert des Ausdrucks bei trace.

while *Ausdruck Anweisung*
while *Ausdruck*
...
next

While hat eine ähnliche Wirkung wie for, allerdings ist hier nur der bedingte Ausdruck für die Schleifenfortsetzung gegeben.

!*Shell-Kommando*

Sofortiger Sprung in die Shell.

#... Diese Anweisung wird ignoriert. Sie können auf diese Weise einen Kommentar in ein Programm einfügen.

## bs

Syntax der Ausdrücke

*Name* — *Name* dient zur Spezifikation einer Variablen. Die Namen bestehen aus einem Buchstaben (Groß- oder Kleinbuchstabe), an den sich wahlweise weitere Buchstaben und Ziffern anschließen können. Nur die ersten sechs Zeichen eines Namens werden berücksichtigt. Mit Ausnahme der in fun-Anweisungen deklarierten Namen haben alle Namen globale Bedeutung für das Programm. Namen können numerische (double float) Werte oder String-Werte annehmen oder einer Ein-/Ausgabe zugeordnet sein (siehe die Built-in-Funktion open() weiter unten).

*Name*([*Ausdruck* [,*Ausdruck*] ...])
Funktionen können mit einem Namen aufgerufen werden, an den sich die weiteren Argumente, in Klammern eingeschlossen und durch Kommata getrennt, anschließen. Außer bei Built-in-Funktionen (unten aufgeführt) muß der Name mit einer fun-Anweisung definiert werden. Argumente für Funktionen werden mit ihrem Wert übergeben.

*Name*[*Ausdruck* [,*Ausdruck*] ...]
Mit dieser Syntax werden Felder (Arrays) oder Tabellen angesprochen (siehe eingebaute Tabellenfunktionen unten). Bei Feldern (Arrays) wird jeder Ausdruck zu einer Ganzzahl verkürzt und als Hinweis auf den Namen verwendet. Die so entstehende Bezugnahme auf ein Feld (Array) ist syntaktisch mit einem Namen identisch; a[1,2] ist gleichbedeutend mit a[1] [2]. Die verkürzten Ausdrücke können Werte im Bereich von 0 bis 32767 haben.

*Zahl* — Mit einer Zahl wird ein konstanter Wert dargestellt. Eine Zahl wird im Fortran-Format geschrieben und enthält Ziffern, einen wahlfreien Dezimalpunkt und möglicherweise einen Skalierfaktor, der aus einem e und einem gegebenenfalls mit Vorzeichen dargestellten Exponenten besteht.

## bs

*Zeichenkette*
Zeichenketten werden in doppelte Hochkommata eingeschlossen. Durch Voranstellung des Escape-Zeichens \ können Anführungszeichen (\"), Neue-Zeile (\n), CR (\r), Backspace (\b) und Tab (\t) in einer Zeichenkette auftreten. Ansonsten steht \ für sich selbst.

(*Ausdruck*) Durch runde Klammern wird die normale Auswertungsreihenfolge geändert.

(*Ausdruck*, *Ausdruck* [,*Ausdruck* ...]) [*Ausdruck*]
Der in eckigen Klammern eingeschlossene Ausdruck dient als Index, der einen durch Kommata abgetrennten Ausdruck aus der in runden Klammern eingeschlossenen Liste auswählt. Die Listenelemente werden von links nach rechts, beginnend mit Null, numeriert. Der Ausdruck

(False, True)[a == b]

hat den Wert wahr, wenn der Vergleich wahr ist.

?*Ausdruck* Der Abfrageoperator fragt den Erfolg eines Ausdrucks und nicht seinen Wert ab. Derzeit kann der Operator verwendet werden, um das Dateiende (siehe den Abschnitt mit Programmierhinweisen weiter unten), das Ergebnis der Built-in-Funktion eval und den Rücksprung aus benutzereigenen Funktionen (siehe freturn) abzufragen. Wird ein Trap erkannt (Dateiende usw.), so verzweigt das Programm sofort zur letzten Abfrage, wobei möglicherweise Zuweisungsanweisungen oder dazwischenliegende Funktionsebenen übersprungen werden.

-*Ausdruck* Das Ergebnis ist die Negation von Ausdruck.

++*Name* Erhöht den Wert der Variable (bzw. Array-Referenz). Das Ergebnis ist der neue Wert.

--*Name* Vermindert den Wert der Variable. Das Ergebnis ist der neue Wert.

!*Ausdruck* Die logische Negation von Ausdruck. Vorsicht wegen des Escape-Kommandos der Shell.

## bs

*Ausdruck Operator Ausdruck*

Gemeinsame Funktionen von zwei Argumenten werden abgekürzt, indem die beiden Argumente durch einen Operator getrennt werden, der die Funktion bezeichnet. Außer bei den Operatoren für Zuweisung, Verkettung und Vergleich werden beide Operanden in numerisches Format konvertiert, bevor die Funktion ausgeführt wird.

**Binäre Operatoren** (in der Reihenfolge zunehmender Priorität)

`=` Zuweisungsoperator. Der linke Operand muß ein Name oder ein Array-Element sein. Das Ergebnis ist der rechte Operand. Die Zuweisung bindet von rechts nach links, während alle anderen Operatoren von links nach rechts binden.

`_` Verkettungsoperator.

`& |` & (logisch Und) liefert das Ergebnis Null, wenn eins der Argumente Null ist. Der Operator liefert das Ergebnis Eins, wenn beide Argumente ungleich Null sind; | (logisch Oder) liefert das Ergebnis Null, wenn beide Argumente Null sind. Der Operator liefert das Ergebnis Eins, wenn eins der Argumente ungleich Null ist. Beide Operatoren behandeln einen leeren String als Null.

`< <= > >= == !=`
Die Vergleichsoperatoren (< kleiner, <= kleiner oder gleich, > größer, >= größer oder gleich, == gleich, != ungleich) liefern das Ergebnis Eins, wenn zwischen ihren Argumenten die angegebene Relation besteht. Anderenfalls liefern sie das Ergebnis Null. Vergleichsoperatoren auf derselben Stufe greifen wie folgt durch: a>b>c ist gleichbedeutend mit a>b&b>c. Ein String-Vergleich wird durchgeführt, wenn beide Operanden Strings sind.

`+ -` Addition und Subtraktion.

`* / %` Multiplikation, Division und Rest (modulo).

`^` Potenzierung.

## bs

### Built-in-Funktionen

1. Behandlung von Argumenten

arg($i$) ist der Wert des $i$-ten Aktualparameters auf der aktuellen Stufe des Funktionsaufrufes. Auf Stufe Null liefert arg das $i$-te Kommandozeilen-Argument zurück (arg(0) liefert bs zurück).

narg() liefert die Anzahl der übergebenen Argumente zurück. Auf Stufe Null wird die Anzahl der Argumente im Kommando zurückgeliefert.

2. Mathematische Funktionen

abs($x$) Absolutwert von $x$.

atan($x$) Arkustangens von $x$. Sein Wert liegt zwischen $-\pi/2$ und $\pi/2$.

ceil($x$) liefert die kleinste Ganzzahl – nicht kleiner als $x$ – zurück.

cos($x$) Kosinus von x (Bogenmaß).

exp($x$) Exponentialfunktion von $x$.

floor($x$) Größte Ganzzahl – nicht größer als x – zurück.

log($x$) Natürlicher Logarithmus von $x$.

rand() Gleichverteilte Zufallszahl zwischen Null und Eins.

sin($x$) Sinus von $x$ (Bogenmaß).

sqrt($x$) Quadratwurzel von $x$.

3. String-Operationen

size($s$) Liefert die Länge von $s$ (in Bytes) zurück.

format($f$,$a$) Liefert den formatierten Wert von $a$ zurück. $F$ wird als Formatspezifikation in der Art und Weise von printf interpretiert. Die einzigen sicheren Typen sind %...f, %...e und %...s.

index($x$,$y$) Liefert die Nummer der ersten Position in $x$ zurück, die mit irgendeinem Zeichen aus $y$ übereinstimmt. Bei fehlender Übereinstimmung wird Null zurückgeliefert.

trans(*s*,*f*,*t*) Übersetzt Zeichen aus der Quelldatei *s* von übereinstimmenden Zeichen aus *f* in ein Zeichen auf derselben Position in *t*. Quelldateizeichen, die nicht in *f* auftreten, werden in das Ergebnis kopiert. Ist der String *f* länger als *t*, erscheinen Quelldateizeichen, für die es im überhängenden Teil von *f* eine Übereinstimmung gibt, nicht im Ergebnis.

subst(*s*,*Anfang*,*Breite*)
Liefert den Sub-String von *s* zurück, der durch *Anfang* und *Breite* definiert ist.

match(*String*,*Muster*)
mstrng(*n*) Das Muster gleicht in der Syntax dem regulären Ausdruck des ed-Kommandos. Die Zeichen ., [, ], ^ (innerhalb eckiger Klammern), * und $ haben Spezialbedeutung. Die Funktion mstring liefert den *n*-ten (1 <= *n* <= 10) Sub-String des Subjekts zurück, der bei dem letzten Aufruf, bei dem eine Übereinstimmung festgestellt wurde, zwischen Paaren der Mustersymbole \( und \) aufgetreten ist. Eine Übereinstimmung wird nur dann erkannt, wenn das Muster mit dem Anfang des Strings übereinstimmt (so als ob alle Muster mit ^ beginnen würden). Die Funktion liefert die Anzahl der übereinstimmenden Zeichen zurück.

Beispiel:

```
match("a123ab123", ".*\([a-z]\)") == 6
mstring(1) == "b"
```

4. Dateibehandlung

open(*Name*,*Datei*,*Funktion*)
close(*Name*)
Das Argument *Name* muß ein bs-Variablenname sein (der als String übergeben wird). Beim Open können Sie das Dateiargument folgendermaßen angeben:

1. eine 0 (Null), 1 oder 2 für Standardeingabe, Standardausgabe bzw. Standardfehlerausgabe,
2. einen String, der einen Dateinamen darstellt, oder
3. einen String, der mit einem ! beginnt und ein auszuführendes Kommando darstellt (über sh -c).

Das Funktionsargument muß r (read – Lesen), w (write – Schreiben), W (Schreiben ohne Neue-Zeile) oder a (append – Anfügen) sein. Nach einem Close wird *Name* wieder eine gewöhnliche Variable. Die anfänglichen Kombinationen sind folgende:

```
open("get",0,"r")
open("put",1,"w")
open("puterr",2,"w")
```

access(s,m) Führt den Systemaufruf access aus.

ftype(s) Liefert den Dateityp in Form eines einzelnen Zeichens zurück: f für Datendatei, p für FIFO (d. h. eine benannte Pipe), d für Verzeichnis, b für blockorientierte Gerätedatei und c für zeichenorientierte Gerätedatei.

5. Tabellen

table(*Name*,*Größe*)
Bei bs ist eine Tabelle ein assoziativ angesprochenes, eindimensionales Array. Indizes (als Schlüssel bezeichnet) sind Strings (Zahlen werden konvertiert). Das Argument *Name* muß ein bs-Variablenname sein (der als String übergeben wird). Das Argument *Größe* gibt die Mindestanzahl der zuzuordnenden Elemente an. Bei einem Tabellenüberlauf gibt bs eine Fehlermeldung aus und stoppt.

item(*Name*,*i*)
key() Mit der Funktion item werden Tabellenelemente sequentiell angesprochen (beim normalen Gebrauch gibt es keine geregelte Folge von Schlüsselwerten). Greift die Funktion item auf Werte zu, spricht die Funktion key den Index des vorangehenden item-Aufrufs an. Das Argument *Name* darf nicht in Anführungszeichen eingeschlossen werden. Da keine genauen Tabellengrößen definiert sind, sollten Sie das Ende der Tabelle mit dem Abfrageoperator abfragen.

Beispiel:

```
table("t", 100)
...
# Wenn ein Wort „party" enthält,
# erhöht der folgende Ausdruck den
# Zählerstand für das Wort um eins:
++t[Wort]
...
# Die Schlüssel/Wert-Paare ausgeben:
for i = 0, ?(s = item(t, i)), ++i if key()
put = key()_":"_s
```

iskey(*Name*,*Wort*)
Die Funktion iskey prüft, ob der *Schlüssel* Wort in der Tabelle *Name* vorhanden ist. Wenn ja, liefert er Eins zurück, anderenfalls Null.

6. Verschiedenes

eval(*s*) Das String-Argument wird als bs-Ausdruck ausgewertet. Diese Funktion bietet sich für das Konvertieren numerischer Strings in die interne numerische Form an. Eval kann auch zur indirekten Adressierung verwendet werden, wie etwa in:

```
Name = "xyz"
eval("++" _Name)
```

zur Erhöhung der Variablen xyz. Außerdem können Sie eval mit dem vorangestellten Abfrageoperator verwenden, um bs-Fehlerbedingungen zu steuern.

Beispiel:

```
?eval("open(\"X\", \"XXX\", \"r\")")
```

liefert den Wert Null zurück, wenn keine Datei mit dem Namen "XXX" existiert. Das folgende Beispiel verzweigt zu der Marke L (falls diese existiert):

```
label="L"
if !(?eval("goto"_label)) puterr = "no label"
```

last() Im sofortigen Ausführungsmodus liefert last den zuletzt errechneten Wert zurück.

## bs

### Programmierhinweise

Bs als Rechner:

```
$ bs
# Entfernung (in Zoll), die das Licht in einer Nanosekunde zurücklegt.
186000 * 5280 * 12 / 1e9
11.78496
...

# Zinseszins (6% für 5 Jahre auf DM 1.000).
int = .06 / 4
bal = 1000
for i = 1 5*4 bal = bal + bal*int
bal - 1000
346.855007
...
exit
```

Aufbau eines typischen bs-Programms:

```
# Initialisierungen:
var1 = 1
open("read", "infile", "r")
...
# Rechnen:
while ?(str = read)
...
next
# Beenden:
close("read")
...
# Letzte ausgeführte Anweisung (exit oder stop):
exit
# Letzte Eingabezeile:
run
```

## bs

Beispiele für Eingabe/Ausgabe:

```
# "Datei1" in "Datei2" kopieren.
open("read", "Datei1", "r")
open("write", "Datei2", "w")
...
while ?(write = read)
...
# "read" und "write" schließen:
close("read")
close("write")

# Pipe zwischen Kommandos:
open("ls", "!ls*", "r")
open("pr", "!pr -2 -h 'List'", "w")
while ?(pr = ls) ...
...
# Unbedingt schließen (warten):
close("ls")
close("pr")
```

SIEHE AUCH:

ed, sh

C

## cal

### cal – Kalenderanzeige

SYNTAX:

cal [*Monat*] *Jahr*

BESCHREIBUNG:

Cal zeigt Ihnen den Kalender für das angegebene Jahr an. Geben Sie zusätzlich einen Monat an, wird nur der Kalender für diesen Monat ausgegeben.
Die Jahreszahl kann zwischen 1 und 9999 angegeben werden, Monatszahlen zwischen 1 und 12. Ausgegeben wird der Kalender für England und seine Kolonien.

HINWEIS:

Der Aufruf cal 85 bezieht sich auf das Jahr 85 n. Chr. und nicht auf 1985.

C

## calendar

calendar – Erinnerungs-Service

SYNTAX:

calendar [-]

BESCHREIBUNG:

Calendar bearbeitet die Datei calendar in Ihrem aktuellen Verzeichnis und zeigt die Zeilen an, die das heutige oder morgige Datum enthalten. Jedes Datum in amerikanischer Darstellung wird akzeptiert, z. B. 'Dec. 7,', 'december 7,', '12/7' etc.; jedoch nicht '7 December' oder '7/12'.

Wird das Argument '-' angegeben, bearbeitet der Befehl calendar für jeden Benutzer, der in seinem Login-Verzeichnis die Datei calendar angelegt hat, diese Datei und sendet ihm – bei positivem Ergebnis – die entsprechenden Informationen durch die mail-Funktion. Normalerweise geschieht dies täglich beim Systemstart.

DATEIEN:

/usr/lib/calprog

/etc/passwd

/tmp/cal/*

SIEHE AUCH:

mail

C

## cancel

cancel – Druckauftrag an Drucker-Spooler löschen

SYNTAX:

cancel [*Auftragsnummer* ...] [*Drucker* ...]

BESCHREIBUNG:

Cancel löscht Druckaufträge, die mit lp erzeugt wurden. Argumente der Kommandozeile sind entweder Auftragsnummern wie sie von lp zurückgemeldet werden oder Druckernamen. (Eine vollständige Liste von Druckernamen erhalten Sie mit lpstat.)

Geben Sie die Auftragsnummer an, wird dieser Auftrag gelöscht, auch wenn er gerade gedruckt wird.

Wird ein Drucker spezifiziert, wird der Auftrag gelöscht, der gerade auf diesem Drucker in Bearbeitung ist.

In jedem Fall wird ein Drucker frei zur Bearbeitung des nächsten Auftrags, wenn ein gerade in Bearbeitung befindlicher Auftrag gelöscht wird.

DATEIEN:

/usr/spool/lp/*

SIEHE AUCH:

enable, lp, lpstat

C

## cat

### cat – Verketten und Anzeigen

SYNTAX:

cat [-u] [-s] [*Datei* ...]

BESCHREIBUNG:

Cat liest die Dateien in der angegebenen Reihenfolge und schreibt sie in die Standardausgabe.

Der Befehl

cat *Datei1 Datei2* > *Datei3*

verkettet also die Inhalte von *Datei1* und *Datei2* und stellt das Ergebnis in *Datei3* ab.

Bitte beachten Sie, daß bei einer Umleitung der Standardausgabe zuerst die Ausgabedatei angelegt wird; anschließend werden die angegebenen Dateien dorthin geschrieben. Wenn Sie also eine Eingabedatei gleichzeitig als Ausgabedatei benennen, wird die Eingabedatei zuerst gelöscht.

Ist keine Eingabedatei angegeben oder wenn Sie das Argument '-' benutzen, liest cat aus der Standardeingabe.

Optionen:

-u Ausgabe wird nicht gepuffert.

-s Die Anzeige über nicht existierende Dateien wird unterdrückt.

SIEHE AUCH:

cp, pr

C

## cb

cb – Formatieren von C-Programmen

SYNTAX:

cb [-s] [-j] [-l*Länge*] [*Datei* ...]

BESCHREIBUNG:

Cb liest C-Quellcode aus der Standardeingabe oder aus den angegebenen Dateien und gibt sie nach Standardausgabe mit den Zwischenräumen und Einrückungen aus, die die Struktur des C-Codes widerspiegeln. Ohne Optionen verändert cb den vom Benutzer angegebenen Zeilenumbruch nicht.

Optionen:

-s Ausgabe in dem von Kerningham und Ritchie vorgegebenen Stil (The C Programming Language).

-j Getrennte Zeilen werden zusammengefügt.

-l*Länge* Zeilen, die die angegebene Länge überschreiten, werden getrennt.

SIEHE AUCH:

cc
Systemliteratur TARGON /35: „Die Sprache C"

C

## cc

### cc – C-Compiler

SYNTAX:

cc [*Optionen*] *Datei* ...

BESCHREIBUNG:

Cc ist der C-Compiler der TARGON /35. Er akzeptiert mehrere Argumententypen.

Dateien, deren Namen auf .c enden, werden als C-Quellprogramme angesehen. Sie werden kompiliert und der Objektcode wird in Dateien mit dem gleichen Namen und der Endung .o gespeichert. Besteht das C-Programm nur aus einer Quelldatei, wird die .o-Datei gelöscht, wenn das Programm kompiliert und gelinkt ist.
Dateien deren Namen auf .s enden, werden als Assembler-Quellprogramme erkannt und assembliert. Die Ausgabe wird – analog der Ausgabe von C-Kompilierungsläufen – in Dateien mit der Endung .o gespeichert.

Optionen:

-c Unterdrücken der Link-Phase nach der Kompilierung. Eine Objektdatei mit der Endung .o wird angelegt, auch wenn nur eine Datei kompiliert wird.

-p Die einzelnen Funktionsaufrufe werden gezählt. Außerdem wird nach dem Linken die Standard-Startfunktion durch eine Funktion ersetzt, die automatisch ein Monitoring aufruft. Ein Ausführungsprofil des Objektprogramms kann dann mit Hilfe von prof erstellt werden.

-g Generierung von zusätzlichen Informationen für sdb.

-w Warnmeldungen werden unterdrückt.

-O Aufruf des Objektcode-Optimierers.

-S Kompiliert die angesprochenen C-Programme und erzeugt dabei Assembler-Quellcode in Dateien, deren Namen auf .s enden.

cc

-E Es wird nur der Preprozessor für die angegebenen Dateien aufgerufen. Das Ergebnis wird in die Standardausgabe geschrieben.

-C Kommentare werden vom Preprozessor nicht entfernt.

-o*Name* Das ausführbare Programm erhält den angegebenen Namen statt a.out.

-D*Name*[=*Wert*]
Für den Preprozessor wird auf die gleiche Weise wie bei #define ein Name definiert. Geben Sie *Wert* nicht an, wird standardmäßig 1 eingesetzt.

-U*Name* Die #define-Anweisung für *Name* wird aufgehoben.

-I*Verzeichnis*
Ändert den Suchalgorithmus für #include-Dateien, deren Namen nicht mit einem Schrägstrich (/) beginnen. Zuerst wird im Verzeichnis der bei cc angegebenen Dateien gesucht, anschließend in dem in der -I-Option angegebenen Verzeichnis und zuletzt in den Standard-Verzeichnissen.

-B*Zeichenkette*
Suchen von Ersatz-Compilerschritten in den Dateien, deren Namen aus *Zeichenkette* und den Endungen cpp, c0 (oder ccom), c1, c2, as und ld bestehen. Wird keine Datei angegeben, wird eine Standard-Backupversion benutzt.

-t[pO12al] Nur die angegebenen Compilerschritte werden in den Dateien, deren Namen von einer -B-Option konstruiert wurden, gesucht. Fehlt die -B-Option, wird als *Zeichenkette* /usr/c angenommen.

-W*c**Arg 1*[,*Arg2*...]
Übergabe der angegebenen *Argumente* an die Compilerphase *c*. *c* können Sie folgendermaßen angeben:

p = Preprozessor
O = Optimierer
1 = erster Compilerschritt
2 = zweiter Compilerschritt
a = Assembler
l = Linker

cc

C

Andere Argumente als .c-Dateien, .s-Dateien und die nachstehend aufgeführten Optionen werden entweder als Link-Optionsargumente, als C-kompatible Objekt-Programme, die aus einem früheren cc-Lauf stammen, oder auch als Bibliotheken von C-kompatiblen Routinen angenommen.

DATEIEN:

| | |
|---|---|
| Datei.c | Eingabedatei |
| Datei.o | Objektdatei |
| a.out | ausführbares Programm |
| /tmp/ctm/? | temporäre Dateien |
| /lib/cpp | C Preprozessor |
| /bin/ccom | Compiler |
| /usr/c/occom | Backup-Compiler |
| /usr/c/ocpp | Backup-Preprozessor |
| /lib/c2 | Optimierer |
| /lib/crt0.o | Laufzeitdatei |
| /lib/mcrt0.o | Profilerdatei |
| /lib/libc.a | C-Standardbibliothek |
| /usr/lib/libc_p.a | Profilversionen der Bibliotheken |
| /usr/include | Standardverzeichnis für #include-Dateien |
| mon.out | Datei für prof |

SIEHE AUCH:

as, ld, prof, sdb
Systemliteratur TARGON /35: „Die Sprache C"

C

## cd

cd – Wechseln des aktuellen Verzeichnisses

SYNTAX:

cd [*Verzeichnis*]

BESCHREIBUNG:

Das angegebene Verzeichnis wird Ihr neues aktuelles Verzeichnis. Voraussetzung ist allerdings, daß Sie eine Ausführungserlaubnis für dieses Verzeichnis haben. Der Verzeichnisname kann als voller oder relativer Pfadname angegeben werden. Mit cd .. gelangen Sie in das nächsthöhere Verzeichnis.

Cd ohne Argument bringt Sie aus einem beliebigen Verzeichnis in das Verzeichnis, welches in der Shell-Variablen $HOME (Home-Verzeichnis) eingetragen ist. Normalerweise ist dies Ihr Login-Verzeichnis.

SIEHE AUCH:

pwd, sh

C

## cdc

### cdc – Ändern des Delta-Kommentars eines SCCS-Delta

SYNTAX:

cdc -r*SID* [-m[*MR-Liste*]] [-y[*Kommentar*]] *Datei* ...

BESCHREIBUNG:

Cdc ändert den Delta-Kommentar für die durch -r spezifizierten *SID* jeder angegebenen SCCS-Datei.

Delta-Kommentar ist als Modifikationsanforderung (Modification Request=MR) und Kommentarinformation definiert, die normalerweise über das delta-Kommando angegeben wird (-m und -y-Optionen).

Geben Sie ein Verzeichnis an, hat dies die gleiche Wirkung, als ob jede Datei im Verzeichnis angegeben wäre. Nicht-SCCS-Dateien und nichtlesbare Dateien werden jedoch ignoriert. Ist '-' als Dateiname angegeben, liest cdc die Standardeingabe. Jede Zeile der Standardeingabe wird als Name einer zu verarbeitenden SCCS-Datei interpretiert.

Die Argumente können in beliebiger Reihenfolge erscheinen und bestehen aus Optionen und Dateinamen.

Alle beschriebenen Optionsargumente werden separat auf jede angeführte Datei angewendet.

Optionen:

-r*SID* Wird benutzt, um die SCCS-Identifikationszeichenkette (SID) für ein Delta anzugeben, bei dem der Delta-Kommentar geändert werden soll.

-m[*MR-Liste*] Ist bei der SCCS-Datei die -v-Option gesetzt, können Sie eine Liste von MR-Nummern angeben, die im Delta-Kommentar des durch die -r-Option spezifizierten *SID* geliefert werden. Eine leere MR-Liste hat keine Wirkung.

## cdc

MR-Einträge werden der MR-Liste auf die gleiche Weise hinzugefügt, wie beim delta-Befehl. Um ein MR zu löschen, muß der MR-Nummer ein Ausrufungszeichen vorangestellt sein (s. Beispiele). Ist der zu löschende MR momentan in der MR-Liste enthalten, wird er entfernt und in eine Kommentarzeile verwandelt. Eine Liste aller gelöschten MRs wird in den Kommentarbereich der Delta-Dokumentation gestellt und mit einer Kommentarzeile versehen, die angibt, daß sie gelöscht wurden.

Ist -m nicht angegeben und die Standardeingabe ist das Terminal, wird der Text 'MRs?' auf dem Bildschirm ausgegeben, bevor die Eingabe gelesen wird. Ist das Eingabemedium nicht die Standardeingabe, wird kein Text angezeigt.
Der Text 'MRs?' geht immer der Ausgabe 'comments?' voraus (siehe -y-Option).

MRs in einer Liste werden durch Blanks und/oder Tabulatorzeichen getrennt. Die Liste wird durch das Zeilenende abgeschlossen.

Ist das 'v'-Flag gesetzt (s. admin), wird dieses als Name eines Programms (oder einer Shell-Prozedur) genommen, das die Korrektheit der MR-Nummern überprüft. Liefert dieses Programm einen Rückgabe-Code ungleich 0, wird cdc beendet und der Delta-Kommentar bleibt unverändert.

-y[*Kommentar*] Ein beliebiger Text wird benutzt, um die Kommentare zu ersetzen, die schon für das mit der -r-Option spezifizierte Delta existieren. Die vorherigen Kommentare werden beibehalten und von einer Kommentarzeile angeführt, die besagt, daß sie geändert wurden. Ein leerer Kommentar wird ignoriert.

Haben Sie -y nicht angegeben und die Standardeingabe ist angesprochen, wird der Text 'comments?' auf dem Bildschirm ausgegeben, bevor die Eingabe gelesen wird. Der Kommentartext wird durch ein Zeilenende-Zeichen beendet.

C

## cdc

Einfach ausgedrückt können Sie den Delta-Kommentar ändern, wenn Sie

1. das Delta selbst erzeugt haben oder
2. Besitzer der Datei bzw. des Verzeichnisses sind.

Beispiele:

cdc -r1.6 -m"bl78-12345 !bl77-54321 bl79-00001" -yFehler s.*Datei*

fügt bl78-12345 und bl79-00001 in die MR-Liste ein, entfernt bl77-54321 aus der MR-Liste und integriert den Kommentar „Fehler" in Delta 1.6 von s.*Datei*.

cdc -r1.6 s.*Datei*
MRs? !bl77-54321 bl78-12345 bl79-00001
comments? Fehler

hat die gleiche Wirkung.

SIEHE AUCH:

admin, delta, get, help, prs
Systemliteratur TARGON: „Programmentwicklungs-Tools"

C

## cflow

cflow – Kontrollflußdiagramme von C-Programmen

SYNTAX:

cflow [-r] [-ix] [-i_] [-d*Tiefe*] *Datei* ...

BESCHREIBUNG:

Cflow versucht die externen Referenzen der angegebenen Dateien grafisch darzustellen, um so den statischen Kontrollfluß des Programms wiederzugeben. Die Darstellung erfolgt auf Standardausgabe. Cflow verarbeitet Dateien, die auf .y, .l, .c, .i, .a und .o enden.

Jede Ausgabezeile beginnt mit einer Zeilennummer, gefolgt von dem Variablen- oder Funktionsnamen der externen Referenz (Standard: nur Funktionen). Je nach Schachtelungstiefe der Referenz wird der Name um entsprechende Tabulatorpositionen eingerückt. An den Namen schließt sich ein Doppelpunkt und der Typ an (z. B. char *() = Funktion, die einen 'char'-Pointer liefert) und als letztes in spitzen Klammern die Quelldatei und die Zeile, in der sich die Definition dieses Elements befindet.

Bei Definitionen, die in gelinkten Objektdateien vorkommen, werden statt der Zeilennummern die Adressen (location counter) in der Objektdatei angegeben.

Bei mehrmaligem Auftreten der gleichen Referenz wird im folgenden in den spitzen Klammern nur die Zeilennummer des ersten Auftretens angegeben. Ist eine Referenz undefiniert, bleibt die spitze Klammer leer.

## cflow

Beispiel: Datei test.c

```
main()
{
        f();
        g();
        f();
}

f()
{
        i = h();
}
```

Das Kommando cflow test.c erzeugt folgende Ausgabe:

```
1       main: int(), <test.c 1>
2               f: int(), <test.c 8>
3                       h: < >
4               g: < >
5               f: <2>
```

Optionen:

-r Die Beziehung aufrufende/aufgerufene Funktion wird umgekehrt, d. h. es wird eine sortierte Liste der Aufrufer jeder Funktion erzeugt.

-ix Nicht nur Funktionen, sondern auch externe und statische Daten werden in die Darstellung mit aufgenommen.

-i_ Auch Funktionen, deren Name mit _ beginnt (i. a. Systemfunktionen), werden mit aufgenommen.

-d*Tiefe* Angabe einer maximalen Schachtelungstiefe für Referenzen, die ausgegeben werden (Standard: sehr groß).

SIEHE AUCH:

as, cc, cpp, lex, lint, nm, pr, yacc
Systemliteratur TARGON /35: „Die Sprache C"

C

## chmod

chmod – Ändern der Zugriffsrechte

SYNTAX:

chmod *Modus Datei* ...

BESCHREIBUNG:

Chmod ändert die Zugriffsrechte der angegebenen Dateien. Sie können den Modus absolut oder symbolisch angegeben.

Die Angabe absoluter Modi geschieht mit Hilfe von Oktalzahlen und zwar folgendermaßen:

4000 Set-User-ID-Bit wird auf Ausführung gesetzt.

2000 Set-Group-ID-Bit wird auf Ausführung gesetzt.

1000 Das Textsegment-Bit wird gesetzt.

0400 Leseerlaubnis für den Eigentümer.

0200 Schreiberlaubnis für den Eigentümer.

0100 Ausführungserlaubnis (Sucherlaubnis im Verzeichnis) für den Eigentümer.

0070 Lese-, Schreib- und Ausführungserlaubnis (gleichzeitig Sucherlaubnis) für die Gruppe des Eigentümers.

0007 Lese-, Schreib- und Ausführungserlaubnis (gleichzeitig Sucherlaubnis) für den „Rest der Welt".

Die symbolische Angabe hat folgende Form:

[*wer*] *Symbol Erlaubnis*, ...

*wer* ist eine Kombination der Buchstaben 'u' (user = eigene Erlaubnis), 'g' (Gruppe) und 'o' (others = alle anderen). 'a' steht für 'ugo'. Wird *wer* nicht angegeben, ist der Default-Wert 'a'. Sie können mehrere symbolische Modi, getrennt durch Kommata, spezifizieren.

## chmod

Symbol-Codes sind:

- \+ Hinzufügen einer Erlaubnis
- \- Löschen einer Erlaubnis
- = Zuordnung einer absoluten Erlaubnis

*Erlaubnis* ist jede Kombination der Buchstaben r (read = lesen), w (write = schreiben), x (execute = ausführen), t (Code bleibt im Swap-Bereich und s (bei Ausführung wird Gruppen- oder Benutzernummer des Eigentümers benutzt). *Erlaubnis* nicht anzugeben ist nur sinnvoll, wenn '=' zum Entziehen jeglicher Rechte verwendet wird.

Beispiel 1:

chmod o-w *Datei*

Erlaubnis zum Schreiben hat nur noch der Eigentümer der Datei sowie die Angehörigen seiner Gruppe. Allen anderen Benutzern ist dieses Recht verwehrt.

Beispiel 2:

chmod +x *Datei*

Erlaubnis zum Ausführen von *Datei* ist allen Benutzern erteilt.

Mehrere symbolische Modi, getrennt durch Kommata, können angegeben werden.

Nur der Eigentümer einer Datei oder der Superuser kann die zugehörigen Zugriffsrechte ändern und nur der Superuser sollte das Textsegment-Bit setzen.

SIEHE AUCH:

ls
Systemliteratur TARGON: „Systemschnittstellen und Programmierung"

C

## chown, chgrp

chown, chgrp – Ändern der Eigentümer oder der Gruppe

SYNTAX:

chown *Eigentümer Datei...*

chgrp *Gruppe Datei ...*

BESCHREIBUNG:

**Chown** ändert den ursprünglichen Eigentümer der angegebenen Datei in *Eigentümer*. Die Angabe des neuen Eigentümers kann sowohl als dezimale Benutzer-ID erfolgen, als auch der Login-Name – wie in /etc/passwd eingetragen – sein.

Nur der Superuser oder der Eigentümer der betroffenen Dateien darf diesen Befehl aufrufen.

**Chgrp** ändert die Gruppen-ID von *Datei* in die Gruppen-ID von *Gruppe*. *Gruppe* kann sowohl eine dezimale Gruppen-ID als auch ein Gruppenname sein, der in der Datei /etc/group definiert ist.

DATEIEN:

/etc/passwd

/etc/group

SIEHE AUCH:

chmod

C

## cmp

cmp – Vergleichen zweier Dateien

SYNTAX:

cmp [-l] [-s] *Datei1 Datei2*

BESCHREIBUNG:

*Datei1* und *Datei2* werden miteinander verglichen. Wird anstelle *Datei1* ein '-' aufgerufen, wird die Standardeingabe gelesen. Dem cmp-Befehl folgt keine Ausgabe, wenn der Inhalt der verglichenen Dateien identisch ist. Sind Abweichungen vorhanden, werden die entsprechende Byte- und Zeilennummer angezeigt, in denen die Abweichungen beginnen.
Ist eine Datei der erste Teil der anderen, wird dies ebenfalls angezeigt.

Optionen:

-l Anzeigen der Bytenummer (dezimal) und der differierenden Bytes (oktal) für jede Abweichung.

-s Keine Anzeige von Abweichungen. Es werden nur Codes zurückgegeben:

0 = Die angegebenen Dateien sind identisch.

1 = Die angegebenen Dateien differieren.

2 = Fehlerhafte Eingabe.

SIEHE AUCH:

comm, diff

C

## col

col – Herausfiltern von Zeilenvorschüben

SYNTAX:

col [-bfpx]

BESCHREIBUNG:

Col liest aus der Standardeingabe und schreibt in die Standardausgabe. Es filtert die von umgekehrten Zeilenvorschüben (ASCII-Code ESC-7) und halbzeiligen Vorschüben in Vorwärts- und Rückwärtsrichtung (ESC-9 bzw. ESC-8) angeforderten Zeilenüberlagerungen aus. Col ist besonders nützlich zum Filtern mehrspaltiger Ausgabedaten, die vom nroff-Kommando .rt erstellt wurden, sowie von Ausgabedaten, die vom Preprozessor tbl erzeugt werden.

Ist die Option -b angegeben, geht col davon aus, daß das verwendete Ausgabegerät keinen Backspace ausführen kann. Sollen in diesem Fall zwei oder mehr Zeichen an derselben Stelle erscheinen, wird nur das letzte Zeichen gelesen und ausgegeben.

Col übernimmt halbzeilige Vorschübe in der Eingabe, gibt diese aber normalerweise nicht an die Ausgabe weiter. Stattdessen wird der Text, der eigentlich zwischen den Zeilen stehen würde, in die nächsttiefere volle Zeile gestellt. Diese Verfahrensweise können Sie mit der Option -f außer Kraft setzen; jetzt kann die Ausgabe von col halbzeilige Zeilentransporte in Vorwärtsrichtung (ESC-9) enthalten, während ein Zeilentransport in Rückwärtsrichtung nach wie vor unterdrückt wird.

Ohne Angabe der Option -x konvertiert col – überall wo dies möglich ist – zeichenfreie Stellen in Tabs, um die Druckzeit zu verkürzen.

Die ASCII-Steuerzeichen SO (\016) und SI (\017) versteht col so, daß sie einen Text in einem alternativen Zeichensatz einleiten bzw. beenden. Col merkt sich den Zeichensatz, zu dem die einzelnen Eingabezeichen gehören, und bei der Ausgabe werden die Zeichen SI und SO in der entsprechenden Form generiert, so daß jedes Zeichen im richtigen Zeichensatz ausgegeben wird.

## col

In der Eingabe sind nur die Steuerzeichen Space, Backspace, Tab, Return, Newline, SI, SO, VT (\013) und ESC mit 7, 8 oder 9 zulässig. Das Zeichen VT ist eine andere Form eines ganzen umgekehrten Zeilenvorschubs, das aus Gründen der Kompatibilität mit einigen älteren Programmen dieser Art vorgesehen ist. Alle anderen nicht-druckbaren Zeichen werden ignoriert.

Normalerweise ignoriert col alle in der Eingabe gefundenen Escape-Folgen, die es nicht kennt; mit der Option -p kann man erreichen, daß col diese Sequenzen als reguläre Zeichen ausgibt, die dann auch bei umgekehrten Zeilenvorschüben überschrieben werden können. Es wird dringend davon abgeraten, diese Option zu verwenden, es sei denn, Sie sind sich voll und ganz im klaren über die Position der Escape-Sequenzen im Text.

Das von col akzeptierte Eingabeformat entspricht der von nroff mit den Optionen -T37 oder -Tlp erstellten Ausgabe. Die Option -T37 (und die Option -f von col) sollten verwendet werden, wenn die Ausgabedaten von col auf ein Gerät geschrieben werden, das halbzeilige Vorschübe vornehmen kann. Anderenfalls benutzen Sie die Option -Tlp.

HINWEISE:

Ein Rückwärtsgehen um mehr als 128 Zeilen ist nicht möglich.
Eine Zeile kann maximal 800 Zeichen einschließlich der Rückschritte enthalten.
Lokale vertikale Bewegungen, die zu einer Position vor der ersten Zeile des Dokuments führen würden, werden ignoriert. Daher dürfen in der ersten Zeile keine hochgestellten Zeichen enthalten sein.

SIEHE AUCH:

nroff, tbl
Systemliteratur TARGON /35: „Textformatierer nroff"

C

## comb

comb – Zusammenfassung von SCCS-Deltas

SYNTAX:

comb [-o] [-s] [-p*SID*] [-c*Liste*] *Datei* ...

BESCHREIBUNG:

Comb erzeugt eine Shell-Prozedur, die in der Lage ist, die angegebenen SCCS-Dateien wiederherzustellen. Die rekonstruierten Dateien werden i. a. kleiner als die Originaldateien.
Die Argumente können in beliebiger Reihenfolge angegeben werden, aber alle Optionen werden auf **allen** angegebenen SCCS-Dateien angewendet. Geben Sie ein Verzeichnis an, tut comb so, als ob jede Datei im Verzeichnis namentlich genannt wäre. Nicht-SCCS-Dateien (letzte Komponente des Pfadnamens beginnt nicht mit s.) sowie nicht-lesbare Dateien werden jedoch ignoriert. Geben Sie '-' statt eines Dateinamens an, wird aus der Standardeingabe gelesen; jede Zeile der Standardeingabe wird als Name einer zu verarbeitenden SCCS-Datei betrachtet.

Die erzeugte Shell-Prozedur wird in die Standardausgabe geschrieben.

Die folgenden Optionen werden so beschrieben, als ob nur **eine** aufgeführte Datei bearbeitet werden soll; die Wirkung jedes Optionsarguments betrifft jedoch jede einzelne Datei.

-p*SID* Die SCCS-Identifikationszeichenkette des ältesten Deltas, das aufbewahrt werden soll. Alle noch älteren Dateien werden in der rekonstruierten Datei gelöscht.

-c*Liste* Eine Liste von Deltas, die erhalten bleiben sollen (s. get bzgl. der Syntax einer Liste). Alle anderen Deltas werden gelöscht.

-o Für jedes erzeugte get -e bewirkt diese Option, daß auf die rekonstruierte Datei in der Version des zu erzeugenden Deltas zugegriffen wird. Sonst würde auf die rekonstruierte Datei in der Version des jüngsten Vorgängers zugegriffen. Die Verwendung von -o kann die Größe der rekonstruierten SCCS-Datei vermindern. Sie kann auch die Form des Deltabaums der Originaldateien verändern.

## comb

-s Dieses Argument bewirkt die Erzeugung einer Shell-Prozedur. Diese gibt eine Übersicht aus, die für jede Datei folgende Informationen enthält: Dateiname, -größe (in Blöcken) nach der Zusammenfassung, Originalgröße (ebenfalls in Blöcken) und die prozentuale Veränderung, berechnet aus

100*(Originalgröße-komb.Größe)/Originalgröße

Sie sollten diese Option verwenden, um genau zu bestimmen, wieviel Platz durch den Vorgang eingespart werden kann, bevor Sie tatsächlich zusammenfassen.

Geben Sie keine Schlüsselargumente an, behält comb nur die Delta-Blattelemente und die Mindestzahl von Vorgängern bei, die benötigt werden, um den Baum zu erhalten.

SIEHE AUCH:

admin, delta, get, help, prs, sh
Systemliteratur TARGON: „Programmentwicklungs-Tools"

C

## comm

comm – Suchen gleicher Zeilen in zwei Dateien

SYNTAX:

comm [-123] *Datei1 Datei2*

BESCHREIBUNG:

Comm liest *Datei1* und *Datei2*. Die Zeilen der Dateien müssen in ASCII-Reihenfolge sortiert sein. Anschließend werden die Zeilen dieser Dateien dreispaltig aufgelistet; die erste Spalte enthält die Zeilen, die nur in *Datei1* vorkommen, die zweite Spalte enthält die Zeilen, welche nur in *Datei2* enthalten sind und die dritte Spalte schließlich enthält die Zeilen, die in beiden Dateien vorkommen.

Die Optionen [-123] unterdrücken die Anzeige bestimmter Spalten.

Der Befehl:

comm -12 *Datei1 Datei2*

zeigt nur die Zeilen an, die in beiden Dateien vorhanden sind.

Der Befehl:

comm -23 *Datei1 Datei2*

zeigt die Zeilen, die nur in *Datei1* vorhanden sind, an.

SIEHE AUCH:

cmp, diff, sort, uniq

C

## cp

cp – Kopieren

SYNTAX:

cp *Datei1* [ *Datei2* ...] *Ziel*

BESCHREIBUNG:

Der Inhalt der angegebenen Dateien wird in *Ziel* kopiert. Ist *Ziel* als Datei bereits vorhanden, bleiben Modus und Eigentümer von *Ziel* erhalten.

Der cp-Befehl erlaubt kein Kopieren, wenn die Zieldatei mit der Quelldatei identisch ist.

SIEHE AUCH:

cpio, rm

C

## cpio

cpio – Kopieren von Dateien

SYNTAX:

cpio -o [acBvV]

cpio -i [BcdmrtuvVfsSb6] [*Muster*] ...

cpio -p [adlmruvV] *Verzeichnis*

BESCHREIBUNG:

Cpio ist ein Programm zum Auslagern und Wiedereinspielen von Dateien.

Cpio -o liest aus der Standardeingabe die Namen der zu kopierenden Dateien und schreibt die Kopien – zusammen mit Pfadnamen und Statusinformationen der kopierten Dateien – in die Standardausgabe.

Cpio -i kopiert Dateien aus der Standardeingabe. Hierbei wird angenommen, daß diese Dateien das Ergebnis eines vorhergegangenen cpio -o-Laufs sind. Die Dateinamen können durch Metazeichen verkürzt angegeben werden. Sie können mehrere Muster spezifizieren. Geben Sie kein Muster an, ist der Standardwert '*', d. h. alle Dateien werden kopiert.

Cpio -p liest aus der Standardeingabe die Namen der gewünschten Dateien, die dann in das angegebene Verzeichnis kopiert werden.

Optionen:

a Nach dem Kopieren der Dateien wird das Dateizugriffsdatum wieder zurückgesetzt.

B Die Blockgröße der Ein-/Ausgabe beträgt 5120 Bytes. Diese Option ist nur bei der Verwendung von Magnetbändern oder Streamer-Kassetten sinnvoll.

d Benötigte Verzeichnisse werden automatisch angelegt.

c Der Kennsatz wird aus Portabilitätsgründen in ASCII-Zeichen geschrieben.

## cpio

r Dateien werden interaktiv umbenannt. Drücken Sie <CR>, wird die angezeigte Datei übersprungen.

t Das Inhaltsverzeichnis des angegebenen Datenträgers wird angezeigt. Ein Kopieren findet nicht statt.

u Unbedingtes Kopieren. Ohne diese Option werden Dateien jüngeren Datums nicht durch ältere Dateien mit gleichem Namen überschrieben.

v Ausgabe eines Inhaltsverzeichnisses. Wird diese Option zusammen mit der -t-Option benutzt, ist die Ausgabe des Inhaltsverzeichnisses gleich der Ausgabe des Kommandos ls -l.

V Es werden VAX-lesbare Kopfzeilen eingefügt.

l Wenn irgendmöglich, werden Dateien gelinkt und nicht kopiert. Kann nur gemeinsam mit cpio -p angewandt werden.

m Das letzte Änderungsdatum der Dateien soll erhalten bleiben. Diese Option wirkt nicht auf kopierte Verzeichnisse.

f Alle Dateien – außer diejenigen, deren Namen den angegebenen Mustern entsprechen – werden eingespielt.

s Byteweises Kopieren. Kann nur mit cpio -i angegeben werden.

S Halbwortweises Kopieren. Kann nur mit cpio -i angegeben werden.

b Kopieren von Bytes und Halbwörtern. Kann nur mit cpio -i angegeben werden.

6 Bearbeiten von Dateien, die älteren UNIX-Formaten entsprechen (z. B. Version 6).

C

## cpio

Beispiele:

```
ls | cpio -o >/dev/mt0
```

Der Inhalt des aktuellen Verzeichnisses wird auf den angegebenen Datenträger kopiert.

```
cd Verzeichnis
find .-depth -print | cpio -pdl Verzeichnis 1
```

Die angegebene Verzeichnishierarchie wird kopiert.

HINWEISE:

Pfadnamen dürfen nicht länger als 128 Zeichen sein.
Nur der Superuser kann Spezialdateien kopieren.
Cpio wurde geringfügig durch Nixdorf Computer AG modifiziert, um die Behandlung von bedingten symbolischen Verknüpfungen zu gewährleisten.

SIEHE AUCH:

ar, find, ls

C

## cpp

cpp – C Preprozessor

SYNTAX:

/lib/cpp [*Option* ...] [*Eingabedatei* [*Ausgabedatei*]]

BESCHREIBUNG:

Cpp ist der C Preprozessor, der als erster Schritt einer C-Kompilierung aufgerufen wird. Daher ist die Ausgabe von cpp die Eingabe für den nächsten Kompilierungsschritt.
Empfehlenswert ist der Aufruf von cpp als Teil des cc-Kommandos, da dieses in seiner Funktionalität den Änderungen der Sprache C angepaßt wird.

Geben Sie keine *Ein-* bzw. *Ausgabedatei* an, werden Standardein- und -ausgabe genommen.

Optionen:

-P Die Eingabe wird bearbeitet, ohne jedoch die Zeilenkontrollinformationen zu erstellen, die beim nächsten Kompilierschritt benötigt werden.

-C Kommentare werden nicht entfernt.

-U*Name* Alle Definitionen von *Name* werden entfernt. *Name* ist ein reserviertes Symbol. Die Liste dieser möglicherweise reservierten Symbole beinhaltet zur Zeit:

Betriebssystem: unix

Hardware: Nixdorf

-D*Name*[=*Wert*]
Für den Preprozessor wird auf die gleiche Weise wie bei #define ein Name definiert. Geben Sie *Wert* nicht an, wird standardmäßig 1 eingesetzt.

## cpp

-I*Verzeichnis*
: Ändert den Such-Algorithmus für #include-Dateien, deren Namen nicht mit '/' beginnen. Zuerst wird im angegebenen Verzeichnis gesucht und dann erst in den Standardverzeichnissen. #include-Dateien, deren Namen in Anführungszeichen eingeschlossen sind, werden zuerst im Verzeichnis nach angegebenen Dateien gesucht, anschließend in dem in der -I-Option angegebenen Verzeichnis und zuletzt in den Standard-Verzeichnissen. Genauso wird verfahren, wenn die Namen der #include-Dateien in spitzen Klammern (<>) eingeschlossen sind, nur wird das Verzeichnis der bei cc angegebenen Dateien nicht durchsucht.

Von cpp werden zwei spezielle Namen verstanden. LINE ist die aktuelle Zeilennummer (Ziffer) und FILE ist der aktuelle Dateiname (C-Zeichenkette). LINE und FILE können überall – auch in Makros – wie jeder andere definierte Name benutzt werden.

Alle cpp-Anweisungen beginnen mit dem Zeichen #. Folgende Anweisungen sind erlaubt:

#define *Name Ersetzungszeichenkette*
: Ersetze nachfolgende Vorkommen von *Name* durch *Ersetzungszeichenkette*.

#define *Name(Arg,...,Arg) Ersetzungszeichenkette*
: Beachten Sie, daß zwischen *Name* und der öffnenden Klammer kein Blank angegeben werden darf!
Ersetze nachfolgende Vorkommen von *Name*, gefolgt von (, einer durch Kommata getrennten Liste von Argumenten und einer schließenden Klammer durch *Ersetzungszeichenkette*.

#undef *Name*
: *Name* wird im nachfolgenden Programm nicht mehr ersetzt.

## cpp



#include "*Datei*"
#include <*Datei*>

Ersetze diese Anweisung durch den Inhalt von *Datei*. Bei der Angabe von *Datei* in spitzen Klammern vermutet der Preprozessor eine Systemdatei aus dem Verzeichnis /usr/include, bei der Angabe in Hochkommata wird eine private Datei vermutet (s. auch Option -I*Verzeichnis*).

#line *Konstante*[ "*Datei*"]

Die Zeilennummer der nächsten Anweisung wird auf den Wert von *Konstante* gesetzt. Geben Sie zusätzlich *Datei* an, wird der Name der aktuellen Quelldatei für den C-Compiler in *Datei* geändert.

#endif Diese Anweisung beendet die bedingte Übersetzung von Programmteilen. Diese bedingte Übersetzung muß durch eine der folgenden Anweisungen eingeleitet werden: #if, #ifdef oder #ifndef.

#ifdef *Name*

Diese Bedingung gilt als erfüllt, wenn an dieser Stelle *Name* in einer #define-Anweisung aufgetreten war und noch nicht mit #undef *Name* unwirksam gemacht wurde.

#ifndef *Name*

Diese Bedingung gilt als erfüllt, wenn *Name* noch nicht in einer #define-Anweisung vorgekommen ist bzw. mit #undef *Name* unwirksam gemacht wurde.

#if *konst. Ausdruck*

Ist der *konst. Ausdruck* ungleich 0 (wahr), werden die nachstehenden Anweisungen ausgeführt. Alle in C legalen Operatoren können in *konst. Ausdruck* angegeben werden. Zusätzlich ist ein unärer Operator definiert, der folgendermaßen angegeben werden kann: defined (*Name*) oder defined *Name*. Dies ermöglicht die Nutzung von #ifdef und #ifndef in einer #if-Anweisung.
Der sizeof-Operator ist nicht verfügbar.

#else Ist der *konst. Ausdruck* nach #if = 0 (falsch), werden die Anweisungen zwischen #if und #else ignoriert.

## cpp

DATEIEN:

/usr/include Standardverzeichnis für #include-Dateien

SIEHE AUCH:

cc, m4
Systemliteratur TARGON /35: „Die Sprache C"
Systemliteratur TARGON: „Programmentwicklungs-Tools"

C

## csplit

csplit – Aufspalten von Zusammenhängen

SYNTAX:

csplit [-s] [-k] [-f*Vorsilbe*] *Datei Arg 1* [... *Argn*]

BESCHREIBUNG:

Csplit liest die angegebene Datei und teilt sie in n+1 Abschnitte auf, wie in Argumenten *Arg 1*...*Argn* festgelegt. Standardmäßig werden diese Abschnitte in Dateien abgestellt, für die folgende Namenskonventionen gelten: xx00...xx*n* (*n* darf nicht größer als 99 sein). Diese Ausgabedateien enthalten folgende Teile der Eingabedatei:

00: Dateianfang bis zu (aber nicht einschließlich) der Zeile, die Sie in *Arg 1* angegeben haben.

01: Ab der in *Arg 1* angegebenen Zeile bis zu der in *Arg 2* genannten Zeile.

*n*+1: Ab der in *Argn* genannten Zeile bis zum Dateiende.

Optionen:

-s Normalerweise wird für jede erzeugte Datei die enthaltene Anzahl Zeichen ausgedruckt. Dies können Sie mit der Option -s unterbinden.

-k Bei Auftritt von Fehlern, werden die angelegten Dateien wieder gelöscht. Mit der Option -k können Sie dies verhindern.

-f*Vorsilbe* Die Dateien erhalten die Bezeichnung *Vorsilbe*00...*Vorsilben*. Standard ist xx00...xxn.

## csplit

Folgende Argumente können Sie beim Aufruf von csplit verwenden und miteinander kombinieren:

*/Ausdruck/* Für den Abschnitt von der aktuellen Zeile bis zu (aber nicht einschließlich) der Zeile mit dem angegebenen regulären Ausdruck muß eine Datei angelegt werden. Die Zeile mit dem regulären Ausdruck wird die aktuelle Zeile.
Auf dieses Argument kann eine positive (+) oder negative (-) Zeilenanzahl folgen, um die die aktuelle Zeile verändert wird.

*%Ausdruck%* Dieses Argument entspricht */Ausdruck/*; es wird jedoch für den entsprechenden Abschnitt keine Datei erzeugt.

*Zeilennummer* Eine Datei muß ab der aktuellen Zeile bis zu (aber nicht einschließlich) *Zeilennummer* angelegt werden. Die aktuelle Zeile ist dann *Zeilennummer*.

*{Anzahl}* Dieses Argument kann im Anschluß an jedes der o. a. Argumente eingesetzt werden. Folgt es auf ein *Ausdruck*-Argument, so wird es so oft ausgeführt, wie in *Anzahl* angegeben.
Folgt es auf *Zeilennummer*, wird die Datei nach jeweils der Anzahl Zeilen, die in *Zeilennummer* angegeben sind, geteilt. Dies wird *Anzahl*-mal ausgeführt.

Alle *Ausdruck*-Argumente, die Leerzeichen oder andere in Shell bedeutungsvolle Zeichen enthalten, müssen in entsprechende 'quotes' gesetzt werden. Reguläre Ausdrücke dürfen keine Zeilenende-Zeichen enthalten.

Die ursprüngliche Datei wird von csplit nicht berührt; soll sie gelöscht werden, müssen Sie dies selbst veranlassen.

## csplit



Beispiele:

csplit -k *Datei* 100 {99}

*Datei* wird bei jeder 100sten Zeile (bis zur 10.000sten Zeile) geteilt. Die Option -k sorgt dafür, daß die erzeugten Dateien erhalten bleiben, auch wenn keine 10.000 Zeilen vorhanden sind; allerdings erfolgt in diesem Fall eine Fehlermeldung.

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

Unter der Voraussetzung, daß prog.c – entsprechend den C-Codierungsregeln – Funktionen mit einem '}' am Zeilenanfang abschließt, werden mit diesem Beispiel Dateien erzeugt, die jede einzelne Funktion (bis zu 21) von prog.c enthalten.

SIEHE AUCH:

ed, sh

C

## cu

### cu – Anrufen und Anmelden in ein anderes System

SYNTAX:

cu [-s*Baud*] [-l*Leitung*] [-h] [-t] [-d] [-m] [-o] [-e] *Rufnummer* | *Verzeichnis*

BESCHREIBUNG:

Mit cu wird ein anderes UNIX-System, ein Terminal oder ein Nicht-UNIX-System angerufen. Cu ermöglicht den interaktiven Dialog sowie einen Transfer von Text-Dateien.

Sie können folgende Optionen setzen:

-s*Baud* Als Baud-Rate sind die Werte 110, 150, 300, 600, 1200, 4800 und 9600 zugelassen. Standardmäßig ist 300 eingestellt.

-l*Leitung* Mit dieser Option können Sie einen speziellen Leitungsanschluß auswählen. In diesem Fall wird die Leitungsgeschwindigkeit der Datei /usr/lib/uucp/L-devices entnommen und der ggf. mit der Option -s gewählte Wert überschrieben. Geben Sie keinen Leitungsanschluß an, wird anhand der vorgegebenen Baud-Rate ein passender Anschluß aus der Datei /usr/lib/uucp/L-devices ermittelt. (Nur bei Wählleitungen sinnvoll.)

-h Local-Echo-Mode wird emuliert. Dadurch ist es möglich, andere Computer-Systeme anzurufen, die Terminals im Halb-Duplex-Mode erwarten.

-t Carriage-return wird durch carriage-return-new-line ersetzt.

-d Aus verschiedenen Stellen des Programmablaufs werden Diagnose- und Ablaufmeldungen auf der Standardfehlerausgabe erzeugt.

-m Nutzen Sie eine Leitung, die über eine Modemstrecke führt, ist diese Option anzugeben, damit ggf. auf die Fertigmeldung des Modems gewartet wird. Ansonsten gilt das Fremdsystem als direkt angeschlossen (Null-Modem).

## cu

-o Erzeugt zu den ASCII-Zeichen auf der Verbindungsstrecke ein Parity-Bit (o-ODD).

-e Erzeugt zu den ASCII-Zeichen auf der Verbindungsstrecke ein Parity-Bit (e-EVEN).

*Rufnummer* oder *Verzeichnis*
Bei Wählverbindungen wird die Rufnummer verlangt, mit der das andere System zu erreichen ist. *Verzeichnis* ist erforderlich, falls die Systeme direkt verbunden sind. In diesem Fall wird die Vorgabe eines Leitungsanschluß durch die Option -l verlangt.

Nachdem die Leitungsverbindung aufgebaut ist, teilt sich cu, und zwei cu-Prozesse laufen parallel als Sende- und Empfangsprozeß. Der Sendeprozeß liest die Standardeingabe (lokale Tastatur) und sendet diese, bis auf die Zeilen, die mit dem Zeichen '~' beginnen, über die Leitung zum anderen System. Der Empfangsprozeß empfängt die Daten des Remote-Systems und gibt diese, bis auf die Zeilen, die mit dem Zeichen '~' beginnen, auf der Standardausgabe (lokales Display) aus.

Zur Datenflußkontrolle des Fremdsystems wird ein DC3/DC1-Protokoll verwendet, durch das ein Überlaufen des Eingabepuffers vermieden wird.

Zeilen, die mit dem Zeichen '~' beginnen, haben sowohl für den Sende- als auch für den Empfangsprozeß spezielle Bedeutung.

Sendeprozeß:

~. Ende der Verbindung und Abbruch von cu.

~EOT Ende der Verbindung und Abbruch von cu.

~<*Datei* Der Inhalt der angegebenen Datei wird so an das Remote-System gesendet, als ob er am lokalen Terminal eingegeben wurde.

~! Aufruf des interaktiven Kommandointerpreters Shell im lokalen System.

~!*Kommando*
Starten des angegebenen Kommandos auf dem lokalen System (per sh -c ...).

C

## cu

~$*Kommando*
Starten des angegebenen Kommandos auf dem lokalen System, die Ausgabe wird jedoch an das Remote-System gesendet.

~%take *Datei1* [*Datei2*]
Kopieren von *Datei1* (Datei des Remote-Systems) in *Datei2* (Datei des lokalen Systems). Geben Sie *Datei2* nicht an, wird *Datei1* auch als Zieldateiname angenommen.

~%put *Datei1* [*Datei2*]
Kopieren von *Datei1* (Datei des lokalen Systems) in *Datei2* (Datei des Remote-Systems). Geben Sie *Datei2* nicht an, wird *Datei1* auch als Zieldateiname angenommen.

~~... Sendet die Zeile '~...'.

~%nostop Abschalten des Datenflußprotokolls DC3/DC1. Dies ist nur sinnvoll, falls das Remote-System nicht richtig auf die Zeichen 'DC3' und 'DC1' reagiert.

Empfangsprozeß:

Zeilen, die das Remote-System sendet und die mit den Zeichen '~>' beginnen, bedeuten für den Empfangsprozeß Start oder Ende einer Umlenkung von Daten in eine Datei. Die vollständige Sequenz ist folgendermaßen:

~>[>]:*Datei*
Keine oder mehrere Zeilen, die in *Datei* eingetragen werden sollen
~>

Die Daten des Remote-Systems werden in die angegebene Datei umgelenkt bzw. an die bestehende Datei angehängt, falls in der ersten Zeile die Zeichenfolge '~>>:*Datei*' steht. Durch die Zeile '~>' wird die Umlenkung beendet.

cu

Das Kommando ~%put erfordert vom Remote-System die Kommandos stty und cat. Ebenso wird vorausgesetzt, daß die Belegung der Tastatursteuerzeichen ERASE und KILL des Remote-Systems mit der Tastaturbelegung des lokalen Systems übereinstimmt. Um diese Zeichen zu übertragen, werden an den entsprechenden Stellen die Steuerzeichen durch einen Backslash (\) neutralisiert.

Das Kommando ~%take erfordert vom Remote-System die Kommandos echo und cat.

DATEIEN:

/usr/lib/uucp/L-devices

/usr/spool/uucp/LCK..(Leitungsname)

/dev/null

HINWEISE:

Eingabedaten werden intern gepuffert.
Durch eine eingebaute Sendeverzögerung während der ~%put-Ausführung ist ein Datenverlust unwahrscheinlich.

SIEHE AUCH:

cat, echo, stty, uname, uucp

C

## cut

cut – Ausgabe vorgegebener Felder aus Zeilen einer Datei

SYNTAX:

cut -c*Liste* [*Datei1 Datei2* ...]

cut -f*Liste* [-d*Zeichen*] [-s] [*Datei1 Datei2* ...]

BESCHREIBUNG:

Mit cut können Sie bestimmte Spalten einer Tabelle oder definierte Felder aus Dateizeilen ausgeben. Die Felder, die in *Liste* definiert werden, können festgelegte Längen haben (-c), Sie können aber auch die Länge von Zeile zu Zeile variieren, indem Sie ein Trennzeichen (-f) angeben.
Sie können cut auch als Filter benutzen; geben Sie keine Eingabedateien an, wird die Standardeingabe gelesen.

Optionen:

*Liste* Die Bezeichnung der auszugebenden Felder wird, durch Kommata getrennt, in Ziffern angegeben (aufsteigende Reihenfolge). Mit einem optionalen Minuszeichen (-) können Sie auch ganze Bereiche angeben. Z. B.: 1,4,7; 1-3,8; -5,10 als kürzere Schreibweise für 1-5,10 oder 3- (drittes bis letztes Feld).

-c*Liste* Die *Liste* in Verbindung mit der -c-Option (ohne Leerzeichen) dient der Angabe von Zeichenpositionen. Z. B. erhalten Sie bei Angabe von -c1-72 die Ausgabe der ersten 72 Zeichen jeder Dateizeile.

-f*Liste* Diese *Liste* bezeichnet Felder, die in der angegebenen Datei jedoch durch Trennzeichen abgegrenzt sein müssen (s. -d). Z. B. bewirkt die Angabe von -f1,7 die Ausgabe der Felder 1 und 7. Zeilen ohne Trennzeichen werden in voller Länge ausgegeben (sinnvoll bei Überschriften), wenn Sie nicht die -s-Option benutzen.

## cut

-d*Zeichen* Angabe des Trennzeichens. Standard ist das Tabulatorzeichen. Diese Option ist nur in Verbindung mit -f wirksam. Benutzen Sie als Trennzeichen ein Sonderzeichen der Shell, müssen Sie einen Backslash voranstellen.

-s In Verbindung mit der -f-Option werden Zeilen ohne Trennzeichen ignoriert.

Beim Aufruf von cut muß entweder die -c-Option oder die -f-Option angegeben werden.

Beispiele:

```
cut -d: -f1,5 /etc/passwd
```

Ausgabe von Benutzer-IDs und Benutzernamen aus der Datei /etc/passwd.

```
name='who am i | cut -f1 -d" "'
```

Der Variablen name wird der Benutzername zugewiesen.

MELDUNGEN:

Hat eine Zeile mehr als 511 Zeichen oder Felder, erfolgt eine Meldung.

Haben Sie beim Aufruf von cut weder die -c noch die -f-Option angegeben oder wurde *Liste* unkorrekt definiert, erhalten Sie eine Fehlermeldung.

SIEHE AUCH:

grep, paste

C

## cxref

cxref – Crossreferenzliste von C-Programmen

SYNTAX:

cxref [-cst] [-w*Breite*] [-o*Datei*] *Datei* ...

BESCHREIBUNG:

Cxref analysiert die angegebenen C-Quelldateien und erstellt auf Standardausgabe eine Crossreferenzliste sämtlicher auftretender Symbole. Ohne -c-Option wird die Liste für jede Datei einzeln erstellt, mit -c wird eine kombinierte Liste für alle Dateien erstellt. Die Stelle, an der ein Symbol definiert wird, ist mit einem '*' gekennzeichnet.

Cxref schließt einen Preprozessorlauf mit ein.

Optionen:

| | |
|---|---|
| -c | Kombinierte Liste für alle Dateien. |
| -w*Breite* | Ändern der Ausgabebreite (Standard: 80). |
| -o*Datei* | Umleiten der Ausgabe in die angegebene Datei. |
| -s | Unterdrücken der Ausgabe der Quell-Dateinamen. |
| -t | Ausgabebreite 80 Zeichen. |

SIEHE AUCH:

cc
Systemliteratur TARGON /35: „Die Sprache C"

# date

## date – Anzeigen und Einstellen des Datums und der Uhrzeit

SYNTAX:

date [*mmddhhmm*[*yy*]] [+*Format*]

BESCHREIBUNG:

Geben Sie kein Argument an, werden aktuelles Datum und Uhrzeit angezeigt.

Werden Argumente angegeben, wird das Datum und die Uhrzeit eingestellt. Das erste *mm* ist die Zahl des Monats, *dd* ist die Spezifikation des entsprechenden Tages. Mit *hh* werden die Stunden angegeben (24-Stunden-System), das zweite *mm* bezeichnet die Minuten. Argument *yy* ist optional und bezeichnet die letzten beiden Ziffern der Jahreszahl. Folgender Befehl:

date 10080045

stellt Datum und Uhrzeit auf den 8. Oktober 00.45h ein. Die Angabe von Jahr, Monat und Tag kann entfallen. Es werden dann die aktuellen Werte als Default-Werte angenommen.

Mit der Angabe von +*Format* kann der Benutzer die Form der Ausgabe vorgeben. Jedes Feld besteht aus dem %-Zeichen, gefolgt von einem Kennbuchstaben für die Feldbeschreibung.

Feldbeschreibung:

n Einfügen eines New-line-Zeichens

t Einfügen eines Tabulator-Zeichens

m Monat(01-12)

d Tag (01-31)

y Die letzten zwei Ziffern der Jahreszahl (00-99)

D Datum in mm/dd/yy

## date

H Stunden (0-23)

M Minuten (00-59)

S Sekunden (00-59)

T Zeit in der Form HH:MM:SS

j Tag des Jahres

w Wochentag (0=Sonntag)

a Abkürzung für Tagesnamen (Sun-Sat)

h Abkürzung für Monate (Jan-Dec)

r Zeit in AM/PM-Notation

Beispiel:

```
date '+Datum: %m/%d/%y%nZeit: %H:%M:%S'
```

generiert folgende Form der Ausgabe:

```
Datum: 08/01/76
Zeit: 14:45:05
```

Nur der Superuser kann das aktuelle Datum verändern.

# dc

## dc – Tischrechner

SYNTAX:

dc [*Datei*]

BESCHREIBUNG:

Dc ist eine Rechenprogramm mit beliebiger Genauigkeit. Normalerweise arbeitet es mit dezimalen Ganzzahlen, aber der Benutzer kann eine Eingabebasis (Zahlensystem-Grundzahl), eine Ausgabebasis und die Anzahl der zu führenden Nachkommastellen spezifizieren. (Siehe bc, ein Preprozessor für dc, der eine Infix-Schreibweise und eine C-ähnliche Syntax bietet, in der Funktionen implementiert sind. Bc bietet außerdem brauchbare Kontrollstrukturen für Programme.) In der Gesamtstruktur ist dc eine Stacking-Maschine (umgekehrte polnische Notation). Haben Sie eine Datei angegeben, werden Eingabedaten aus dieser Datei übernommen, bis das Dateiende erreicht ist; anschließend werden Eingabedaten von der Standardeingabe übernommen. Die folgenden Konstruktionen werden erkannt:

*Zahl* Der Wert der Zahl wird auf dem Stack abgelegt. Eine Zahl ist eine ununterbrochene Folge der Ziffern 0-9. Den Ziffern kann ein Unterstrich (_) als negatives Vorzeichen vorangestellt sein. Zahlen können Dezimalpunkte enthalten.

+ - / * % ^ Die beiden oberen Werte auf dem Stack werden addiert (+), subtrahiert (-), multipliziert (*), dividiert (/), es wird der Rest gebildet (%) oder sie werden potenziert (^). Die beiden obersten Werte werden vom Stack geholt. An deren Stelle wird das Ergebnis auf dem Stack abgelegt. Ein eventueller Nachkommateil eines Exponenten wird ignoriert.

s*x* Der oberste Wert vom Stack wird genommen und in einem Register namens *x* abgelegt, wobei *x* ein beliebiges Zeichen sein kann. Wird s als Großbuchstabe eingegeben, so wird *x* als Stack behandelt, und der Wert wird auf diesem Stack abgelegt.

## dc

| | |
|---|---|
| l*x* | Der Wert des Registers *x* wird auf dem Stack abgelegt. Das Register *x* bleibt unverändert. Alle Register haben anfangs den Wert Null. Wird das l als Großbuchstabe geschrieben, so wird Register *x* als Stack behandelt, und sein oberster Wert wird oben auf dem Haupt-Stack abgelegt. |
| d | Der oberste Wert auf dem Stack wird dupliziert. |
| p | Der oberste Wert im Stack wird ausgegeben, bleibt jedoch unverändert. P interpretiert den obersten Wert des Stack als ASCII-Zeichenfolge, entfernt ihn vom Stack und gibt ihn aus. |
| f | Alle Werte im Stack werden ausgegeben. |
| q | Verlassen des Programms. Wenn eine Zeichenfolge ausgeführt wird, wird die Rekursionsstufe um zwei Stufen im Stack vermindert. Geben Sie q als Großbuchstabe an, wird der oberste Wert vom Stack entfernt und die Ausführungsstufe der Zeichenfolge um diese Anzahl Stufen vermindert. |
| x | Das oberste Element auf dem Stack als Zeichenfolge behandeln und als Folge von dc-Kommandos ausführen. |
| X | Die oberste Zahl im Stack durch ihren Skalierfaktor ersetzen. |
| [...] | Den in eckigen Klammern stehenden ASCII-String oben auf dem Stack ablegen. |
| <*x* >*x* =*x* | Die beiden oberen Elemente im Stack werden entfernt und verglichen. Register *x* wird ausgewertet, wenn sie die angegebene Relation erfüllen. |
| v | Das oberste Element im Stack durch seine Quadratwurzel ersetzen. Ein gegebenenfalls vorhandener Nachkommateil des Arguments wird berücksichtigt, aber ansonsten wird der Skalierfaktor ignoriert. |
| ! | Den Rest der Zeile als UNIX-Kommando interpretieren. |
| c | Alle Werte im Stack werden entfernt. |

## dc



| | |
|---|---|
| i | Der oberste Wert im Stack wird entfernt und als Zahlenbasis für die weitere Eingabe verwendet. I legt die Eingabebasiszahl oben auf dem Stack ab. |
| o | Der oberste Wert wird vom Stack entfernt und als Zahlenbasis für die weitere Ausgabe verwendet. |
| O | Die Ausgabebasiszahl oben auf dem Stack ablegen. |
| k | Der oberste Wert wird vom Stack entfernt und als nichtnegativer Skalierfaktor verwendet: Die entsprechende Anzahl Stellen wird in den Ausgabedaten, bei der Multiplikation, Division und Potenzierung berücksichtigt. Das Zusammenwirken von Skalierfaktor, Eingabebasis und Ausgabebasis ist problemlos, wenn alle zusammen geändert werden. |
| z | Die Stack-Stufe wird oben auf dem Stack abgelegt. |
| Z | Die oberste Zahl im Stack durch ihre Länge ersetzen. |
| ? | Eine Eingabezeile wird von der Eingabe (normalerweise vom Terminal) übernommen und ausgeführt. |
| ;: | Wird von bc für Matrixoperationen verwendet. |

Beispiel:

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

```
[la1+dsa*pla10>y]sy
Osa1
lyx
```

## dc

DIAGNOSE:

| | |
|---|---|
| *x* is unimplemented | *x* ist eine Oktalzahl. |
| stack empty | Es sind nicht genügend Elemente im Stack vorhanden, um die gewünschte Operation auszuführen. |
| Out of space | Die Freiliste ist leer (zu viele Stellen). |
| Out of headers | Es müssen zu viele Zahlen gespeichert werden. |
| Out of pushdown | Es sind zu viele Elemente im Stack. |
| Nesting Depth | Zu große Schachtelungstiefe bei der Ausführung. |

SIEHE AUCH:

bc

## dd

dd – Konvertieren und Kopieren von Dateien

SYNTAX:

dd [*Option=Wert*]

BESCHREIBUNG:

Dd konvertiert Dateien gemäß einer Eingabespezifikation.

Dieser Befehl ist eine große Hilfe beim Ein-/Ausgeben fremder Magnetbänder oder Dateien, die nicht den Spezifikationen des Zielsystems entsprechen.

Die möglichen Optionen mit den zugehörigen Werten sind:

| | |
|---|---|
| if=*Datei* | Eingabedatei; Standard ist die Standardeingabe. |
| of=*Datei* | Ausgabedatei; Standard ist die Standardausgabe. |
| ibs=*n* | Eingabe-Blockgröße ist *n* Bytes; Standard ist 512 Bytes. |
| obs=*n* | Ausgabe-Blockgröße ist *n* Bytes; Standard ist 512 Bytes. |
| bs=*n* | Ein- und Ausgabe-Blockgröße sind *n* Bytes. Diese Option überlagert 'ibs' und 'obs'. |
| cbs=*n* | Größe des Konvertierungspuffers. |
| skip=*n* | *n* Sätze, vom Dateianfang gerechnet, werden überlesen. |
| seek=*n* | *n* Sätze der Ausgabedatei werden übersprungen, bevor das Kopieren beginnt. |
| count=*n* | Nur *n* Sätze werden kopiert. |

## dd

| | |
|---|---|
| conv=ascii | EBCDIC nach ASCII. |
| ebcdic | ASCII nach EBCDIC. |
| ibm | ASCII nach EBCDIC gemäß IBM. |
| lcase | Buchstaben werden klein geschrieben. |
| ucase | Buchstaben werden groß geschrieben. |
| swab | Jedes Paar Bytes wird vertauscht. |
| noerror | Bei Fehlern wird die Verarbeitung fortgesetzt. |
| sync | Jeder Eingabesatz wird auf 'ibs'-Länge vergrößert. |

Bei 'conv' können beliebig viele, durch Kommata getrennte Optionen eingegeben werden.

Wo Größen festzulegen sind, muß die entsprechende Anzahl Bytes angegeben werden, u. U. gefolgt von den Zeichen 'k', 'b' oder 'w' zur Kennzeichnung des Multiplikators:

k = 1024 (Kilo-Byte)

b = 512 (Blöcke)

w = 2 (Worte)

Zwei Zahlen können durch 'x' miteinander multipliziert werden, um anzudeuten, daß es sich um ein Produkt handelt.

Cbs wird nur bei 'ascii'- bzw. 'ebcdic'-Konvertierungen verwendet. Im ersten Fall werden die 'cbs'-Zeichen in den Konvertierungspuffer eingelesen, nach ASCII konvertiert, hinten anhängende Blanks abgeschnitten und ein Zeilenende-Zeichen (end-of-line) hinzugefügt, bevor die Zeile ausgegeben wird.
Im zweiten Fall werden die ASCII-Zeichen in den Konvertierungspuffer eingelesen, nach EBCDIC konvertiert und der Datensatz für die Ausgabe durch Hinzufügen von Blanks der 'cbs'-Länge angepaßt.

Wenn die Verarbeitung abgeschlossen ist, wird sowohl für die Eingabe als auch für die Ausgabe die Anzahl der vollständig und teilweise belegten Blöcke angezeigt.

## dd

Beispiel:

Ein Magnetband mit einer Blockung von zehn 80-Byte EBCDIC-Sätzen pro Block soll in eine ASCII-Datei x eingelesen werden:

```
dd if=/dev/rmt0 of=x ibs=800 cbs=80 conv=ascii,lcase
```

Der dd-Befehl eignet sich hervorragend für I/O's auf unformatierten Speichermedien, da in beliebigen Satzlängen gelesen und geschrieben werden kann.

SIEHE AUCH:

cp

## delta

delta – Einbringen eines Deltas (Verändern) in SCCS-Dateien

SYNTAX:

delta [-r*SID*] [-s] [-n] [-g*Liste*] [-m[*MR-Liste*]]
[-y[*Kommentar*]] [-*p*] *Datei* ...

BESCHREIBUNG:

Delta fügt in die genannte SCCS-Datei die Änderungen ein, die in Dateien, die von get bearbeitet wurden (g-Dateien oder generierte Dateien), enthalten sind.

Es wird ein Delta für jede genannte SCCS-Datei erzeugt. Geben Sie ein Verzeichnis an, verhält delta sich so, als ob jede in dem Verzeichnis enthaltene Datei spezifiziert worden sei. Ignoriert werden lediglich nichtlesbare Dateien sowie Dateien, deren Namen nicht mit 's.' (keine SCCS-Dateien) beginnen. Wird '-' als Dateiname angegeben, liest delta die Standardeingabe; jede Eingabezeile wird dann als Name einer SCCS-Datei interpretiert.

In einigen Fällen erscheinen nach dem Aufruf von delta Zeichen (Prompts) auf dem Bildschirm. Dies hängt von bestimmten Optionen ab, die in SCCS-Dateien enthalten sind.

Optionen werden unabhängig von der Reihenfolge ihrer Angabe auf jede spezifizierte Datei angewendet.

-r*SID* Stellt ausschließlich fest, welches Delta für die SCCS-Datei erzeugt werden soll. Der Einsatz dieser Option ist nur nötig, wenn zwei oder mehr ausstehende gets (get -e) eines Benutzers (Login-Name) auf die gleiche SCCS-Datei vorhanden sind. Der SID-Wert ist entweder der angegebene Wert in der get-Kommandozeile oder die Versionsnummer, die von get zurückgegeben wird. Fehler treten auf, wenn die angegebene Versionsnummer mehrdeutig ist oder wenn sie in der Kommandozeile angegeben werden muß, aber dort fehlt.

## delta

-s
Unterdrückt die Anzeige der angelegten Versionsnummer sowie der Anzahl der eingefügten, gelöschten und unveränderten Zeilen in der SCCS-Datei.

-n
Die editierte g-Datei bleibt erhalten und wird nicht – wie im Normalfall – nach Ausführung von delta gelöscht.

-g*Liste*
Anlegen einer Liste (siehe auch den get-Befehl bzgl. der Listen-Definition) der Deltas, die ignoriert werden sollen, wenn auf die Datei mit der SCCS-Identifikationsnummer, die von diesem Befehl angelegt wurde, zugegriffen wird.

-m[*MR-Liste*]
Ist in der SCCS-Datei eine -v-Option enthalten, müssen Sie eine MR-Nummer (Modification Request) angegeben, um das neue Delta zu kreieren.
Geben Sie die -m-Option nicht an und delta liest aus der Standardeingabe, wird 'MRs?' am Bildschirm angezeigt, bevor die Eingabe gelesen wird. Die Frage 'MRs?' geht immer der Frage nach 'comments?' (Kommentar) voraus (siehe -y-Option).

Eine Liste von MRs wird durch Leerzeichen oder Tabulatoren getrennt. Durch '\' am Ende einer Zeile wird die Auflistung in der nächsten Zeile fortgeführt. <CR> beendet die Liste. Es ist zu beachten, daß der Wert der Option -v als Name eines Programms oder einer Shell-Prozedur interpretiert wird, welche die Richtigkeit der MR-Nummern prüft. Gibt dieses Programm einen Code ungleich 0 zurück, wird delta mit der Annahme beendet, daß nicht alle MR-Nummern gültig sind.

-y*Kommentar*
Ein beliebiger Text, der die Gründe für ein Delta beschreibt. Eine leere Zeichenkette wird als gültiger Kommentar akzeptiert.

Geben Sie die -y-Option nicht an und delta liest aus der Standardeingabe, wird die Frage 'comments?' am Bildschirm ausgegeben, bevor die Eingabe gelesen wird. Das Zeilenende beendet den kommentierenden Text.

## delta

-p Anzeige der Unterschiede in der SCCS-Datei – im Format der diff-Ausgabe – bevor und nachdem delta ausgeführt wurde.

DATEIEN:

g-Datei Die Datei bestand bereits vor der Ausführung von delta. Nach beendigter Ausführung wird sie gelöscht.

p-Datei Die Datei bestand bereits vor der Ausführung von delta. Sie kann nach beendigter Ausführung weiterbestehen.

q-Datei Die Datei wird während der Ausführung von delta angelegt. Nach der Ausführung wird sie gelöscht.

x-Datei Die Datei wird während der Ausführung von delta angelegt. Nach beendigter Ausführung wird sie in eine SCCS-Datei umbenannt.

z-Datei Die Datei wird während der Ausführung von delta angelegt und gelöscht.

d-Datei Die Datei wird während der Ausführung von delta angelegt und nach beendigter Ausführung gelöscht.

HINWEISE:

Zeilen, die mit dem ASCII-Zeichen SOH beginnen, können nicht in einer SCCS-Datei untergebracht werden. Das Zeichen hat für SCCS eine spezielle Bedeutung, daher muß ein Backslash vorangestellt werden.

Ein get auf mehrere SCCS-Dateien, gefolgt von einem delta auf die gleichen Dateien, sollte vermieden werden, wenn get eine größere Menge von Daten zu generieren hat. Stattdessen sollten mehrere get/delta-Sequenzen benutzt werden.

Wird beim Aufruf von delta die Standardeingabe '-' angegeben, müssen die Optionen -m und -y ebenfalls benutzt werden. Das Weglassen dieser Optionen begründet Fehler.

SIEHE AUCH:

admin, bdiff, cdc, get, help, prs, rmdel
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# deroff

deroff – Entfernen von nroff-, tbl- und eqn-Konstrukten

SYNTAX:

deroff [-w] [*Dateien*]

BESCHREIBUNG:

Deroff entfernt aus den angegebenen Dateien alle nroff-Steuerzeichen, Makro-Aufrufe, Backslash-Konstrukte, eqn-Konstrukte (zwischen den Zeilen, die mit .EQ und .EN beginnen) sowie Tabellenbeschreibungen. In den meisten Fällen werden an Stelle der entfernten Konstrukte Leerzeichen oder Leerzeilen in den verbleibenden Text – der auf der Standardausgabe ausgegeben wird – eingefügt. Deroff bearbeitet ebenfalls Dateien, die mit den nroff-Steuerzeichen .so und .nx eingebunden wurden. Trifft deroff bei der Bearbeitung der Eingabedateien zum zweiten Mal auf die gleichen eingebundenen Dateien wird eine mit .so eingebundene Datei ignoriert. Bei einer mit .nx eingebundenen Datei beendet deroff die Ausführung.

Geben Sie die -w-Option an, erfolgt die Ausgabe als Liste von „Worten", jedes „Wort" in einer Zeile. Alle anderen Zeichen werden gelöscht. Im Eingabetext ist ein „Wort" jede Zeichenkette, die wenigstens zwei Buchstaben enthält. Die Zeichenkette kann aus Buchstaben, Ziffern, Ampersands (&) und einfachen Hochkommata (') bestehen. In einem Makroaufruf ist ein „Wort" eine Zeichenkette, die mit mindestens zwei Buchstaben beginnt und mindestens drei Buchstaben enthält. Nachfolgende Ampersands und Hochkommata werden aus dem „Wort" entfernt.

SIEHE AUCH:

eqn, nroff, tbl
Systemliteratur TARGON /35: „Textformatierer nroff"

# df

## df – Ermittlung des freien Plattenspeicherplatzes (disk free)

SYNTAX:

df [-i] [-l] [*Dateisystem* ...] [*Datei* ...]

BESCHREIBUNG:

Df gibt den Umfang des freien Plattenspeicherplatzes aus, der in dem angegebenen Dateisystem bzw. in dem Dateisystem, in dem die angegebene Datei enthalten ist, verfügbar ist. Geben Sie kein Dateisystem an, so wird der freie Speicherplatz in allen eingehängten Dateisystemen ausgegeben. Die ausgegebenen Werte verstehen sich in KByte.

Optionen:

-i Anzahl der belegten und freien I-Knoten auswerten.

-l Untersuchung der Liste freier Blöcke und damit Gegenprüfung, ob die Gesamtanzahl im Superblock des Dateisystems korrekt ist.

DATEIEN:

/etc/fstab Liste der normalerweise eingehängten Dateisysteme

# diff





diff – Ermittlung von Dateiunterschieden

SYNTAX:

diff [-efbh] *Datei1 Datei2*

BESCHREIBUNG:

Diff vergleicht *Datei1* und *Datei2* zeilenweise und zeigt die Unterschiede an. Wird *Datei1* oder *Datei2* mit '-' angegeben, wird die Standardeingabe gelesen. Ist *Datei1* (*Datei2*) ein Verzeichnis, wird die Datei in dem Verzeichnis verglichen, welche den gleichen Namen wie *Datei2 (Datei1)* hat.

Die Ausgabe enthält Zeilen des folgenden Formats:

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

Diese Zeilen ähneln ed-Kommandos, um *Datei1* in *Datei2* umzuwandeln. Die Zahlen (n3 und n4) hinter den Buchstaben (a, d und c) sind Zeilenangaben in *Datei2*. Indem Sie 'a' durch 'd' ersetzen und die Zeile rückwärts lesen, erhalten Sie die Hinweise, um *Datei2* in *Datei1* umzuwandeln. Wie bei ed werden identische Paare (n1 = n2 oder n3 = n4) als einzelne Zahl dargestellt.
Jeder dieser Zeilen folgen die Zeilen, die in der ersten Datei betroffen sind. Sie werden durch '<' gekennzeichnet. Anschließend folgen die entsprechenden Zeilen der zweiten Datei, gekennzeichnet durch '>'.

-e Erstellung eines Scripts von 'a'-, 'c'- und 'd'-Kommandos für den Editor ed, um *Datei2* aus *Datei1* zu rekonstruieren.

-f Anzeige der Unterschiede in umgekehrter Reihenfolge.

-b Abschließende Leerstellen (Zwischenräume und Tabs) werden ignoriert. Andere Folgen von Leerzeichen werden als gleichlang angenommen.

## diff

-h Für Dateien „unbegrenzter" Länge. Arbeitet sehr schnell, jedoch nur, wenn die geänderten Passagen kurz und zudem noch deutlich voneinander abgegrenzt sind. Die Optionen -e und -f können nicht mit dieser Option kombiniert werden.

DATEIEN:

/tmp/d?????

/usr/lib/diffh für die Option -h

HINWEIS:

Fehlt in einer Eingabedatei am Ende das Zeilenende-Zeichen, wird ein entsprechender Hinweis ausgegeben. Auch wenn die beiden Zeilen der Dateien unterschiedlich sind, erfolgt die Ausgabe so, als wären sie gleich.

SIEHE AUCH:

cmp, comm, ed

# diff3

## diff3 – Vergleich von drei Dateiversionen

SYNTAX:

diff3 [-ex3] *Datei1 Datei2 Datei3*

BESCHREIBUNG:

Diff3 vergleicht drei Versionen einer Datei und zeigt differierende Textstellen – symbolisiert durch folgende Codes – an:

==== Alle drei Dateien differieren untereinander.

====1 *Datei1* differiert von *Datei2* und *Datei3*.

====2 *Datei2* differiert von *Datei1* und *Datei3*.

====3 *Datei3* differiert von *Datei1* und *Datei2*.

Die Änderungen der unterschiedlichen Dateien können folgendermaßen vorgenommen werden:

*D*:*n1* a Der Text wird hinter *n1* in der Datei *D*, wobei *D* = 1, 2 oder 3 sein kann.

*D*:*n1* ,*n2* c Der Text wird in dem angegebenen Zeilenbereich – also zwischen *n1* und *n2* geändert. Ist *n1* = *n2* wird der Bereich auf *n1* verkürzt.

Optionen:

-e Für den Editor ed wird ein Script erstellt, das alle Unterschiede zwischen *Datei2* und *Datei3* enthält (==== und ====3). Dieses Script kann in *Datei1* eingefügt werden.

-x Erstellung eines Scripts für den Editor ed, indem die Unterschiede aller Dateien (====) enthalten sind.

-3 Erstellung eines Scripts für den Editor ed, indem die Unterschiede zwischen *Datei3* (====3) und den beiden anderen Dateien enthalten sind.

## diff3

Mit dem folgenden Kommando können Sie das ed-Script in *Datei1* einfügen:

(cat script; echo '1,$p') | ed - *Datei1*

DATEIEN:

/tmp/d3*

/usr/lib/diff3prog

HINWEISE:

Textzeilen, die lediglich aus einem Punkt bestehen, heben die -e-Option auf.
Dateien, die länger als 64 Kilobytes sind, können von diff3 nicht bearbeitet werden.

SIEHE AUCH:

diff

## diffmk

diffmk – Markieren von Differenzen zwischen Dateien für nroff/troff

SYNTAX:

diffmk *Name1 Name2 Name3*

BESCHREIBUNG:

Diffmk vergleicht zwei Versionen einer Datei und erstellt eine dritte Datei, die „change mark"-Kommandos (Änderungsmarkierungen) für nroff/troff enthält. *Name1* und *Name2* sind die alte und die neue Version der Datei. Diffmk generiert *Name3*, die die Zeilen von *Name2* und eingefügte „change mark"-Anforderungen (.mc) enthält. Beim Formatieren von *Name3* wird geänderter oder eingefügter Text durch einen senkrechten Strich (|) am rechten Rand jeder Zeile markiert. Stellen, an denen Text gelöscht wurde, werden durch einen einzelnen Stern (*) markiert.

Diffmk kann auch herangezogen werden, um Listings von C- oder anderen Programmen zu erstellen, in denen die Änderungen markiert sind. Eine typische Befehlszeile für einen solchen Fall würde etwa wie folgt aussehen:

diffmk alt.c neu.c tmp; nroff macs tmp | pr

wobei die Datei macs folgende Befehle enthält:

```
.pl 72
.ll 77
.nf
.eo
.nc '
```

Mit dem Befehl .ll könnte je nach Art des zu druckenden Programms eine andere Zeilenlänge angegeben werden. Die Befehle .eo und .nc sind in der Regel nur für C-Programme erforderlich.

## diffmk

Erscheinen Ihnen die Zeichen | und * für die Markierung von Änderungen ungeeignet, können Sie diese in einer Kopie von diffmk ändern (diffmk ist eine Shell-Prozedur).

FEHLER:

Aus ästhetischen Gründen kann es notwendig sein, Ausgabedaten manuell nachzuformatieren. Bestehen zwischen Dateien Differenzen, bei denen es sich nur um Formatierungsbefehle handelt, kann dies zu unerwünschten Ergebnissen führen. Wird beispielsweise .sp durch .sp 2 ersetzt, so steht die Änderungsmarkierung in der vorangehenden oder nachfolgenden Zeile in der Ausgabe.

SIEHE AUCH:

diff, nroff
Systemliteratur TARGON /35: „Textformatierer nroff"

# dircmp

dircmp – Vergleich zweier Verzeichnisse

SYNTAX:

dircmp [-d] [-s] *Verzeichnis1 Verzeichnis2*

BESCHREIBUNG:

Dircmp überprüft die angegebenen Verzeichnisse und generiert verschiedene tabellarische Informationen über ihren Inhalt. Z. B. wird eine Liste der Dateien ausgegeben, die in beiden Verzeichnissen vorkommen und eine Liste die anzeigt, ob die Dateien, die in beiden Verzeichnissen vorhanden sind, auch den gleichen Inhalt haben.

Optionen:

-d Vergleicht den Inhalt zweier Dateien mit demselben Namen in den angegebenen Verzeichnissen. Die angegebene Liste zeigt die Änderungen an, die vorgenommen werden müssen, um die Dateien in Übereinstimmung zu bringen.

-s Meldungen über identische Dateien werden unterdrückt.

SIEHE AUCH:

cmp, diff

# du

## du – Ermittlung der Speicherbelegung

SYNTAX:

du [-ars] [*Name* ...]

BESCHREIBUNG:

Mit diesem Befehl können Sie den gesamten beanspruchten Speicherplatz sowie die Größe einzelner Dateien und Verzeichnisse am Bildschirm abfragen. Im Falle eines Verzeichnisses wird der von diesem Verzeichnis selbst belegte Platz sowie der sämtlicher zugehörigen Unterverzeichnisse angezeigt.
Die Angaben beziehen sich dabei auf Blöcke, d. h. der du-Befehl ist damit dem Befehl ls mit der Option -s vergleichbar.

Dateien mit zwei und mehr Links werden nur einmal gezählt.

Optionen:

-a Für jede vorhandene Datei wird die Anzahl der Blöcke einzeln angezeigt. Diese Option müssen Sie auch dann verwenden, wenn Sie sich über die Größe einer einzelnen Datei informieren wollen.

-s Die Gesamtzahl aller beanspruchten Blöcke wird angezeigt.

-r Normalerweise werden Verzeichnisse, die nicht gelesen, und Dateien, die nicht eröffnet werden können, bei der Ermittlung des Speicherplatzes nicht berücksichtigt. Bei der Verwendung der Option -r erhalten Sie in solchen Fällen eine Fehlermeldung.

Wird du ohne Option eingegeben, erhalten Sie die Anzahl Blöcke je Verzeichnis.

## du

HINWEISE:

Geben Sie die -a-Option nicht an, werden nur Verzeichnisse ausgewertet.
Wird eine gewisse Anzahl von gelinkten Dateien überschritten, zählt du den Speicherplatz mehr als ein Mal.
Bei Dateien, die Leerräume enthalten, werden die belegten Blöcke eventuell nicht ganz korrekt angegeben.

# echo

## echo – Anzeige von Argumenten

SYNTAX:

echo [*Argument* ...]

BESCHREIBUNG:

Echo zeigt die Argumente des Befehls – getrennt durch Leerzeichen – am Terminal an. Es interpretiert C-ähnliche Escape-Konventionen:

| | |
|---|---|
| \b | Backspace |
| \c | Kein Zeilenvorschub |
| \f | Formularvorschub |
| \n | Zeilenvorschub |
| \r | Neue Zeile (CR) |
| \t | Tabulator |
| \v | Vertikaler Tabulator |
| \\ | Backslash (\) |

Echo wird dazu benutzt, um Diagnosemeldungen aus Shell-Programmen auszugeben und feststehende Daten in Pipes zu übertragen.

SIEHE AUCH:

sh

## ed, red

ed, red – Standard-Editor

SYNTAX:

ed [-] [*Datei*]

red [-] [*Datei*]

BESCHREIBUNG:

Mit dem Text-Editor **ed** können Sie Textdateien erstellen, anzeigen und bearbeiten. Dieser Editor ist nicht bildschirmorientiert, d. h. die Bearbeitung (Löschen, Einfügen, Anfügen, Verschieben usw.) erfolgt nicht mit Hilfe des Cursors direkt im Text, sondern durch Angabe des entsprechenden Befehls zeilenweise außerhalb des Textes.

**Red** ist eine eingeschränkte Version von ed. Mit red können Sie nur Dateien im aktuellen Verzeichnis bearbeiten. Shell-Kommandos (Aufruf: !*Kommando*) werden nicht ausgeführt.

Beim Aufruf des Editors wird ein Arbeitsbereich, der sogenannte Editierpuffer, eingerichtet, in dem die Texterstellung und -bearbeitung stattfindet.
Die Änderung bereits vorhandener Texte erfolgt dabei in einer Kopie, die beim Aufruf des Programms von der gewünschten Textdatei automatisch angefertigt und in den Puffer eingelesen wird.

Da der Editierpuffer nur für die Dauer des Editierprogramms besteht und der Inhalt des Puffers beim Verlassen des Editors gelöscht wird, müssen Sie unbedingt vor Beendigung des Programms den Text mit dem Befehl w (write) sichern, d. h. in der entsprechenden Textdatei speichern.

Der Editor enthält einen Text- und einen Befehlsmodus, in denen die Eingaben entweder ausschließlich als Text oder als Befehle interpretiert werden.
Beim Aufruf des Programms befinden Sie sich im Befehlsmodus, aus dem heraus Sie mit einem Textbearbeitungsbefehl in den Textmodus gelangen.

## ed, red

Die Befehle sind einfach, und ihre Struktur ist immer gleich. Im allgemeinen ist nur ein Befehl pro Zeile erlaubt. Einige Befehle ermöglichen es, mehrere Texte gleichzeitig in den Puffer zu stellen.

Aus dem Textmodus zurück in den Befehlsmodus gelangen Sie durch Eingabe eines Punktes (.) als erstes Zeichen einer Zeile. Als Bestätigung des erfolgten Wechsels wird die letzte Textzeile am Bildschirm angezeigt.

Geben Sie beim Aufruf des Editors die Option - an, unterbleibt die Anzeige der Anzahl Zeichen in der Textdatei, die sonst erfolgt.

### Adressierung

Der Text im Editierungspuffer ist in fortlaufend numerierte Zeilen aufgeteilt, die durch <CR> (neue Zeile) voneinander abgegrenzt sind. Das Programm ändert automatisch die Numerierung, wenn Textzeilen eingefügt oder gelöscht werden. Der Editor führt stets eine sogenannte „aktuelle Zeile", d. i. die zuletzt von einem Befehl betroffene Zeile.

Die zur Verfügung stehenden Befehle enthalten einen Adreßbereich, gefolgt von dem aus einem Zeichen bestehenden Befehl, an den sich noch Parameter anschließen können.

Der Adreßbereich eines Befehls besteht aus einer oder zwei vorangestellten Zeilennummern, Adressen genannt. Sie geben an, auf welche Zeilen des im Puffer gespeicherten Textes der Befehl sich bezieht.
Ist nur eine Zeilennummer angegeben, so gilt der Befehl nur für die betreffende Textzeile. Zwei Zeilennummern beziehen sich auf einen Zeilenbereich des Textes, der die beiden adressierten Zeilen einschließt.
Werden mehr Adressen angegeben als der Befehl akzeptiert, berücksichtigt das Programm höchstens die beiden letzten. Geben Sie keine Adresse an, gilt der Befehl immer für die aktuelle Zeile.

Einige Befehle enthalten keinen Adreßbereich und reagieren auf die Eingabe von Zeilennummern mit einer Fehlermeldung.

ed, red

Adressen werden durch Kommata voneinander getrennt. Wird als Trennzeichen ein Semikolon verwendet, interpretiert das Programm den Befehl sequentiell zuerst für die erste und anschließend für die zweite Adresse.

Zu beachten ist, daß bei der Angabe von zwei Adressen die niedrigere Zeilennummer immer zuerst genannt werden muß.

Im folgenden sind die wesentlichen Adressierungsregeln zusammengefaßt:

1. Der Punkt (.) adressiert die aktuelle Zeile.
2. Das '$'-Zeichen adressiert die letzte Zeile im Puffer.
3. Eine Dezimalzahl adressiert die $n$-te Zeile im Puffer.
4. $x$ adressiert die Zeilen, die zuvor mit $x$ gekennzeichnet wurden. Das für $x$ stehende Zeichen muß ein Kleinbuchstabe sein.
5. Ein regulärer Ausdruck, der von Schrägstrichen eingeschlossen ist, adressiert eine Zeile mit einer höheren Zeilennummer als die aktuelle Zeile und einer durch den regulären Ausdruck beschriebenen Zeichenkette. Dies ist die nächsthöhere passende Zeile gegenüber der aktuellen Zeile.
6. Ein in Fragezeichen eingeschlossener regulärer Ausdruck adressiert eine Zeile mit einer niedrigeren Zeilennummer als die aktuelle Zeile und einer durch den regulären Ausdruck beschriebenen Zeichenkette.
7. Eine Adresse, gefolgt von einem Plus- oder Minuszeichen, an das sich eine Dezimalzahl anschließt, entspricht der in der Adresse genannten Zeile plus oder minus der mit der Dezimalzahl angegebenen Anzahl Zeilen. Das Pluszeichen kann weggelassen werden.
8. Beginnt eine Adresse mit einem Plus- oder Minuszeichen, erfolgt die Addition bzw. Subtraktion von der aktuellen Zeile aus.
9. Endet eine Adresse mit einem Plus- oder Minuszeichen, wird von der aktuellen Zeile aus 1 addiert bzw. subtrahiert. Mehrere Plus- oder Minuszeichen bewirken eine ihrer Anzahl entsprechende Addition bzw. Subtraktion.
10. In Adressen haben die Zeichen '^' und '-' dieselbe Bedeutung.

## ed, red

### Regeln für die Befehlsnotation

Beim Standard-Editor werden in begrenztem Umfang in Adressen reguläre Ausdrücke zur Spezifikation von Textzeilen verwendet. Im folgenden sind die Regeln hierfür beschrieben:

1. Jedes Zeichen, ausgenommen die Sonderzeichen, steht für sich selbst. Sonderzeichen sind das Abgrenzungszeichen des regulären Ausdrucks, '\', '[', '.' und manchmal '~', '*' und '$'.
2. Ein Punkt steht für irgendein beliebiges Zeichen.
3. Ein '\', gefolgt von irgendeinem beliebigen Zeichen (auch Sonderzeichen), ausgenommen Ziffern oder runden Klammern, steht für das betreffende Zeichen.
4. Eine in eckige Klammern gesetzte nicht-leere Zeichenkette steht für irgendein beliebiges Zeichen in dieser Zeichenkette. Ein '~' als erstes Zeichen bedeutet, daß alle Zeichen, außer den in der Zeichenkette angegebenen, gemeint sind.
   Die rechte eckige Klammer beendet die Zeichenkette nicht, wenn sie als erstes Zeichen angegeben wird. Z. B. betrifft das Suchmuster [ ]a-f] die rechte eckige Klammer sowie die Buchstaben a-f. Ein '\' hat keine besondere Bedeutung.
5. Ein regulärer Ausdruck der Form 1.-4., gefolgt von einem '*', steht für eine beliebige Wiederholung des regulären Ausdrucks.
6. Teile des Suchmusters können durch '\(...\)' geklammert werden.
7. Ein '\', gefolgt von einer Ziffer $n$, ist eine Abkürzung für eine Wiederholung eines in '\(...\)' eingeklammerten regulären Ausdrucks. Die Ziffer $n$ gibt an, welche Klammer im aktuellen Ausdruck, von links gezählt, gemeint ist.
8. Ein regulärer Ausdruck $x$ der Form 1.-8., gefolgt von einem regulären Ausdruck $y$ der Form 1.-7., erkennt die längste mögliche Form von $x$, solange $y$ auch noch getroffen wird.
9. Ein regulärer Ausdruck der Form 1.-8., dem entweder ein '~' vorausgeht oder ein '$' folgt, steht für den Anfang oder das Ende einer Zeile.

10. Ein regulärer Ausdruck der Form 1.-9. erkennt die längste, am weitesten links stehende Zeichenkette einer Zeile.
11. Falls der reguläre Ausdruck fehlt, wird der letzte vorangehende reguläre Ausdruck genommen.

In dem Begriff „Zeichen" sind in der o. a. Beschreibung alle möglichen Zeichen, mit Ausnahme von Zeilenende, enthalten.

E



### Die Befehle des Standard-Editors

a Mit diesem Befehl können Sie Text hinter eine adressierte Zeile anfügen (append). Die aktuelle Zeile ist im Anschluß daran entweder die letzte Eingabezeile oder – falls keine Eingabe erfolgt ist – die zuvor adressierte Zeile.

c Mit dem Befehl c (change) können Sie mehrere Textzeilen gegen andere austauschen. Dabei werden die in der Adresse angegebenen „alten" Zeilen gelöscht. Die aktuelle Zeile ist dann die letzte Eingabezeile oder – falls keine Eingabe erfolgt ist – die letzte Zeile vor dem gelöschten Textbereich.

d Mit dem Befehl d (delete) können Sie Textzeilen im Editierpuffer löschen. Die Zeile, die unmittelbar dem gelöschten Textbereich folgt, wird die aktuelle Zeile. Standen die gelöschten Zeilen am Ende des Textes, wird die neue letzte Zeile die aktuelle Zeile.

e*Datei* Durch diesen Befehl können Sie den Pufferinhalt löschen und durch den Inhalt der angegebenen Datei ersetzen. Die aktuelle Zeile ist die letzte Pufferzeile; die Anzahl der eingelesenen Zeichen wird angezeigt.
Der angegebene Dateiname wird für einen möglichen r- (read) oder w- (write) Befehl als Standard-Dateiname gespeichert. Geben Sie keinen Dateinamen an, bleibt der Name der ursprünglich im Puffer gespeicherten Textdatei bestehen.
Geben Sie statt eines Dateinamens ein Ausrufungszeichen an, wird der Rest der Eingabezeile als Shell-Kommando interpretiert und ausgeführt. Die Ausgabe dieses Kommandos wird in den Puffer geschrieben. Ein solches Shell-Kommando bleibt nicht als Dateiname erhalten.

ed, red

E*Datei* Wie e, mit der Ausnahme, daß keine Fehlermeldung erfolgt, wenn nach der letzten Änderung der Text nicht mit w gesichert wird.

f*Datei* Mit dem Befehl f (file) wird der zuletzt gespeicherte Dateiname angezeigt. Geben Sie bei diesem Befehl einen Dateinamen ein, wird dieser anstelle des alten Names gespeichert.

g/*Regulärer Ausdruck*/*Befehlsliste*
Bei dem globalen Befehl g werden zunächst die zu dem angegebenen regulären Ausdruck passenden Zeilen gekennzeichnet und anschließend Zeile für Zeile die Befehle der Befehlsliste ausgeführt.
Ein einzelner Befehl oder der erste von mehreren Befehlen der Befehlsliste muß auf derselben Zeile wie der globale Befehl g erscheinen.
Im Falle einer Mehrzeile-Liste müssen sämtliche Zeilen – mit Ausnahme der letzten – mit einem Backslash (\) beendet werden.
Die Befehle a, i und e sowie zugeordnete Eingaben sind erlaubt. Der Punkt, mit dem der Eingabemodus beendet wird, kann auf der letzten Zeile der Befehlsliste entfallen.
Die Befehle g und v dürfen in der Befehlsliste nicht verwendet werden.

G/*regulärer Ausdruck*
Dieses Kommando handelt interaktiv. Im ersten Schritt werden alle Zeilen markiert, auf die der angegebene reguläre Ausdruck zutrifft. Dann wird die erste dieser markierten Zeilen am Bildschirm angezeigt, gleichzeitig wird sie zur aktuellen Zeile. Sie haben jetzt die Möglichkeit, irgendein Kommando (außer a, c, i, g, G, v oder V) anzugeben, welches ausgeführt wird. Anschließend wird die nächste markierte Zeile angezeigt usw. Durch Eingabe eines & wird das letzte ausgeführte Kommando rückgängig gemacht.
Beachten Sie, daß Sie in diesem Kommando-Modus jeden Teil der Datei adressieren und bearbeiten können.
Das Kommando G wird durch Eingabe eines Abbruchsignals verlassen.

h Das Kommando h (help) gibt Ihnen Erklärungen zu den häufigsten Fehlermeldungen.

## ed, red

H — Mit diesem Kommando stellt ed Ihnen einen Modus zur Verfügung, der Ihnen bei allen folgenden Fehlermeldungen Erklärungen liefert. Sind vor Aufruf dieses Kommandos bereits Fehlermeldungen erfolgt, werden diese ebenfalls nachträglich erklärt.

i — Mit dem Befehl i (insert) können Sie Text vor der angegebenen Zeile einfügen. Die letzte Eingabezeile oder, wenn keine Eingabe erfolgt ist, die Zeile vor der adressierten Zeile wird die aktuelle Zeile.
Dieser Befehl unterscheidet sich von dem Befehl a nur dadurch, daß der Text an anderer Stelle eingefügt wird.

j — Mit diesem Befehl (join) werden die angegebenen Zeilen zu einer einzigen Zeile zusammengefaßt; dazwischenliegende Zeilenende-Zeichen werden gelöscht.

k*x* — Der Markierungsbefehl kennzeichnet die adressierten Zeilen mit dem gewünschten Buchstaben; *x* muß als Kleinbuchstabe angegeben werden.

l — Mit dem Befehl l (list) werden die angegebenen Textzeilen, einschließlich nichtdarstellbarer Zeichen, am Bildschirm angezeigt. Die nichtdarstellbaren Zeichen werden dabei als zweistellige Oktalzahl ausgegeben und zu lange Zeilen werden in der nächsten Zeile fortgesetzt. Das Kommando l kann mit jedem anderen Kommando kombiniert werden, außer mit e, f, r oder w.

m*a* — Mit dem Befehl m (move) können Sie angegebene Textzeilen hinter die mit *a* adressierte Zeile verschieben. Diese darf nicht im Bereich der zu verschiebenden Zeilen liegen. Die letzte umgestellte Zeile wird die aktuelle Zeile.

n — Dieses Kommando zeigt Ihnen die adressierten Zeilen am Bildschirm an. Jeder Zeile wird ihre Zeilennummer und ein Tabulatorzeichen vorangestellt. Die letzte angezeigte Zeile wird die aktuelle Zeile. Der n-Befehl darf in derselben Zeile zusammen mit anderen Kommandos, außer e, f, r und w, stehen.

p — Mit dem Befehl p können Sie Textzeilen am Bildschirm ausgeben. Die letzte angezeigte Zeile wird die aktuelle Zeile. Der Befehl p darf in derselben Zeile zusammen mit anderen Befehlen, außer e, f, r und w, stehen.

## ed, red

P Wie p.

q Mit dem Befehl q (quit) beenden Sie das Editor-Programm. Es erfolgt kein automatisches Schreiben (w) der Datei aus dem Puffer.

Q Wie q, mit der Ausnahme, daß keine Fehlermeldung erfolgt, wenn nach der letzten Änderung des im Puffer stehenden Textes das Programm ohne den Befehl w beendet wird.

r*Datei* Mit dem Befehl r (read) können Sie den Text der angegebenen Datei hinter die adressierte Zeile einlesen. Geben Sie keinen Dateinamen an, greift das Programm auf die unter dem gespeicherten Namen angelegte Datei zu (s. Befehle e und f). Der bestehende Dateiname wird nicht geändert, es sei denn, *Datei* ist die erste angesprochene Datei seit dem Editoraufruf. Ist die Einspielung der gewünschten Textdatei abgeschlossen, wird die Anzahl der eingelesenen Zeichen am Bildschirm angezeigt. Die letzte eingelesene Zeile wird die aktuelle Zeile. Geben Sie statt einer Datei ein Ausrufungszeichen an, wird der Rest der Zeile als Shell-Kommando interpretiert und ausgeführt. Die Ausgabe dieses Kommandos erfolgt in den Puffer.

s/*regulärer Ausdruck*/*Zeichenkette*
s/*regulärer Ausdruck*/*Zeichenkette*/g
Der Ersetzungsbefehl durchsucht die in den Adressen angegebenen Zeilen auf Vorhandensein des *regulären Ausdrucks*. Überall dort, wo dies der Fall ist, wird er durch *Zeichenkette* substituiert, wenn der globale Indikator g hinter dem Befehl steht.
Falls in keiner der adressierten Zeilen eine Substitution erfolgt, reagiert ed mit einer Fehlermeldung.
Als Grenzzeichen für reguläre Ausdrücke und die Zeichenkette darf jedes beliebige Zeichen – ausgenommen Leerstellen und Zeilenende-Zeichen – anstelle des Schrägstrichs (/) verwendet werden. Die letzte substituierte Zeile wird die aktuelle Zeile.
Ein & in *Zeichenkette* wird durch die Zeichenkette ersetzt, auf die der reguläre Ausdruck wirkt. Die besondere Bedeutung, die das Zeichen & in diesem Zusammenhang besitzt, kann durch einen vorangestellten Backslash (\) aufgehoben werden.

Die Zeichen \$n$ ($n$ ist eine Ziffer) werden durch den Text ersetzt, auf den der $n$-te in '\(...\)' gesetzte reguläre Ausdruck wirkt.
Wenn in Klammern gesetzte geschachtelte Teilausdrücke vorhanden sind, wird $n$ durch Zählen der vorkommenden Klammerungen von links nach rechts bestimmt.
Zeilen können durch Zeilenende-Zeichen unterteilt werden. Dem Zeilenende-Zeichen in der Ersatzzeichenkette muß ein Backslash vorausgehen.

t$a$ Dieser Befehl wirkt wie der Befehl m, mit der Ausnahme, daß die adressierten Zeilen hinter die mit $a$ angegebene Zeile kopiert werden ($a$ kann 0 sein). Die letzte kopierte Zeile wird die aktuelle Zeile.

u Mit dem Befehl u (undo) können Sie den letzten Befehl in der aktuellen Zeile rückgängig machen.

v/*regulärer Ausdruck*/*Befehlsliste*
Dieser Befehl wirkt wie der globale Befehl g, mit der Ausnahme, daß die Befehlsliste global für alle Zeilen, außer für diejenigen, auf die der angegebene reguläre Ausdruck zutrifft, ausgeführt wird.

V/*regulärer Ausdruck*/
Dieser Befehl wirkt wie der interaktive globale Befehl G, mit der Ausnahme, daß alle die Zeilen markiert werden, auf die der angegebene reguläre Ausdruck nicht zutrifft.

w*Datei* Mit dem Befehl w (write) können Sie die adressierten Zeilen sichern, d. h. in die angegebene Datei schreiben. Ist die genannte Datei nicht vorhanden, wird sie unter dem Modus 666, d. h. mit Schreib- und Leserecht für jeden Benutzer, angelegt. Dieser Dateiname wird gespeichert, falls noch kein Name vorhanden war.
Geben Sie keinen Dateinamen an, greift das Programm auf die unter dem etwaigen gespeicherten Dateinamen vorhandene Datei zu (s. Befehle e und f).
Ist der Befehl ausgeführt, wird die Anzahl der geschriebenen Zeichen am Bildschirm angezeigt.

($)= Die Zeilennummer der adressierten Zeile wird angezeigt. Die aktuelle Zeile bleibt dieselbe.

## ed, red

!*Kommando*
: Die Zeichenkette hinter ! wird als Shell-Kommando interpretiert und ausgeführt. Geben Sie als erstes Zeichen des Kommandonamens ein ! an, wird das Ausrufezeichen durch das zuletzt aufgerufene Shell-Kommando ersetzt. Durch !! wird also die Ausführung des letzten Kommandos wiederholt.

Neue-Zeile-Zeichen
: Dieses Zeichen allein in einer Zeile ist gleich der Eingabe von .+1p.

Geben Sie ein Unterbrechungssignal ein, antwortet ed mit einem Fragezeichen und kehrt in den Kommando-Modus zurück.

DATEIEN:

| | |
|---|---|
| /tmp/e# | Temporäre Datei; # ist die Prozeßnummer |
| ed.hup | Sicherungsdatei |

SIEHE AUCH:

grep, sed, sh, stty

## edit

edit – Texteditor (Variante von ex für gelegentliche Benutzer)

SYNTAX:

edit [-r] *Name* ...

BESCHREIBUNG:

Edit ist eine Variante des Texteditors ex und kann von neuen oder ungeübten Benutzern verwendet werden, die mit einem kommandoorientierten Editor arbeiten möchten. Die folgende kurze Einführung genügt schon, um die Arbeit mit edit beginnen zu können. Steht Ihnen jedoch ein Bildschirmterminal zur Verfügung, ist es unter Umständen vorteilhafter, mit dem Bildschirmeditor vi zu arbeiten.

KURZE EINFÜHRUNG:

Wollen Sie den Inhalt einer vorhandenen Datei bearbeiten, geben Sie zunächst das Kommando

edit *Dateiname*

ein. Edit legt eine Kopie der aufgerufenen Datei an, die vom Benutzer überarbeitet werden kann, und teilt dabei mit, wie viele Zeilen und Zeichen in der Datei enthalten sind. Wollen Sie eine neue Datei anlegen, denken Sie sich einen Namen für die Datei aus und versuchen Sie, diese mit edit zu bearbeiten; dies führt zu einer Fehlermeldung, die Sie aber einfach ignorieren.

Mit dem Prompt-Zeichen ':' verlangt edit Kommandos. Dieses Aufforderungszeichen müßte nach dem Starten des Editors zu sehen sein. Wenn Sie eine vorhandene Datei editieren, stehen einige Zeilen im Editorpuffer (der von edit vergebene Name für die Kopie der zu bearbeitenden Datei). Die meisten Kommandos beziehen sich auf die „aktuelle Zeile", wenn Sie dem Editor nicht mitteilen, welche Zeile zu bearbeiten ist. Geben Sie also den Befehl print (kann zu p abgekürzt werden) ein (alle Kommandos an edit müssen mit Zeilenschaltung bestätigt werden), wird diese aktuelle Zeile gedruckt. Löschen Sie die aktuelle Zeile mit d (delete), gibt edit die neue aktuelle Zeile aus. Beginnen Sie mit

edit

dem Editieren, ist die letzte Zeile in der Datei zunächst die aktuelle Zeile. Wird diese letzte Zeile gelöscht, wird die neue letzte Zeile zur aktuellen Zeile. Grundsätzlich gilt, daß nach dem Löschen einer Zeile die nächste Zeile in der Datei die aktuelle Zeile wird. (Das Löschen der letzten Zeile ist ein Sonderfall.)

Beginnen Sie mit einer leeren Datei oder wollen in eine bestehende Datei Zeilen einfügen, geben Sie das Kommando a (append – Anfügen) ein. Nach Eingabe dieses Kommando liest edit Zeilen von Ihrem Terminal, bis eine Zeile erscheint, auf der nur ein Punkt (.) steht. Die neuen Zeilen werden hinter die aktuelle Zeile gestellt. Die letzte eingegebene Zeile wird dann die aktuelle Zeile. Das Kommando i (insert – Einfügen) wirkt ähnlich wie a (append), stellt die eingegebenen Zeilen jedoch vor und nicht hinter die aktuelle Zeile.

Edit numeriert die Zeilen im Puffer von 1 aufwärts. Geben Sie das Kommando 1 ein, gibt edit die erste Zeile aus. Wird dann der Löschbefehl d eingegeben, löscht edit die erste Zeile, Zeile 2 wird zu Zeile 1, und edit gibt die aktuelle Zeile (die neue Zeile 1) aus, so daß Sie sehen, wo Sie sich befinden. Grundsätzlich ist die aktuelle Zeile die letzte von einem Kommando beeinflußte Zeile.

Wollen Sie Text in der aktuellen Zeile ändern, so verwenden Sie das Kommando s (substitute – Ersetzen). Sie schreiben s/*alt*/*neu*/, wobei *alt* für die alten zu ersetzenden Zeichen und *neu* für die neuen einzusetzenden Zeichen steht.

Das Kommando f (file – Datei) gibt Auskunft darüber, wieviele Zeilen der Editorpuffer enthält. Die Meldung „[Modified]" wird ausgegeben, wenn der Puffer geändert wurde. Nach dem Ändern einer Datei können Sie mit dem Kommando w (write – Zurückschreiben) den Pufferinhalt in die Datei zurückschreiben, so daß diese geändert ist. Anschließend können Sie die Editorsitzung mit dem Kommando q (quit – Verlassen) beenden. Haben Sie eine Datei mit edit aufgerufen aber nicht verändert, ist es nicht notwendig, obgleich unschädlich, sie zurückzuschreiben. Versuchen Sie den Editor nach dem Ändern des Puffers zu verlassen, ohne den Puffer zurückzuschreiben, erscheint die Warnmeldung „No write since last change" (Seit letzter Änderung nicht zurückgeschrieben), und edit erwartet ein weiteres Kommando. Soll der Puf-

edit

ferinhalt nicht zurückgeschrieben und die Datei somit nicht geändert werden, müssen Sie ein weiteres Kommando q (quit) absetzen. Damit ist der Pufferinhalt unwiderruflich verloren, und Sie befinden sich wieder in der Kommandoebene der Shell.

Mit Hilfe der Kommandos zum Löschen (d) und Anfügen (a) und durch Angabe der Zeilennummern zum Ausgeben der Zeilen in der Datei kann der Benutzer alle gewünschten Änderungen vornehmen. Einige weitere Funktionen sollten Sie jedoch kennen, wenn Sie des öfteren mit edit arbeiten.

Das Kommando c (change – Ändern) ändert die aktuelle Zeile in eine Folge von Zeilen, die der Benutzer eingibt (wie beim Kommando a werden diese Zeilen durch eine Zeile, die nur aus einem Punkt besteht, geändert). Sie können im Kommando c mehrere zu ändernde Zeilen angeben. Schreiben Sie dazu die Nummern der zu ändernden Zeilen vor das Kommando, z. B. „3,5c". Nach demselben Prinzip können mehrere Zeilen ausgegeben werden. Das Kommando „1,23p" gibt die ersten 23 Zeilen der Datei aus.

Mit dem Kommando u (undo – Rückgängigmachen) können Sie die Wirkung des letzten Kommandos, das den Puffer geändert hat, wieder rückgängig machen. Geben Sie beispielsweise einen Befehl zum Ersetzen ein, stellen dann aber fest, daß dies doch nicht richtig war, geben Sie einfach das Kommando u (undo) ein, und schon steht die Zeile in der ursprünglichen Form wieder da. Selbst die Wirkung des Kommandos u kann wieder ungeschehen gemacht werden. Beeinflußt ein Kommando mehrere Zeilen im Puffer, gibt edit eine Warnmeldung aus. Wenn der Umfang der Änderungen unangemessen groß erscheint, ist es sinnvoll, mit dem Kommando u die Wirkung wieder rückgängig zu machen und noch einmal zu überprüfen. Ist die Änderung in Ordnung, können Sie erneut das Kommando u eingeben, so daß die Änderung doch durchgeführt wird. Kommandos wie w (Wegschreiben) und q (Verlassen) können nicht mit u rückgängig gemacht werden.

Wollen Sie sich die nächste Zeile im Puffer ansehen, drücken Sie einfach die Zeilenschaltungstaste. Wollen Sie mehrere Zeilen sehen, halten Sie die Taste CTRL nieder, betätigen gleichzeitig die Taste D und lassen dann beide los. Dadurch werden auf einem Bildschirm ein hal-

## edit

ber Bildschirminhalt und auf einem Druckerterminal 12 Zeilen ausgegeben. Sie können sich den Text in der Umgebung Ihrer aktuellen Position anschauen, indem Sie das Kommando z. eingeben. Die letzte ausgegebene Zeile ist dann die aktuelle Zeile. Mit dem Kommando " kommen Sie zu der Zeile zurück, in der Sie sich vor Ausführung des Kommandos z. befanden. Hinter z können auch andere Zeichen eingegeben werden: z- gibt einen Bildschirminhalt (bzw. 24 Zeilen) mit Text aus, die mit der aktuellen Zeile enden; z+ gibt den nächsten Bildschirminhalt aus. Wenn weniger als ein Bildschirminhalt ausgegeben werden soll, geben Sie beispielsweise z.12 ein, um insgesamt 12 Zeilen zu bekommen. Diese Methode der Wiederholungsfaktoren ist allgemein anwendbar. Beispielsweise können fnf Zeilen ab der aktuellen Zeile mit dem Kommando delete 5 gelöscht werden.

Wollen Sie eine bestimmte Stelle in der Datei ansteuern, können Sie die entsprechende Zeilennummer eingeben. Da sich die Zeilennummern beim Einfügen und Löschen von Zeilen ändern, ist diese Methode allerdings nicht immer sehr zielgenau. Zeichenfolgen können in der Datei in Vorwärts- oder in Rückwärtsrichtung gesucht werden. Das Kommando */Text/* sucht vorwärts nach dem angegebenen Text, und das Kommando ?*Text*? sucht rückwärts. Wird der gesuchte Text nicht gefunden, läuft der Prozeß zyklisch weiter bis zur aktuellen Zeile. Eine nützliche Erweiterung ist ein Suchbefehl in der Form /~*Text*/, der nach dem Text am Anfang einer Zeile sucht. Analog dazu sucht /*Text*$/ nach dem Text am Ende einer Zeile. Bei diesen Kommandos kann das abschließende Zeichen / bzw. ? weggelassen werden.

Die aktuelle Zeile hat den symbolischen Namen „.". Dies ist bei einer Folge von Zeilen sehr nützlich. Das Kommando „.,$print" gibt beispielsweise die restlichen Zeilen in der Datei aus. Die letzte Zeile in der Datei wird mit dem symbolischen Namen „$" angesprochen. Das Kommando „$d" löscht beispielsweise die letzte Zeile in der Datei, gleichgültig in welcher Zeile Sie sich vorher befanden. Die Zeilenadressen können auch durch arithmetische Ausdrücke spezifiziert werden. Die Zeile „$-5" ist beispielsweise die fünftletzte Zeile, und „.+20" ist die zwanzigste Zeile nach der aktuellen Zeile.

## edit

Wollen Sie wissen, in welcher Zeile Sie sich gerade befinden, geben Sie „=" ein. Dies ist wichtig, wenn Sie einen Textabschnitt innerhalb einer Datei oder zwischen mehreren Dateien versetzen oder kopieren wollen. Dazu stellen Sie die erste und letzte Zeilennummer des zu kopierenden bzw. zu versetzenden Abschnitts fest (z. B. 10 bis 20). Um die Zeilen zu versetzen, geben Sie den Befehl "10,20delete a" ein. Dadurch werden diese Zeilen aus der Datei gelöscht und in einem Puffer mit dem Namen a gespeichert. Edit besitzt 26 benannte Puffer mit den Namen a bis z. Später können Sie diese Zeilen wieder holen, indem Sie den Pufferinhalt mit „put a" hinter die aktuelle Zeile stellen. Wollen Sie eine Reihe von Zeilen von einer Datei in eine andere versetzen oder kopieren, geben Sie nach dem Kopieren der Zeilen den Befehl e (edit) und spezifizieren dahinter den Namen der anderen zu bearbeitenden Datei, z. B. „edit Kapitel2". Wenn Sie den Befehl delete in den bisherigen Ausführungen durch yank (ohne Löschung in Puffer stellen) ersetzen, können Sie Zeilen kopieren, d. h. die Zeilen bleiben an der ursprünglichen Stelle ebenfalls erhalten. Befindet sich der zu versetzende bzw. zu kopierende Text in einer einzigen Datei, können Sie beispielsweise einfach „10,20move $" eingeben. In diesem Fall brauchen Sie keine benannten Puffer benutzen (obwohl dies durchaus möglich ist).

SIEHE AUCH:

ex
Systemliteratur TARGON /35: „Einführung in den Editor vi"

## enable, disable

enable, disable – Drucker-Status auf betriebsbereit bzw. nicht betriebsbereit setzen

SYNTAX:

enable *Drucker* ...

disable [-c] [-r[*Grund*]] *Drucker* ...

BESCHREIBUNG:

Enable aktiviert die angegebenen Drucker, damit durch lp abgesetzte Aufträge gedruckt werden können.

Disable deaktiviert die angegebenen Drucker. Durch lp vergebene Druckaufträge werden von den Druckern nicht mehr verarbeitet. Aufträge, die gerade bearbeitet werden, werden nochmals vollständig auf dem gleichen Drucker oder einem anderen Drucker der gleichen Klasse gedruckt.

Folgende Optionen können Sie beim Aufruf von disable angeben:

-c Löscht alle Aufträge, die für einen der angegebenen Drucker bestimmt sind.

-r[*Grund*] Hinweis auf den Grund der Deaktivierung. Eine -r-Option ist gültig bis zur nächsten -r-Option in der Kommandozeile. Geben Sie -r nicht oder ohne *Grund* an, wird ein Standardtext verwendet. *Grund* wird beim Aufruf von lpstat mit ausgegeben.

DATEIEN:

/usr/spool/lp/*

SIEHE AUCH:

lp, lpstat

## env

env – Änderung der Umgebung bei Ausführung von Kommandos

SYNTAX:

env [-] [*Name=Wert*] ... [*Kommando ...*]

BESCHREIBUNG:

Env führt das als Parameter angegebene Kommando in einer erweiterten oder veränderten Umgebung aus. Argumente in der Form *Name=Wert* werden zu der ursprünglichen Umgebung hinzugefügt bzw. überlagern den ursprünglichen Wert von *Name*, bevor das Kommando ausgeführt wird.
Die Option '-' bewirkt, daß die ursprüngliche Umgebung vollkommen ignoriert wird, so daß das Kommando in der spezifizierten Umgebung ausgeführt wird.

Rufen Sie env ohne Argumente auf, wird die bestehende Umgebung angezeigt. Jedes Name=Wert-Paar wird in einer Zeile angedruckt.

SIEHE AUCH:

sh

## eqn, neqn, checkeq

eqn, neqn, checkeq – Preprozessor für nroff/troff

SYNTAX:

eqn [-d*xy*] [-p*n*] [-s*n*] [-f*n*] [*Dateien*]

neqn [-d*xy*] [-p*n*] [-s*n*] [-f*n*] [*Dateien*]

checkeq [*Dateien*]

BESCHREIBUNG:

Eqn ist ein Preprozessor für troff, während neqn die Ausgabe für nroff aufbereitet. Beide Programme arbeiten gleich und erlauben dem Benutzer, in einer recht einfachen Form mathematische Formeln zu schreiben. Im nachfolgenden Text steht eqn synonym für beide Programme. Der Aufruf dieser Preprozessoren erfolgt folgendermaßen:

eqn *Dateien* | troff
neqn *Dateien* | nroff

Geben Sie keine Dateien oder nur - an, liest eqn aus der Standardeingabe. Anweisungen, die von eqn bearbeitet werden sollen, sind entsprechend zu kennzeichnen. Dies geschieht in einer Textdatei folgendermaßen:

.EQ
*Formel_Text*
.EN

Der Preprozessor analysiert die zwischen .EQ und .EN stehenden Zeilen und erzeugt danach Anweisungen für den Textformatierer. Dabei werden die Zeilen mit .EQ und .EN ebenfalls weitergereicht. Alle Angaben zwischen diesen beiden Kommandos werden als **eine** Formel betrachtet, die im Normalfall eine eigene Zeile einnimmt. Wollen Sie Formeln mitten in einer Textzeile verwenden, können Sie entweder gleich beim Aufruf des Preprozessors mit der Option -d*xy*, oder zwischen .EQ und .EN mit delim *xy*, Begrenzungszeichen definieren. Der Text zwischen den Begrenzungszeichen wird dann als eqn-Eingabe erkannt. Die Begrenzungszeichen dürfen gleich sein, häufig wird das $-Zeichen genommen. Durch delim off können Sie die Begrenzungszeichen wieder außer Kraft setzen.

## eqn, neqn, checkeq

Das Programm checkeq meldet fehlende oder inkonsistente Angaben von Begrenzungszeichen und .EQ/.EN-Paaren.

Zeichengröße und Zeichensatz können Sie durch Angabe von size *n* oder size ±*n*, roman, italic, bold und font *n* ändern. Diese Änderungen gelten nur für das nachfolgende Element. Sollen die Änderungen für alle nachfolgenden Formeln des Dokuments gelten, so verwenden Sie entweder gsize *n* oder gfont *n* oder geben beim Preprozessor-Aufruf die Optionen -s*n* und -f*n* an.

Durch die Option -p*n* können Sie den Zeichengrößen-Unterschied zwischen vorgegebener Größe und Zeichengröße von höher und tiefer gestelltem Exponent bzw. Index beeinflussen (Standard -3).

SIEHE AUCH:

nroff, tbl

# ex

ex – Texteditor

SYNTAX:

ex [-] [-v] [-t*Marke*] [-r] [-R] [+*Kommando*] [-x] *Name* ...

BESCHREIBUNG:

Ex ist eine Erweiterung des UNIX-Standardeditors ed. Die wichtigste Erweiterung ist die Möglichkeit des bildschirmorientierten Editierens.

FÜR ED-BENUTZER:

Wenn Sie bisher mit ed gearbeitet haben, werden Sie feststellen, daß ex eine Reihe neuer Funktionen bietet, die besonders gut für Bildschirmterminals geeignet sind. Der Editor ex nutzt die Leistungsmöglichkeiten von Terminals wesentlich besser aus als ed und stellt aufgrund der Terminalfunktionsdatei terminfo und des in der Variablen TERM in der Umgebung angegebenen Terminaltyps fest, wie das betreffende Terminal am günstigsten anzusteuern ist.

Ex enthält eine Reihe neuer Funktionen, die das Sichtbarmachen des Textes einer Datei erleichtern. Mit dem Kommando z können Sie gezielt auf Textbereiche (Fenster) zugreifen. Das Kommando CTRL-D veranlaßt den Editor, den Text um ein halbes Fenster vorwärts zu rollen. Dieser Befehl ist zum schnellen Durchsehen einer Datei günstiger als die Zeilenschaltungstaste. Im bildschirmorientierten Editiermodus hat man selbstverständlich ständig den zu editierenden Kontext im Blick.

Ex unterstützt Sie in wesentlich verbesserter Weise, wenn Ihnen Fehler unterlaufen. Mit dem Kommando u (undo – Rückgängig machen) können Sie die letzte unerwünschte Änderung rückgängig machen. Außerdem gibt Ihnen ex zahlreiche Rückmeldungen. Geänderte Zeilen werden normalerweise ausgegeben. Ex zeigt auch an, wenn ein Kommando eine größere Anzahl Zeilen beeinflußt. Dies ist ein zusätzliches Maß an Sicherheit, so daß Sie noch einmal überprüfen können, ob tatsächlich so viele Zeilen verändert werden sollen.

## ex

Der Editor verhindert das Überschreiben vorhandener Dateien mit Ausnahme der Dateien, die von Ihnen bearbeitet wurden. Dadurch wird gewährleistet, daß durch das Zurückschreiben nicht versehentlich eine andere Datei als die soeben editierte zerstört wird. Stürzt das System (oder der Editor) ab, kann die Arbeit mit dem Recover-Kommando des Editors wiederhergestellt werden. Dadurch kommen Sie beinahe wieder an die Unterbrechungsstelle zurück, und es gehen höchstens einige wenige Zeilen verloren.

Ex bietet Ihnen mehrere Funktionen, die das gleichzeitige Bearbeiten mehrerer Dateien erleichtern. Sie können in der Kommandozeile eine Liste der gewünschten Dateien angeben und mit dem Kommando n (next) jeweils in die nächste Datei umschalten. Dem Kommando n (next) kann auch eine Liste mit Dateinamen oder eine Namensangabe mit von der Shell benutzten Metazeichen beigegeben werden, um eine neue Gruppe von zu bearbeitenden Dateien zu spezifizieren. Grundsätzlich können die Dateinamen im Editor gemäß der vollständigen Metasyntax der Shell aufgebaut werden. Das Metazeichen % kann ebenfalls zur Bildung von Dateinamen herangezogen werden und wird jeweils durch den Namen der aktuellen Datei ersetzt.

Für das Versetzen von Text innerhalb einer Datei oder von einer Datei zur anderen verfügt der Editor über eine Reihe von benannten Puffern mit den Namen von a bis z. Der Benutzer kann Texte in den benannten Puffern ablegen und diese mit in eine andere Datei nehmen.

Das Kommando & wiederholt das letzte Ersetzungskommando. Außerdem gibt es ein überwachtes Ersetzungskommando. Der Benutzer gibt eine Reihe von Ersetzungswünschen ein, und der Editor fragt jeweils interaktiv ab, ob die betreffende Ersetzung hier gewünscht wird.

Beim Suchen und Ersetzen können Groß- und Kleinbuchstaben gleichbehandelt werden. Ferner können mit ex reguläre Ausdrücke aufgebaut werden, die mit Wörtern übereinstimmen. Dies ist beispielsweise sinnvoll, wenn man das Wort „Marke" sucht und das Dokument auch das Wort „Markenzeichen" enthält.

## ex

Ex bietet eine Reihe von Optionen, mit deren Hilfe Sie den Editor auf Ihre Bedürfnisse zuschneiden können. Eine sehr nützliche Option ist autoindent (automatische Einrückung), bei der der Editor am Zeilenanfang automatisch Leerstellen einfügt, um Absätze einzurücken. Dann kann man die Taste CTRL-D als Rückwärtstab verwenden und mit der Leertaste und Tabs vorwärtsgehen, um neue Eingabedaten in der gewünschten Form anzuordnen.

Es gibt einige weitere nützliche Funktionen. Das intelligente Kommando j (join – Zusammenfügen) setzt beispielsweise das Leerzeichen zwischen zusammengefügten Zeilen automatisch ein, die Kommandos < und > stellen Gruppen von Zeilen um, und Teile des Puffers können mit Kommandos wie etwa sort gefiltert werden.

AUFRUFOPTIONEN:

Die folgenden Aufrufoptionen werden von ex interpretiert:

| | |
|---|---|
| - | Alle interaktiven Rückmeldungen werden unterdrückt. Dies ist sinnvoll bei der Verarbeitung von Editor-Scripts. |
| -v | Ruft vi auf. |
| -t*Marke*R | Die Datei, die die angegebene Marke enthält, bearbeiten, und den Editor auf diese Definition positionieren. |
| -r[*Datei*] | Die angegebene Datei nach einem Absturz des Editors oder Systems wiederherstellen. Geben Sie keine Datei an, wird eine Liste aller gespeicherten Dateien ausgegeben. |
| -R | Lesemodus (readonly). Verhindert das versehentliche Überschreiben einer Datei. |
| +*Kommando* | Mit dem Editieren beginnen, indem das angegebene Such- oder Positionierungskommando des Editors ausgeführt wird. |

Das Argument *Name* gibt zu editierende Dateien an.

## ex

### Modi von ex

| | |
|---|---|
| Kommandomodus | Modus beim Aufruf von ex. Eingaben werden mit mit dem Zeichen : angefordert. Mit dem Kill-Zeichen Ihres Terminals wird ein unvollständiges Kommando storniert. |
| Einfügemodus | Aufruf durch a, i und c. In diesem Modus können Sie beliebigen Text eingeben. Dieser Modus wird normal durch eine Zeile, die nur aus einem Punkt besteht, beendet. Durch Eingabe des Unterbrechungssignals wird er abgebrochen. |
| Visual-Modus | Aufruf durch vi. Beendigung durch Q oder CTRL-\. |

Ex – Kommandonamen und -abkürzungen

| | | | | | |
|---|---|---|---|---|---|
| abbrev | ab | next | n | unabbrev | una |
| append | a | number | nu | undo | u |
| args | ar | | | unmap | unm |
| change | c | preserve | pre | version | ve |
| copy | co | print | p | visual | vi |
| delete | d | put | pu | write | w |
| edit | e | quit | q | xit | x |
| file | f | read | re | yank | ya |
| global | g | recover | rec | window | z |
| insert | i | rewind | rew | escape | ! |
| join | j | set | se | lshift | < |
| list | l | shell | sh | print next | CR |
| map | map | source | so | resubst | & |
| mark | ma | stop | st | rshift | > |
| move | m | substitute | s | scroll | CTRL-D |

## ex

| Ex – Kommandoadressierung | | | |
|---|---|---|---|
| *n* | Zeile *n* | */Muster* | nächste mit *Muster* |
| . | aktuelle | ?*Muster* | letzte mit *Muster* |
| $ | letzte | *x*-*n* | *n* vor *x* |
| + | nächste | *x*,*y* | *x* bis *y* |
| - | vorangehende | 'x | markiert mit *x* |
| +*n* | *n* vorwärts | '' | vorangehender Kontext |
| % | 1,$ | | |

### Initialisierungsoptionen

| | |
|---|---|
| EXINIT | Gesetzte Optionen hier in Umgebungsvariable plazieren. |
| $HOME/.exrc | Editor-Initialisierungsdatei. |
| set *x* | Option aktivieren. |
| set no*x* | Option deaktivieren. |
| set *x*=*Wert* | Werte zuordnen. |
| set | Geänderte Optionen anzeigen. |
| set all | Alle Optionen anzeigen. |
| set *x*? | Den Wert von *x* anzeigen. |

ex

| Wichtige Optionen | | |
|---|---|---|
| autoindent | ai | Automatische Einrückung |
| autowrite | aw | Vor dem Ändern von Dateien zurückschreiben |
| ignorecase | ic | Gleichbehandlung von Groß- und Kleinbuchstaben |
| magic | . [ * | Sonderbedeutung in Suchmustern |
| number | nu | Zeilen numerieren |
| paragraphs | para | Makronamen, die einen neuen Abschnitt beginnen |
| redraw | | Intelligentes Terminal simulieren |
| scroll | | Zeilen scrollen |
| sections | sect | Makronamen, die einen neuen Abschnitt beginnen |
| shiftwidth | sw | Bei < > und Eingabe CTRL-D |
| showmatch | sm | Bei ) und } je nach Eingabe |
| showmode | smd | Einfügemodus im vi anzeigen |
| slowopen | slow | Änderungen beim Einfügen stoppen |
| window | | Zeilen im visual-Modus |
| wrapscan | ws | Puffer bei Ende zyklisch weiter durchsuchen? |
| wrapmargin | wm | Automatische Zeilenschaltung |

| Bildung von Suchmustern | |
|---|---|
| ^ | Zeilenanfang |
| $ | Zeilenende |
| . | Beliebiges Zeichen |
| < | Wortanfang |
| > | Wortende |
| [*String*] | Beliebiges Zeichen in *String* |
| [↑*String*] | Beliebiges Zeichen nicht in *String* |
| [x-y] | Beliebiges Zeichen von x bis y |
| * | Beliebige Wiederholungen der vorangehenden Angaben |

## ex

HINWEISE:

Das Kommando u (undo) löscht alle Markierungen in Zeilen, die geändert und dann wiederhergestellt wurden, sofern die markierten Zeilen verändert wurden.

Durch undo wird der Zustand „Puffer verändert" nicht zurückgesetzt.

Das Kommando z gibt die Anzahl der logischen und nicht der physischen Zeilen aus. Wenn überlange Zeilen vorkommen, kann der Ausgabeumfang über einen vollen Bildschirminhalt hinausgehen.

Bei Datei-Ein-/Ausgabefehlern wird kein Name ausgegeben, wenn Sie die Kommandooption '-' verwenden.

Es gibt keine einfache Methode, einen einzelnen Suchlauf bei Gleichbehandlung von Groß- und Kleinbuchtstaben durchzuführen.

Der Editor gibt keine Warnung aus, wenn Text in benannten Puffern abgelegt und vor dem Verlassen des Editors nicht mehr verwendet wird.

Nullzeichen werden in Eingabedateien unberücksichtigt gelassen und können in Dateien nicht erscheinen.

SIEHE AUCH:

awk, ed, edit, grep, sed, vi
Systemliteratur TARGON /35: „Einführung in den Editor vi"

# expr

expr – Berechnung von Argumenten

SYNTAX:

expr *Argument* ...

BESCHREIBUNG:

Expr errechnet das Ergebnis der angegebenen Argumente. Das Resultat wird auf der Standardausgabe ausgegeben. Jedes angegebene Zeichen wird als separates Argument angenommen.

Die Operatoren und Schlüsselwörter sind nachstehend aufgeführt. Die Liste ist in aufsteigender Prioritätenfolge geordnet, Operatoren mit gleicher Priorität sind zusammengefaßt.

*Ausdruck* | *Ausdruck*
: Gibt den ersten *Ausdruck* zurück, wenn er weder ein Leerstring noch 0 ist, anderenfalls wird der zweite *Ausdruck* zurückgegeben.

*Ausdruck* & *Ausdruck*
: Gibt den ersten Ausdruck zurück, wenn kein Ausdruck einen Leerstring oder 0 ergibt, anderenfalls wird 0 zurückgegeben.

*Ausdruck Zeichen Ausdruck*
: Geben Sie eines der Zeichen <, <=, =, !=, >=, > an und beide Argumente werden als Zahlen angegeben, ist der Vergleich numerisch, sonst lexikalisch.

*Ausdruck* + *Ausdruck*
: Addition der Argumente.

*Ausdruck* - *Ausdruck*
: Subtraktion der Argumente.

*Ausdruck* * *Ausdruck*
: Multiplikation der Argumente.

*Ausdruck* / *Ausdruck*
: Division der Argumente.

*Ausdruck* % *Ausdruck*
: Divisionsrestberechnung (Modulo-Funktion) der Argumente.

## expr

*Ausdruck* : *Ausdruck*
: Der Operator ':' vergleicht das erste Argument mit dem zweiten Argument. Dieses muß ein regulärer Ausdruck sein. Die Syntax regulärer Ausdrücke entspricht der Syntax von ed, außer daß alle Muster (Pattern) mit dem Zeichen '^' beginnen. Daher gilt '^' in diesem Zusammenhang nicht als ein spezielles Zeichen. Standardmäßig gibt diese Option die Anzahl der betroffenen Zeichen zurück. Alternativ können Sie jedoch auch das Pattern-Symbol (...) angeben; dann ist der Rückgabewert ein Teil des ersten Arguments.

(*Ausdrücke*)
: Die Klammern dienen zum Ändern der Ausführungsreihenfolge bei der Bildung von Ausdrücken: Ausdrücke innerhalb der Klammern werden zuerst ausgewertet.

Wenn Sie Operatoren benutzen, die für die Shell Sonderzeichen darstellen, muß diesen ein Backslash (\) vorangestellt werden.

Beispiel:

```
a = 'expr $a + 1'
```

addiert 1 zu der Shell-Variablen a.

RÜCKGABE-CODES:

0 = Wenn der Ausdruck weder ein Leerstring noch 0 ist.
1 = Wenn der Ausdruck ein Leerstring oder 0 ist.
2 = Wenn der Ausdruck ungültig ist.

SIEHE AUCH:

ed, sh

# f77

## f77 – Fortran 77 Compiler

SYNTAX:

*f77 [Optionen] Dateien*

BESCHREIBUNG:

F77 ist der Fortran 77-Compiler des UNIX-Systems. Ihm können verschiedene Arten von Dateiargumenten vorgegeben werden:

Argumente, deren Namen mit .f enden, werden als Fortran 77-Quellprogramme interpretiert; sie werden kompiliert, und die einzelnen Objektprogramme werden im aktuellen Verzeichnis in einer Datei abgelegt, deren Name dem der Quelle entspricht, wobei das .f allerdings durch .o ersetzt ist.

Argumente, deren Namen mit .r oder .e enden, werden als RATFOR- bzw. EFL-Quellprogramme interpretiert. Diese werden zunächst von dem entsprechenden Preprozessor transformiert, dann von f77 kompiliert und in Dateien mit dem Suffix .o abgelegt.

Analog werden Argumente, deren Namen mit .c oder .s enden, als C- bzw. Assembler-Quellprogramme interpretiert und kompiliert bzw. assembliert. Die Objektprogramme werden in Dateien mit dem Suffix .o abgelegt.

Die folgenden Optionen haben dieselbe Bedeutung wie bei cc (Optionen für den Link-Editor siehe ld):

-c Den Binderlauf unterdrücken und für jede Quelldatei eine .o-Datei erzeugen.

-p Objektdateien für das Profilieren vorbereiten (siehe prof).

-O Einen Objektcode-Optimierer aufrufen.

-S Die genannten Programme kompilieren und die Ausgabe in Assemblersprache in entsprechenden Dateien ablegen, deren Namen das Suffix .s bekommen (es werden keine .o-Dateien angelegt).

## f77

-o*Ausgabe*
Der Name der Ausgabedatei soll *Ausgabe* und nicht a.out lauten.

-f Auf Systemen ohne Gleitpunkt-Hardware eine Version von f77 verwenden, die Gleitpunktkonstanten behandelt und das Objektprogramm mit dem Gleitpunkt-Interpreter linkt.

-g Zusätzliche Informationen für sdb erzeugen.

Die folgenden Optionen sind speziell für f77 ausgelegt:

-onetrip DO-Schleifen kompilieren, die mindestens einmal ausgeführt werden, wenn sie erreicht werden. (DO-Schleifen bei Fortran 77 werden nicht ausgeführt, wenn die obere Grenze kleiner ist als die untere Grenze.)

-1 Diese Option hat dieselbe Bedeutung wie -onetrip.

-66 Erweiterungen, die die Fortran 66-Kompatibilität verbessern, unterdrücken.

-C Code für die Index-Bereichsprüfung zur Laufzeit generieren.

-I[24s] Änderung der Standardgröße von Integer-Variablen. Bei Eingabe von -I2 bekommen alle Integers die Größe von zwei Bytes, bei -I4 die Größe von vier Bytes. -Is bewirkt die Änderung der Standardgröße von Subscript-Ausdrücken in zwei Bytes.

-u Der Standardtyp einer Variablen ist undefiniert, und es werden nicht die üblichen Fortran-Regeln angewandt.

-w Alle Warnmeldungen unterdrücken. Die Option -w66 unterdrückt nur die Warnmeldungen, die sich auf die Fortran 66-Kompatibilität beziehen.

-F Den EFL- bzw. RATFOR-Preprozessor auf die entsprechenden Dateien anwenden und das Ergebnis in Dateien ablegen, deren Suffixe in .f abgeändert sind (es werden keine .o-Dateien angelegt).

-m Den M4-Preprozessor auf jede EFL- bzw. RATFOR-Quelldatei anwenden, bevor eine Transformation mit dem Prozessor ratfor bzw. efl vorgenommen wird.

-E Die übrigen Zeichen im Argument werden beim Verarbeiten einer .e-Datei als EFL-Flag-Argument verwendet.

-R Die übrigen Zeichen im Argument werden beim Verarbeiten einer .r-Datei als RATFOR-Flag-Argument verwendet.

Weitere Argumente werden entweder als Optionsargumente des Link-Editors oder als von f77 kompilierbare Objektprogramme (die normalerweise in einem früheren Durchlauf erzeugt wurden) oder als Bibliotheken von Routinen, die von f77 kompilierbar sind, interpretiert. Diese Programme werden zusammen mit den Ausgabedaten ggf. spezifizierter Kompilierungen (in der angegebenen Reihenfolge) gelinkt, so daß ein ausführbares Programm mit dem standardmäßigen Namen a.out entsteht.

DATEIEN:

| | |
|---|---|
| file.[fresc] | Eingabedatei |
| *Datei*.o | Objektdatei |
| a.out | Gelinkte Ausgabe |
| ./fort[*pid*].? | Temporäre Datei |
| /usr/lib/f77pass1 | Compiler |
| /lib/c1 | Pass 2 |
| /lib/c2 | Optimierer |
| /usr/lib/libF77.a | Bibliothek der internen Standardfunktionen |
| /usr/lib/libI77.a | Fortran-E/A-Bibliothek |
| /lib/libc.a | C-Bibliothek |

DIAGNOSE:

Die von f77 ausgegebenen Diagnosemeldungen sind selbsterklärend. Gelegentlich können Meldungen vom Link-Editor ld ausgegeben werden.

SIEHE AUCH:

asa, cc, ld, m4, prof, ratfor, sdb

# factor

factor – Zerlegung einer Zahl in ihre Faktoren

SYNTAX:

factor [*Zahl*]

BESCHREIBUNG:

Rufen Sie factor ohne Argument auf, wartet das Programm auf die Eingabe einer Zahl. Geben Sie eine Zahl ein, die kleiner als $2^{56}$ sein muß, wird diese in ihre Primfaktoren zerlegt. Jeder Primfaktor wird so oft angezeigt, wie er tatsächlich vorkommt (Beispiel: 12 = 2 2 3). Nach der Ausgabe erwartet das Programm die nächste Eingabe. Das Programm wird beendet, wenn Sie eine 0 oder ein nicht-numerisches Zeichen eingeben.

Rufen Sie factor mit einem Argument auf, wird die Zahl – wie oben beschrieben – zerlegt und das Programm beendet.

# false

false – Lieferung des Rückgabe-Codes 1 (unwahr)

SYNTAX:

fálse

BESCHREIBUNG:

Der false-Befehl hat keine andere Aufgabe, als den Rückgabe-Code 1 zu liefern. False wird in Shell-Kommandoprozeduren benutzt.

Beispiel:

```
until false
do
    Kommando
done
```

SIEHE AUCH:

sh, true

# file

## file – Ermittlung von Dateitypen

SYNTAX:

file [-c] [-f] [-m*Datei*] *Datei* ...

BESCHREIBUNG:

File führt für jede angegebene Datei eine Serie von Tests durch, um festzustellen, welchen Inhalt die Dateien haben. Wird eine ASCII-Datei erkannt, prüft file die ersten 512 Bytes und versucht, die Sprache zu erkennen.
Ist die angegebene Datei ein ausführbares Programm, zeigt file die Versionsnummer an, wenn sie größer als 0 ist.

File benutzt die Datei /etc/magic um solche Dateien zu identifizieren, die „magische Zahlen" (magic numbers) beinhalten. „Magische Zahlen" sind numerische oder String-Konstanten, die bei bestimmten Dateien (z. B. Objekt-Dateien) den Dateityp kennzeichnen. Ein Kommentar am Anfang der Datei /etc/magic erklärt ihr Format.

Optionen:

-f Das nächste Argument wird als eine Datei interpretiert, die die Namen der zu prüfenden Dateien enthält.

-m*Datei* File benutzt statt der Datei /etc/magic die angegebene Datei (magische Datei), um „magische Zahlen" zu identifizieren.

-c File überprüft die magische Datei auf Formatfehler. Diese Überprüfung geschieht aus Laufzeitgründen nicht automatisch. Eine Ermittlung des Dateityps wird nicht vorgenommen.

SIEHE AUCH:

ld

## find

find – Suchen von Dateien

SYNTAX:

find *Pfadnamen-Liste Bedingungen*

BESCHREIBUNG:

Find durchsucht die Verzeichnisse, die in *Pfadnamen-Liste* angegeben wurden sowie deren Unterverzeichnisse nach Dateien, die die angegebenen *Bedingungen* erfüllen. In *Pfadnamen-Liste* werden ein oder mehrere Pfadnamen angegeben, die durch Leerstellen oder Tabs getrennt sein müssen.

Bedingungen:

| | |
|---|---|
| -name*Dateiname* | Namensangabe der zu suchenden Datei. |
| -perm*Oktalzahl* | Suche von Dateien, deren Zugriffsrechte der angegebenen Oktalzahl entsprechen. |
| -type*x* | Angabe des Dateityps:<br>b = Gerätedatei (blockorientiert)<br>c = Gerätedatei (zeichenorientiert)<br>d = Verzeichnis<br>p = Fifodatei<br>f = Datendatei |
| -links*n* | Suche von Dateien mit *n* Links. |
| -user*Name* | Suche von Dateien des angegebenen Benutzers. Wird *Name* numerisch angegeben und ist in dieser Form nicht als ein Login-Name in der Datei /etc/passwd eingetragen, wird die Zahl als User-ID interpretiert. |
| -group*Name* | Suche von Dateien der angegebenen Gruppe. Wird *Name* numerisch angegeben und ist in dieser Form nicht als ein Gruppen-Name in der Datei /etc/group eingetragen, wird die Zahl als Gruppen-ID interpretiert. |

## find

| | |
|---|---|
| -size*n* | Suche von Dateien mit *n* Blöcken. |
| -atime*n* | Suche von Dateien, auf die vor *n* Tagen zugegriffen wurde. |
| -mtime*n* | Suche von Dateien, die vor *n* Tagen modifiziert wurden. |
| -ctime*n* | Suche von Dateien, deren I-Knoten vor *n* Tagen geändert wurden. |
| -exec*Befehl*[{}]\; | Suche der angegebenen Dateien und Ausführung von *Befehl*.<br>Das Kommandoargument {} wird durch den aktuellen Pfadnamen ersetzt. |
| -ok*Befehl*[{}]\; | Wie '-exec', jedoch wird vor Ausführung des Befehls Ihre Bestätigung durch Eingabe von 'y' gefordert. |
| -print | Anzeige der Pfadnamen der Dateien, die die angegebenen Bedingungen erfüllen. |
| -cpio*Gerät* | Ausgabe der Datei auf das angegebene Gerät. Die Ausgabe erfolgt im cpio-Format (5120 Bytes/Satz). |
| -newer*Datei* | Suche von Dateien, die zu einem späteren Zeitpunkt geändert wurden als die angegebene Datei. |
| \(*Ausdruck*\) | Rückgabe-Code 0, wenn der in Klammern angegebene Ausdruck wahr ist. Da die Klammern für die Shell spezielle Zeichen bedeuten, müssen sie mit einem vorangestellten Backslash (\) angegeben werden. |

## find

Parameter der Bedingungen:

*n* = Dezimalzahl (genau *n*).

-*n* = Weniger als *n*.

+*n* = Mehr als *n*.

-a = Bei der Angabe mehrerer Bedingungen werden nur die Dateien angezeigt, die alle Bedingungen gleichzeitig erfüllen.

() = Werden Bedingungen in Klammern angegeben, müssen die gesuchten Dateien diese gleichzeitig erfüllen.

-o = Wird eine von mehreren geforderten Bedingungen erfüllt, so gilt die Datei als gefunden (Oder-Funktion).

! = Sucht alle Dateien, die die nachgestellten Bedingungen nicht erfüllen.

Beispiel:

```
find / \( -name a.out -o -name '*.o \) -atime +7 -exec rm {} \;
```

Alle Dateien mit Namen a.out oder *.o, auf die seit einer Woche nicht zugegriffen wurde, werden gelöscht.

DATEIEN:

/etc/passwd

/etc/group

SIEHE AUCH:

cpio, sh, test
Systemliteratur TARGON: „Systemschnittstellen und Programmierung"

# get

## get – Erzeugen von Versionen einer SCCS-Datei

SYNTAX:

get [-r*SID*] [-c*Abschnitt*] [-i*Liste*] [-x*Liste*] [-a*Sequ.Nr.*]
[-k] [-e] [-l[p]] [-p] [-m] [-n] [-s] [-b] [-g] [-t] *Datei* ...

BESCHREIBUNG:

Get erzeugt eine ASCII-Textdatei – in Übereinstimmung mit den angegebenen Optionen – aus jeder aufgeführten SCCS-Datei. Die Argumente können in beliebiger Reihenfolge angegeben werden, aber **alle** Optionsargumente beziehen sich auf jede der angeführten SCCS-Dateien.
Geben Sie ein Verzeichnis an, werden alle darin enthaltenen Dateien bearbeitet; Nicht-SCCS-Dateien (die letzte Komponente des Pfadnamens beginnt nicht mit 's.') und nicht-lesbare Dateien werden jedoch übergangen. Ist der angegebene Dateiname '-', wird aus der Standardeingabe gelesen. Jede Eingabezeile wird dann als Name einer zu verarbeitenden SCCS-Datei genommen. Auch hier gelten die gleichen Regeln wie oben.

Der erzeugte Text wird normalerweise in eine g-*Datei* geschrieben. Deren Name wird aus dem SCCS-Dateinamen einfach durch Entfernen des führenden 's.' gebildet.

Jede der nachfolgenden Optionen ist so erklärt, als ob nur **eine** SCCS-Datei verarbeitet werden soll, aber die Wirkung eines jeden beliebigen Arguments bezieht sich auf alle angegebenen Dateien.

-r*SID* Die SCCS-Identifikation der Deltaversion einer SCCS-Datei, die aufgefunden werden soll. Tabelle 1 zeigt für die meisten Fälle, welche Version einer SCCS-Datei als Ergebnis des angegebenen *SID* wiedergefunden wird (wie auch den *SID* der Version, die dann schließlich von delta erzeugt wird, wenn Sie die -e-Option ebenfalls angegeben haben).

## get

-c*Abschnitt* Datum-Zeit-Angabe, in der Form

*JJ*[*MM*[*TT*[*HH*[*MM*[*SS*] ] ] ] ]

Die Veränderungen (Deltas) an der SCCS-Datei, die nach dem angegebenen *Abschnitt* erzeugt wurden, werden nicht in der generierten ASCII-Textdatei aufgenommen. Zeiteinheiten, die bei der Datum-Zeit-Angabe ausgelassen werden, erhalten standardmäßig ihre größtmöglichen Werte. Z. B. ist '-c8402' gleichwertig mit '-c840229235959'. Eine beliebige Anzahl von nicht-numerischen Zeichen kann die verschiedenen Ziffernpaare der Datum-Zeit-Angabe untereinander trennen. Diese Möglichkeit läßt eine cutoff-Angabe in der Form '-c84/2/2 9:22:25' zu. Das bedeutet, daß Sie die '%E%' und '%U%'-Schlüsselwörter (s. unten) für geschachtelte get-Befehle benutzen können, z. B. als Eingabe für ein send-Kommando:

~!get "-c%E% %U%" s.*Datei*

-e Anweisung, daß get zum Editieren oder Ändern (Herstellen eines Deltas) der SCCS-Datei über einen anschließenden Aufruf von delta bestimmt ist.
Die -e-Option, die bei einem Aufruf von get für eine bestimmte Version (SID) benutzt wird, verhindert, daß kein weiteres get zum Editieren abgesetzt wird, bis delta ausgeführt oder der j-Schalter in der SCCS-Datei gesetzt ist (s. admin). Gleichzeitiges Benutzen von get -e für verschiedene SIDs ist jederzeit zulässig.

Wird die g-Datei, die von get -e erzeugt wurde, beim Editierprozeß zufällig zerstört, können Sie sie durch den Aufruf von get -k rekonstruieren.

Der SCCS-Dateischutz, der in der SCCS-Datei über den „floor" (niedrigste Version), das „ceiling" (höchste Version) und die Liste der autorisierten Benutzer angegeben wird, tritt in Kraft, wenn Sie die -e-Option benutzen.

## get

-b Diese Option wird – kombiniert mit -e – benutzt, um zu veranlassen, daß das neue Delta eine SID in einem neuen Zweig bekommt, wie in Tabelle 1 dargestellt. Diese Option wird ignoriert, wenn kein b-Schalter in der SCCS-Datei gesetzt ist (s. admin) oder wenn das aufgefundene Delta kein Blattelement des Baumes ist (ein Blattelement hat keinen Nachfolger im Baum).

-i*Liste* Liste von Deltas, die bei der Erzeugung der SCCS-Datei mit eingeschlossen werden sollen. *Liste* hat die folgende Form:

*Liste* ::= *Bereich* | *Liste* , *Bereich*
*Bereich* ::= *SID* | *SID* - *SID*

Die SCCS-Identifikationszeichenkette SID kann in der Form sein, wie sie in der Spalte „SID-Spezifikation" in Tabelle 1 angegeben ist. Teil-SIDs werden so interpretiert, wie in der Spalte „Gefundener SID" in Tabelle 1 dargestellt.

-x*Liste* Liste von Deltas, die bei der Erzeugung der SCCS-Datei nicht berücksichtigt werden sollen (s. -i-Option bzgl. des Listenformats).

-k Unterdrückt im gefundenen Text die Ersetzung von Identifikationsschlüsselworten (s. unten) durch ihren Wert. Diese Option wird durch -e impliziert.

-l[p] Durch Eingabe von -l veranlassen Sie die Ausgabe einer Delta-Zusammenfassung in die Datei l-*Datei*. Geben Sie -lp an, wird l-*Datei* nicht erzeugt; die Ausgabe der Delta-Zusammenfassung erfolgt dann auf der Standardausgabe.

-p Der Text aus der SCCS-Datei wird in die Standardausgabe geschrieben. Eine g-*Datei* wird nicht kreiert. Die gesamte Ausgabe von get, die normalerweise auf der Standardausgabe erfolgt, wird in die Standardfehlerausgabe umgeleitet. Benutzen Sie die -s-Option, wird die Ausgabe unterdrückt.

## get

-s Unterdrückt den normalerweise in die Standardausgabe geschriebenen Text. Fehlermeldungen bleiben davon unberührt, da sie immer in die Standardfehlerausgabe gehen.

-m Jeder Textzeile, die aus der SCCS-Datei geholt wird, wird die SID desjenigen Deltas vorangestellt, welches die Textzeile der SCCS-Datei als letztes beeinflußt hat. Das Format lautet: SID, gefolgt von einem Tabulatorzeichen, dem eine Textzeile folgt.

-n Jeder erzeugten Textzeile wird das %M%-Identifikations-Schlüsselwort vorangestellt (s. unten). Das Format lautet: %M%-Wert, gefolgt von einem Tabulatorzeichen, dem eine Textzeile folgt. Geben Sie sowohl die -m als auch die -n-Option an, lautet das Format: %M%-Wert, gefolgt von einem Tabulatorzeichen, gefolgt von dem durch die -m-Option erzeugten Format.

-g Es wird keine Datei erstellt, sondern die SID der letzten Stammversion ermittelt. Diese Option sollten Sie benutzen, um eine I-*Datei* zu erzeugen oder um das Vorhandensein einer bestimmten SID zu überprüfen.

-t Zugriff auf das zuletzt erzeugte Delta (top) in einer bestimmten Version (z. B. -r1) oder Version und Stufe (z. B. -r1.2) zu.

-a*Sequ.Nr.* Delta-Sequenz-Nummer desjenigen Deltas, das aufgesucht werden soll. Diese Option wird vom comb-Befehl benutzt; sie sollte vom Benutzer nicht aufgerufen werden. Geben Sie sowohl die -r als auch die -a-Option an, wird -a angenommen. Bei der Verwendung von -a zusammen mit -e ist Vorsicht geboten, da der SID des zu erzeugenden Deltas u. U. nicht den Erwartungen entspricht. Die -r-Option kann mit -a und -e kombiniert werden, um die Benennung des SID des zu erzeugenden Deltas zu steuern.

Nach jeder verarbeiteten Datei gibt get die SID, auf die zugegriffen wurde, sowie die Anzahl Zeilen, die aus der SCCS-Datei geholt wurden, auf der Standardausgabe aus.

## get

Verwenden Sie die -e-Option, so erscheint die SID des zu erzeugenden Deltas hinter der SID, auf die zugegriffen wird und vor der Anzahl der ausgegebenen Zeilen. Geben Sie mehr als eine Datei, ein Verzeichnis oder die Standardeingabe an, wird jeder Dateiname vor der Verarbeitung auf einer separaten Zeile aufgelistet. Bei Verwendung der -i-Option werden die einzuschließenden Deltas nach der Meldung 'Included:' aufgelistet; ist die -x-Option angegeben, so werden die auszuschließenden Deltas nach der Meldung 'Excluded:' angezeigt.

Tabelle 1 – Bestimmung des SID

| SID* Spezifik. | -b-Option benutzt**** | sonstige Bedingungen | Gefundener SID | SID des zu erzeug. Deltas |
|---|---|---|---|---|
| keine***** | nein | R standardm. mR | mR.mL | mR.(mL+1) |
| keine***** | ja | R standardm. mR | mR.mL | mR.mL.(mB+1).1 |
| R | nein | R > mR | mR.mL | R.1*** |
| R | nein | R = mR | mR.mL | mR.(mL+1) |
| R | ja | R > mR | mR.mL | mR.mL.(mB+1).1 |
| R | ja | R = mR | mR.mL | mR.mL.(mB+1).1 |
| R | – | R < mR und R existiert nicht | hR.mL** | hR.mL.(mB+1).1 |
| R | – | Stammnachfolger in Version > R und R existiert | R.mL | R.mL.(mB+1).1 |
| R.L | nein | kein Stammnachf. | R.L | R.(L+1) |
| R.L | ja | kein Stammnachf. | R.L | R.L.(mB+1).1 |
| R.L | – | Stammnachf. in Version ≥ R | R.L | R.L.(mB+1).1 |
| R.L.B | nein | kein Stammnachf. | R.L.B.mS | R.L.B.(mS+1) |
| R.L.B | ja | kein Stammnachf. | R.L.B.mS | R.L.(mB+1).1 |
| R.L.B.S | nein | kein Stammnachf. | R.L.B.S | R.L.B.(S+1) |
| R.L.B.S | ja | kein Stammnachf. | R.L.B.S | R.L.(mB+1).1 |
| R.L.B.S | – | Stammnachfolger | R.L.B.S | R.L.(mB+1).1 |

* R, L, B und S sind die Komponenten Release (Version), Level (Stufe), Branch (Zweig) und Sequence (Sequenz) der SID. Das kleine m steht für Maximum. So bedeutet etwa R.mL die maximale Stufennummer innerhalb der Version R; R.L.(mB+1).1 bezeichnet die erste Sequenznummer des neuen Zweiges (d. h. maximale Zweignummer plus eins) der Stufe L innerhalb von Version R. Es ist zu beachten, daß jede der spezifizierten Komponenten existieren muß, wenn die SID die Form R.L, R.L.B oder R.L.S.B hat.

** hR bezeichnet die höchste existierende Version, die niedriger ist als die angegebene **nichtexistierende** Version R.

*** Erzwingt die Generierung des ersten Deltas in einer neuen Version.

**** Die -b-Option ist nur wirksam, wenn der b-Schalter in der Datei gesetzt ist. '-' bedeutet „irrelevant".

***** Trifft zu, wenn der d-Schalter (Standard-SID) in der Datei nicht gesetzt ist. Ist er gesetzt, so wird die aus der d-Angabe gewonnene SID so behandelt, als wäre sie auf der Kommandozeile spezifiziert. Dann trifft ein anderer Fall dieser Tabelle zu.

### Identifikations-Schlüsselwörter

Identifizierende Informationen werden in den aus der SCCS-Datei geholten Text integriert, indem die ID-Schlüsselwörter – wo immer sie vorkommen – durch ihren Wert ersetzt werden.

Die folgenden Schlüsselwörter können Sie im SCCS-Dateitext verwenden:

| Schlüsselwort | Bedeutung |
|---|---|
| %M% | Modulname: entweder der Wert des m-Schalters in der Datei oder – falls dieser nicht vorhanden ist – der Name der SCCS-Datei ohne führendes 's.'. |
| %I% | SCCS-Identifikationszeichenkette (SID = %R%.%L%.%B%.%S%) des aufgefundenen Textes. |

## get

| | |
|---|---|
| %R% | Version (Release). |
| %L% | Stufe (Level). |
| %B% | Zweig (Branch). |
| %S% | Sequenz. |
| %D% | Aktuelles Datum (Format: JJ/MM/TT). |
| %H% | Aktuelles Datum (Format: MM/TT/JJ). |
| %T% | Uhrzeit (Format: HH:MM:SS). |
| %E% | Datum, an dem das neueste Delta erzeugt wurde (Format. JJ/MM/TT). |
| %G% | Datum, an dem das neueste Delta erzeugt wurde (Format: MM/TT/JJ). |
| %U% | Uhrzeit, zu der das neueste Delta erzeugt wurde (Format: HH:MM:SS). |
| %Y% | Modultyp: Wert des t-Schalters in der SCCS-Datei (s. admin). |
| %F% | SCCS-Dateiname. |
| %P% | Vollständiger SCCS-Dateiname. |
| %Q% | Wert des q-Schalters (s. admin). |
| %C% | Laufende Zeilennummer. Dieses Schlüsselwort ist dafür bestimmt, um vom Programm ausgegebene Meldungen zu identifizieren. Es ist **nicht** dazu bestimmt, auf jeder Zeile als Sequenznummer zu erscheinen. |
| %Z% | Zeichenkette der Länge 4, die von what erkannt wird. |
| %W% | Abkürzende Schreibweise, um what-Zeichenketten für UNIX-Programmdateien zu erzeugen. Format: %W% = %Z%%M% Tabulatorzeichen %I% |
| %A% | Abkürzende Schreibweise, um what-Zeichenketten für Nicht-UNIX-Programmdateien zu erzeugen. Format: %A% = %Z%%Y% %M% %I%%Z% |

## get

DATEIEN:

Verschiedene Hilfsdateien können von get erzeugt werden. Diese sind allgemein als g-*Datei*, l-*Datei*, p-*Datei* und z-*Datei* bekannt. Der Buchstabe vor dem Bindestrich wird Kennzeichen genannt. Der Name der Hilfsdatei wird aus dem SCCS-Dateinamen gebildet: Die letzte Komponente aller SCCS-Dateinamen muß die Form 's.', gefolgt vom Modulnamen, haben.
Die Hilfsdateinamen werden durch Ersetzen des führenden 's.' durch das Kennzeichen gebildet. g-*Datei* bildet hier eine Ausnahme: sie wird durch Entfernen des führenden 's.' mit dem SCCS-Dateinamen benannt. Beispielsweise heißen bei einem Dateinamen s.xyz.c die Namen der Hilfsdateien xyz.c, l.xyz.c, p.xyz.c und z.xyz.c.

Die g-*Datei*, die den erzeugten Text enthält, wird im aktuellen Verzeichnis eingetragen (es sei denn, daß Sie die -p-Option benutzt haben). g-*Datei* wird in jedem Fall erzeugt, egal ob Textzeilen von get generiert wurden oder nicht. Der tatsächliche Benutzer ist auch der Eigentümer der Datei. Geben Sie die -k-Option an, oder wird sie implizit gesetzt, ist der Zugriffsmodus 644 (rw-r--r--), falls er nicht auf 444 (r--r--r--) gesetzt wird. Nur der tatsächliche Benutzer braucht Schreiberlaubnis im aktuellen Verzeichnis.

Die l-*Datei* enthält eine Tabelle, die anzeigt, welche Deltas beim Erzeugen des wiedergewonnenen Textes benutzt wurden. Die l-*Datei* wird im aktuellen Verzeichnis angelegt, wenn Sie die -l-Option benutzen. Ihr Modus ist 444 (r--r--r--) und der tatsächliche Benutzer ist auch ihr Eigentümer. Nur er braucht Schreiberlaubnis im aktuellen Verzeichnis.

Zeilen in einer l-*Datei* haben das folgende Format:

a. Ein Leerzeichen (Blank), wenn das Delta mit eingeschlossen wurde. '*' in allen anderen Fällen.

b. Ein Leerzeichen (Blank), wenn das Delta eingeschlossen oder wenn es ausgeschlossen und ignoriert wurde. '*', wenn es nicht benutzt und nicht ignoriert wurde.

c. Code, der anzeigt, warum das Delta benutzt wurde oder nicht.
   'I': eingeschlossen (Included)
   'X': ausgeschlossen (Excluded)
   'C': abgeschnitten (cut off, -c-Option)

## get

d. Leerzeichen.

e. SID.

f. Tabulatorzeichen

g. Datum und Uhrzeit der Erzeugung (Format: JJ/MM/TT HH:MM:SS).

h. Leerzeichen.

i. Login-Name des Benutzers, der das Delta erzeugt hat.

Die Kommentare und MR-Daten folgen – um eine Tabulatorposition eingerückt – auf den nächsten Zeilen. Eine Leerzeile beendet jeden Eintrag.

p-*Datei* wird benutzt, um die aus einem get -e stammenden Informationen an das delta-Kommando zu übergeben. Ihr Inhalt wird auch dazu benutzt, um die nachfolgende Ausführung von get -e für denselben SID solange zu verhindern, bis delta ausgeführt ist oder der j-Schalter (s. admin) in der SCCS-Datei gesetzt ist.
p-*Datei* wird in dem Verzeichnis erzeugt, das die SCCS-Datei enthält; der momentane Benutzer muß in diesem Verzeichnis Schreiberlaubnis haben. Der Zugriffsmodus von p-*Datei* ist 644 (rw-r--r--). Der momentane Benutzer ist auch gleichzeitig der Besitzer. Das Format von p-*Datei* ist:

- Erzeugte SID.
- SID, die das neue Delta nach der Erzeugung hat.
- Login-Name des tatsächlichen Benutzers.
- Datum und Uhrzeit, zu der get ausgeführt wurde.
- Optionsargument -i, falls vorhanden.
- Optionsargument -x, falls vorhanden.

Die p-*Datei* kann jederzeit eine beliebige Anzahl von Zeilen enthalten; jedoch kann nur eine Zeile dieselbe SID für das neue Delta enthalten.

## get

Die z-*Datei* dient als Ausschlußmechanismus gegen gleichzeitige Veränderungen. Sie enthält die zwei Bytes lange Prozeßnummer des Kommandos, das den Prozeß erzeugte.
z-*Datei* wird in dem Verzeichnis erzeugt, das die SCCS-Datei während der Dauer von get enthält. Dieselben Schutzmechanismen wie für die p-*Datei* treffen auch für die z-*Datei* zu. Sie wird mit dem Zugriffsmodus 644 (rw-r--r--) erzeugt.

SIEHE AUCH:

admin, delta, help, prs, what
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# getopt

## getopt – Analyse von Kommando-Optionen

SYNTAX:

set -- 'getopt *Zeichenkette* $*'

BESCHREIBUNG:

Getopt wird benutzt, um Optionen durch Shell-Prozeduren zu analysieren und Kommandozeilen auf legale Optionen zu überprüfen.

*Zeichenkette* enthält die Optionen, die angegeben werden dürfen. Ein Doppelpunkt hinter einem Optionsbuchstaben symbolisiert, daß ein Argument auf diese Option folgen muß. Das Argument kann direkt hinter der Option stehen, es ist aber auch ein Zwischenraum erlaubt.

Die spezielle Option '--' begrenzt das Ende der Optionen. Sie kann explizit angegeben werden, anderenfalls wird sie von getopt generiert. Auf jeden Fall wird '--' an das Ende der Optionen gesetzt. Die Positionsparameter der Shell ($1, $2, ...) werden neu gesetzt, so daß jede Option von einem Minuszeichen (-) angeführt wird und in ihrem eigenen Positionsparameter steht. Jedes Optionsargument wird ebenfalls in einem eigenen Positionsparameter abgestellt.

## getopt

Der folgende Programmteil zeigt, wie Sie die Argumente für ein Kommando prüfen können, welches die Optionen -a oder -b erkennt sowie die Option -o, welche ein Argument verlangt.

```
set -- 'getopt abo: $*'
if [$? != 0]
then
        echo $USAGE
        exit 2
fi
for i in $*
do
        case $i in
          -a | -b)    FLAG=$i; shift;;
          -o)         OARG=$2; shift2;;
          --)         shift; break;;
        esac
done
```

Dieses Programm akzeptiert dann die Eingabe eines Kommandos in folgenden Formen:

```
Kommando -aoArgument Datei Datei
Kommando -a -o Argument Datei Datei
Kommando -a -oArgument -- Datei Datei
```

MELDUNGEN:

Wird eine Option erkannt, die nicht in *Zeichenkette* enthalten ist, gibt getopt eine Fehlermeldung aus.

SIEHE AUCH:

sh

# greek

## greek – Auswahl eines Terminalfilters

SYNTAX:

greek [-T*Terminal*]

BESCHREIBUNG:

Greek ist ein Filter, der den erweiterten Zeichensatz sowie die Rückwärts- und Halbzeilenschaltungen eines TELETYPE-Terminals Modell 37 (des Standardterminals für nroff) für bestimmte andere Terminals neu interpretiert. Sonderzeichen werden durch Überschreiben simuliert, sofern dies notwendig und möglich ist. Geben Sie -T nicht an, versucht greek auf die Umgebungsvariable $TERM zurückzugreifen. Derzeit werden die folgenden Terminals erkannt:

| | |
|---|---|
| 300 | DASI 300 |
| 300-12 | DASI 300 mit 12 pitch (Zeichen pro Inch) |
| 300s | DASI 300s |
| 300s-12 | DASI 300s mit 12-pitch |
| 450 | DASI 450 |
| 450-12 | DASI 450 mit 12-pitch |
| 1620 | Diablo 1620 (alias DASI 450) |
| 1620-12 | Diablo 1620 (alias DASI 450) mit 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 |

# grep, egrep, fgrep

grep, egrep, fgrep – Durchsuchen von Dateien nach Suchmustern

SYNTAX:

grep [Option ...] *Suchmuster* [*Datei* ...]

egrep [Option ...] [*Suchmuster*] [*Datei* ...]

fgrep [Option ...] [*Zeichenkette* ...] [*Datei* ...]

BESCHREIBUNG:

Die grep-Kommandos durchsuchen die Eingabedateien (Defaultwert ist die Standardeingabe) nach Zeilen, die das angegebene Suchmuster enthalten.
Jede so gefundene Zeile wird auf der Standardausgabe angezeigt. Grep-Suchmuster sind auf reguläre Ausdrücke im Sinne von ed beschränkt. Suchmuster bei egrep sind ebenfalls reguläre Ausdrücke. Im Gegensatz zu grep benutzt egrep einen schnellen Algorithmus, der exponentiell anwachsenden Speicherplatz belegt. Fgrep-Suchmuster sind feste Zeichenketten; das Programm ist schnell und belegt konstanten Speicher.

Optionen:

-v Anzeige der Zeilen, in denen das Suchmuster **nicht** enthalten ist.

-x Nur Zeilen, die in ihrer Gesamtheit dem Suchmuster entsprechen, werden angezeigt (nur fgrep).

-c Anzeige der Anzahl der gefundenen Zeilen.

-l Nur die Namen der Dateien, die Zeilen enthalten, auf die das Suchmuster paßt, werden ausgegeben.

-n Jeder ausgegebenen Zeile wird die entsprechende Zeilennummer innerhalb ihrer Datei vorangestellt.

-b Jeder ausgegebenen Zeile wird die entsprechende Blocknummer vorangestellt.

-s Die Fehlermeldungen für nicht-existierende oder nicht-lesbare Dateien werden unterdrückt (nur grep).

## grep, egrep, fgrep

-e*Suchmuster*
Beginnt ein Suchmuster mit einem Bindestrich, muß -e vorangestellt werden, um das Suchmuster von einer Option zu unterscheiden (nur egrep und fgrep).

-f*Datei* Der reguläre Ausdruck (egrep) oder die Zeichenkette (fgrep) wird aus *Datei* entnommen.

In allen Fällen wird der Dateiname ausgegeben, wenn mehr als eine Eingabedatei vorhanden ist.
Vorsicht ist bei der Angabe folgender Zeichen in *Suchmuster* geboten: $, *, [, ^, |, (, ) und \ haben für die Shell eine besondere Bedeutung. *Suchmuster*, die solche Zeichen enthalten, müssen daher in einfache Hochkommata eingeschlossen werden.

Fgrep sucht nach Zeilen, die eine der angegebenen Zeichenketten – getrennt durch ein Zeilenende-Zeichen – enthalten.

Egrep akzeptiert reguläre Ausdrücke wie bei ed, außer \( und \). Hinzu kommt:

1. Von einem '+' gefolgte reguläre Ausdrücke passen auf ein oder mehrere Vorkommen des regulären Ausdrucks im Text.
2. Ein '?' nach einem regulären Ausdruck entspricht entweder keinem oder genau einem Vorkommen dieses Ausdrucks.
3. Zwei reguläre Ausdrücke, die durch '|' oder Zeilenvorschub getrennt sind, stellen Alternativen dar, d. h. es wird nach Übereinstimmung mit dem einen oder dem anderen Ausdruck gesucht.
4. Ein regulärer Ausdruck kann zum Zweck der Gruppierung mit runden Klammern versehen werden.

Die Prioritätenfolge der Operatoren ist '[ ]', dann '*?+', die Verkettung '|' und schließlich das Zeilenvorschubzeichen.

## grep, egrep, fgrep

MELDUNGEN:

Wenn Übereinstimmungen gefunden werden, ist der Ergebnisstatus 0, wenn nicht, 1.
Bei Syntaxfehlern oder bei nicht-lesbaren Dateien ist der Rückgabe-Code 2.

HINWEISE:

Zeilen sollten auf „BUFSIZE"-Größe begrenzt sein, da sie sonst bei der Verarbeitung verstümmelt werden. (BUFSIZE ist in /usr/include/stdio.h definiert.)
Egrep erkennt keine Zusammenfassung von Zeichen (z. B. [a-z]).

SIEHE AUCH:

ed, sed, sh

# help

help – Hilfe

SYNTAX:

help [*Argument* ...]

BESCHREIBUNG:

Help gibt Erklärungen über Fehlermeldungen sowie die Handhabung von Befehlen. Es können mehrere Argumente angegeben werden. Geben Sie kein Argument an, fordert help welche.

Argumente können entweder Nachrichtennummern (werden gewöhnlich in Klammern hinter den Nachrichten angezeigt) oder Kommandos sein. Die Argumente sind in folgende Typen eingeteilt:

Typ 1 Beginnt mit Buchstaben und endet mit Zahlen. Die Buchstaben sind Abkürzungen eines Programms. Z. B. bedeutet ge6 die Fehlermeldung 6 des get-Kommandos.

Typ 2 Enthält keine Zahlen, lediglich Kommandos.

Typ 3 Enthält nur Zahlen.

Die Antwort von help ist eine erklärende Information bezüglich der angegebenen Argumente.

DATEIEN:

| | |
|---|---|
| /usr/lib/help | Verzeichnis, das die Dateien mit den Nachrichtentexten enthält. |
| /usr/lib/help/helploc | Datei, die die Pfadnamen der help-Dateien enthält, die nicht in /usr/lib/help enthalten sind. |

SIEHE AUCH:

Systemliteratur TARGON: „Programmentwicklungs-Tools"

## hyphen

hyphen – Überprüfen von Trennstellen

SYNTAX:

hyphen [*Dateien*]

BESCHREIBUNG:

Hyphen überprüft die angegebenen Dateien und gibt alle Worte aus, die am Zeilenende einen Bindestrich enthalten. Geben Sie keine Datei an, liest hyphen aus der Standardeingabe. Daher kann hyphen auch als Filter benutzt werden.

Im folgenden Beispiel überprüft hyphen eine mit nroff formatierte Textdatei:

nroff *Datei* | hyphen

SIEHE AUCH:

Systemliteratur TARGON /35: „Textformatierer nroff"

# id

id – Anzeige von Benutzer- und Gruppenkennung

SYNTAX:

id

BESCHREIBUNG:

Id zeigt an der Standardausgabe Ihre Benutzer- und Gruppen-Nummer sowie die zugehörigen Namen an.

SIEHE AUCH:

logname

# ipcrm

ipcrm – IDs für Nachrichtenwarteschlange, Semaphorgruppe oder gemeinsam benutzten Speicher löschen

SYNTAX:

ipcrm [*Optionen*]

BESCHREIBUNG:

Ipcrm löscht eine oder mehrere angegebene Nachrichten oder IDs für Semaphore und gemeinsam benutzten Speicher. Die IDs werden von den folgenden Optionen spezifiziert:

-q*msqid* Löscht die angegebene Nachrichtenwarteschlangen-ID msqid aus dem System und entfernt die dazugehörende Nachrichtenwarteschlange und Datenstruktur.

-m*shmid* Löscht die angegebene ID für gemeinsam benutzten Speicher aus dem System. Nach der letzten Abtrennung (detach) werden das zugehörige gemeinsam benutzte Speichersegment und die Datenstruktur entfernt.

-s*semid* Löscht die angegebene Semaphor-ID aus dem System und entfernt die zugehörige Semaphorgruppe und Datenstruktur.

-Q*msgkey* Löscht die mit dem Schlüssel *msgkey* erstellte Nachrichtenwarteschlangen-ID aus dem System und entfernt die zugehörige Nachrichtenwarteschlange und Datenstruktur.

-M*shmkey* Löscht die mit dem Schlüssel *shmkey* erstellte gemeinsam benutzte Speicher-ID aus dem System. Das zugehörige gemeinsam benutzte Speichersegment und die Datenstruktur werden nach der letzten Abtrennung (detach) entfernt.

-S*semkey* Löscht die mit dem Schlüssel *semkey* erstellte Semaphor-ID aus dem System und entfernt die zugehörige Semaphorgruppe und Datenstruktur.

Die IDs und Schlüssel können mit ipcs gefunden werden.

SIEHE AUCH:

ipcs

# ipcs

## ipcs – Melden des Status von Interprozeß-Kommunikationsfunktionen

SYNTAX:

ipcs [*Optionen*]

BESCHREIBUNG:

Ipcs gibt bestimmte Informationen über aktive Interprozeß-Kommunikationsfunktionen aus. Ohne Angabe von Optionen werden Informationen im Kurzformat über die derzeit im System aktiven Nachrichtenwarteschlangen, den gemeinsam benutzten Speicher und Semaphore ausgegegeben. Art und Umfang der ausgegebenen Informationen können durch folgende Optionen gesteuert werden:

-q Informationen über aktive Nachrichtenwarteschlangen ausgeben.

-m Informationen über aktive gemeinsam benutzte Speichersegmente (shared memory) ausgeben.

-s Informationen über aktive Semaphore ausgeben.

Geben Sie eine oder mehrere der Optionen -q, -m und -s an, so werden nur die entsprechenden Informationen ausgegeben. Geben Sie keine der Optionen an, so erhalten Sie Informationen über alle drei Bereiche.

-b Gibt Informationen über die maximal zulässige Größe aus. (Maximale Anzahl Bytes in den Nachrichten der Warteschlangen bei Nachrichtenwarteschlangen, Größe der Segmente bei gemeinsam benutztem Speicher bzw. Anzahl der Semaphore in einer Semaphormenge).

-c Login-Name und Gruppenname des Erstellers ausgeben.

-o Informationen über noch offenstehende Aufgaben ausgeben. (Anzahl Nachrichten in der Warteschlange und Gesamtzahl Bytes in den Nachrichten der Warteschlange bzw. Anzahl der an die gemeinsam benutzten Speichersegmente gekoppelten Prozesse.)

-p Informationen über die Prozeßnummern ausgeben. (Prozeß-ID des letzten Prozesses, der eine Nachricht gesendet hat, und Prozeß-ID des letzten Prozesses, der eine Nachricht aus Nachrich-

tenwarteschlangen empfangen hat, bzw. Prozeß-ID des erstellenden Prozesses und Prozeß-ID des letzten Prozesses, der sich bei gemeinsam benutzten Speichersegmenten angeschlossen oder abgekoppelt hat.)

-t Zeitinformationen ausgeben.

-a Alle Ausgabeoptionen verwenden. (Dies ist eine abgekürzte Schreibweise für -b, -c, -o, -p und -t.)

-C*Coredatei*
Die Datei *Coredatei* anstelle von /dev/kmem verwenden.

-N*Namensliste*
Dieses Argument wird als Name einer alternativen Namensliste interpretiert (Standard ist /unix).

Im folgenden sind die Spaltenüberschriften in einer vom Kommando ipcs ausgegebenen Liste und ihre Bedeutungen erklärt. Die Buchstaben in Klammern geben die Optionen an, die das Erscheinen der entsprechenden Spaltenüberschrift bewirken; alle bedeutet, daß die Überschrift immer erscheint. Hier sei noch einmal darauf hingewiesen, daß diese Optionen nur festlegen, welche Informationen für eine Funktion bereitgestellt werden; sie bestimmen nicht, welche Funktionen gelistet werden.

T (alle) Art der Funktion:

- q Nachrichtenwarteschlange
- m gemeinsam benutztes Speichersegment (shared memory)
- s Semaphor

ID (alle) Kennummer des Eintrags für die Funktion.

KEY (alle) Der Schlüssel, der als Argument für msgget, semget bzw. shmget für die Erstellung des Funktionseintrags verwendet wird.

Hinweis: Der Schlüssel eines gemeinsam benutzten Speichersegments wird in IPC_PRIVATE abgeändert, wenn das Segment gelöscht wurde, bis alle an das Segment angeschlossenen Prozesse sich davon abkoppeln.

## ipcs

MODE (alle) Die Zugriffsmodi und Flags der Funktion: Der Modus besteht aus 11 Zeichen mit folgender Bedeutung: Die ersten beiden Zeichen sind:

- R wenn ein Prozeß bei einem 'msgrcv' wartet,
- S wenn ein Prozeß bei einem 'msgsnd' wartet,
- D wenn das zugehörige gemeinsam benutzte Speichersegment gelöscht wurde. Es verschwindet, wenn der letzte an das Segment angeschlossene Prozeß dieses abkoppelt,
- C wenn das zugehörige gemeinsam benutzte Speichersegment gelöscht werden soll, sobald die erste Anschließung (attach) ausgeführt wird,
- \- wenn das entsprechende spezielle Flag nicht gesetzt ist.

Die nächsten neun Zeichen werden als drei Gruppen zu je drei Bits interpretiert. Die erste Gruppe bezieht sich auf die Zugriffsberechtigungen des Benutzers; die nächste Gruppe bezieht sich auf die übrigen Mitglieder der Benutzergruppe des Funktionseintrags; die letzte Gruppe bezieht sich auf alle anderen Benutzer. Innerhalb jeder Gruppe gibt das erste Zeichen die Leseberechtigung und das zweite Zeichen die Schreib- oder Änderungsberechtigung für den Funktionseintrag an, und das letzte Zeichen ist derzeit noch nicht belegt.

Die Berechtigungen werden wie folgt angezeigt:

- r Leseberechtigung ist erteilt.
- w Schreibberechtigung ist erteilt.
- a Änderungsberechtigung ist erteilt.
- \- Die entsprechende Berechtigung ist nicht erteilt.

OWNER (alle) Der Login-Name des Eigentümers des Funktionseintrages.

GROUP (alle) Der Gruppenname der Gruppe des Eigentümers des Funktionseintrages.

ipcs

| | |
|---|---|
| CREATOR (a,c) | Der Login-Name des Erstellers des Funktionseintrages. |
| CGROUP (a,c) | Der Gruppenname der Gruppe des Erstellers des Funktionseintrages. |
| CBYTES (a,o) | Die Anzahl der Bytes in Nachrichten, die in der zugehörigen Nachrichtenwarteschlange derzeit offenstehen. |
| QNUM (a,o) | Die Anzahl der Nachrichten, die in der zugehörigen Nachrichtenwarteschlange derzeit offenstehen. |
| QBYTES (a,b) | Die maximale Anzahl Bytes, die in offenstehenden Nachrichten in der zugehörigen Nachrichtenwarteschlange zulässig ist. |
| LSPID (a,p) | Die Prozeß-ID des letzten Prozesses, der eine Nachricht an die zugehörige Warteschlange abgesetzt hat. |
| LRPID (a,p) | Die Prozeß-ID des letzten Prozesses, der eine Nachricht aus der zugehörigen Warteschlange empfangen hat. |
| STIME (a,t) | Der Zeitpunkt, an dem die letzte Nachricht an die zugehörige Warteschlange gesendet wurde. |
| RTIME (a,t) | Der Zeitpunkt, an dem die letzte Nachricht aus der zugehörigen Warteschlange empfangen wurde. |
| CTIME (a,t) | Der Zeitpunkt, an dem der zugehörige Eintrag erstellt oder geändert wurde. |
| NATTCH (a,o) | Die Anzahl der Prozesse, die an das zugehörige gemeinsam benutzte Speichersegment angeschlossen sind. |
| SEGSZ (a,b) | Die Größe des zugehörigen gemeinsam benutzten Speichersegments. |
| CPID (a,p) | Die Prozeß-ID des Erstellers des Eintrags für gemeinsam benutzten Speicher. |
| LPID (a,p) | Die Prozeß-ID des letzten Prozesses, der das gemeinsam benutzte Speichersegment angeschlossen (attach) oder abgetrennt (detach) hat. |

## ipcs

ATIME (a,t) Der Zeitpunkt, an dem die letzte Anschließung (attach) an das zugehörige gemeinsam benutzte Speichersegment durchgeführt wurde.

DTIME (a,t) Der Zeitpunkt, an dem die letzte Abtrennung (detach) von dem zugehörigen gemeinsam benutzten Speichersegment durchgeführt wurde.

NSEMS (a,b) Die Anzahl der Semaphore in der zu dem Semaphoreintrag gehörenden Menge.

OTIME (a,t) Der Zeitpunkt, an dem die letzte Semaphor-Operation mit der zu dem Semaphoreintrag gehörenden Menge durchgeführt wurde.

DATEIEN:

| | |
|---|---|
| /unix | System-Namensliste |
| /dev/kmem | Speicher |
| /etc/passwd | Benutzernamen |
| /etc/group | Gruppennamen |

HINWEIS:

Während der Ausführung von ipcs können sich die Verhältnisse ändern, das von ipcs wiedergegebene Bild spiegelt die tatsächlichen Verhältnisse aber annäherungsweise wieder.

# join

join – Relationaler Datenbank-Operator

SYNTAX:

join [*Optionen*] *Datei1 Datei2*

BESCHREIBUNG:

Join fügt die zwei Relationen, die in den Zeilen von *Datei1* und *Datei2* spezifiziert sind, in der Standardausgabe zusammen. Geben Sie statt *Datei1* '-' an, wird die Standardeingabe benutzt.

Die angegebenen Dateien müssen in aufsteigender ASCII-Reihenfolge in Bezug auf die Felder, durch die sie miteinander verbunden werden sollen, sortiert sein. In der Regel ist dies das erste Feld in jeder Zeile.

Für jedes Zeilenpaar gibt es eine Ausgabezeile mit identischen 'join'-Feldern. Die Ausgabezeile enthält normalerweise das gemeinsame Feld, den Rest der Zeile aus *Datei1* und abschließend den Rest der Zeile aus *Datei2*.

Die Felder sind durch Leerzeichen, Tabulatoren oder Zeilenende-Zeichen voneinander getrennt. Mehrere dieser Trennzeichen werden als ein Zeichen gezählt; führende Trennzeichen werden gelöscht.

Folgende Optionen können Sie verwenden:

-a*n* Für jede Zeile in der Datei *n* (*n* = 1 oder 2), mit der kein Zeilenpaar gebildet werden kann, wird zusätzlich zur normalen Ausgabe eine Zeile erzeugt.

-e*Zeichenkette*
Leere Ausgabefelder werden durch *Zeichenkette* ersetzt.

-j[*n*]*m* Die Dateien werden auf dem *m*ten Feld der Datei *n* zusammengefügt. Wenn *n* fehlt, wird das *m*te Feld aus jeder Datei verwendet.

-o*Liste* Jede Ausgabezeile enthält die in *Liste* spezifizierten Felder, von denen jedes Element die Form *n.m* hat, wobei *n* eine Dateinummer und *m* eine Feldnummer darstellt.

## join

-tc Das Zeichen, das Sie für *c* angeben, wird als Trennzeichen verwendet (Tabulatorzeichen).

Beispiel:

```
join -j1 4 -j2 3 -o 1.1 2.1 1.6 -t: /etc/passwd /etc/group
```

verbindet die Dateien /etc/passwd und /etc/group. Das Verbindungsfeld ist die Gruppen-ID. Ausgegeben werden Login-Namen, Gruppen-Namen und Login-Verzeichnis. Voraussetzung ist jedoch, daß die Dateien nach Gruppen-ID's sortiert sind.

SIEHE AUCH:

awk, comm, sort, uniq

## kill

kill – Interrupt senden

SYNTAX:

kill [-*Signal*] *Prozeßnummer* ...

BESCHREIBUNG:

Der kill-Befehl sendet das angegebene Signal an den spezifizierten Prozeß. Zum Beispiel bricht der Befehl:

kill -9 *Prozeßnummer*

auch die Prozesse ab, die andere Signale nicht empfangen. Das Signal 9 kann nicht ignoriert werden. Geben Sie kein Signal an, wird standardmäßig Signal 15 gesendet. Dies führt normalerweise zum Prozeßabbruch (Standardverhalten).

Ein Prozeß kann nur von seinem Besitzer oder vom Superuser abgebrochen werden.

Beim Starten von Prozessen im Hintergrund (&) zeigt die interaktive Shell die Prozeßnummer an. Rufen Sie den ps-Befehl auf, werden ebenfalls die Prozeßnummern der laufenden Prozesse angezeigt.

SIEHE AUCH:

ps, sh
Systemliteratur TARGON: „Systemschnittstellen und Programmierung"

# ld

## ld – Binder (Link-Editor)

SYNTAX:

ld [*Optionen*] *Datei* ...

BESCHREIBUNG:

Das Kommando ld verbindet mehrere Objektprogramme zu einem einzigen Programm, löst externe Symbole auf und durchsucht Bibliotheken. Im einfachsten Fall sind mehrere Objektprogramme vorgegeben, die ld miteinander verbindet und daraus ein Objektmodul erstellt, das entweder ausgeführt oder als Eingabe für einen weiteren Durchlauf von ld verwendet werden kann. (Im letzteren Fall muß die Option -r angegeben werden, damit die Relocation-Bits erhalten bleiben.) Die Ausgabe von ld wird in a.out geschrieben. Diese Datei kann nur dann ausgeführt werden, wenn beim Laden keine Fehler aufgetreten sind.

Die in den Argumenten angegebenen Routinen werden in der angegebenen Reihenfolge verkettet. Die Einsprungstelle der Ausgabe ist der Anfang der ersten Routine (außer wenn die Option -e angegeben ist).

Ist ein Argument eine Bibliothek, wird diese an der Stelle, an der sie in der Argumentenliste gefunden wird, genau einmal durchsucht. Es werden nur die Routinen geladen, die eine nicht aufgelöste externe Referenz definieren. Wenn eine Routine aus einer Bibliothek eine andere Routine in der Bibliothek anspricht und die Bibliothek nicht von ranlib bearbeitet wurde, muß die angesprochene Routine in der Bibliothek hinter der ansprechenden Routine erscheinen. Die Reihenfolge der Programme in den Bibliotheken kann daher von Bedeutung sein. Die erste Komponente einer Bibliothek sollte eine Datei mit dem Namen __.SYMDEF sein, bei der es sich um ein Dictionary (Wörterbuch) für die von ranlib erstellte Bibliothek handelt; das Dictionary wird iterativ durchsucht, um so viele Referenzen wie möglich zu erfüllen.

## ld

Die Symbole _etext, _edata und -end (etext, edata und end in C) sind reserviert. Wenn sie angesprochen werden, werden sie auf die erste Speicherposition oberhalb des Programms bzw. die erste Speicherposition oberhalb aller Daten gesetzt. Es ist falsch, diese Symbole zu definieren.

Ld kann mehrere Optionen auswerten. Mit Ausnahme von -l sollten die Optionen vor den Dateinamen stehen.

Optionen:

-A Diese Option bedeutet inkrementelles Laden, d. h. daß Linken soll so erfolgen, daß das daraus entstehende Objekt in ein bereits laufendes Programm eingelesen werden kann. Das nächste Argument ist der Name einer Datei, deren Symboltabelle als Grundlage für die Definition weiterer Symbole verwendet wird. Nur neu gelinkte Teile werden in den Text- und Datenteil von a.out eingefügt, aber die neue Symboltabelle berücksichtigt alle Symbole, die vor und nach dem inkrementalen Laden definiert wurden. Dieses Argument muß vor jeder anderen Objektdatei in der Argumentenliste stehen. Die Option -T kann ebenfalls verwendet werden und bedeutet, daß das neu gelinkte Segment auf der entsprechenden Adresse (die ein Vielfaches von 2048 sein muß) beginnt. Standard ist der alte Wert von _end.

-D Das nächste Argument wird als Hexadezimalzahl interpretiert und das Datensegment bis zur angegebenen Länge mit Null-Bytes aufgefüllt.

-d Die Definition eines gemeinsamen Speicherbereichs wird zwingend herbeigeführt, auch wenn die Option -r angegeben ist.

-e Das folgende Argument wird als Name der Einsprungstelle des geladenen Programms interpretiert; Adresse 0 ist Standardwert.

-l*x* Diese Option ist eine Abkürzung für den Bibliotheksnamen /lib/lib*x*.a, wobei *x* eine Zeichenfolge ist. Existiert diese Bibliothek nicht, versucht ld es weiter in /usr/lib/lib*x*.a. Eine Bibliothek wird in dem Augenblick durchsucht, da ihr Name erkannt wird. Deshalb ist die Plazierung von -l von Bedeutung.

## ld

-M Erstellung einer einfachen Ladetabelle, in der die Namen der zu ladenden Dateien verzeichnet sind.

-N Den Textteil nicht schreibgeschützt (read-only) oder gemeinsam benutzbar (sharable) machen. (Die „magische Zahl" 0407 verwenden.)

-n Dadurch, daß der Ausgabedatei die „magische Zahl" 0410 zugeordnet wird, wird dafür gesorgt, daß bei Ausführung der Ausgabedatei der Textteil schreibgeschützt (read-only) ist und von allen Benutzern, die die Datei ausführen, gemeinsam benutzt wird. Dabei werden auch die Datenbereiche bis zur ersten möglichen 2048-Byte-Grenze hinter dem Ende des Textes verschoben.

-o Das Argument hinter -o ist der Name der Ausgabedatei von ld, der anstelle von a.out verwendet wird.

-r Erzeugung von Relocation-Bits in der Ausgabedatei, so daß sie erneut von ld bearbeitet werden kann. Diese Option verhindert außerdem, daß gemeinsamen Symbolen endgültige Definitionen gegeben werden, und sie unterdrückt die Diagnose nicht-definierter Symbole.

-S In der Ausgabe alle Symbole mit Ausnahme lokaler und globaler Symbole entfernen.

-s In der Ausgabe die Symboltabelle und die Relocation-Bits entfernen, um Speicherplatz zu sparen (wonach allerdings die Debugger weniger wirksam sind). Diese Informationen können auch mit strip entfernt werden.

-T Das nächste Argument ist eine Hexadezimalzahl, die die Startadresse des nächsten Segments festlegt. Standardwert für die Startadressse ist 0.

-t Bei der Bearbeitung einer Datei jeweils deren Namen ausgeben (Trace).

-u Das folgende Argument als Symbol interpretieren und dieses als nicht-definiert in die Symboltabelle eintragen. Dies ist hilfreich beim ausschließlichen Laden aus einer Bibliothek, da die Symboltabelle anfangs leer ist, aber eine nicht aufgelöste Referenz erforderlich ist, um das Laden der ersten Routine herbeizuführen.

ld

-X Lokale Symbole mit Ausnahme derjenigen, deren Namen mit L beginnen, speichern. Diese Option wird von cc benutzt, um intern generierte Marken wegfallen zu lassen, während lokale Symbole innerhalb von Routinen beibehalten werden.

-x Lokale (non-.globl) Symbole sollen in der ausgegebenen Symboltabelle nicht erhalten bleiben; nur externe Symbole eintragen. Diese Option spart Speicherplatz in der Ausgabedatei.

-y*sym* Alle Dateien, in denen *sym* auftritt, mit ihrem Typ angeben und mitteilen, ob die Datei dieses Symbol definiert oder anspricht. Sollen mehrere Symbole überwacht werden, so können mehrere Optionen dieser Form angegeben werden. (In der Regel muß *sym* mit einem _ beginnen, da externe Variablen bei C, Fortran und Pascal mit Unterstrichen beginnen.)

-z Der Prozeß soll auf Anforderung aus der entstehenden ausführbaren Datei (Format 413) geladen und nicht vorab geladen werden (Standard). Dies führt zu einem 2048 Byte großen Header in der Ausgabedatei, an den sich ein Text- und ein Datensegment anschließt, deren Größe jeweils ein ganzes Vielfaches von 2048 Bytes beträgt (wobei die Datei gegebenenfalls mit Null-Zeichen aufgefüllt wird). Bei diesem Format können die ersten wenigen Symbole des BBS-Segments effektiv so aussehen (aufgrund der Ausgabe von size), als ob sie im Datensegment angesiedelt sind; dadurch soll unnötige Belegung von Speicherplatz vermieden werden, die sich durch das Zusammenstellen der Größe des Datensegments ergeben kann.

DATEIEN:

| | |
|---|---|
| /lib/lib*.a | Bibliotheken |
| /usr/lib/lib*.a | Bibliotheken |
| /usr/local/lib/lib*.a | Bibliotheken |
| a.out | Ausgabedatei |

## ld

HINWEIS:

Es gibt keine Möglichkeit, die Daten nach Seitengrenzen auszurichten. Ld füllt Abbilder, die bei Bedarf aus dem Dateisystem geladen werden, bis zur nächsten Seitengrenze auf, um einen Fehler im System zu vermeiden.

SIEHE AUCH:

as, ar, cc

## lex

lex – Generierung von Programmen für eine einfache lexikalische Analyse

SYNTAX:

lex [-rctvn] [*Datei*] ...

BESCHREIBUNG:

Lex generiert Programme für eine einfache lexikalische Analyse von Texten.

Die Eingabedateien (Standard: Standardeingabe) enthalten Zeichenfolgen und Ausdrücke, nach denen gesucht wird, und C-Programmcode, der auszuführen ist, wenn diese Suche erfolgreich war.

Es wird eine Datei lex.yy.c erzeugt, die, wenn sie mit der Bibliothek geladen wird, die Eingabe in die Ausgabe kopiert, außer wenn eine in der Datei angegebene Zeichenkette gefunden wird; in diesem Fall wird der entsprechende Programmtext ausgeführt. Die eigentliche gefundene Zeichenkette wird in yytext abgestellt, einem externen Zeichen-Array. Der Vergleich erfolgt in der Reihenfolge der Zeichenketten in der Datei. Die Zeichenfolgen können eckige Klammern enthalten, in die Zeichenklassen eingeschlossen sind. So bedeutet [abx-z] beispielsweise die Zeichen a, b, x, y und z. Der Operator * steht für eine beliebige nicht-negative Anzahl, der Operator + für eine beliebige positive Anzahl und der Operator ? für null oder ein Exemplar des vorangehenden Zeichens oder der Zeichenklasse. Das Zeichen . steht für die Klasse aller ASCII-Zeichen mit Ausnahme des Neue-Zeile-Zeichens. Runde Klammern für die Gruppierung und der senkrechte Strich für die logische Oder-Verknüpfung sind ebenfalls zulässig. Die Konstruktion $r\{d,e\}$ in einer Regel bedeutet $d$ bis $e$ Wiederholungen des regulären Ausdrucks $r$. Diese Konstruktion hat eine höhere Priorität als |, aber eine niedrigere Priorität als *, ?, + und Verkettung (concatenation). Das Zeichen ^ am Anfang eines Ausdrucks sorgt dafür, daß eine Zeichenfolge nur unmittelbar nach einem Neue-Zeile-Zeichen erkannt wird, und das Zeichen $ am Ende eines Ausdrucks verlangt ein abschließendes Neue-Zeile-Zeichen. Das Zeichen / in einem Ausdruck bedeutet nachfolgender Kontext; nur der Teil des Ausdrucks bis zum Schrägstrich wird in yytext zurückgeliefert, aber der

## lex

übrige Teil des Ausdrucks muß im Eingabestrom darauf folgen. Ein Operatorzeichen kann als gewöhnliches Textzeichen verwendet werden, indem man es in doppelte Hochkommata einschließt oder einen Backslash (\) voranstellt.
Die Codierung [a-zA-Z]+ erkennt eine Folge von Buchstaben.

Es werden drei als Makros definierte Unterprogramme erwartet: input() zum Lesen eines Zeichens, unput(c) zum Ersetzen eines gelesenen Zeichens und output(c) zur Ausgabe eines Zeichens. Diese Unterprogramme sind standardmäßig definiert. Sie können sie jedoch außer Kraft setzen. Das generierte Programm trägt den Namen yylex(), und die Bibliothek enthält ein Hauptprogramm main(), das dieses Programm aufruft. Die auf der rechten Seite einer Regel codierte Aktion REJECT bewirkt, daß diese gefundene Übereinstimmung zurückgewiesen und die nächste passende Übereinstimmung ausgeführt wird. Die Funktion yymore() stellt weitere Zeichen in yytext, und die Funktion yyless(p) schiebt den Teil der gefundenen Zeichenfolge ab p wieder zurück, der zwischen yytext und yytext+yyleng sein sollte. Die Makros input und output benutzen die Dateien yyin und yyout zum Lesen bzw. Schreiben. Diese Dateien sind standardmäßig stdin bzw. stdout zugeordnet.

Beginnt eine Zeile mit einem Blank, nimmt das Programm an, daß diese Zeile nur C-Text enthält und wird kopiert. Wenn ihr %% vorangeht, wird sie in den externen Definitionsbereich der Datei lex.yy.c kopiert. Die Regeln sollten – wie bei yacc – dem %% folgen. Zeilen, die vor dem %% stehen und mit einem Zeichen ungleich Blank beginnen, bedeuten, daß die links stehende Zeichenkette den Rest der Zeile definiert; er kann später aufgerufen werden, indem er durch geschweifte Klammern eingeschlossen wird. Beachten Sie bitte, daß geschweifte Klammern keine runden Klammern implizieren; es erfolgt lediglich eine String-Substitution.

## lex

BEISPIEL:

```
D       [0-9]
%%
if      printf("IF-Anweisung\n");
[a-z]+  printf("Textwert %s\n",yytext);
O{D}+   printf("Oktalzahl %s\n",yytext);
{D}+    printf("Dezimalzahl %s\n",yytext);
"++"    printf("unärer Operator\n");
"+"     printf("binärer Operator\n");
"/*"    {       loop:
                while (input() !='*');
                switch (input())
                        {
                        case'/':break;
                        case'*':unput('*');
                        default:go to loop;
                        }
        }
```

Die von lex generierten externen Namen beginnen alle mit dem Präfix yy oder YY. Die Optionen müssen vor den Dateien angegeben werden.

Optionen:

-r RATFOR-Aktionen.

-c C-Aktionen (Standard).

-t Ausgabe des Programms lex.yy.c auf Standardausgabe.

-v Erstellung einer einzeiligen statistischen Zusammenfassung des generierten Automaten.

-n Ausgabe der Zusammenfassung wird unterdrückt.

Mehrere Dateien werden als eine Datei behandelt. Geben Sie keine Dateien an, wird die Standardeingabe gelesen.

## lex

Bestimmte Tabellengrößen für den entstehenden endlichen Automaten können im Definitionsteil gesetzt werden:

%p $n$ Anzahl Positionen ist $n$ (Standard: 2000).

%n $n$ Anzahl Zustände ist $n$ (Standard: 500).

%t $n$ Anzahl der Knoten im Parser-Baum ist $n$ (Standard: 1000).

%a $n$ Anzahl der Übergänge ist $n$ (Standard: 3000).

Geben Sie mindestens eine der o. g. Angaben vor, so impliziert dies automatisch -v, außer wenn die Option -n angegeben wird.

SIEHE AUCH:

yacc
Systemliteratur TARGON: „Textmustererkennung und Verarbeitung"

# line

line – Lesen einer Zeile

SYNTAX:

line

BESCHREIBUNG:

Line kopiert eine Zeile aus der Standardeingabe und schreibt sie in die Standardausgabe.
Nach Eingabe von EOF liefert line den Rückgabe-Code 1.

Line wird in Shell-Dateien häufig benutzt, um Eingaben des Benutzers zu lesen.

SIEHE AUCH:

sh

# lint

lint – Prüfung der Syntax und Semantik von C-Programmen

SYNTAX:

lint [*Optionen*] [*Datei* ...]

BESCHREIBUNG:

Das Dienstprogramm lint überprüft C-Programme und ermittelt Fehlerursachen und nicht portierbare oder überflüssige Teile. Es überprüft die Typverknüpfungen strenger als der C-Compiler selbst. Außerdem meldet es u. a. unerreichbare Befehle; Schleifen, die nicht an ihrem Anfang gestartet werden, nicht benutzte, aber definierte Variablen der Speicherklasse auto und logische Ausdrücke, deren Wert konstant ist. Der Aufruf von Funktionen wird ebenfalls überprüft. Hier wird im einzelnen festgestellt, ob Funktionen in einigen Fällen Werte zurückliefern und in anderen nicht, ob Funktionen mit unterschiedlicher Anzahl von Parametern aufgerufen werden und ob Funktionswerte nicht benutzt werden.

Standardmäßig wird angenommen, daß alle Dateien zusammen ein Programm bilden, sie werden auf gegenseitige Kompatibilität überprüft. Normalerweise benutzt lint die Funktionsdefinitionen aus der lint-Standardbibliothek llib-lc.ln, haben Sie die -p-Option angegeben, werden die Funktionsdefinitionen aus der portablen lint-Bibliothek llib-port.ln benutzt.

Im lint-Kommando können beliebig viele Optionen in beliebiger Reihenfolge angegeben werden; jede Option schließt bestimmte Arten von lint-Beanstandungen aus:

-a Keine Beanstandung von Zuweisungen von 'long'-Werten an nicht-'long'-Variablen.

-b Keine Beanstandung von break-Anweisungen, die nicht erreicht werden (lex- und yacc-Programme erzeugen häufig diese Meldung).

-h Mit dieser Option werden heuristische Tests unterdrückt, die Fehler erkennen, Überflüssiges reduzieren und den Programmierstil verbessern.

## lint

-l*x* Einbinden von zusätzlichen lint-Bibliotheken. Z. B. binden Sie durch Eingabe von -lm die mathematische Bibliothek llib-lm.ln ein. Die Standardnutzung von llib-lc.ln bleibt jedoch – auch bei Angabe von -l*x* – erhalten.

-n Keine Überprüfung der Kompatibilität mit der Standard- bzw. portablen lint-Bibliothek.

-p Überprüfung der Portabilität auf andere C-Dialekte.

-u Keine Beanstandung von verwendeten, aber nicht deklarierten und deklarierten, aber nicht verwendeten Funktionen und externen Variablen (sinnvoll, wenn lint für Dateien aufgerufen wird, die Teil eines größeren Programms sind).

-v Keine Beanstandung von unbenutzten Argumenten in Funktionen.

-x Keine Beanstandung von deklarierten, aber unbenutzten externen Variablen.

Auch die -D, -U und -I-Optionen des cc-Kommandos werden von lint berücksichtigt.

Lint reagiert auch auf einige bestimmte Kommentare in Ihrem C-Programm:

/*NOTREACHED*/ Ab dieser Stelle wird die Beanstandung von unerreichbaren Anweisungen gestoppt.

/*VARARGS*n*/ Die normale Überprüfung auf variable Anzahl von Parametern in der folgenden Funktionsdeklaration wird unterdrückt. Die Datentypen der ersten *n* Argumente werden geprüft; ein fehlendes *n* wird als 0 interpretiert.

/*ARGSUSED*/ Die -v-Option wird für die nächste Funktion in Kraft gesetzt.

/*LINTLIBRARY*/ Dieser Kommentar am Anfang einer Datei unterdrückt Meldungen über unbenutzte Funktionen in dieser Datei.

## lint

DATEIEN:

| | |
|---|---|
| /usr/lib | Verzeichnis für die lint-Bibliotheken |
| /usr/lib/lint[12] | Erster und zweiter Schritt von lint |
| /usr/lib/llib.lc.ln | Deklarationen für C-Bibliotheksfunktionen (Format ist binär; die Quelle steht in /usr/lib/llib-lc) |
| /usr/lib/llib-port.ln | Deklarationen für portable Funktionen (Format ist binär; die Quelle steht in /usr/lib/llib-port) |
| /usr/lib/llib-lm.ln | Deklarationen für mathematische Bibliotheksfunktionen (Format ist binär; die Quelle steht in /usr/lib/llib-lm) |
| /usr/tmp/*lint* | Temporäre Dateien |

SIEHE AUCH:

cc
Systemliteratur TARGON /35: „Die Sprache C"

# ln

## ln – Erstellen einer Verknüpfung (Link)

SYNTAX:

ln [-s] *Name1* [*Name2*]

ln [-c] *u1Name:Name1* [*u2Name:Name2*] ... *Name*

ln *Name* ... *Verzeichnis*

BESCHREIBUNG:

Eine Verknüpfung (Link) ist ein Verzeichniseintrag, der sich auf eine Datei bezieht; jede einzelne Datei (einschließlich ihrer Größe, ihrer Zugriffsrechte usw.) kann mehrere Verknüpfungen haben. Derzeit gibt es drei Arten von Verknüpfungen: feste Verknüpfungen, symbolische Verknüpfungen und bedingte symbolische Verknüpfungen.

Standardmäßig stellt ln feste Verknüpfungen her. Es gibt keine Möglichkeit, einen festen Link von dem ursprünglichen Verzeichniseintrag einer Datei zu unterscheiden. Jede Veränderung einer Datei bezieht sich automatisch auf alle zugehörigen Verknüpfungen, gleichgültig mit welchem Namen die Datei angesprochen wird. Feste Verknüpfungen können nicht über Dateisysteme hinwegreichen und können sich nicht auf Verzeichnisse beziehen.

Mit der Option -s erstellt ln symbolische Verknüpfungen (Links). Ein symbolischer Link enthält den Namen der Datei, mit der er verknüpft ist. Die angesprochene Datei wird benutzt, wenn für die Verknüpfung ein open (Systemaufruf) oder creat (Systemaufruf) ausgeführt wird. Ein stat (Systemaufruf) für einen symbolischen Link gibt die verknüpfte Datei zurück; ein lstat (Systemaufruf) muß abgesetzt werden, um Informationen über die Verknüpfung zu erhalten. Mit readlink (Systemaufruf) kann der Inhalt einer symbolischen Verknüpfung ausgelesen werden. Symbolische Verknüpfungen können über Dateisysteme hinausgehen und können sich auf ein Verzeichnis beziehen.

## ln

Wenn ein oder zwei Argumente gegeben sind, stellt ln eine Verknüpfung mit einer vorhandenen Datei *Name1* her. Ist *Name2* angegeben, bekommt die Verknüpfung diesen Namen; *Name* kann auch ein Verzeichnis sein, in das die Verknüpfung eingetragen werden soll; anderenfalls wird die Verknüpfung in das aktuelle Verzeichnis eingetragen. Wenn nur das Verzeichnis angegeben ist, wird die Verknüpfung mit der letzten Komponente von *Name1* hergestellt.

Haben Sie mehr als zwei Argumente angegeben, stellt ln Verknüpfungen mit allen angegebenen Dateien im angegebenen Verzeichnis her. Die Verknüpfungen bekommen dieselben Namen wie die Dateien, die verknüpft werden.

Bei Angabe der Option -c stellt ln eine bedingte symbolische Verknüpfung wie folgt her:

ln -c *u1Name:Name1* [*u2Name:Name2*] ... *Name*

Sind beispielsweise zwei Umgebungen „att" und „ucb" vorhanden, kann man das Verzeichnis /bin wie folgt mit beiden Umgebungen verknüpfen:

ln -c att:/.attbin ucb:/.ucbbin /bin

Auf diese Weise wird /bin – wenn Sie sich im att-Universum befinden – durch /.attbin ersetzt und im ucb-Universum durch /.ucbbin. In diesem Beispiel ist /bin eine bedingte symbolische Verknüpfung zu den tatsächlichen Inhaltsverzeichnissen /.attbin und /.ucbbin.

SIEHE AUCH:

cp, rm, mv

# login

## login – Anmelden in das System

SYNTAX:

login [*Name* [ *Umgebungsvariable* ...]]

BESCHREIBUNG:

Mit dem Kommando login beginnen Sie eine Terminalsession und identifizieren sich gegenüber dem System. Login kann als Kommando oder vom System aufgerufen werden, wenn eine Verbindung zum ersten Mal hergestellt wird. Es wird auch vom System aufgerufen, wenn ein Benutzer die anfängliche Shell durch die Eingabe von CTRL-d beendet hat, um ein EOF zu signalisieren.

Rufen Sie login als Kommando auf, muß es den anfänglichen Kommando-Interpreter ersetzen. Dazu müssen Sie folgenden Befehl angeben:

exec login

Login wird von getty zu folgenden Zwecken aufgerufen:

1. Paßwortprüfung des Benutzers. Erlaubt oder verbietet dem Benutzer den Systemzugang.

   Rufen Sie login ohne Argument auf, fragt es nach dem Benutzernamen und ggf. nach dem Paßwort. Das eingegebene Paßwort wird (wenn möglich) nicht ausgegeben und erscheint somit nicht im gedruckten Protokoll der Sitzung.

   An einigen Anlagen kann eine Option aktiviert werden, die die Eingabe eines zweiten Wähl-Paßworts (dialup password) verlangt. Dies kommt nur bei Wählverbindungen vor und wird mit der Meldung "dialup password:" angefordert. Zum erfolgreichen Anmelden müssen beide Paßwörter eingegeben werden.

   Ist die Anmeldung nicht innerhalb einer gewissen Frist (z. B. einer Minute) erfolgreich abgeschlossen, wird die Verbindung zum System unter Umständen kommentarlos getrennt.

2. Einstellen des Login-Universums für den Benutzer.

## login

3. Prüfen, ob Post vorhanden ist.

   Nach erfolgreicher Anmeldung werden Abrechnungsdateien aktualisiert, und dem Benutzer wird mitgeteilt, ob Post für ihn vorhanden ist.

4. Ausgeben der Nachricht des Tages.

   Nach einer erfolgreichen Anmeldung wird die Nachricht des Tages ausgegeben.

5. Starten der Shell des Benutzers (in der Regel /bin/sh).

   Login initialisiert die Benutzer- und Gruppen-ID sowie das Arbeitsverzeichnis und führt sodann entsprechend den Spezifikationen in der Paßwort-Datei einen Kommando-Interpreter (normalerweise sh) aus. Das Argument 0 des Kommando-Interpreters ist -sh oder, allgemeiner gesagt, der Name des Kommando-Interpreters mit einem vorangestellten Minuszeichen (-). Login wird von sh erkannt.

Desweiteren führt login die Prozedur /etc/profile durch. Die Datei .profile im Arbeitsverzeichnis wird, sofern sie existiert, ausgeführt. Diese Spezifikationen finden sich in dem Eintrag für den Benutzer in der Datei /etc/passwd. Der Name des Kommando-Interpreters lautet -, an den sich die letzte Komponente des Pfadnamens des Interpreters anschließt (d. h. -sh). Wenn dieses Feld in der Paßwortdatei leer ist, wird der standardmäßige Kommando-Interpreter /bin/sh verwendet.

Login greift auf die folgenden Dateien zu, um die entsprechenden Aufgaben auszuführen:

| | |
|---|---|
| /etc/ttytype | Stellt die TERM-Variable bereit. Login setzt TERM aufgrund dieser Datei. TERM wird auf su gesetzt, wenn für die Leitung kein Eintrag vorhanden ist. |
| /etc/ttys | Login benutzt diese Datei für die Abrechnung der Login-Sätze für das ucb-Universum (/etc/utmp). Ist der Terminalname in /etc/inittab und /etc/ttys nicht identisch, weist login möglicherweise den Anmeldeversuch des Benutzers zurück. |

## login

/etc/u_universe — Aufgrund dieser Datei setzt login das Login-Universum des Benutzers. Der Benutzer wird automatisch im ucb-Universum angemeldet, wenn:

1. kein Eintrag für den Benutzer vorhanden ist,
2. der Name des Universums weder att noch ucb ist,
3. die Datei /etc/u_universe fehlt.

/etc/utmp — Enthält Informationen über angemeldete Benutzer. /etc/utmp ist eine bedingte symbolische Verknüpfung mit /etc/.ucbutmp. Das utmp von System V gibt außerdem an:

1. wann die Maschine gebootet wurde und
2. wieviele getty-Prozesse darauf warten, daß sie login-Prozesse werden.

/etc/wtmp — Enthält kumulative Abrechnungssätze. Die Programme init, getty, login und shutdown schreiben Sätze in /etc/wtmp.

Ferner werden die folgenden Dateien verwendet:

| | |
|---|---|
| /etc/mail | Post |
| /etc/motd | Nachricht des Tages |
| /etc/passwd | Paßwort-Informationen |

Die Basis-Umgebung wird wie folgt initialisiert:

| | |
|---|---|
| HOME | Ihr Login-Verzeichnis |
| PATH | :/bin:/usr/bin |
| SHELL | Eintrag für letztes Feld des Paßworts |
| MAIL | /usr/mail/Ihr Login-Name |
| TZ | Angabe der Zeitzone |

## login

Die Umgebung kann erweitert oder abgeändert werden, indem zur Ausführungszeit oder wenn login Ihren Login-Namen anfordert, weitere Argumente an login übergeben werden. Die Argumente können die Form *xxx* oder *xxx=yyy* haben. Argumente ohne Gleichheitszeichen werden als

L*n*=*xxx*

in die Umgebung eingebracht, wobei *n* eine Nummer ist, die bei 0 beginnt und jedesmal erhöht wird, wenn ein neuer Variablenname benötigt wird. Variablen mit Gleichheitszeichen werden unverändert in die Umgebung eingebracht. Wenn sie bereits in der Umgebung vorhanden sind, ersetzen sie den älteren Wert. Hierzu gibt es zwei Ausnahmen. Die Variablen PATH und SHELL können nicht verändert werden. Dadurch wird verhindert, daß Personen, die sich in einer eingeschränkten Shell-Umgebungen anmelden, uneingeschränkte sekundäre Shells hervorbringen. Bei login und getty kann die Spezialbedeutung einiger Zeichen mit einem vorangestellten Backslash aufgehoben werden. Dadurch ist es beispielsweise möglich, Leerzeichen und Tabs zu schreiben.

DIAGNOSE:

Login incorrect:
Wenn der Benutzername oder das Paßwort nicht gefunden wird.

No shell, cannot open password file oder no directory:
Lassen Sie sich von einem Fachmann für UNIX-Systemprogrammierung beraten.

No utmp entry. You must exec login from the lowest level sh:
Wenn Sie versucht haben, login als Kommando auszuführen, ohne das interne Shell-Kommando exec zu benutzen, oder wenn versucht wurde, sich aus einer anderen als der anfänglichen Shell anzumelden.

SIEHE AUCH:

mail, newgrp, sh, su

# logname

logname – Anzeige des Login-Namens

SYNTAX:

logname

BESCHREIBUNG:

Logname zeigt Ihnen den Login-Namen, mit dem Sie sich in das System angemeldet haben, an.

DATEI:

/etc/profile

SIEHE AUCH:

env, login

## lorder

lorder – Ausgabe von Referenzlisten für Objekt- oder Archivdateien

SYNTAX:

lorder *Datei* ...

BESCHREIBUNG:

Als Eingabe dienen eine oder mehrere Objekt- oder Archivdateien. Standardausgabe ist eine Tabelle mit Objektdatei-Paaren, von denen die erste Datei auf eine externe Referenz (identifier) hinweist, die in der zweiten Spalte definiert ist. Diese Ausgabe können Sie mit dem Befehl tsort sortieren.

Normalerweise benötigt der Link-Editor beim Aufbau eines Archives nicht den Befehl lorder; jedoch wird hiermit während des Link-Editor-Prozesses ein noch schnellerer Zugriff auf das Archiv ermöglicht.

Im folgenden Beispiel wird eine neue Bibliothek aus vorhandenen Dateien, deren Namen auf .o enden, erstellt:

ar cr *Bibliothek* 'lorder *.o | tsort'

HINWEIS:

Enden Namen von Objektdateien nicht auf .o, werden sie – sogar wenn sie in Bibliotheksarchiven enthalten sind – übersehen. Ihre Symbole und Referenzen werden anderen Dateien zugeschrieben.

SIEHE AUCH:

ar, ld, tsort

# lp

## lp – Druckauftrag an Drucker-Spooler senden

SYNTAX:

lp [-c] [-d*Ziel*] [-m] [-n*Anzahl*] [-o*Option*] [-s] [-t*Titel*] [-w] [*Datei* ...]

BESCHREIBUNG:

Die mit lp abgesetzten Druckaufträge für die angegebenen Dateien werden vom Drucker-Spooler übernommen und an einen Drucker weitergeleitet. Geben Sie keine Dateien an, wird die Standardeingabe gelesen. Die Dateien werden in der von Ihnen angegebenen Reihenfolge gedruckt.

Lp erzeugt für jeden angenommenen Druckauftrag eine eindeutige Auftragsnummer und gibt diese nach Standardausgabe aus. Diese Nummer können Sie dazu verwenden, den Auftrag zu stornieren (s. cancel) oder den Status des Auftrags festzustellen (s. lpstat).

Die folgenden Optionen können in beliebiger Reihenfolge für verschiedene Dateinamen verwendet werden:

-c Es werden Kopien von den zu druckenden Dateien erstellt. Geben Sie diese Option nicht an, werden die Dateien nur gelinkt, d. h. jede Veränderung der Datei, die Sie zwischen Druckauftrag und Druck vornehmen, wird in den Ausdruck aufgenommen. Insbesondere sollten Sie keine Datei löschen, bevor sie vollständig gedruckt ist.

-d*Ziel* Ist *Ziel* ein Drucker, wird der Auftrag nur auf dem spezifizierten Drucker ausgedruckt. Ist *Ziel* eine Klasse von Druckern, wird der Auftrag auf dem ersten freien Drucker dieser Klasse ausgegeben.

Unter verschiedenen Bedingungen werden Aufträge für ein *Ziel* abgelehnt (s. lpstat). Geben Sie die -d-Option nicht an, wird *Ziel* aus der Umgebungsvariablen LPDEST entnommen (wenn sie gesetzt ist). Anderenfalls wird ein System-Default-Ziel benutzt (wenn es definiert ist). Die Namen für *Ziel* sind von System zu System verschieden.

lp

-m Nach Beendigung des Druckauftrags wird durch mail eine entsprechende Nachricht an den Auftraggeber gesendet. Standardmäßig erfolgt bei korrekter Ausführung des Auftrags keine Nachricht.

-n*Anzahl* Der Ausdruck erfolgt mit der angegebenen Anzahl Kopien (Standard = 1).

-o*Option* Angabe von speziellen drucker- bzw. klassenabhängigen Optionen.

-s Nachrichten von lp werden unterdrückt.

-t*Titel* Ausdruck des angegebenen Titels auf der ersten Seite.

-w Durch write wird eine Meldung über die Beendigung des Druckauftrags auf Ihr Terminal geschrieben. Sind Sie zu diesem Zeitpunkt nicht angemeldet, wird die entsprechende Nachricht durch mail gesendet.

DATEIEN:

/usr/spool/lp/*

SIEHE AUCH:

cancel, enable, lpstat, mail

# lpstat

lpstat – Spoolersystem-Statusinformationen anfordern

SYNTAX:

lpstat [*Optionen*]

BESCHREIBUNG:

Lpstat gibt Informationen über den aktuellen Status des LP-Spoolersystems.
Geben Sie keine Option an, werden die Stati Ihrer – mit lp abgesetzten – Druckaufträge aufgelistet.

Jedes angegebene Argument, daß keine Option ist, wird als Auftragsnummer (request id) interpretiert. Lpstat listet dann den Status dieser Aufträge.
Optionen können Sie in beliebiger Reihenfolge und zwischen Auftragsnummern angeben. Einige Optionen bieten die Möglichkeit, mit optionalen Listen zu arbeiten. Diese Listen können Sie in folgenden Formaten angeben:

- Liste von Argumenten, die durch Kommata voneinander getrennt werden.
- Liste von Argumenten, die durch Kommata und/oder Leerzeichen voneinander getrennt werden. Diese Liste muß in doppelte Hochkommata eingeschlossen werden.

Beispiel:

-u"*Benutzer1*, *Benutzer2*, *Benutzer3*"

Eine solche Liste selektiert die verfügbaren Informationen auf Grund der Einträge in dieser Liste, d. h. fehlt die Liste, werden alle Informationen ausgegeben, die durch die Option spezifiziert sind.

## lpstat

Beispiel:

```
lpstat -o
```

listet den Status aller Druckaufträge.

Optionen:

-a[*Liste*] Ausgabe des Akzeptanz-Status der in *Liste* angegebenen Drucker. *Liste* ist eine Liste von Druckernamen bzw. Drukkerklassen.

-c[*Liste*] Ausgabe der Klassennamen und ihrer zugehörigen Drucker. *Liste* ist eine Liste von Klassennamen.

-d Ausgabe des System-Default-Ziels.

-o[*Liste*] Ausgabe einer Statusübersicht der Druckaufträge. In *Liste* können Sie Druckernamen, Klassennamen und Auftragsnummern angeben.

-p[*Liste*] Status-Ausgabe der in *Liste* angegebenen Drucker.

-r Status-Ausgabe des Spooler-Schedulers.

-s Ausgabe einer Status-Übersicht, die folgende Informationen enthält:

- Status des Spooler-Schedulers,
- Status des System-Default-Ziels,
- Liste von Klassen und der zugehörigen Drucker,
- Liste von Druckern und der zugeordneten Geräte.

–t Ausgabe aller Statusinformationen.

-u[*Liste*] Status-Ausgabe der Druckaufträge für die in *Liste* angegebenen Benutzer. In *Liste* tragen Sie die gewünschten Log-in-Namen ein.

-v[*Liste*] Ausgabe der in *Liste* angegebenen Druckernamen sowie die Pfadnamen der zugeordneten Geräte.

## lpstat

DATEIEN:

/usr/spool/lp/*

SIEHE AUCH:

cancel, enable, lp

# ls

## ls – Auflisten des Inhalts von Verzeichnissen

SYNTAX:

ls [-logtasdLrucifp] [*Name*] ...

BESCHREIBUNG:

Übergeben Sie dem ls-Befehl einen Verzeichnisnamen als Argument, wird der Inhalt dieses Verzeichnisses – jeder einzelne Eintrag in einer Zeile – in alphabetischer Reihenfolge aufgelistet. Wird ein Dateiname angegeben, wiederholt ls diesen Namen und gibt zusätzlich die gewünschten Informationen aus. Geben Sie kein Argument an, wird der Inhalt des aktuellen Verzeichnisses als Standardargument eingesetzt. Bei Eingabe mehrerer Argumente werden diese zuerst sortiert, da ls die Dateiargumente vor den Verzeichnisargumenten verarbeitet.

Mit folgenden Optionen können zusätzliche Informationen aufgerufen werden:

-l Auflisten in Lang-Format. Angezeigt werden die Zugriffsrechte, Anzahl der Links, Gruppe, Eigentümer, Größe in Bytes und Zeitpunkt der letzten Änderung für jede Datei. Handelt es sich um Gerätedateien, wird statt der Größe die Geräteklasse (major) und die Geräteeinheit (minor) angezeigt. Ist eine Datei eine symbolische Verknüpfung, werden Informationen über das Ziel dieser Verknüpfung ausgegeben.

-o Identisch mit -l, jedoch wird die Gruppe nicht angezeigt.

-g Identisch mit -l, jedoch wird der Eigentümer nicht angezeigt.

-t Auflisten in zeitlicher Reihenfolge der letzten Änderungen (die letzte Änderung wird zuerst angezeigt).

-a Auflisten aller Einträge einschließlich der mit '.' beginnenden.

-s Angabe der Größe in Blöcken.

-d Ist das angegebene Argument ein Verzeichnis, wird nur der Name angezeigt. In Verbindung mit der -l-Option erhält man die Statusanzeigen des Verzeichnisses.

ls

-L Ist das Argument eine symbolische Verknüpfung, wird die Verknüpfung selbst und nicht die Datei oder das Verzeichnis, auf das sich die Verknüpfung bezieht, ausgegeben.

-r Auflisten in umgekehrter Reihenfolge des Alphabets bzw. in umgekehrter Reihenfolge des Änderungsdatums, wenn zusätzlich -t angegeben wird.

-u Auflisten in zeitlicher Reihenfolge des letzten Zugriffs (in Verbindung mit -t wird der letzte Zugriff zuerst angezeigt).

-c Sortierkriterium ist die zeitliche Reihenfolge der Änderung der Dateibeschreibung statt des letzten Änderungsdatums der Dateien.

-i Anzeigen der Dateibeschreibung in der ersten Spalte.

-f Jedes angegebene Argument wird als Verzeichnis angenommen und der Inhalt – wird als Verzeichniseinträge interpretiert – ausgegeben. Diese Option setzt -l, -t, -s und -r außer Kraft und verhält sich wie die -a-Option.

-p Anzeige des Inhalts der angegebenen Verzeichnisse. Der Name von Unterverzeichnissen wird durch einen angehängten Schrägstrich (/) gekennzeichnet.

Die einzelnen Optionen können auch kombiniert werden, um gewünschte Einzelinformationen zusammen zu erhalten.

Wird zur Auflistung die -l-Option angegeben, werden die Zugriffsrechte angezeigt. Sie umfassen zehn Zeichen:

erstes Zeichen:

d = Verzeichnis

b = Gerätedatei (blockorientiert)

c = Gerätedatei (zeichenorientiert)

p = Fifo-Datei (benannte Pipe)

\- = Datendatei

## ls

Die nächsten neun Zeichen sind als drei Sätze zu jeweils drei Zeichen zu verstehen. Der erste Satz zeigt die Rechte des Eigentümers an, der nächste Satz die Rechte der Gruppe zu der der Eigentümer gehört und der dritte Satz zeigt die Rechte der übrigen Benutzer. In jedem Satz werden die Rechte in der gleichen Reihenfolge angegeben. So ist jedem einzelnen Satz zu entnehmen, welche Rechte erteilt wurden. Folgende Rechte können erteilt oder entzogen werden:

r = Leseerlaubnis (read).

w = Schreiberlaubnis (write).

x = Ausführungserlaubnis (execute).

\- = Erlaubnis ist nicht erteilt.

t = Der Code eines Programms bleibt auch nach seiner Ausführung im Swap-Bereich (Textsegment-Bit).

s = Steht statt der Ausführungserlaubnis des Eigentümers oder der Gruppe ein 's', wird bei der Ausführung der Datei die Benutzernummer bzw. die Gruppennummer des Eigentümers benutzt und nicht die des Aufrufenden.

Bezogen auf ein Verzeichnis bezeichnet 'x' die Berechtigung, dieses Verzeichnis nach einem bestimmten Dateinamen zu durchsuchen.
Die Angabe von 's' und 't' erfolgt in Großbuchstaben, wenn die entsprechende Ausführungserlaubnis nicht erteilt wurde.

DATEIEN:

/etc/passwd Enthält die Benutzernummern

/etc/group Enthält die Gruppennummern

SIEHE AUCH:

chmod, find

# m4

## m4 – Makro-Prozessor

SYNTAX:

m4 [*Optionen*] [*Dateien*]

BESCHREIBUNG:

M4 erleichtert die Handhabung von Programmiersprachen und ermöglicht Programme lesbarer zu gestalten. M4 wird vor der Übersetzung eines Quellprogramms aufgerufen und ist programmiersprachenunabhängig. Man kann ihn z. B. sowohl für C als auch für COBOL verwenden.

Die beim Aufruf angegebenen Optionen beziehen sich auf alle zu bearbeitende Dateien. Geben Sie statt eines Dateinamens ein '-' ein, so wird die Standardeingabe verwendet.

Beim Aufruf des Makro-Prozessors m4 sind folgende Optionen zulässig:

-e Arbeitet interaktiv; Fehler führen nicht zum Abbruch. Die Ausgabe erfolgt ungepuffert. Die Verwendung dieser Option erfordert fortgeschrittene Kenntnisse über den m4-Makro-Prozessor.

-s Ermöglicht sync Ausgaben an den C-Compiler zu machen (#line...).

-B*Wert* Ändert die Größe des Rückgabe- und Argumentenpuffers vom Standardwert 4096.

-H*Wert* Ändert die Größe der symbolischen Hashtabelle vom Standardwert 199. Die Größe sollte eine Primzahl sein.

-s*Wert* Ändert die Größe des „Call-Stacks" (Standard = 100 Plätze). Makros belegen drei Plätze und Nichtmakros belegen einen Platz.

-T*Wert* Ändert die Größe des Zeichenpuffers (Standard = 512 Bytes).

## m4

Die voranstehenden Optionen sind nur wirksam, wenn sie vor den Dateinamen und auch vor den folgenden Flags angegeben werden.

-D oder -U-Flags

-D*Name*[=*Inhalt*] Definiert *Name* für das val-Makro. Ist das val-Makro nicht aufgerufen, so ist das Flag bedeutungslos.

-U*Name* Redefiniert *Name*.

Makroaufrufe haben folgendes Format:

*name(arg1,arg2,...,argn)*

Die offene Klammer muß dem Makronamen unmittelbar folgen. Wenn dem Namen eines definierten Makros kein „(" folgt, wird der Makroaufruf ohne Argumente angenommen. Als Bestandteil von Makronamen sind Buchstaben, Ziffern und Unterstriche zulässig. Das erste Zeichen eines Makronamens darf allerdings keine Ziffer sein.

Bei der Argumenteneingabe werden Leerstellen, Tabulatoren und Zeilenende nicht ausgewertet. Hochkommata haben für m4 eine besondere Bedeutung. M4 unterscheidet zwischen dem anführenden Hochkomma (') und dem schließenden Hochkomma ('). Die Hochkommata sind nicht Inhalt der Zeichenkette.

Wird ein Makroname erkannt, so werden die Argumente durch Klammerung identifiziert. Werden beim Makroaufruf weniger Argumente übergeben, als bei der Definition, so werden alle fehlenden Argumente als leer übergeben.

Makroaufrufe bestehen normalerweise aus:

1. dem Makronamen,
2. Argumenten, die durch Kommata getrennt sind.

Die beiden Elemente werden differenziert, indem man die Argumente mit einer Klammerung versieht.

## m4

Nach der Sammlung der Argumente wird der Makroname durch den Inhalt des Makros in den Eingabezeilen ersetzt. Die Eingabezeilen werden anschließend noch solange durchsucht und geändert, bis kein Makroname mehr gefunden wird.

M4 stellt die folgenden eingebauten Makros zur Verfügung. Diese Makros können durchaus redefiniert werden. Allerdings geht in diesem Fall die alte Bedeutung für den aktuellen m4-Lauf verloren.

define — Das erste Argument ist der Makroname, der durch das zweite Argument ersetzt wird. Jedes $\$n$ ($n$ ist eine Ziffer) in dem Ersetzungstext wird durch das $n$-te Argument ersetzt. Das Argument 0 ist der Makroname. Nicht angegebene Argumente werden durch eine Leerzeichenfolge ersetzt; $# wird durch die Anzahl Argumente, $* durch die Liste der Argumente, durch Kommata getrennt, ersetzt. Die Bedeutung von $@ stimmt mit der Bedeutung von $* überein, allerdings werden alle Argumente mit Anführungsstrichen versehen.

undefine — Löscht das als Argument übergebene Makro.

defn — Gibt die Definition des Arguments in Klammern zurück. Dieses Makro wird im Zusammenhang mit der Neubenennung von eingebauten Makros verwendet.

pushdef — Arbeitet wie define; allerdings wird die vorhergehende Definition gesichert.

popdef — Löscht die aktuelle Definition des Arguments bzw. der Argumente; popdef gibt die vorausgehende Definition, soweit vorhanden, aus.

ifdef — Ist das erste Argument definiert, so ist das zweite Argument der Wert, anderenfalls das dritte. Ist kein drittes Argument vorhanden, ist der Wert leer. Das Wort „unix" ist automatisch vordefiniert.

shift — Gibt alle Argumente bis auf das erste zurück. „Schiebt" die Argumentenliste um ein Element.

## m4

changequote — Tauscht die als Hochkommata definierten Symbole durch das 1. und 2. Argument aus. Die neuen Symbole dürfen höchstens fünf Zeichen lang sein. Werden keine Argumente angegeben, so werden die originalen Symbole des m4 hergestellt (also 'und').

changecom — Ändert die Kommentarkennzeichnungen (Standard = # und Zeilenende). Wird kein Argument angegeben, so ist keine Kommentarkennzeichnung möglich. Wird nur ein Argument angegeben, so wird die linke Kommentarkennzeichnung dem Argument entnommen, der rechten wird der Standardwert zugeordnet (Zeilenende). Auch hier liegt die Längenbegrenzung der Symbole bei fünf Zeichen.

divert — M4 kann zehn Ausgabeumleitungen mit den Leitungsnummern 0-9 verwalten. Als Argument kann eine Leitungsnummer angegeben werden, in die dann alle nachfolgenden Ausgaben erfolgen. Für jede angesteuerte Leitung wird eine temporäre Datei angelegt. Alle durch divert umgeleiteten Ausgaben werden zunächst in diesen temporären Dateien abgelegt. Wird das divert-Makro ohne Argumente aufgerufen, so erfolgen die folgenden Ausgaben in die Standardumleitung mit der Leitungsnummer 0.

undivert — Alle umgeleiteten Ausgaben werden in der Folge der Leitungsnummern in die Standardausgabe gestellt. Als Argument kann man eine Leitungsnummer angeben, so daß nur die Ausgaben der entsprechenden Leitungsnummer erfolgen.

divnum — Gibt die aktuelle Leitungsnummer an.

dnl — Löscht die nächsten Zeichen einschließlich Leerzeichen bis zur nächsten Textzeile.

## m4

| | |
|---|---|
| ifelse | Hat mindestens drei Argumente. Stimmen die ersten beiden Argumente überein, so ist das dritte Argument der Inhalt. Sind vier Argumente angegeben und stimmen die zwei ersten Argumente nicht überein, so ist das vierte Argument der Inhalt. Sind mehr als vier Argumente angegeben, so werden die ersten zwei Argumente verglichen. Stimmen die Argumente nicht überein, so wird das vierte Argument mit dem fünften verglichen usw. |
| incr | Vermehrt den Wert des Arguments um 1. |
| decr | Vermindert den Wert des Arguments um 1. |
| eval | Ermöglicht die Durchführung von Berechnungen. Als Operatoren sind +, -, *, /, %, ^, bitweises &, \| und ~ zugelassen. Klammern können nach den mathematischen Gesetzen verwendet werden. Oktale und hexadezimale Zahlen können wie bei der C-Programmierung definiert werden. Als zweites Argument kann die Basis angegeben werden. Der Standardwert der Basis ist 10. Das dritte Argument gibt die Mindestzahl der Ergebnisstellen an. |
| len | Liefert die Anzahl Stellen des Arguments. |
| index | Überprüft, ob das zweite Argument im ersten vorkommt. Ist dies der Fall, so wird die Position des ersten übereinstimmenden Zeichens ausgegeben. Die erste Stelle des ersten Arguments ist Position 0. |
| substr | Liefert eine Teilzeichenkette der im ersten Argument angegebenen Zeichenkette. Das zweite Argument liefert die Anfangsposition, ab der die Teilzeichenkette der Zeichenkette entnommen werden soll. Im dritten Argument ist die Anzahl Stellen angegeben, die der Zeichenkette entnommen werden sollen. Ist das dritte Argument nicht angegeben, so wird der gesamte Rest der Zeichenkette übergeben. |

## m4

| | |
|---|---|
| translit | Ermöglicht den Austausch einzelner Zeichen einer Zeichenkette. Im ersten Argument werden alle Zeichen ausgetauscht, die mit Zeichen des zweiten Arguments übereinstimmen. Jedes übereinstimmende Zeichen wird durch ein Zeichen des dritten Arguments ausgetauscht. Die Position des übereinstimmenden Zeichens in Argument zwei ist die Position, von der das Austauschzeichen dem dritten Argument entnommen wird. Ist bei einem Austauschvorgang kein entsprechendes Zeichen in Argument drei angegeben, so wird das Zeichen des ersten Arguments gelöscht. Dies gilt auch wenn kein drittes Argument übergeben wurde. |
| include | Fügt den Inhalt der Datei ein, deren Name als Argument übergeben wurde. |
| sinclude | Ist identisch mit include; allerdings werden keine Meldungen ausgegeben und die Verarbeitung wird fortgesetzt. |
| syscmd | Führt das im ersten Argument genannte Benutzerkommando aus. |
| sysval | Liefert den Rückgabewert des letzten syscmd. |
| maketemp | Fügt in die angegebene Zeichenkette, die als Argument übergeben wird, die Prozeßnummer ein. |
| m4exit | Führt zum unmittelbaren Verlassen von m4. Der Rückgabe-Code ist 1, wenn Sie dies beim Aufruf angeben. Anderenfalls ist der Rückgabe-Code 0. |
| m4wrap | Das erste Argument wird nach Erkennung des Dateiendes der Eingabe-Datei an die Ausgabedatei angehängt. Beispiel: m4wrap ('funktion()') |
| errprint | Gibt den als Argument übergebenen Text auf der Standardfehlerausgabe aus. |
| dumpdef | Gibt die aktuellen Definitionen und Werte für das angegebene Argument aus. Geben Sie kein Argument an, werden alle Definitionen und Werte ausgegeben. |
| traceon | Ruft eine Ablaufverfolgung für alle Makros auf. |
| traceoff | Hebt die Ablaufverfolgung auf. |

## m4

SIEHE AUCH:

cc, cpp
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# machid

nixdorf, pdp11, sun, u3b, u3b5, vax – Ermittlung des Prozessortyps

SYNTAX:

nixdorf

pdp11

sun

u3b

u3b5

vax

BESCHREIBUNG:

Die nachstehend aufgeführten Kommandos liefern folgende Rückgabe-Codes:

| | |
|---|---|
| pdp11 | 0, wenn Sie auf einer PDP-11/45 oder PDP-11/70 arbeiten. |
| u3b | 0, wenn Sie auf einer 3B20S arbeiten. |
| vax | 0, wenn Sie auf einer VAX-11/750 oder VAX-11/780 arbeiten. |

Arbeiten Sie nicht auf einer der o. g. Anlagen, wird ein Rückgabe-Code ungleich 0 geliefert. Diese Kommandos werden häufig bei make in Makedateien oder in Shell-Kommandoprozeduren benutzt, um die Portabilität zu erhöhen.

SIEHE AUCH:

sh, test, true

# mail, rmail

## mail, rmail – Elektronische Post

SYNTAX:

mail [-t] *Benutzername ...*

mail [-e] [-r] [-q] [-p] [-f*Datei*]

rmail [-t] *Benutzername*

BESCHREIBUNG:

Mit diesem Befehl können Sie sich mit den anderen Benutzern des Systems verständigen, indem Sie Mitteilungen schreiben und empfangen.

### Nachrichten empfangen:

Unmittelbar nach Ihrer Anmeldung im System werden Sie über vorhandene Nachrichten informiert. Diese können Sie durch Eingabe des Befehls mail ohne Argument – Zug um Zug – am Bildschirm abfragen. Die Nachrichten werden in der Reihenfolge „last in first out", d. h. die zuletzt eingegangenen Meldungen zuerst, ausgegeben. Nach jeder Mitteilung erscheint ein Fragezeichen, und Sie können angeben, was mit der betreffenden Nachricht geschehen soll:

| | |
|---|---|
| <CR> | Die nächste Nachricht wird angezeigt. |
| + | Wie <CR>. |
| d | Die Nachricht wird gelöscht und die nächste Mitteilung angezeigt. |
| p | Die Nachricht wird noch einmal angezeigt. |
| – | Die vorherige Nachricht wird noch einmal angezeigt. |
| s[*Datei ...*] | Die Nachricht wird in den angegebenen Dateien des aktuellen Verzeichnisses gespeichert. Geben Sie keine Datei an, so wird der Text in der Briefkastendatei mbox abgestellt. Anschließend verzweigt das Programm in die Shell-Kommandoebene zurück. |

## mail, rmail

| | |
|---|---|
| w[*Datei ...*] | Die Nachrichten werden ohne Angabe des Absenders und weiterer Zusätze in den angegebenen Dateien gespeichert. Geben Sie keine Datei an, werden die Texte in der Datei mbox abgestellt. Anschließend verzweigt das Programm in die Shell-Kommandoebene zurück. |
| m[*Benutzername*] | Die Nachricht wird an die angegebene(n) Person(en) weitergeleitet. Falls kein Name angegeben ist, wird die Nachricht an Sie selbst übermittelt. |
| q | Die Nachricht bleibt im elektronischen Briefkasten, und das Programm wird beendet. |
| END-Taste | Wie q. |
| x | Die Nachrichten werden unverändert in den elektronischen Briefkasten zurückgeschrieben, und das Programm wird beendet. |
| !*Shell-Befehl* | Das Programm verzweigt zur Ausführung eines Befehls vorübergehend in die Shell-Ebene zurück. |
| * | Die oben beschriebenen Anweisungen werden am Bildschirm angezeigt und erläutert. |

Die Angabe von Optionen beim Aufruf des Programms mail verändert die Anzeige der Nachrichten folgendermaßen:

| | |
|---|---|
| -r | Die Anzeige der Nachrichten erfolgt in umgekehrter Reihenfolge. Die zuerst eingegangene Post wird auch zuerst ausgegeben. |
| -q | Nach einem Unterbrechungssignal wird das Programm beendet. Standardmäßig wird bei Empfang eines Interrupt nur das Anzeigen der Nachricht abgebrochen. |
| -p | Der gesamte Inhalt des elektronischen Briefkastens wird ohne Fragezeichen am Bildschirm ausgegeben, d. h. das Programm erwartet keine weiteren, die Nachrichten betreffenden, Anweisungen. |

## mail, rmail

-f*Datei* Die angegebene Datei wird so angezeigt, als handele es sich um die Briefkastendatei.

-e Die Post wird nicht angezeigt; es werden lediglich Rückgabe-Codes geliefert. Der Rückgabe-Code ist 0, wenn Sie Post in Ihrem Briefkasten haben; ansonsten 1.

Nachrichten senden:

Wenn Sie eine Mitteilung an andere Benutzer des Systems senden wollen, müssen Sie folgende Syntax verwenden:

mail *Benutzername* ...

Die Mitteilung kann beliebig lang sein; sie wird durch ein Dateiende-Zeichen (EOF) oder eine Zeile, die lediglich einen Punkt enthält, beendet und in der entsprechenden Briefkastendatei gespeichert.
Jeder Meldung wird automatisch ein Kopftext vorangestellt, der den Namen des Absenders sowie weitere Vermerke wie Datum und Uhrzeit enthält. Geben Sie die Option -t an, enthält der vorangestellte Kopftext die Namen der Benutzer, an die diese Nachricht gesendet wurde.

Nachrichten können auch an Benutzer von Remote-Systemen gesendet werden. In diesem Fall wird der betreffende Systemname, gefolgt von einem Ausrufungszeichen (!), dem Benutzernamen vorangestellt. Die Syntax lautet:

mail *System !Benutzername*

Falls Sie keinen direkten Zugriff auf das Empfängersystem haben, können Sie die Nachricht über ein Ihnen zugängliches System umleiten, das seinerseits auf das Empfängersystem zugreifen kann.

Beispiel:

mail a!b!cde

Diese Adressierung bedeutet, daß die Nachricht über das System 'a' an den Empfänger 'cde' auf dem System 'b' gesendet wird.

Rmail dient nur zum Senden von Nachrichten. Uucp benutzt rmail als Vorsichtsmaßnahme.

## mail, rmail

DATEIEN:

| | |
|---|---|
| /etc/passwd | Zur Identifizierung von Absender und Empfänger. |
| /usr/mail/*Benutzer* | Enthält die Eingangspost des *Benutzers* (elektronischer Briefkasten). |
| $HOME/mbox | Enthält die gespeicherten Nachrichten. |
| $MAIL | Variable, die den Pfadnamen des elektronischen Briefkastens enthält. |

SIEHE AUCH:

login, uucp, write

# make

make – Pflege, Aktualisierung und Regenerierung von Programmsystemen

SYNTAX:

make [*Optionen*] [*Makrodefinitionen*] [*Zielobjekte*]

BESCHREIBUNG:

Das Folgende ist eine Kurzbeschreibung aller *Optionen* und einiger spezieller Namen:

-f*Makedatei* Name der Beschreibungsdatei. *Makedatei* ist der benutzerdefinierte Name der Beschreibungsdatei. Der Dateiname '-' kennzeichnet die Standardeingabe. Eingebaute Regeln werden, falls sie vorhanden sind, vom Inhalt der *Makedateien* überschrieben.

-p Die komplette Liste von *Makrodefinitionen* und *Zielobjekt*beschreibungen wird ausgegeben.

-i Fehlerrückgabewerte der aufgerufenen Kommandos werden ignoriert. Dieser Modus ist auch eingestellt, wenn das Schein-*Zielobjekt* .IGNORE in der Beschreibungsdatei auftaucht.

-k Im Fehlerfall werden die Tätigkeiten an diesem Eintrag eingestellt, andere Einträge werden weiterbearbeitet, wenn sie von dem fehlerhaften Eintrag nicht abhängig sind.

-s Stiller Modus. Kommandos werden nicht vor ihrer Ausführung ausgedruckt. Dieser Modus ist auch eingestellt, wenn das Schein-*Zielobjekt* .SILENT in der Beschreibungsdatei auftaucht.

-r Die eingebauten Regeln werden nicht benutzt.

-n Modus des nicht Ausführens. Die Kommandos werden ausgegeben, aber nicht ausgeführt; auch Kommandozeilen, die mit '@' beginnen, werden ausgegeben.

-b Modus zum Erreichen von Kompatibilität alter *Makedateien*.

## make

| | |
|---|---|
| -e | Umgebungsvariablen überschreiben Zuweisungen innerhalb von *Makedateien*. |
| -m | Ein Speicherauszug zeigt die Größe von Text, Daten und Stack. Die Option arbeitet nur auf Systemen mit dem getu-Systemaufruf. |
| -t | Die *Zielobjekte* bekommen ein neues Erstellungsdatum, ohne eine Generierung auszuführen. |
| -d | Modus der Fehlersuche. Eine genaue Information über die Datei und deren Erstellungsdatum wird ausgegeben |
| -q | Frage. Mit dem Rückgabe-Code Null oder ungleich Null antwortet das make-Kommando auf die Frage, ob die *Zielobjekte* aktuell oder nicht aktuell sind. |
| .DEFAULT | Wenn ein *Zielobjekt* erstellt werden muß, aber keine ausdrücklichen Kommandos oder anwendbaren, eingebauten Regeln existieren, werden die mit .DEFAULT beschriebenen Kommandos benutzt, falls sie vorhanden sind. |
| .PRECIOUS | Abhängigkeiten dieses *Zielobjektes* werden nicht gelöscht, wenn die Bearbeitung durch Abbruch oder Unterbrechung beendet wird. |
| .SILENT | Hat dieselbe Wirkung wie die Option -s. |
| .IGNORE | Hat dieselbe Wirkung wie die Option -i. |

Make führt die in der *Makedatei* enthaltenen Kommandos aus, um ein oder mehrere *Zielobjekte* zu aktualisieren. *Zielobjekt* ist typischerweise ein Programm oder ein Programmsystem. Ohne die Option -f wird in der Reihenfolge makefile, Makefile, s.makefile und s.Makefile gesucht. Die Standardeingabe wird genommen, falls *Makedatei* den Namen '-' hat. Es dürfen mehr als ein '-'-*Makedatei* Parameter-Paar auftreten.

## make

Make aktualisiert ein *Zielobjekt* nur dann, wenn abhängige Dateien neueren Datums sind als das *Zielobjekt*. Alle Dateien, die Voraussetzung für das *Zielobjekt* sind, werden rekursiv der Liste der *Zielobjekte* hinzugefügt. Bei fehlenden Dateien wird angenommen, daß sie nicht aktuell sind.

*Makedatei* enthält eine Reihe von Einträgen, die Abhängigkeiten präzisieren. In der ersten Zeile eines Eintrages steht eine nicht-leere Liste von *Zielobjekten*, die durch Leerzeichen getrennt sind, darauffolgend ein Doppelpunkt, anschließend eine möglicherweise leere Liste von vorausgesetzten oder abhängigen Dateien. Text, der nach einem Semikolon steht, und folgende Zeilen, die mit dem Tabulatorzeichen beginnen, werden als Shell-Kommandos zur Aktualisierung des *Zielobjektes* interpretiert. Die erste Zeile, die nicht mit dem Tabulatorzeichen oder dem Nummernzeichen (#) beginnt, beschreibt eine neue Abhängigkeit oder *Makrodefinition*. Shell-Kommandos können mit Hilfe von '\', gefolgt vom Zeilenendezeichen, über mehr als eine Zeile geschrieben werden. Alle durch make ausgegebenen Zeichen (außer dem einleitenden Tabulatorzeichen) werden so, wie sie sind, an die Shell weitergegeben. Deshalb ergibt der Aufruf:

```
echo a\
b
```

die Ausgabe:

```
ab
```

Kommentare werden vom Kommentarzeichen (#) und dem Zeilenendezeichen eingegrenzt.

Die folgende *Makedatei* besagt, daß adr_liste von drei Dateien lese.o, sortiere.o und schreibe.o abhängt und diese von ihren korrespondierenden Quelldateien sowie der gemeinsamen Datei hilf.h:

## make

```
adr_liste:    lese.o sortiere.o schreibe.o
        cc lese.o sortiere.o schreibe.o -o adr_liste
lese.o:           lese.c hilf.h
        cc -c lese.c
sortiere.o:   sortiere.c hilf.h
        cc -c sortiere.c
schreibe.o:   schreibe.c hilf.h
        cc -c schreibe.c
```

Kommandozeilen werden einzeln durch jeweils eine eigene Shell ausgeführt. Das erste oder die ersten beiden Zeichen in einem Kommando können wie folgt sein: '-', '@', '-@' oder '@-'. Bei '@' wird die Ausgabe der Kommandozeile unterdrückt. Auftretende Fehler werden von make bei '-' ignoriert. Eine Kommandozeile wird vor ihrer Ausführung ausgegeben, solange nicht die Option -s oder der Eintrag .SILENT: in *Makedatei* gesetzt ist oder zu Beginn der Zeile ein '@' steht. Die Option -n ruft Kommandozeilenausgabe ohne Ausführung der Kommandos auf; beinhaltet eine Kommandozeile die Zeichenkette $(MAKE), so wird diese Zeile in jedem Fall ausgeführt (siehe auch Diskussion des MAKEFLAGS-Makros unter Systemumgebung). Die Option -t aktualisiert das Erstellungsdatum, ohne daß Kommandos ausgeführt werden.

Kommandos mit einem Rückgabe-Code ungleich Null beenden normalerweise die Ausführung von make. Bei der Option -i, dem Eintrag .IGNORE: in *Makedatei* oder einer vorangestellten Zeichenkette, die '-' enthält, werden auftretende Fehler ignoriert. Bei der Option -k wird im Fehlerfall die Bearbeitung des aktuellen Eintrags beendet, sie wird aber in Einträgen, die von dem fehlerhaften Eintrag nicht abhängig sind, fortgesetzt.

Die Option -b ermöglicht die korrekte Benutzung alter *Makedateien* (solche, die für die alte Version von make geschrieben sind). Der Unterschied zwischen der alten Version von make und dieser Version liegt darin, daß bei dieser Version jeder Abhängigkeitszeile ein leeres oder implizites Kommando folgt. Die vorhergehende Version von make setzte voraus, daß falls kein Kommando ausdrücklich aufgerufen wurde, auch kein Kommando ausgeführt wurde.

Bei Unterbrechung oder Abbruch wird das *Zielobjekt* gelöscht, wenn das *Zielobjekt* nicht von dem speziellen Namen .PRECIOUS abhängt.

# make

## Systemumgebung

Die Systemumgebung wird von make gelesen. Alle Variablen werden als Makrodefinitionen vorausgesetzt und als solche verarbeitet. Die Umgebungsvariablen werden vor jeder *Makedatei* und nach den internen Regeln verarbeitet; deshalb überschreiben Makrozuweisungen in einer *Makedatei* die Umgebungsvariablen. Die Option -e bewirkt das Überschreiben der Makrozuweisungen einer *Makedatei* durch die Systemumgebung.

Die Umgebungsvariable MAKEFLAGS wird von make ausgeführt, als enthielte sie legale Eingabe-Optionen (außer -f, -p und -d), definiert für eine Kommandozeile. Von make wird eine solche Variable „erfunden", wenn es sie nicht gibt. Make stellt die aktuellen Optionen hinein und übergibt sie beim Aufruf von Kommandos. Deshalb enthält MAKEFLAGS immer die aktuellen Optionen. Dies ist nützlich bei „Super-makes". Tatsächlich wird, wie oben beschrieben, bei der Option -n das Kommando $(MAKE) in jedem Fall ausgeführt; damit zeigt ein make -n rekursiv für ein komplettes Software-System an, was noch ausgeführt werden müßte. Das liegt daran, daß die Option -n in MAKEFLAGS gestellt und an die weiteren Aufrufe durch $(MAKE) übergeben wird. So gibt es eine Möglichkeit, um die Makedateien eines Software-Systems auf Fehler zu untersuchen, ohne irgendetwas zu aktualisieren.

## Makros

Einträge der Form *Zeichenkette1=Zeichenkette2* heißen Makrodefinitionen. *Zeichenkette2* ist definiert als alle Zeichen bis zu einem Kommentarzeichen oder einem Zeilenendezeichen. Ein späteres Auftreten von $(*Zeichenkette1*[:*Unterkette1*=[*Unterkette2*]]) wird durch *Zeichenkette2* ersetzt. Die Klammern sind optional, wenn es sich um einen einbuchstabigen Makronamen handelt und es keine Ersetzungszeichenkette gibt. Die Option *Unterkette1=Unterkette2* ist eine Ersetzungszeichenkette. Wenn sie aufgeführt ist, werden alle nicht überlappenden Vorkommen von *Unterkette1* in diesem Makro durch *Unterkette2* ersetzt. Zeichenketten für diesen Zweck sind begrenzt durch Leer-, Tabulator oder Zeilenendezeichen auf der einen und dem Beginn der Zeile auf der anderen Seite. Ein Beispiel für die Benutzung von Ersetzungszeichenketten findet sich im Abschnitt „Bibliotheken".

## make

### Interne Makros

Es gibt fünf intern geführte Makros, die hilfreich beim Schreiben von Regeln zur Bildung von *Zielobjekten* sind.

$* Das Makro $* steht für den Dateinamen der aktuellen Abhängigkeit ohne die Endung. Es wird nur bei Ableitungsregeln ausgewertet.

$@ Das Makro $@ steht für den vollständigen Namen des aktuellen *Zielobjektes*. Es wird nur bei ausdrücklich benannten Abhängigkeiten ausgewertet.

$< Das Makro $< wird nur für Ableitungsregeln und die .DEFAULT-Regel ausgewertet. Es handelt sich um das Modul, welches in Bezug auf das *Zielobjekt* nicht aktuell ist (z. B. den Namen der zu erstellenden abhängigen Datei). So wird in der .c.o-Regel das Makro $< als die .c-Dateien interpretiert. Nachfolgend ein Beispiel für das Herstellen von optimierten .o-Dateien aus .c-Dateien:

```
.c.o:
    cc -c -O $*.c
```

oder

```
.c.o:
    cc -c -O $<
```

$? Das Makro $? wird benutzt, wenn ausdrückliche Regeln der *Makedatei* ausgewertet werden. Dies ergibt eine Liste von Voraussetzungen, die in Bezug auf das *Zielobjekt* nicht aktuell sind; eigentlich sind es die Moduln, die aktualisiert werden müssen.

$% Das Makro $% wird nur dann ausgewertet, wenn das *Zielobjekt* ein Archivbibliotheks-Inhalt der Form adr(lese.o) ist. In diesem Fall wird $ als adr und $% als der Bibliotheksinhalt lese.o ausgewertet.

Vier der fünf Makros können eine Ergänzung haben. Ein großes D oder F, angehängt an eines der vier Makros, ändert die Bedeutung dieses Makros in Verzeichnisteil bei D und Dateiteil für F. So zeigt das Makro $(@D) auf den Verzeichnisteil der Zeichenkette $@. Gibt es keinen Verzeichnisteil, so wird ./ erzeugt. Das Makro $? ist von dieser zusätzlichen Bedeutung ausgeschlossen.

## make

### Endungen

Bestimmte Namen, vor allem solche, die mit .o enden, haben ableitbare Voraussetzungen wie .c, .s etc. Gibt es keine Aktualisierungskommandos für solche Dateien in *Makedatei*, aber eine ableitbare Voraussetzung erfüllt ist, so wird die Voraussetzung zum Herstellen des *Zielobjektes* ausgeführt. Für diesen Fall besitzt make Ableitungsregeln, die das Herstellen von Dateien aus anderen Dateien mit Hilfe der Endungen und nach Ermittlung der geeigneten Ableitungsregel ermöglichen. Zur Zeit gibt es solche Ableitungsregeln für die Endungen:

.c .c~ .sh .sh~ .c.o .c~.o .c~.c .s.o .s~.o .y.o .y~.o
.l.o .l~.o .y.c .y~.c .l.c .c.a .c~.a .s~.a .h~.h

Diese internen Regeln für make befinden sich in der Quelldatei rules.c. Die Regeln können lokal geändert werden. Um die Regeln, die in make zusammengestellt sind, in einer passenden Form für den Gebrauch auszudrucken, wird folgendes Kommando benutzt:

```
make -fp - 2>/dev/null </dev/null
```

Eine Tilde (~) in den o. g. Regeln verweist auf eine SCCS-Datei (siehe sccs). So würde die Regel .c~.o eine SCCS C-Quelldatei in eine Objektdatei (.o) überführen. Da s. eine Kennung von SCCS-Dateien ist, ist sie, vom Standpunkt der make-Endungen ausgehend, unverträglich.

Eine Regel mit nur einer Endung, z. B. .c:, definiert das Herstellen von x aus x.c. Alle anderen Endungen sind hierbei nicht wirksam. Dies ist nützlich für das Herstellen von *Zielobjekten* aus nur einer Quelldatei (z. B. Shell-Prozeduren oder einfache C-Routinen).

Zusätzliche Endungen können als Abhängigkeiten von .SUFFIXES definiert werden. Dabei ist die Reihenfolge maßgeblich; auf den ersten möglichen Namen, für den es sowohl eine Datei als auch eine Regel gibt, wird als Voraussetzung geschlossen. Die Voreinstellungsliste umfaßt:

```
.SUFFIXES: .o .c .y .l .s
```

## make

Auch hierfür druckt das o. g. Kommando die Liste der Endungen, die auf der Maschine implementiert sind. Mehrfache Endungslisten addieren sich; .SUFFIXES: ohne Abhängigkeit löscht die Liste der Endungen.

### Ableitungsregeln

Das erste Beispiel kann in der folgenden Form wesentlich kürzer geschrieben werden:

```
adr_liste:    lese.o sortiere.o schreibe.o
        cc lese.o sortiere.o schreibe.o -o adr_liste
lese.o sortiere.o schreibe.o:    hilf.h
```

Das liegt daran, daß make eine Sammlung interner Regeln zur Bildung von *Zielobjekten* besitzt. Sie können Regeln zu dieser Sammlung hinzufügen, indem Sie sie einfach in die *Makedatei* schreiben.

Bestimmte Makros werden von voreingestellten Ableitungsregeln benutzt, um die Einbeziehung von optionalen Gegebenheiten in die Kommandos zu erlauben. Z. B. enthalten die Makros CFLAGS, LFLAGS und YFLAGS Übersetzeroptionen für cc, lex und yacc. Auch hier wird die o. g. Methode zur Erläuterung der Regeln empfohlen.

Die Ableitung aus Voraussetzungen kann überprüft werden. Die Regel, um eine Datei mit der Endung .o aus einer Datei mit der Endung .c zu erzeugen, wird mit dem Eintrag .c.o: als *Zielobjekt* und ohne Abhängigkeit präzisiert. Shell-Kommandos, verbunden mit dem *Zielobjekt*, definieren die Regel, eine .o-Datei aus einer .c-Datei herzustellen. Jedes *Zielobjekt*, welches keinen Schrägstrich beinhaltet und mit einem Punkt beginnt, wird als Regel und nicht als echtes *Zielobjekt* identifiziert.

### Bibliotheken

Beinhaltet ein *Zielobjekt*- oder Abhängigkeitsname Klammern, wird er als Name einer Archivbibliothek angenommen; die Zeichenkette innerhalb der Klammern verweist auf ein Modul der Bibliothek. So verweisen adr(lese.o) und $(LIB)(lese.o) beide auf ein Modul lese.o in einer Archivbibliothek. (Voraussetzung ist allerdings, daß das Makro LIB vorher definiert wurde.) Der Ausdruck $(LIB)(lese.o sortiere.o) ist nicht zulás-

sig. Regeln, die sich auf Archivbibliotheken beziehen, haben die Form .*XX*a, wobei *XX* die Endung ist, die die Moduln des Archivs haben. Ein Nebenprodukt der aktuellen Implementation fordert, daß *XX* unterschiedlich zur Endung der Archiv-Moduln sein muß. Deshalb darf es keine ausdrückliche Abhängigkeit zwischen adr(lese.o) und lese.o geben. Das folgende Beispiel zeigt die gewöhnliche Benutzung der Archivschnittstelle. Hierbei wird vorausgesetzt, daß alle Dateien C-Quellcode enthalten:

```
CC = cc
LIB = adr
$(LIB): adr(lese.o) adr(sortiere.o) adr(schreibe.o)
        @echo adr ist jetzt aktuell
.c.a:
        $(CC) -c $(CFLAGS) $<
        ar rv $^ $*.o
        rm -f $*.o
```

Tatsächlich ist die oben notierte .c.a-Regel in make eingebaut und deshalb in diesem Beispiel überflüssig. Im folgenden ein sehr interessantes, aber auch begrenztes Beispiel einer Archivbibliotheks-Konstruktion:

```
$(LIB): adr(lese.o) adr(sortiere.o) adr(schreibe.o)
        $(CC) -c $(CFLAGS) $(?:.o=.c)
        ar rv $(LIB) $?
        rm $?
        @echo adr ist jetzt aktuell
.c.a:;
```

Hier wird der Ersetzungsmodus der Makro-Erweiterungen ausgenutzt. Die $?-Liste ist als die Sammlung der Objektdateinamen (innerhalb adr) definiert, deren C-Quelldateien nicht mehr aktuell sind. Der Ersetzungsmodus überführt .o nach .c (Z. Z. kann nicht gleich nach .c~ überführt werden; vielleicht wird dies in der Zukunft möglich sein.) Bemerkenswert ist die hier entbehrliche .c.a:-Regel, die jede Objektdatei, eine nach der anderen, erzeugen würde. Dieses spezielle Konstrukt beschleunigt die Pflege der Archivbibliothek beträchtlich. Das Konstrukt arbeitet wesentlich schwerfälliger, wenn die Archivbibliothek sowohl assemblierte Programme als auch C-Quellcode enthält.

## make

DATEIEN:

[Mm]akedatei und s.[Mm]akedatei

SIEHE AUCH:

cc, cd, lex, sh, yacc
Systemliteratur TARGON: „Programmentwicklungs-Tools"

## makekey

makekey – Generierung eines Verschlüsselungs-Codes

SYNTAX:

/usr/lib/makekey

BESCHREIBUNG:

Durch makekey verbessern Sie Verschlüsselungsschemata, die auf einem Code basieren, da der Zeitaufwand zur Entschlüsselung dieses Codes vergrößert wird.
Makekey liest 10 Bytes aus der Standardeingabe und schreibt 13 Bytes in die Standardausgabe.

Die ersten 8 Eingabe-Bytes (der Eingabe-Schlüssel) können beliebige ASCII-Zeichen sein. Die letzten beiden sollten dem Zeichensatz: Ziffern, Punkt (.) oder Schrägstrich (/) sowie dem Alphabet (Groß- oder Kleinbuchstaben) entnommen werden. Diese Auswahlzeichen werden in den ersten zwei ausgegebenen Zeichen wiederholt. Die restlichen 11 Ausgabezeichen stammen ebenfalls aus diesem Zeichensatz und bilden den Schlüssel.

Die Umwandlung erfolgt folgendermaßen: Die Auswahlzeichen bestimmen eine der 4096 Verschlüsselungsmaschinen, die alle auf dem National Bureau of Standards DES-Algorithmus basieren, der in 4096 verschiedene Möglichkeiten aufgespalten ist. Mit der Eingabe wird eine konstante Zeichenkette in die Maschine eingelesen und mehrere Male bearbeitet. Das Ergebnis dieser Bearbeitung (64 Bits) wird auf die 66 Bits des Ausgabeschlüssels verteilt.

Makekey ist für Programme vorgesehen, die Verschlüsselungen bearbeiten (z. B. ed).

SIEHE AUCH:

ed

## man, manprog

man, manprog – Ausgabe von Manual-Teilen

SYNTAX:

man [*Optionen*] [*Kapitel*] *Titel*

/usr/lib/manprog *Datei*

BESCHREIBUNG:

Das Kommando man lokalisiert und druckt den angegebenen Teil der Systemliteratur. Der Titel wird in Kleinbuchstaben eingetragen. Die Kapitelnummer hat unter Umständen kein alphabetisches Suffix. Ist kein Kapitel angegeben, so wird das gesamte Handbuch nach dem angegebenen Titel durchsucht und alle Einträge mit diesem Namen werden ausgegeben.

Optionen:

-t Ausgabe im Standardformat (8.5 Zoll x 11 Zoll).

-s Ausgabe in Kleinformat (6 Zoll x 9 Zoll).

-T4014 Ausgabe auf einem Terminal Tektronix 4014.

-Ttek Gleichbedeutend mit -T4014.

-Tst Ausgabe auf der MHCC STARE-Einrichtung ausdrucken (diese Option ist auf den meisten Systemen nicht verwendbar).

-Tvp Die gesetzte Ausgabe auf einem Versatec-Drucker ausdrukken; diese Option wird nicht von allen UNIX-Anlagen unterstützt.

-T*term* Den Eintrag mit Hilfe von nroff formatieren und auf der Standardausgabe (normalerweise dem Terminal) ausgeben; *term* ist der Terminaltyp; soll eine Liste der zulässigen Werte von *term* angezeigt werden, help term2 eingeben. Standardwert von *term* ist 450.

-w Auf der Standardausgabe nur die Pfadnamen der Einträge relativ zu /usr/man bzw. bei der Option -d relativ zum aktuellen Verzeichnis ausgeben.

## man, manprog

-d Das aktuelle Verzeichnis und nicht /usr/man durchsuchen; dabei muß der vollständige Dateiname (z. B. cu.1c und nicht nur cu) angegeben werden.

-12 Gibt an, daß der Manual-Eintrag mit einer Zeichendichte von 12 Zeichen pro Zoll erstellt werden soll. Ist zulässig, wenn $TERM (siehe unten) einen der Werte 300, 300s, 450 oder 1620 hat. (Der Schalter für die Zeichendichte an den Terminals DASI 300 und 300s muß manuell auf 12 eingestellt werden, wenn diese Option verwendet wird.)

-c Bewirkt, daß man das Kommando col aufruft; es wird darauf hingewiesen, daß col von man automatisch aufgerufen wird, es sei denn, *term* hat einen der Werte 300, 300s, 450, 37, 4000a, 382, 4014, tek, 1620 und X.

-y Veranlaßt man, die nicht-komprimierte Version der Makros zu verwenden.

Die genannten Optionen – außer -d, -c und -y – schließen sich gegenseitig aus, wobei allerdings die Option -s auch in Verbindung mit den ersten vier Optionen, die mit -T beginnen, verwendet werden kann. Alle anderen Optionen werden an troff, nroff bzw. das Makropaket man übergeben.

Wird mit nroff gearbeitet, untersucht man die Umgebungsvariable $TERM und versucht, Optionen für nroff auszuwählen und darüber hinaus Filter, die die Ausgabe an das jeweilige Terminal anpassen. Die Option -Tterm setzt den Wert von $TERM außer Kraft; insbesondere sollte man -Tlp angeben, wenn die Ausgabe von man an einen Zeilendrucker übergeben wird.

*Kapitel* kann vor jedem *Titel* geändert werden.

Beispiel:

man man

würde auf dem Terminal diese Beschreibung wiedergeben und darüber hinaus alle anderen Einträge mit dem Namen man, die möglicherweise in anderen Kapiteln der Literatur vorkommen.

## man, manprog

Besteht die erste Zeile der Eingabe für einen Eintrag ausschließlich aus dem String:

'\" *x*

wobei *x* eine beliebige Kombination der Zeichen e und t ist und zwischen dem Anführungszeichen (") und dem *x* genau ein Leerzeichen steht, verarbeitet man seine Eingabe mit der entsprechenden Kombination von eqn (neqn bei nroff) bzw. tbl; wenn eqn oder neqn aufgerufen werden, lesen diese automatisch die Datei /usr/pub/eqnchar.

Das Kommando man führt manprog aus, das einen Dateinamen als Argument hat. Manprog errechnet und übergibt eine Folge von drei Registerdefinitionen, die von den Formatierungsroutinen verwendet werden und das letzte Änderungsdatum der Datei angeben. Die zurückgelieferte Zeichenfolge hat die Form

-rdTag -rmMonat -ryJahr

und wird an nroff übergeben, das diesen String als Variable für das Makropaket man einsetzt. Die Monate sind von 0 bis 11 numeriert; deshalb ist Monat immer um 1 kleiner als der tatsächliche Monat. Die man-Makros errechnen den korrekten Monat. Wenn das man-Makropaket als Option für nroff/troff aufgerufen wird (d. h. nroff -man Datei), wird das aktuelle Datum in der Form Tag/Monat/Jahr als Datum gedruckt.

DATEIEN:

| | |
|---|---|
| /usr/man/man[1-8]/* | Das Nixdorf System User's Manual. |
| /usr/man/local/man[1-8]/* | Lokale Hinzufügungen. |
| /usr/lib/manprog | Errechnet die Änderungsdaten der Einträge. |

SIEHE AUCH:

eqn, nroff, tbl

# mesg

mesg – Erlaubt oder Verbietet das Senden von Nachrichten an ein Terminal

SYNTAX:

mesg [n] [y]

BESCHREIBUNG:

Rufen Sie den mesg-Befehl ohne Option auf, wird Ihnen angezeigt, ob an Ihr Terminal zur Zeit Nachrichten mit write gesendet werden dürfen oder nicht. Mit Hilfe der Optionen können Sie diesen Status ändern.

Optionen:

y Erlaubt das Senden von Nachrichten.

n Verbietet das Senden von Nachrichten. Die Schreiberlaubnis für das Terminal ist also nur dem Benutzer erteilt.

SIEHE AUCH:

write

# mkdir

mkdir – Anlegen eines Verzeichnisses

SYNTAX:

mkdir *Name* ...

BESCHREIBUNG:

Mkdir legt Verzeichnisse mit den von Ihnen angegebenen Namen an. Die Zugriffsrechte werden auf 'rwxrwxrwx' festgesetzt (kann mit umask geändert werden). Standardeinträge, wie '.' für das Verzeichnis selbst und '..' für das übergeordnete Verzeichnis, werden automatisch vorgenommen.

Mkdir verlangt Schreiberlaubnis im übergeordneten Verzeichnis.

HINWEIS:

Mkdir liefert den Rückgabewert 0, wenn die Verzeichnisse angelegt wurden. Anderenfalls wird eine Fehlermeldung ausgegeben und der Rückgabewert ist ungleich 0.

SIEHE AUCH:

rm, sh, umask

# mv

mv – Verschieben oder Umbenennen von Dateien oder Verzeichnissen

SYNTAX:

mv [-f] *Datei1* [*Datei2* ...] *Ziel*

BESCHREIBUNG:

Mv gibt *Datei1* den neuen Namen *Ziel*. Ist *Ziel* als Datei bereits vorhanden, wird sie gelöscht und erst dann wird *Datei1* in *Ziel* umbenannt.
Ist *Ziel* ein Verzeichnis, können Sie eine oder mehrere Dateien in dieses Verzeichnis verschieben.

Ist *Ziel* schreibgeschützt, so erscheint das Zugriffsrecht auf dem Bildschirm. Per Tastatur können Sie mit 'y' (yes) angeben, ob die Umbenennung trotzdem durchgeführt werden soll. Geben Sie 'n' (no) ein, wird der mv-Befehl beendet. Die Abfrage unterbleibt, wenn Sie die -f-Option benutzen oder die Standardeingabe nicht Ihr Terminal ist.
Mv kann nicht ausgeführt werden, wenn Quelldatei und Zieldatei identisch sind.

SIEHE AUCH:

cpio, rm

## newform

newform – Ändern des Formats einer Textdatei

SYNTAX:

newform [*Optionen*] [*Datei* ...]

BESCHREIBUNG:

Newform liest Textzeilen aus den angegebenen Dateien oder – wenn Sie keine Dateien angeben – von der Standardeingabe und reproduziert sie in der Standardausgabe. Die Textzeilen werden gemäß den in der Kommandozeile angegebenen Optionen reformatiert.

Mit Ausnahme von -s, das zuerst angegeben werden muß, dürfen die Optionen in jeder beliebigen Reihenfolge angegeben werden. Dabei bestimmt die Reihenfolge die Rangordnung, in der die Programme ausgeführt werden und damit auch das abschließende Ergebnis des Befehlsaufrufes. So unterscheidet sich z. B. das Resultat im Falle von -e15 -l60 von dem Ergebnis, das mit den Optionen -l60 -e15 erzielt wird.
Die in der Kommandozeile genannten Optionen gelten für sämtliche angegebenen Dateien:

-i[*Tabspez*] Eingabe-Tabulatorspezifikation. Die Tabulatoren werden auf die in *Tabspez* angegebene Leerstellenzahl ausgedehnt. *Tabspez* berücksichtigt alle Spezifikationsarten, die in tabs beschrieben sind. Darüber hinaus kann die Spezifikation in Form von '--' angegeben werden, d. h. daß die Tabulatorangabe in der ersten Zeile der Standardeingabe erfolgt (s. fspec). Die Standard-Tabulatorweite beträgt '-8'; sie wird automatisch eingesetzt, wenn *Tabspez* nicht angegeben ist. Die Spezifikation '-0' bedeutet, daß keine Tabulatoren vorhanden sind; werden dennoch welche gefunden, so werden sie wie '-1' behandelt.

-o[*Tabspez*] Ausgabe-Tabulatorspezifikation. Die Tabulatoren werden entsprechend der in dieser Option angegebenen Tabulatorspezifikation ersetzt. Sie haben hier die gleichen Möglichkeiten wie bei der Option -i*Tabspez*. Der Standardwert ist '-8'; von diesem Wert geht das Programm aus,

## newform

wenn *Tabspez* nicht angegeben ist. Die Spezifikation '-0' bedeutet, daß keine Veränderung der eingegebenen Tabulatoren bei der Ausgabe vorgenommen wird.

-l[*n*] Zeilenlänge. Mit dieser Option können Sie die tatsächliche Zeilenlänge auf *n* Zeichen festlegen. Die Zeilenlänge beträgt 72, wenn Sie die Option -l verwenden, ohne *n* anzugeben. Geben Sie diese Option nicht an, wird die Zeilenlänge automatisch auf 80 Zeichen festgesetzt.

-b[*n*] Zeilen verkürzen. Mit dieser Option können Sie angeben, um wieviel (*n*) Zeichen der Zeilenanfang gekürzt werden soll, wenn die Anzahl Zeichen pro Zeile die festgelegte Zeilenlänge übersteigt (s. -l*n*). Standardmäßig werden so viele Zeichen entfernt, bis die festgesetzte Zeilenlänge gerade erreicht ist. Das ist immer dann der Fall, wenn Sie die Option -b ohne Angabe von *n* verwenden.

Diese Option ermöglicht es Ihnen, auf folgende Weise die Numerierung aus einem COBOL-Programm zu löschen.

newform -l1 -b7 Dateiname

-l1 setzt eine Zeilenlänge fest, die kleiner ist als jede andere in der Datei vorhandene Textzeile, so daß -b aktiviert wird.

-e[*n*] Diese Option entspricht -b*n*, mit der Ausnahme, daß die Kürzung am Zeilenende erfolgt.

-c[*k*] Vorsatz/Nachsatz ändern. Der Vorsatz/Nachsatz wird in *k* geändert. Standardzeichen für *k* ist ein Leerzeichen.

-p[*n*] Zeile verlängern. Wenn die Anzahl Zeichen pro Zeile kleiner ist als die festgelegte Zeilenlänge, werden *n* Zeichen an den Zeilenanfang angefügt. Standardmäßig sind dies so viele Zeichen, wie erforderlich sind, um die definierte Zeilenlänge zu erreichen.

-a[*n*] Diese Option entspricht -p*n*, mit der Ausnahme, daß die Zeichen an das Zeilenende angefügt werden.

## newform

-f Die Formatzeile für die Tabulatorspezifikation wird vor allen anderen Zeilen in die Standardausgabe geschrieben. Das Format entspricht der in der letzten -o-Option enthaltenen Angabe. Wenn -o nicht verwendet wurde, enthält die ausgegebene Formatzeile den Standardwert '-8'.

-s Zeichen abtrennen und wieder anfügen. Die ersten Zeichen jeder Zeile werden bis zum ersten Tabulator abgetrennt und bis zu 8 dieser Zeichen wieder an das Zeilenende angefügt. Wurden mehr als 8 Zeichen abgeschnitten (den Tabulator nicht mitgezählt), so wird das achte Zeichen durch '*' ersetzt und jedes weitere Zeichen rechts davon nicht mehr berücksichtigt.
Falls in keiner Zeile der Datei ein Tabulator vorhanden ist, erfolgt eine Fehlermeldung, und das Programm wird beendet. Die abgetrennten Zeichen werden intern gespeichert und an die zuletzt reformatierte Zeile angefügt, wenn die übrigen Optionen für alle angegebenen Dateien ausgeführt sind.

Beispiel:

Umformung einer Datei mit folgenden Merkmalen:

- führende Ziffern,
- mindestens 1 Tabulator,
- Text in jeder Zeile,

in eine Datei mit den Merkmalen:

- Text am Dateianfang,
- alle Tabulatoren – mit Ausnahme des ersten – werden bis zur Spalte 72 um Leerstellen erweitert bzw. verkürzt,
- die führenden Ziffern werden versetzt, beginnend ab Spalte 73:

newform -s -i -l -a -e *Dateiname*.

SIEHE AUCH:

csplit, tabs

## newgrp

newgrp – Anmelden in eine neue Gruppe

SYNTAX:

newgrp [-] [*Gruppe*]

BESCHREIBUNG:

Newgrp ändert Ihre Gruppen-ID. Der Login-Name bleibt erhalten und das aktuelle Verzeichnis wird nicht geändert, aber die Verwaltung der Zugriffsrechte auf Dateien bezieht sich auf die neue Gruppen-ID.

Rufen Sie newgrp ohne Argument auf, wird Ihre Gruppen-ID in die ursprüngliche zurückgeändert.

Die Angabe von '-' bewirkt die Änderung der Umgebung.

Ein Paßwort wird verlangt, wenn die Gruppe ein Gruppenpaßwort besitzt und der Benutzer nicht, oder der Benutzer nicht in der Datei /etc/group als Gruppenmitglied eingetragen ist.

DATEIEN:

/etc/group

/etc/passwd

HINWEISE:

Die Eingabe von Gruppenpaßwörtern wird nicht unterstützt, da sie nur eine geringe Sicherheit bieten. Die Gruppenpaßwörter werden auf weite Sicht wahrscheinlich nicht mehr benutzt werden.

SIEHE AUCH:

login, sh

## news

news – Anzeige des Inhalts der Dateien aus /usr/news

SYNTAX:

news [-ans] [*Datei* ...]

BESCHREIBUNG:

News dient dazu, den Benutzer über aktuelle Ereignisse zu informieren. Diese Ereignisse sind in Dateien des Verzeichnisses /usr/news beschrieben.

Aufgerufen ohne Argumente, zeigt news den Inhalt aller im Verzeichnis /usr/news enthaltenen Dateien an. Dies geschieht in zeitlicher Reihenfolge. Die jüngste Datei wird zuerst ausgegeben. Jede Dateiausgabe wird angeführt von einer zugehörigen Kopfzeile.

News speichert das aktuelle Datum als das Änderungsdatum der Datei .news_time im Home-Verzeichnis jedes Benutzers; nur Dateien die jünger sind als dieses Datum, werden als aktuell eingestuft. So wird gewährleistet, daß jeder Benutzer die Nachrichten nur einmal bekommt.

Optionen:

-a Anzeige aller Dateien, ohne Prüfung der Aktualität. In diesem Fall wird die gespeicherte Zeit nicht geändert.

-n Anzeige der Namen der aktuellen Dateien. Die gespeicherte Zeit wird nicht geändert.

-s Anzeige der Anzahl der aktuellen Dateien. Namen und Inhalt werden nicht ausgegeben; die gespeicherte Zeit wird nicht geändert. Es ist sinnvoll, diesen Aufruf in die .profile-Datei des Benutzers oder in die /etc/profile-Datei des Systems einzubinden.

Alle anderen Argumente werden dahingehend interpretiert, daß sie news-Dateien sind, die angezeigt werden sollen.

## news

Geben Sie während der Anzeige einer Datei ein Abbruchsignal ein, wird diese Ausgabe abgebrochen und die Anzeige der nächsten aktuellen Datei gestartet. Folgt dem ersten Abbruchsignal innerhalb einer Sekunde ein weiteres, wird das Programm abgebrochen.

DATEIEN:

/etc/profile

/usr/news/*

$HOME/.news_time

# nice

## nice – Ausführung eines Kommandos mit niedriger Priorität

SYNTAX:

nice [-*n*] *Kommando* [*Argument* ...]

BESCHREIBUNG:

Nice weist dem angegebenen Kommando eine niedrigere CPU-Priorität zu. Die Priorität kann zwischen 1 und 19 angegeben werden (hoher Prioritätswert bedeutet niedrige Priorität); um diese Zahl wird der aktuelle Prioritätswert erhöht. Der Standardwert für *n* ist 10.

Der Superuser darf negative Werte angeben (z. B. --10), d. h. die Priorität wird entsprechend erhöht.

SIEHE AUCH:

nohup

# nl

nl – Numerierung von Textzeilen

SYNTAX:

nl [-h*Typ*] [-b*Typ*] [-f*Typ*] [-v*Startnr*] [-i*Incr*] [-p] [-l*n*]
[-s*Sep*] [-w*n*] [-n*Format*] [-d*xx*] [*Datei*]

BESCHREIBUNG:

NI liest Zeilen aus der genannten Datei oder – wenn keine Datei angegeben ist – aus der Standardeingabe und kopiert sie in die Standardausgabe. Die Zeilen werden auf der linken Seite entsprechend den verwendeten Optionen numeriert.

Die Zeilennumerierung erfolgt seitenweise, d. h. sie beginnt auf jeder Seite neu.

Eine Seite besteht aus einem Kopf-, Haupt- und Fußteil; leere Teile sind erlaubt.
Verschiedene Zeilennumerierungsoptionen können unabhängig für den Kopf-, Haupt- und Fußteil verwendet werden, wie z. B. keine Numerierung der Kopf- und Fußzeilen; Numerierung auch der Leerzeilen im Hauptteil einer Seite usw.

Jeder Seitenabschnitt (Kopf-, Haupt- und Fußteil) beginnt mit einer Eingabezeile, die ausschließlich folgende Zeichen enthalten darf:

\:\:\: (Kopfteil)
\:\: (Hauptteil)
\: (Fußteil)

Falls andere als die o. a. Zeichen eingegeben werden, interpretiert das Programm den Text vollständig als Hauptteil einer Seite.

Optionen dürfen in beliebiger Reihenfolge angegeben und im Zusammenhang mit einem einzigen optionalen Dateinamen verwendet werden.

## nl

Optionen:

-b*Typ* Diese Option gibt an, welche Hauptteilzeilen numeriert werden sollen:

Typ

| | |
|---|---|
| a | Numerierung aller Zeilen. |
| t | Numerierung nur darstellbarer Textzeilen; dies ist der Standardtyp für die Zeilennumerierung eines Seiten-Hauptteils. |
| n | Keine Numerierung. |
| p*string* | Ausschließliche Numerierung der Zeilen, die den in *string* angegebenen regulären Ausdruck enthalten. |

-h*Typ* Wie -b*Typ*, nur für den Kopfteil. Standardtyp für den Kopfteil ist 'n'.

-f*Typ* Wie -b*Typ*, nur für den Fußteil. Standardtyp für die Numerierung eines Fußteils ist 'n'.

-p Die Zeilennumerierung erfolgt fortlaufend ohne Neubeginn auf neuen Seiten.

-v*Startnr* *Startnr* ist der für die Zeilennumerierung erforderliche Anfangswert. *Startnr* ist eine ganze Zahl (Standardwert ist 1).

-i*Incr* *Incr* ist der Wert, um den die Zeilennummer schrittweise erhöht wird. Standardwert ist 1.

-s*Sep* *Sep* sind ein oder mehrere Zeichen, die die Zeilennummer von der zugehörigen Textzeile trennen (Standard = Tab).

-w*n* Mit *n* wird die Anzahl Zeichen angegeben, die die Zeilennummer enthalten soll. Standard ist 6.

## nl

-n*Format* Diese Option enthält das Zeilennumerierungsformat. Folgende Werte sind möglich:

Format

ln linksbündig, führende Nullen werden unterdrückt;

rn rechtsbündig, führende Nullen werden unterdrückt (Standard);

rz rechtsbündig, führende Nullen bleiben erhalten.

-l*n* *n* gibt die Anzahl Leerzeilen an, die als eine Zeile gelten soll.
Beispiel: -l2 bedeutet, daß – bei Verwendung der Optionen -ha, -ba und/oder -fa – nur jede zweite Leerzeile gezählt wird. Standard ist 1.

-d*xx* *xx* sind Begrenzungszeichen, die den Anfang einer logischen Seite spezifizieren. Wird nur ein Zeichen angegeben, erhält das zweite Zeichen den Defaultwert ':'. Zwischen -d und den Begrenzungszeichen darf kein Blank stehen. Für '\' (Backslash) müssen Sie '\\' eingeben.

Beispiel:

Nl numeriert die Zeile von *Datei*. Die Numerierung beginnt in der zehnten Zeile mit einem Increment von 10. Das Begrenzungszeichen für die logischen Seiten ist '!+':

```
nl -v10 -i10 -d!+ Datei
```

SIEHE AUCH:

pr

# nm

## nm – Anzeige der Namenslisten von Objektdateien

SYNTAX:

nm [*Optionen*] *Dateinamen*

BESCHREIBUNG:

Das Kommando nm zeigt die Symboltabellen aller angegebenen Objektdateien an. *Dateiname* kann eine verschiebbare oder absolute Objektdatei sein; es kann sich aber auch um ein Archiv verschiebbarer oder absoluter Objektdateien handeln. Bei jedem Symbol werden die folgenden Informationen ausgegeben:

Name — Der Name des Symbols.

Value — Der Wert des Symbols, je nach Speicherklasse ausgedrückt als eine Distanz (Offset) oder Adresse.

Class — Die Speicherklasse des Symbols.

Type — Der Typ und abgeleitete Typ des Symbols. Ist das Symbol eine Struktur oder eine Union, wird die Struktur- bzw. Union-Marke (tag) im Anschluß an den Typ angegeben (z. B. struct-tag). Ist das Symbol ein Array, so werden im Anschluß an den Typ die Array-Dimensionen angegeben (z. B. char[*n*][*m*]). Die Objektdatei muß mit der Option -g des Kommandos cc kompiliert worden sein, damit diese Informationen erscheinen.

Size — Die Größe des Symbols in Bytes, falls verfügbar. Die Objektdatei muß mit der Option -g des Kommandos cc kompiliert worden sein, damit diese Informationen erscheinen.

Line — Die Zeilennummer in der Quelle, in der das Symbol definiert ist, falls verfügbar. Die Objektdatei muß mit der Option -g des Kommandos cc kompiliert worden sein, damit diese Informationen erscheinen.

Section — Bei den Speicherklassen static und external des Objektdateiabschnittes, der das Symbol enthält (z. B. text, data oder bss).

## nm

Die Ausgabe von nm kann mit den folgenden Optionen gesteuert werden:

-o Den Wert und die Größe eines Symbols oktal statt dezimal ausgeben.

-x Den Wert und die Größe eines Symbols hexadezimal statt dezimal ausgeben.

-h Die Kopfdaten der Ausgabe nicht anzeigen.

-v Externe Symbole vor der Ausgabe nach Wert sortieren.

-n Externe Symbole vor der Ausgabe nach Namen sortieren.

-e Nur externe und statische Symbole ausgeben.

-f Vollständige Ausgabedaten anzeigen. Auch redundante Symbole (.text, .data und .bss) anzeigen, die normalerweise unterdrückt werden.

-u Nur undefinierte Symbole ausgeben.

-V Die Version des laufenden Kommandos nm in die Standardfehlerausgabe ausgeben.

Die Optionen können in beliebiger Reihenfolge einzeln oder zusammen codiert werden und können an beliebiger Stelle in der Kommandozeile erscheinen. Beispiel: nm Name -e -v und nm -ve Name haben dieselbe Bedeutung und geben die in Name enthaltenen statischen und externen Symbole aus, wobei die externen Symbole nach Wert sortiert werden.

DATEIEN:

/usr/tmp/nm??????

## nm

DIAGNOSE:

| | |
|---|---|
| nm: Name: cannot open | *Name* kann nicht gelesen werden. |
| nm: Name: bad magic | *Name* ist keine geeignete gemeinsame Objektdatei. |
| nm: Name: no symbols | Die Symbole sind aus *Name* entfernt worden. |

SIEHE AUCH:

as, cc, ld

## nohup

nohup – Ausführung eines Kommandos ohne Rücksicht auf Unterbrechungssignale

SYNTAX:

nohup *Kommando* [*Argumente*]

BESCHREIBUNG:

Rufen Sie ein Kommando mit nohup auf, wird dieses – ohne Beachtung von Unterbrechungssignalen – ausgeführt. Leiten Sie die Standardausgabe nicht um, erfolgen Standardfehlerausgabe und Standardausgabe in nohup.out. Haben Sie für nohup.out im aktuellen Verzeichnis keine Schreibberechtigung, wird die Ausgabe nach $HOME/nohup.out umgelenkt.

BEISPIEL:

In vielen Fällen ist der Aufruf von nohup bei Pipes oder Kommandolisten vorteilhaft. Dazu müssen Sie Pipes und Kommandolisten in einer Shell-Prozedur zusammenfassen. Diese Kommandoprozedur können Sie dann mit dem Befehl

nohup sh *Datei*

aufrufen; nohup wirkt dann auf alle Kommandos in dieser Datei. Wird diese Kommandodatei häufig ausgeführt, können Sie sich die Eingabe des Kommandos sh sparen, indem Sie dieser Datei entsprechende Ausführungserlaubnis erteilen.
Bei Ausführungen im Hintergrund durch Angabe von & werden Unterbrechungssignale ebenfalls ignoriert.

## nohup

HINWEISE:

| | |
|---|---|
| nohup *Kommando1; Kommando2* | Nohup wirkt nur auf *Kommando1*. |
| nohup (*Kommando1; Kommando2*) | Dieser Aufruf ist syntaktisch falsch. |

Sie sollten auf Umlenkungen der Standardfehlerausgabe achten. Das folgende Kommando könnte zum Beispiel Fehlermeldungen auf Band schreiben, so daß dieses unlesbar wird:

nohup cpio -o <*Liste* >/dev/rmt1/1m&

Hingegen schreibt

nohup cpio -o <*Liste* >/dev/rmt1/1m 2 >Fehler&

die Fehlermeldungen in die Datei Fehler.

SIEHE AUCH:

chmod, nice, sh

## nroff

### nroff – Textformatierer

SYNTAX:

nroff [*Optionen*] [*Dateien*]

BESCHREIBUNG:

*Dateien* enthalten das Dokument, das formatiert werden soll. Ein einfaches Minuszeichen '-' bezeichnet die Standardeingabe als Datei. Sind garkeine Dateien angegeben, wird ebenfalls die Standardeingabe verwendet.

Folgende Optionen können Sie angeben:

-o*Liste* Nur die Seiten ausgeben, deren Nummern in *Liste* enthalten sind. Die Nummern werden durch Kommata (ohne Leerzeichen) getrennt, dabei sind Bereichsangaben durch *n1-n2* möglich. Eine führende Angabe -*n* bedeutet „Anfang bis Seite *n* einschließlich" und *n*- als letzte Angabe bedeutet „ab Seite *n* bis Ende".

Beispiele: -o1,3,11 -o-3 -o1,3-6,9,14-17,19-

-n*n* Die erste Seite bekommt Seitennummer *n* (statt Standard: 1).

-s*n* Die Ausgabe wird vor der jeweils *n*-ten Seite angehalten (zum Papiernachlegen usw.) und fortgesetzt beim Auslösen der CR-Taste.

-m*Name* Das Makropaket /usr/lib/tmac/tmac.*Name* wird vor den angegebenen Dateien bearbeitet.

-c*Name* Die komprimierten Makropakete /usr/lib/macros/cmp.[nt].[dt].*Name* und /usr/lib/macros/ucmp.[nt].*Name* werden vor den angegebenen Dateien bearbeitet.

-k*Name* Die Makropakete, die mit dieser Option aufgerufen werden, werden komprimiert und das Ergebnis wird im aktuellen Verzeichnis in Dateien [dt].*Name* abgelegt.

## nroff

-r*cn* Das Register *c* erhält zu Beginn den Wert *n*. Es können nur Register angesprochen werden, deren Name aus einem Zeichen besteht.

-i Nach den angegebenen Dateien wird noch die Standardeingabe gelesen.

-q Gleichzeitige Ein- und Ausgabe mit dem .rd-Kommando wird ermöglicht.

-T*Name* Angabe eines bestimmten Ausgabegeräts. Hierdurch werden einige spezifische Eigenschaften ausgenutzt und spezielle Steuerzeichen erzeugt.

Jede Option wird als eigenes Argument angegeben:

```
nroff -o1,3-5 -me dok0 dok1
```

bedeutet, daß der Text aus den Dateien dok0 und dok1 mit nroff formatiert wird. In dem Text können Makros aus dem -me-Makropaket aufgerufen werden, da zuvor das Makropaket /usr/lib/tmac/tmac.e bearbeitet wird. Vom erzeugten formatierten Text werden nur die Seiten 1, 3, 4 und 5 ausgegeben.

DATEIEN:

| | |
|---|---|
| /usr/lib/tmac/tmac.* | Makro-Dateien |
| /usr/lib/macros/* | Makro-Dateien |

SIEHE AUCH:

Systemliteratur TARGON /35: „Textformatierer nroff"

# od

od – Ausdrucken von Dateiinhalten (Dump)

SYNTAX:

od [-bcdosx] [*Datei*] [[+]*Distanz*[.][b]]

BESCHREIBUNG:

Mit dem Befehl od können Sie eine Datei in unterschiedlichen Formen, entsprechend dem ersten angegebenen Argument, darstellen lassen. Fehlen alle diese Argumente, so wird standardmäßig -o eingesetzt.

Zwischen folgenden Möglichkeiten der Darstellung können Sie wählen:

-b Die Bytes werden oktal angegeben.

-c Die Bytes werden im ASCII-Code aufgeführt. Einige nicht-darstellbare Zeichen werden wie in der Programmiersprache C üblich angegeben.

- NUL = \0
- Backspace = \b
- Seitenvorschub = \f
- Neue Zeile = \n
- CR = \r
- Tabulator = \t

Andere nicht-darstellbare Zeichen erscheinen als 3-ziffrige Oktalzahlen.

-d Worte werden als Dezimalzahl ohne Vorzeichen angegeben.

-o Worte werden als Oktalzahl angegeben.

-s Worte werden als Dezimalzahl mit Vorzeichen angegeben.

-x Worte werden als Hexadezimalzahl angegeben.

## od

Als zweites Argument beim Aufruf dieses Befehls geben Sie die Datei an, für die die Ausgabe erstellt werden soll. Geben Sie keine Datei an, nimmt das Programm die Standardeingabe.

Durch die Angabe von *Distanz* können Sie die Stelle innerhalb der Datei festlegen, ab der der Ausdruck erfolgen soll. *Distanz* wird in der Regel in Oktalbytes interpretiert; die Interpretation in Dezimalbytes erfolgt, wenn ein Punkt angefügt wird. Geben Sie -b an, wird die Angabe in 512-Bytes-Blöcken interpretiert.
Geben Sie keine Datei an, muß vor *Distanz* ein Pluszeichen eingegeben werden.

SIEHE AUCH:

dump

P

## pack

pack – Komprimieren von Dateien

SYNTAX:

pack [-] *Datei* ...

BESCHREIBUNG:

Pack komprimiert die angegebenen Dateien nach dem Huffmann-Algorithmus (byteweise mit minimaler Redundanz). Die Ergebnisse werden in Dateien abgelegt, deren Namen durch Anhängen von '.z' an die ursprünglichen Dateinamen erzeugt werden. Alle Dateiattribute (Zugriffsrechte, Erzeugungsdatum usw.) werden von den ursprünglichen Dateien übernommen; diese Dateien werden anschließend gelöscht.

Sie erzielen durch pack Einsparungen von 25-40% bei Texten, aber nur ca. 10% bei ausführbaren Programmen. Bei kleinen Dateien wird normalerweise keine Einsparung erzielt. Ausnahme: es werden nur wenige unterschiedliche Zeichen benutzt, wie z. B. bei Graphik oder Plotter-Ausgaben!

Geben Sie die Option '-' an, werden zusätzlich zum Huffmann-Algorithmus folgende Angaben ausgegeben:

- Anzahl des Auftretens,
- Relative Häufigkeit,
- Interner Code.

Diese Option kann durch wiederholtes Einfügen von '-' zwischen den Dateien ein- und ausgeschaltet werden.

## pack

Pack liefert als Rückgabe-Code die Anzahl der Dateien, die aus einem der folgenden Gründe nicht komprimiert werden konnten:

- Die Datei ist schon komprimiert.
- Der Dateiname hat mehr als 12 Zeichen, so daß die Endung '.z' nicht angehängt werden kann.
- Es existieren Links auf die angegebene Datei.
- *Datei*.z existiert bereits.
- *Datei*.z kann nicht angelegt werden.
- Ein-/Ausgabefehler während der Verarbeitung.

SIEHE AUCH:

pcat, unpack

P

## passwd

passwd – Eingeben oder Ändern des Login-Kennworts

SYNTAX:

passwd [*Paßwort*]

BESCHREIBUNG:

Passwd erlaubt die Eingabe eines Paßworts oder die Änderung eines bereits existierenden Paßworts. Das Paßwort steht in Beziehung zum Benutzernamen.

Bei Änderung fragt das Programm nach Ihrem alten Paßwort. Nach Eingabe des alten Paßworts fragt es nach dem neuen. Dieses muß zweimal angegeben werden, um Eingabefehler zu vermeiden. Die Paßworte werden nicht am Bildschirm angezeigt.

Bei Eingabe des Paßworts ist die Kombination von Groß-, Kleinbuchstaben, Zahlen und Sonderzeichen erlaubt. Die Länge des von Ihnen gewählten Paßworts muß mindestens 6 Zeichen betragen. Das Paßwort kann beliebig lang sein, aber nur die ersten acht Zeichen dienen der Identifikation.

Nur der Eigentümer kann sein Kennwort ändern und nur der Superuser kann Paßwörter löschen.

DATEI:

/etc/passwd

SIEHE AUCH:

id, login, su

P

## paste

paste – Mischen von gleichen Zeilen mehrerer Dateien oder zusammengehörenden Zeilen einer Datei

SYNTAX:

paste *Datei1 Datei2* ...

paste -d*Liste Datei1 Datei2* ...

paste -s [-d*Liste*] *Datei1 Datei2* ...

BESCHREIBUNG:

In den ersten beiden Formen verkettet paste korrespondierende Zeilen der Eingabedateien *Datei1*, *Datei2* usw.
Paste behandelt jede Datei als einzelne Spalte oder als Spalten einer Tabelle und fügt sie horizontal zusammen (paralleles Mischen). Paste ist also als Gegenstück zu cat anzusehen, das Dateien vertikal verkettet, d. h. eine Datei wird hinter der anderen angehängt.

Die letzte Befehlsform bewirkt die Zusammenfassung aufeinanderfolgender Zeilen der Eingabedatei zu einer Zeile (serielles Mischen).

In allen o. g. Fällen werden die Zeilen entweder durch ein Tabulatorzeichen oder durch das optional in *Liste* angegebene Zeichen getrennt.

Die Ausgabe erfolgt auf der Standardausgabe, so daß Sie paste als Teil einer Pipe oder als Filter nutzen können, wenn Sie statt einer Eingabedatei ein Minuszeichen (-) angeben.

Optionen:

-d Ohne diese Option werden die Zeilenende-Zeichen der angegebenen Dateien – außer in der letzten Datei (bei der -s-Option der letzten Zeile) – durch ein Tabulatorzeichen ersetzt. Durch Angabe dieser Option können Sie die Tabulatorzeichen durch andere Zeichen ersetzen (s. *Liste*).

## paste

*Liste* Ein oder mehrere Zeichen, die Sie ohne Leerzeichen direkt hinter der -d-Option angeben müssen, ersetzen das Tabulatorzeichen als das Standardverkettungszeichen. Beim parallelen Mischen (gilt nicht für die -s-Option) werden die Zeilen der letzten Eingabedatei grundsätzlich durch das Zeilenende-Zeichen beendet und nicht durch die Zeichen, die Sie in *Liste* angegeben haben. *Liste* darf auch folgende Zeichen enthalten: \n (Zeilenende), \t (Tabulator), \\ (Backslash) und \0 (leere Zeichenkette). Sie müssen Fluchtsymbole verwenden, wenn die angegebenen Zeichen für die Shell eine spezielle Bedeutung haben, z. B. müssen Sie \\\\ eingeben, um einen Backslash zu erhalten.

-s Durch Angabe dieser Option werden meherere aufeinanderfolgende Zeilen – statt jeweils einer – der Eingabedateien gemischt. Das Tabulatorzeichen wird zur Verkettung benutzt, es sei denn, Sie haben in Kombination mit der -s-Option eine Liste anderer Zeichen festgelegt. Auf jeden Fall ist jedoch das letzte Zeichen einer Datei das Zeilenende-Zeichen.

- Durch die Angabe des Minuszeichens, anstelle von Dateinamen, wird die Standardeingabe gelesen.

Beispiele:

```
ls | paste -d" " -
```

Das aktuelle Verzeichnis wird in einer Spalte aufgelistet.

```
ls | paste - - - -
```

Das aktuelle Verzeichnis wird vierspaltig aufgelistet.

```
paste -s -d"\t\n" Datei
```

Zusammenfügen von Zeilenpaaren zu einer Zeile.

P

## paste

MELDUNGEN:

Da die Länge der Ausgabezeilen maximal 511 Zeichen beträgt, erfolgt eine Meldung, wenn diese Länge überschritten wird.

Außer bei Angabe der -s-Option, verarbeitet paste nicht mehr als zwölf Eingabedateien. Geben Sie mehr Dateien an, erfolgt eine Fehlermeldung.

SIEHE AUCH:

cut, grep, pr

P

## pcat

pcat – Dekomprimieren von Dateien

SYNTAX:

pcat *Datei* ...

BESCHREIBUNG:

Pcat dekomprimiert die angegebenen Dateien, ohne die komprimierten Versionen zu beseitigen. Pcat wirkt auf komprimierte Dateien wie cat auf normale Dateien, kann aber nicht als Filter benutzt werden.

Pcat liefert als Rückgabe-Code die Anzahl der Dateien, die aus einem der folgenden Gründe nicht dekomprimiert werden konnten:

- Die Datei wurde nicht von pack erzeugt.
- Der angegebene Dateiname hat mehr als 12 Zeichen.
- Die angegebene Datei kann nicht eröffnet werden.

SIEHE AUCH:

pack, unpack

P

# pr

## pr – Dateien ausgeben

SYNTAX:

pr [*Optionen*] [*Datei* ...]

BESCHREIBUNG:

Pr gibt die angegebenen Dateien auf der Standardausgabe aus. Ist *Datei* gleich '-', wird die Standardeingabe gelesen. Standardmäßig wird die Ausgabe in Seiten unterteilt, wobei jede mit Seitennummer, Datum, Zeit und Dateinamen versehen wird.

Mehrspaltige Ausgabe ist möglich. Standardmäßig sind die Spalten von gleicher Breite und werden durch mindestens ein Leerzeichen getrennt; nicht passende Zeilen werden abgeschnitten. Wenn die -s-Option verwendet wird, werden die Zeilen nicht abgeschnitten und die Spalten werden durch das angegebene Trennzeichen unterteilt.

Ist die Standardausgabe einem Terminal zugeordnet, werden Fehlermeldungen solange zurückgehalten, bis die Ausgabe abgeschlossen ist.

Die folgenden Optionen können einzeln oder in beliebiger Reihenfolge angewandt werden:

| | |
|---|---|
| +*k* | Der Ausdruck wird mit Seite *k* begonnen (Standard 1). |
| -*k* | Es wird ein *k*-spaltiger Ausdruck erzeugt (Standard 1). Die Optionen -e und -i werden für mehrspaltigen Ausdruck ausgewertet. |
| -a | Mehrspaltiger Ausdruck. |
| -m | Mische und drucke alle angegebenen Dateien gleichzeitig, je eine pro Spalte (übersteuert die -k und die -a-Option). |
| -d | Die Ausgabe wird jeweils mit einer Leerzeile Zwischenraum gedruckt. |

pr

-ec*k* Die Eingabetabulatoren positionieren die Ausgabe auf die Zeichenpositionen *k*+1, 2∗*k*+1 usw. Wenn *k* gleich 0 ist oder weggelassen wird, werden Standardtabulatoren an jeder achten Position angenommen.

Tabulatorzeichen im Eingabestrom werden auf die entsprechende Zahl von Leerzeichen expandiert. Ist *c* angegeben, wird es als Eingabetabulatorzeichen behandelt. Standard für *c* ist das ASCII-Tabulatorzeichen.

-ic*k* In der Ausgabe wird „white space" an den Stellen *k*+1, 2∗*k*+1 usw. durch Tabulatorzeichen ersetzt. Wenn *k* gleich 0 ist oder ausgelassen wird, werden als Standardpositionen jede achte Position angenommen.

Ist *c* (irgendein Zeichen, das keine Ziffer ist) angegeben, wird es als Ausgabetabulatorzeichen verwendet. Standard für *c* ist das ASCII-Tabulatorzeichen.

-nc*k* Eine Zeilennumerierung von *k* Ziffern (Standard für *k* ist 5) wird erzeugt. Die Nummer nimmt die ersten *k*+1 Stellen jeder Spalte des normalen Ausdrucks oder jeder Zeile der -m-Ausgabe ein. Wenn *c* angegeben ist (irgendein Zeichen, das keine Ziffer ist), wird es an die Zeilennummer angehängt, um sie vom folgenden Text zu trennen (Standard ist hier wiederum das ASCII-Tabulatorzeichen).

-w*k* Die Breite einer Zeile wird auf *k* Zeichenpositionen festgesetzt (Standard ist 72 für gleichbreite, mehrspaltige Ausgabe, ansonsten besteht keine Grenze).

-o*k* Jede Zeile wird um *k* Zeichenpositionen eingerückt (Standard: 0). Die Anzahl der Zeichenpositionen pro Zeile ist die Summe aus Breite und Einrückung.

-l*k* Die Seitenlänge wird auf *k* Zeilen festgesetzt (Standard ist 66 Zeilen pro Seite).

-h*Name* *Name* wird anstelle des Dateinamens als Kopfzeile genommen.

pr

P

-p Wenn die Ausgabe am Terminal erfolgt, wird vor jedem Seitenanfang angehalten. Pr gibt ein akustisches Signal aus. Die Ausgabe der nächsten Seite wird fortgesetzt, wenn Sie <CR> drücken.

-f Um eine neue Seite anzufangen, wird das ASCII-Zeichen für Seitenvorschub (FF = Form Feed) genommen. Standardmäßig wird eine Reihe von Zeilenvorschüben erzeugt. Ansonsten verhält sich die Option wie -p.

-r Bei nicht vorhandenen Dateien werden keine Fehlermeldungen ausgegeben.

-t Weder der 5-zeilige Kopfzeilenbereich noch der fünfzeilige Fußzeilenbereich wird ausgedruckt. Nach der letzten Zeile jeder Datei wird die Ausgabe gestoppt, ohne zum Ende der Seite vorzurücken.

-sc Die Spalten werden anstelle der passenden Anzahl von Leerzeichen durch das einzelne Zeichen *c* getrennt. Standard ist das ASCII-Tabulatorzeichen.

Beispiele:

Drucke *Datei1* und *Datei2* als 3-spaltigen Ausdruck mit der Kopfzeile „Dateiliste" und mit eingestreuten Leerzeilen.

pr -3dh "Dateiliste" *Datei1 Datei2*

Kopiere *Datei1* nach *Datei2* und expandiere Tabulatorzeichen auf die Positionen 10, 19, 28, 37, ...

pr -e9 -t <*Datei1* >*Datei2*

SIEHE AUCH:

cat

P

## prof

### prof – Anzeigen von Profildateien

SYNTAX:

prof [-tcan] [-ox] [-g] [-z] [-h] [-s] [-m*DateiM*] [*Programm*]

BESCHREIBUNG:

Prof interpretiert eine von der Monitor-Funktion erstellte Profildatei. Die Symboltabelle in der Objektdatei *Programm* (standardmäßig a.out) wird gelesen und mit einer Profildatei (standardmäßig mon.out) korreliert. Für jedes externe Textsymbol wird der prozentuale Anteil der Zeit, die für die Ausführung zwischen der Adresse dieses Symbols und der Adresse des nächsten Symbols aufgewandt wurde, sowie die Anzahl der Aufrufe dieser Funktion und die durchschnittliche Anzahl Millisekunden pro Aufruf angezeigt.

Die sich gegenseitig ausschließenden Optionen t, c, a und n legen fest, wie die Ausgabezeilen sortiert werden:

-t In absteigender Reihenfolge nach dem Prozentsatz der Gesamtzeit sortieren (Standard).

-c In absteigender Reihenfolge nach der Anzahl der Aufrufe sortieren.

-a In aufsteigender Reihenfolge nach Symboladressen sortieren.

-n Lexikalisch nach Symbolnamen sortieren.

Die sich gegenseitig ausschließenden Optionen -o und -x legen fest, in welcher Form die Adressen der überwachten Symbole ausgegeben werden:

-o Die Symboladressen (oktal) zusammen mit dem Symbolnamen ausgeben.

-x Die Symboladressen (hexadezimal) zusammen mit dem Symbolnamen ausgeben.

## prof

Die folgenden Optionen können zusammen angegeben werden:

-g Nicht-globale Symbole (statische Funktionen) berücksichtigen.

-z Alle Symbole im Profilbereich berücksichtigen, auch wenn die Anzahl der Aufrufe und die Zeit Null ist.

-h Den normalerweise in der Auswertung erscheinenden Kopf wegfallen lassen. (Dies ist sinnvoll, wenn die Auswertung weiterverarbeitet werden soll.)

-s Eine Zusammenfassung mehrerer der Überwachungsparameter und Statistiken auf der Standardfehlerausgabe ausgeben.

-m*DateiM* Die Datei *DateiM* anstelle von mon.out als Eingabe-Profildatei verwenden.

Ein Programm legt eine Profildatei an, wenn es mit der Option -p von cc kompiliert wurde. Diese Option des Kommandos cc sorgt für Aufrufe an den Monitor zu Beginn und am Ende der Ausführung. Der Aufruf an den Monitor am Ende der Ausführung bewirkt das Schreiben der Profildatei.

DATEIEN:

mon.out für das Profil

a.out für die Namensliste

HINWEIS:

Während der Programmausführung können für maximal 300 Funktionen Aufrufzähler eingerichtet werden. Beim Überschreiten dieser Grenze können Daten überschrieben werden und die Datei mon.out enthält verfälschte Daten. Die Anzahl der Aufrufzähler wird vom prof-Kommando automatisch gemeldet, sobald die Anzahl von 250 überschritten wird.

SIEHE AUCH:

cc

P

## prs

### prs – Ausdrucken von SCCS-Dateien

SYNTAX:

prs [-d[*Datenspezifikation*]] [-r[*SID*]] [-e] [-l] [-a] *Datei* ...

BESCHREIBUNG:

Prs gibt auf der Standardausgabe eine SCCS-Datei teilweise oder als Ganzes in dem von Ihnen angegebenen Format aus. Spezifizieren Sie ein Verzeichnis, verhält sich prs so, als ob jede Datei dieses Verzeichnis aufgeführt wäre. Nicht-SCCS-Dateien und nicht-lesbare Dateien werden stillschweigend übergangen.

Argumente können in beliebiger Reihenfolge angegeben werden und bestehen aus Optionen und Dateinamen.

Alle Optionen wirken unabhängig voneinander auf jede genannte Datei:

-d[*Datenspezifikation*]
: Spezifizierung der Ausgabedaten. Die Datenspezifikation ist eine Zeichenkette, die aus SCCS-Datenschlüsselwörtern, gemischt mit zusätzlichem Benutzertext, besteht.

-r[*SID*]
: Angabe der SCCS-Identifikation (SID) desjenigen Deltas, über das Informationen gewünscht werden. Geben Sie keine SID an, wird die SID des zuletzt erzeugten Deltas benutzt.

-e
: Informationen über alle Deltas, die früher als das durch die -r-Option bezeichnete Delta erzeugt wurden (dieses ist jedoch eingeschlossen).

-l
: Informationen über alle Deltas, die später als das durch die -r-Option bezeichnete Delta erzeugt wurden (dieses ist jedoch eingeschlossen).

-a
: Informationen über gelöschte (Deltatyp = R, s. rmdel) und existierende (Deltatyp = D) Deltas. Geben Sie -a nicht an, werden nur Informationen über existierende Deltas geliefert.

## prs

Datenschlüsselworte:

Sie spezifizieren, welche Teile einer SCCS-Datei gesucht und ausgegeben werden sollen. Alle Teile einer SCCS-Datei haben ein ihnen zugeordnetes Schlüsselwort. Die Häufigkeit, mit der ein Schlüsselwort auftreten kann, ist nicht begrenzt.

Die von prs gedruckten Informationen bestehen aus:

1. dem vom Benutzer angegebenen Text,
2. den aus der SCCS-Datei entnommenen, für die erkannten Datenschlüsselworte eingesetzten Werte in der Reihenfolge ihres Auftretens in der Datenspezifikation.

Das Format eines Schlüsselwortwertes ist entweder „simple" (S), bei dem die Ersetzung direkt erfolgt, oder „multi-line" (M), wobei die Ersetzung von einem Zeilenvorschub gefolgt wird.

Benutzertext ist jeglicher Text außer erkannten Datenschlüsselwörtern. Ein Tabulatorzeichen wird durch \t bezeichnet, ein Zeilenvorschub durch \n.

SIEHE AUCH:

admin, delta, get, help
Systemliteratur TARGON: „Programmentwicklungs-Tools"

P

## ps

### ps – Anzeige Prozeß-Status

SYNTAX:

ps [*Optionen*]

BESCHREIBUNG:

Ps gibt bestimmte Informationen über aktive Prozesse aus. Bei fehlenden Optionen werden Informationen über die dem aktuellen Terminal zugeordneten Prozesse angezeigt. Diese Ausgabedaten bestehen aus einer Liste, die lediglich die Prozeßnummer, die Terminalkennung, die kumulierte Ausführungszeit und den Kommandonamen enthält. Der Umfang der angezeigten Informationen kann durch Optionen gesteuert werden.

Bei Optionen, die in Verbindung mit Argumentlisten angegeben werden, können Sie diese Listen folgendermaßen spezifizieren:

1. Argumente werden durch Kommata voneinander getrennt.
2. Argumentliste wird in doppelte Hochkommata eingeschlossen. Die einzelnen Argumente werden durch Kommata und/oder Leerzeichen getrennt.

Optionen:

-e Ausgabe von Informationen über alle Prozesse.

-d Ausgabe von Informationen über alle Prozesse, exklusive der Führungsprozesse von Prozeßgruppen.

-a Ausgabe von Informationen über alle Prozesse, exklusive der Führungsprozesse von Prozeßgruppen und solcher Prozesse, die keinem Terminal zugeordnet sind.

-f Ausgabe von zusätzlichen Informationen. Die Erläuterung dieser zusätzlichen Informationen finden Sie weiter unten.

-l Ausgabe von zusätzlichen Informationen. Die Erläuterung dieser zusätzlichen Informationen finden Sie weiter unten.

## ps

-c*Speicherdatei*
Anstelle der Standarddatei /dev/mem wird *Speicherdatei* benutzt.

-s*Swapdatei*
Anstelle der Standarddatei /dev/swap wird *Swapdatei* benutzt. Diese Option ist zum Überprüfen von Speicherdateien sinnvoll.

-n*Namenliste*
*Namenliste* wird als alternative Namenlisten-Datei interpretiert.

-t*Terminalliste*
Es werden nur Daten über die Prozesse ausgegeben, die den in der Liste aufgeführten Terminals zugeordnet sind. Terminalkennungen können auf zweierlei Weise spezifiziert werden, und zwar durch Kommata getrennt oder eine in doppelte Hochkommata eingeschlossene Liste, deren einzelne Elemente durch Kommata und/oder Leerzeichen getrennt werden.

-p*Prozeßliste*
Es werden nur Informationen über die Prozesse ausgegeben, deren Prozeßnummern in der Liste angegeben sind. Die Prozeßliste wird in der gleichen Form angegeben wie die Terminalliste.

-u*Benutzer-ID-Liste*
Es werden nur Daten über die Prozesse ausgegeben, deren Benutzer-IDs oder Login-Namen in der Liste angegeben sind. In der Ausgabe erscheint die numerische Benutzer-ID.
Geben Sie die -f-Option an, wird der Login-Name ausgegeben.

-g*Gruppenliste*
Ausgabe von Informationen über die Prozeßgruppen, deren Führungsprozesse in der Liste angegeben sind.

Nachfolgend werden die Spaltenüberschriften und die Bedeutung der Spalten in einer ps-Liste erläutert. Die Buchstaben f und l geben die Option an, bei der die entsprechende Überschrift in der Liste erscheint. „Alle" bedeutet, das die Spalte immer erscheint. Die Optionen -f und -l legen nur fest, welche Informationen über einen Prozeß angezeigt werden. Sie haben keinen Einfluß darauf, welche Prozesse gelistet werden.

P

## ps

F (l) Dem Prozeß zugeordnete Flags:

- 01 im Hauptspeicher;
- 02 Systemprozeß;
- 04 im Speicher gesperrt (z. B. für physische E/A);
- 10 wird gerade ein- oder ausgelagert;
- 20 wird von einem anderen Prozeß überwacht;
- 40 Trace-Flag.

S (l) Prozeßstatus:

- O nicht existent;
- S schlafend;
- W wartend;
- R in Ausführung;
- I Zwischenstatus;
- Z beendet;
- T gestoppt;
- X wachsend.

UID (f,l) Die Benutzer-ID des Prozeßeigentümers. Bei Angabe der -f-Option erhalten Sie den Benutzernamen.

PID (alle) Prozeßnummer. Haben Sie diese Angabe, können Sie einen Prozeß mit kill abbrechen.

PPID (f,l) Prozeßnummer des Vaterprozesses.

C (f,l) Prozessorauslastung für Scheduling.

STIME (f) Startzeit des Prozesses.

PRI (l) Prozeßpriorität. Je höher die Zahl, desto geringer ist die Priorität.

NI (l) Nice-Wert; wird zur Berechnung der Priorität herangezogen.

ps

| | |
|---|---|
| ADDR (l) | Adresse des Prozesses, wenn er im Speicher resident ist, anderenfalls die Plattenadresse. |
| SZ (l) | Blockgröße des Prozeß-Core-Images. |
| WCHAN (l) | Das Ereignis, auf das der Prozeß wartet. Ist dieses Feld leer, wird der Prozeß gerade ausgeführt. |
| TTY (alle) | Kontroll-Terminal des Prozesses. |
| TIME (alle) | Kumulierte Ausführungszeit des Prozesses. |
| CMD (alle) | Kommandoname. Der volle Kommandoname inklusive seiner Argumente wird bei Verwendung der -f-Option ausgegeben. |

Ein Prozeß, der beendet wurde und einen Vaterprozeß besitzt, der nicht auf ihn gewartet hat, wird als <defunct> markiert.

Geben Sie die Option -f an, versucht ps, den bei der Erzeugung des Prozesses angegebenen Kommandonamen inklusive seiner Argumente durch Überprüfung des Hauptspeichers oder des Swap-Bereiches festzustellen. Ist dies nicht möglich, wird lediglich der Kommandoname – eingeschlossen in eckige Klammern – angezeigt.

DATEIEN:

| | |
|---|---|
| /unix | System-Namenliste |
| /dev/mem | Speicher |
| /dev/swap | Swap-Gerät (Standard) |
| /etc/passwd | Stellt Informationen über Benutzer-IDs bereit |
| /etc/ps_data | Interne Datenstruktur |
| /dev | Verzeichnis, das die Gerätedateien enthält |

SIEHE AUCH:

kill, nice

P

# pwd

pwd – Anzeige aktuelles Verzeichnis

SYNTAX:

pwd

BESCHREIBUNG:

Pwd zeigt den vollen Pfadnamen des aktuellen Verzeichnisses an.

HINWEISE:

Bekommen Sie beim Aufruf von pwd Hinweise, daß Dateien nicht eröffnet oder gelesen werden können, weist dies auf Inkonsistenzen im Dateisystem hin. In solchen Fällen sollten Sie Ihren Systemverwalter einschalten.

SIEHE AUCH:

cd

## ratfor

ratfor – Fortran Preprozessor

SYNTAX:

ratfor [*Optionen*] [*Dateien*]

BESCHREIBUNG:

Ratfor ist ein Preprozessor für Fortran 77. Er ermöglicht strukturierte Programmierung und setzt diese Programme in Standard-Fortran um. Die Kontrollfluß-Konstruktionen von ratfor entsprechen im wesentlichen denen von C:

Gruppierung von Anweisungen:
{*Anweisung;Anweisung;Anweisung*}

Sprungverteiler:
if (*Bedingung*) *Anweisung* [else *Anweisung*]
switch (*ganzzahliger Wert*) {
case *Ganzzahl*:*Anweisung*
...
[default:] *Anweisung*
}

Schleifen:
while (*Bedingung*) *Anweisung*
for (*Ausdruck; Bedingung; Ausdruck*) *Anweisung*
*do Grenzen Anweisung*
repeat *Anweisung* [until (*Bedingung*)]
break
next

## ratfor

Sie haben folgende Möglichkeiten, die Ihnen die Programmerstellung erleichtern und die Programme übersichtlicher machen:

Formatfreie Eingabe:
Mehrere Anweisungen pro Zeile; automatische Fortsetzung in Folgezeilen.

Kommentare:
# dies ist ein Kommentar.

Übersetzung von Vergleichsoperatoren:
>, >= usw. werden zu .GT., .GE. usw.

Rückgabe eines Ausdrucks von der Funktion an die aufrufende Komponente:
return (*Ausdruck*)

Define: define *Name Ersetzung*

Include: include *Datei*

Optionen:

-h Umwandlung von in Anführungszeichen eingeschlossenen Zeichenketten in 27H-Konstrukte.

-C Kommentare werden in die Ausgabe kopiert und übersichtlich geordnet.

-6x Fortsetzungszeilen sind normalerweise durch ein & in Spalte 1 gekennzeichnet. Durch diese Option wird das Fortsetzungszeichen als x definiert und in Spalte 6 plaziert.

SIEHE AUCH:

f77

## regcmp

regcmp – Kompilieren regulärer Ausdrücke

SYNTAX:

regcmp [-] *Dateien*

BESCHREIBUNG:

Benutzen Sie regcmp, ist es in den meisten Fällen nicht notwendig, regcmp aus C-Programmen aufzurufen. Dadurch wird sowohl die Ausführungszeit als auch die Programmgröße reduziert. Regcmp kompiliert die regulären Ausdrücke in *Datei* und schreibt die Ausgabedaten in *Datei*.i. Geben Sie die Option '-' an, so werden die Ausgabedaten in *Datei*.c geschrieben. Die Einträge in *Datei* bestehen aus einem Namen (C-Variable), dann folgen ein oder mehrere Blanks und danach ein in doppelte Hochkommata eingeschlossener regulärer Ausdruck. Regcmp gibt C-Quellcode aus. Kompilierte reguläre Ausdrücke werden als Zeiger auf Vektoren vom Typ „char" dargestellt. Die .i-Dateien können daher in C-Programme eingebunden und .c-Dateien können kompiliert und später geladen werden. In dem C-Programm, das die Ausgabe von regcmp benutzt, wendet regex(abc,*Zeile*) den regulären Ausdruck mit dem Namen abc auf *Zeile* an. Die Diagnosemeldungen sind selbsterklärend.

BEISPIELE:

Name "([A-Za-z][A-Za-z0-9]*)$0"

telno "\({0,1}([2-9][01][1-9])$0\){0,1} *"
"([2-9][0-9]{2})$1[-]{0,1}"
"([0-9]{4})$2"

In dem C-Programm, das die Ausgabedaten von regcmp benutzt, wendet

regex(telno,Zeile,area,exch,rest)

den regulären Ausdruck mit dem Namen telno auf *Zeile* an.

## rm, rmdir

rm, rmdir – Löschen von Dateien; Löschen von Verzeichnissen

SYNTAX:

rm [-fri] *Datei* ...

rmdir *Verzeichnis* ...

BESCHREIBUNG:

Rm löscht die Eintragung in ein Verzeichnis für eine oder mehrere Dateien. War dieser Eintrag die letzte Verknüpfung (Link) einer Datei, wird die Datei selbst gelöscht. Das Löschen eines Eintrags verlangt die Schreiberlaubnis für ihr Verzeichnis, aber weder Schreib- noch Leseerlaubnis für die Datei selbst.

Liegt für *Datei* keine Schreiberlaubnis vor und die Standardeingabe ist die Tastatur, so werden die Zugriffsrechte am Bildschirm angezeigt. Der rm-Befehl erwartet dann die Eingabe von 'y' (yes) um den Befehl auszuführen, anderenfalls wird der Befehl terminiert.

Optionen:

-f Kein Anzeigen der Zugriffsrechte.

-r Falls beim rm-Befehl ein Verzeichnis angesprochen wird, löscht rm nacheinander alle Datei-Einträge des Verzeichnisses und das Verzeichnis selbst.

-i Fragt vor jeder Ausführung, ob die Datei gelöscht werden soll und erwartet Ihre Entscheidung (i = interaktiv). In Verbindung mit der -r-Option gilt dies auch für Verzeichnisse.

Rmdir löscht den Eintrag von Verzeichnissen. Die Verzeichnisse müssen leer sein.

## rmdel

rmdel – Entfernen eines Deltas aus einer SCCS-Datei

SYNTAX:

rmdel -r*SID Datei* ...

BESCHREIBUNG:

Rmdel entfernt das durch die SID spezifizierte Delta aus jeder aufgeführten SCCS-Datei. Das zu entfernende Delta muß das jüngste Delta seines Zweiges oder des Stammes im Deltabaum sein. Zusätzlich darf das angegebene Delta nicht als Basis für Änderungen dienen, die durch ein get -e eingeleitet, aber noch nicht mit delta eingebunden wurden. (Wenn eine p-*Datei* für die aufgeführte SCCS-Datei existiert, darf das spezifizierte Delta nicht in irgendeinem Eintrag der p-*Datei* erscheinen.)

Geben Sie ein Verzeichnis an, wird rmdel auf jede lesbare SCCS-Datei in diesem Verzeichnis angewendet.
Ist '-' als Dateiname angegeben, liest rmdel aus der Standardeingabe; jede Zeile der Standardeingabe wird als Name einer SCCS-Datei betrachtet.

Bei den benötigten Rechten zum Entfernen eines Deltas handelt es sich – einfach ausgedrückt – um folgende Fälle:

1. Wer ein Delta anlegen kann, kann es auch löschen und
2. Eigentümer von Datei und Verzeichnis kann ein Delta löschen.

SIEHE AUCH:

delta, get, help, prs
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# sact

sact – Anzeigen der Editieraktivitäten der aktuellen SCCS-Datei

SYNTAX:

sact *Datei* ...

BESCHREIBUNG:

Durch sact können Sie sich über die anstehenden Deltas einer SCCS-Datei informieren. Dies ist immer dann sinnvoll, wenn der Befehl get -e ausgeführt wurde, ohne daß anschließend delta aufgerufen wird.

Geben Sie bei diesem Befehl ein Verzeichnis an, so gilt er für jede in diesem Verzeichnis enthaltene Datei, ausgenommen Nicht-SCCS-Dateien und nicht-lesbare Dateien.

Wurde als Eingabedatei '-' angegeben, liest das Programm die Standardeingabe. Jede Zeile wird dann als Name einer zu bearbeitenden SCCS-Datei interpretiert.

Für jede der genannten Dateien werden insgesamt fünf, durch Leerstellen voneinander getrennte Felder ausgegeben:

Feld 1 Dieses Feld spezifiziert die SID des aktuellen Deltas in der SCCS-Datei, die geändert werden soll, um ein neues Delta zu erzeugen.

Feld 2 Dieses Feld spezifiziert die SID des neu anzulegenden Deltas.

Feld 3 Dieses Feld enthält den Benutzernamen des Anwenders, der das Delta erzeugen will.

Feld 4 Dieses Feld enthält das Datum, an dem der Befehl get -e ausgeführt wurde.

Feld 5 Dieses Feld enthält die Uhrzeit, an dem get -e ausgeführt wurde.

SIEHE AUCH:

delta, get, unget
Systemliteratur TARGON: „Programmentwicklungs-Tools"

## sar

### sar – Auswertung der Systemaktivitäten (system activity reporter)

SYNTAX:

sar [-ubdycwpaqvmA] [-o*Datei*] *t* [*n*]

sar [-ubdycwpaqvmA] [-s*Zeit*] [-e*Zeit*] [-i*Sek*] [-f*Datei*]

BESCHREIBUNG:

Im ersten Fall fragt sar kumulative Aktivitätenzähler im Betriebssystem in *n* Intervallen je *t* Sekunden ab. Geben Sie die Option -o an, speichert es die Abfragewerte im Binärformat in *Datei*. Der Standardwert von *n* ist 1. Im zweiten Fall ist kein Abfrageintervall angegeben. Hier holt sar Daten aus einer zuvor aufgezeichneten Datei, bei der es sich entweder um die bei der Option -f angegebene Datei oder, wenn nichts angegeben ist, um die täglich erstellte Standard-Systemaktivitätendatei /usr/adm/sa/sa*dd* für den aktuellen Tag dd handelt. Die Anfangs- und die Endzeit der Auswertung können mit den Zeitargumenten -s und -e in der Form *hh*[:*mm*[:*ss*]] eingegrenzt werden. Die Option -i bewirkt, daß Datensätze in Intervallen von *Sek* Sekunden ausgewählt werden. Anderenfalls werden alle in der Datei gefundenen Intervalle ausgewertet.

In jedem Fall können Untermengen der auszugebenden Daten durch Optionen spezifiziert werden:

-u CPU-Auslastung auswerten (Standardwert):

%usr, %sys, %idle – Anteil der CPU-Zeiten für Benutzermodus, Systemmodus und Leerzeiten.

-b Auswertung der Pufferaktivität:

bread/s, bwrit/s – Anzahl Datentransfers pro Sekunde zwischen Systempuffern und Magnetplatte oder anderen blockorientierten Geräten;

lread/s, lwrit/s – Zugriffe auf die Systempuffer;

%rcache, %wcache – Cache-Trefferraten, z. B. 1 - bread/lread;

pread/s, pwrit/s – Transfers über den physikalischen Gerätemechanismus.

**sar**

-d Auswertung der Aktivitäten aller blockorientierten Geräte, z. B. Magnetplatte oder Bandlaufwerk:

%busy, avque – Zeitanteil, währenddessen das Gerät mit dem Bedienen einer Transferanforderung beschäftigt war, durchschnittliche Anzahl der während dieser Zeit ausstehenden Anforderungen;

r+w/s, blks/s - Anzahl der Datentransfers vom oder zum Gerät, Anzahl der transferierten Daten in Einheiten von 2048 Bytes;

avwait, avserv – Durchschnittliche Zeit in Millisekunden, während die Transferanforderungen inaktiv in der Warteschlange warten, und durchschnittliche Zeit für die Bedienung (die bei Magnetplatten die Positionierzeit, die Latenzzeit und die Datentransferzeit umfaßt).

-y Auswertung der TTY-Geräteaktivität:

rawch/s, canch/s, outch/s – Eingabe-Zeichenrate, von 'Canon' bearbeitete Eingabe-Zeichenrate, Ausgabe-Zeichenrate;

rcvin/s, xmtin/s, mdmin/s – Raten der Empfangs-, Sende- und Modemunterbrechungen.

-c Auswertung der Systemaufrufe:

scall/s – Systemaufrufe aller Art;

sread/s, swrit/s, fork/s, vfork/s, exec/s – Bestimmte Systemaufrufe;

rchar/s, wchar/s – Von Systemaufrufen für Lesen und Schreiben übertragene Zeichen.

-w Auswertung der Swapping- und Prozeßwechselaktivitäten:

swpin/s, swpot/s, pswin/s, pswot/s – Anzahl der Transfers und Anzahl der 2048 Bytes großen Einheiten (Pages), die bei Programmeinlagerungen (swap-ins) (einschließlich des anfänglichen Ladens eines Programms) und Programmauslagerungen (swap-outs) übertragen wurden;

pswch/s – Prozeßwechsel.

## sar

-p Auswertung der Paging-Aktivitäten:

pgin/s, pgot/s, ppgin/s, ppgot/s – Anzahl der Übertragungen und Anzahl der Pages (2048 Bytes pro Seite) die für Pagein und Pageout transferiert wurden. (Hier ist auch das anfängliche Laden einiger Programme enthalten.)

-a Auswertung der Benutzung der Systemroutinen für den Dateizugriff: iget/s, namei/s, dirblk/s.

-q Auswertung der durchschnittlichen Warteschlangenlänge während der Belegung und prozentualer Zeitanteil der Belegung:

runq-sz, %runocc – Warteschlange der im Speicher befindlichen und lauffähigen Prozesse;

swpq-sz, %swpocc – Swap-Warteschlange der ausgelagerten aber ablaufbereiten Prozesse.

-v Auswertung des Status der Text-, Prozeß-, Inode- und Dateitabellen:

text-sz, proc-sz, inod-sz, file-sz – Einträge/Größe pro Tabelle, die einmal am Abfragepunkt ausgewertet wird;

text-ov, proc-ov, inod-ov, file-ov – Zwischen den Abfragepunkten auftretende Überläufe.

-m Auswertung der Nachrichten- und Semaphor-Aktivitäten:

msg/s, sema/s – Grundelemente (primitives) pro Sekunde.

-A Auswertung aller Daten. Gleichbedeutend mit -udqbwpcayvm.

## sar

BEISPIELE:

Auswertung der CPU-Tagesaktivitäten bis zum aktuellen Zeitpunkt:

```
sar
```

Die CPU-Aktivität zehn Mal in sechzig Sekunden überwachen und die Daten sichern:

```
sar -o Datei 60 10
```

Später die Platten- und Bandaktivitäten desselben Zeitraums überprüfen:

```
sar -d -f Datei
```

DATEIEN:

| | |
|---|---|
| /usr/adm/sa/sa*dd* | Tägliche Datendatei, wobei *dd* zwei Ziffern sind, die den aktuellen Tag im Monat angeben. |

## sccsdiff

sccsdiff – Vergleich zweier Versionen einer SCCS-Datei

SYNTAX:

sccsdiff -r*SID1* -r*SID2* [-p] [-s*n*] *Datei* ...

BESCHREIBUNG:

Mit sccsdiff können Sie die Unterschiede zwischen zwei Versionen einer SCCS-Datei ermitteln. Die Anzahl der angegebenen SCCS-Dateien ist beliebig groß, aber für jede Datei müssen Sie Argumente angeben.

-r*SID?* *SID?* spezifizieren die Deltas, die miteinander verglichen werden sollen. Die betreffenden Versionen werden in der angegebenen Reihenfolge automatisch an den Befehl bdiff übertragen.

-p Durch pr (print) erfolgt für jede der betreffenden Dateien die Ausgabe des Ergebnisses.

-s*n* *n* entspricht der Puffergröße der Datei, die bdiff an diff weitergibt (*n* = ganze Zahl). Dies ist dann von Nutzen, wenn diff aufgrund starker Systemauslastung nicht ausgeführt werden kann.

SIEHE AUCH:

bdiff, get, help, pr
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# sdb

## sdb – Symbolischer Debugger

SYNTAX:

sdb [-w] [-W] [-x] [*Objektdatei* [*Core-Datei* [*Verzeichnis*]]]

BESCHREIBUNG:

Sdb ist ein symbolisches Debug-Programm, das für Programme in C und F77 verwendet werden kann. Mit sdb können entsprechende Objektdateien und Core-Dateien untersucht und eine kontrollierte Umgebung für deren Ausführung bereitgestellt werden.

*Objektdatei* ist normalerweise eine ausführbare Programmdatei, die mit der Option -g (debug) kompiliert wurde. Wurde sie nicht mit der Option -g kompiliert oder handelt es sich nicht um eine ausführbare Datei, sind die symbolischen Testmöglichkeiten von sdb begrenzt, aber die Datei kann trotzdem untersucht und das Programm ausgetestet werden. Standard für *Objektdatei* ist a.out. *Core-Datei* ist eine Core-Image-Datei, die nach Ausführung von *Objektdatei* erstellt wird. Der standardmäßige Name für *Core-Datei* ist core. Die Core-Datei braucht nicht vorhanden zu sein. Geben Sie anstelle von *Core-Datei* ein '-' an, so ignoriert sdb eine gegebenenfalls vorhandene Core-Image-Datei. Namen von Quelldateien in *Objektdatei* werden in Bezug zum angegebenen Verzeichnis interpretiert.

Sie sollten wissen, daß es stets eine aktuelle Zeile und eine aktuelle Datei gibt. Ist *Core-Datei* vorhanden, ist diese Position anfangs die Zeile und die Datei, die die Quellanweisung enthält, bei der der Prozeß beendet wurde. Anderenfalls wird die Position auf die erste Zeile in main() gesetzt. Aktuelle Zeile und Datei können mit Hilfe der Kommandos für die Untersuchung von Quelldateien geändert werden.

Standardmäßig werden Warnmeldungen ausgegeben, wenn die Quelldateien, aus denen *Objektdatei* erstellt werden soll, nicht gefunden werden oder neuer sind als *Objektdatei*. Diese Prüffunktion und die entsprechenden Warnmeldungen können Sie durch Angabe der -W-Option deaktivieren.

## sdb

Namen von Variablen werden genauso wie in C oder F77 geschrieben. Variablen, die lokale Bedeutung in einer Prozedur haben, können in der Form *Prozedur:Variable* angesprochen werden. Geben Sie keinen Prozedurname an, wird standardmäßig die Prozedur genommen, die die aktuelle Zeile enthält.

Es ist auch möglich, Strukturkomponenten als *Variable.Komponente*, Zeiger auf Strukturkomponenten als *Variable->Komponente* und Array-Elemente als *Variable*[*Nummer*] anzusprechen. Zeiger können durch die Form *Zeiger*[0] dereferiert werden. Kombinationen dieser Formen sind möglich. Anstelle des Namens einer Strukturvariablen können Sie eine Zahl angeben. In diesem Fall gilt die Zahl als Adresse der Struktur, und als Schablone für die Struktur wird diejenige der letzten von sdb angesprochenen Struktur verwendet. Bei verschiedenen Kommandos kann auch eine nicht-qualifizierte Strukturvariable verwendet werden. Grundsätzlich interpretiert sdb eine Struktur als eine Gruppe von Variablen. Daher gibt sdb die Werte aller Elemente einer Struktur aus, wenn es eine Struktur anzeigen soll. Eine Ausnahme hiervon bildet die Ausgabe von Variablenadressen. Eine gesamte Struktur hat eine Adresse. Sdb zeigt diesen Wert an und nicht die Adressen der einzelnen Elemente.

Elemente eines mehrdimensionalen Arrays können durch *Variable*[*Nummer*][*Nummer*][*Nummer*]... oder als *Variable*[*Nummer*,*Nummer*,...] angesprochen werden. Statt *Nummer* können Sie auch *Nummer;Nummer* angeben, um einen Wertebereich zu spezifizieren. Durch * können Sie alle zulässigen Werte für einen Index angeben. Sie können Indizes jedoch auch ganz weglassen, wenn es sich um die letzten Indizes handelt und der gesamte Wertebereich gewünscht wird. Wie bei Strukturen zeigt sdb die Werte eines Arrays oder des Abschnitts eines Arrays an, wenn die abschließenden Indizes weggelassen werden. Es zeigt nur die Adresse des Arrays selbst bzw. des von Ihnen angegebenen Abschnitts an, wenn keine Indizes spezifiziert wurden. Ein mehrdimensionaler Parameter in einem F77-Programm kann nicht als Array angezeigt werden, sondern dies ist effektiv ein Zeiger, dessen Wert die Position des Arrays ist. Der Array selbst kann aus der aufrufenden Funktion heraus symbolisch angesprochen werden.

## sdb

Ein bestimmtes Exemplar einer Variablen auf dem Stack kann in der Form *Prozedur:Variable,Nummer* angesprochen werden. Sie können alle genannten Variationen zur Benennung von Variablen verwenden. *Nummer* ist das Vorkommen der angegebenen Prozedur auf dem Stack, wobei die oberste, d. h. jüngste Variable als erste gezählt wird. Ist keine Prozedur angegeben, wird standardmäßig die derzeit laufende Prozedur genommen.

Es ist auch möglich, eine Variable durch ihre Adresse zu spezifizieren. Sie können alle Formen ganzzahliger Konstanten, die in C gültig sind, verwenden, so daß Adressen dezimal, oktal oder hexadezimal eingegeben werden können.

Zeilennummern im Quellprogramm werden in der Form *Dateiname:Nummer* oder *Prozedur:Nummer* angesprochen. In jedem Fall wird die Zeilennummer relativ vom Anfang der Datei gezählt. Geben Sie keine Prozedur und keinen Dateinamen an, so wird standardmäßig die aktuelle Datei zugrundegelegt. Ist keine Nummer angegeben, so wird die erste Zeile der angegebenen Prozedur bzw. Datei angesprochen.

Während ein Prozeß unter sdb läuft, beziehen sich alle Adressen auf das laufende Programm; anderenfalls beziehen sie sich auf *Objektdatei* bzw. *Core-Datei*. Haben Sie die Option -w aufgerufen, können Adressen in *Objektdatei* überschrieben werden.

### Adressen

Die Adresse in einer Datei, die einer geschriebenen Adresse zugeordnet ist, wird durch einen der Datei zugeordneten Abbildungsmechanismus bestimmt. Jede Abbildung wird durch zwei Tripels (*b1, e1, f1*) und (*b2, e2, f2*) dargestellt, und die einer geschriebenen Adresse entsprechende Dateiadresse wird wie folgt errechnet:

*b1Adresse<e1*
Dateiadresse=*Adresse+f1-b1*

anderenfalls

*b2Adresse<e2*
Dateiadresse=*Adresse+f2-b2*

## sdb

Anderenfalls ist die gewünschte Adresse unzulässig. In einigen Fällen (z. B. bei Programmen mit getrenntem Befehls- und Datenbereich) können sich die beiden Segmente einer Datei überlappen.

Die anfängliche Einstellung der beiden Abbildungsmechanismen ist für normale a.out- und Core-Dateien geeignet. Hat eine der Dateien nicht den erwarteten Typ, so wird für diese Datei *b 1* auf 0, *e 1* auf die maximale Dateigröße und *f 1* auf 0 gesetzt; somit kann die gesamte Datei ohne Adreßumsetzung untersucht werden.

Damit sdb auch für große Dateien verwendet werden kann, werden alle entsprechenden Werte als 32-Bit-Ganzzahlen geführt.

### Kommandos

Folgende Kommandos stehen zum Untersuchen von Daten im Programm zur Verfügung:

t Ausgabe eines Stack-Trace des beendeten oder angehaltenen Programms.

T Ausgabe der ersten Zeile des Stack-Trace.

*Variable/lm*
Ausgabe des Wertes von *Variable* entsprechend der Länge *l* und dem Format *m*. Geben Sie *l* und *m* nicht an, wählt sdb Länge und Format passend für den Variablentyp, so wie er im Programm deklariert ist. Die Länge können Sie durch folgende Angaben spezifizieren:

- b ein Byte
- h zwei Bytes (Halbwort)
- l vier Bytes (Wort)

Für *m* sind folgende Werte zulässig:

- c Zeichen
- d Dezimal
- u Dezimal, vorzeichenlos
- o Oktal

x Hexadezimal

f 32-Bit-Gleitkommazahl, einfache Genauigkeit

g 64-Bit-Gleitkommazahl, doppelte Genauigkeit

s Die Variable wird als Pointer interpretiert. Zeichen werden ab der Adresse, auf die die Variable verweist, ausgegeben.

a Ausgabe von Zeichen ab der Adresse der Variablen. Dieses Format können Sie bei Registervariablen nicht verwenden.

p Pointer auf eine Prozedur.

i Instruktionen in Maschinensprache werden disassembliert und Adressen symbolisch ausgegeben.

l Instruktionen in Maschinensprache werden disassembliert und Adressen numerisch ausgegeben.

Die Angabe von Längen ist nur bei den Formaten d, u, o und x wirksam. Geben Sie eines dieser Formate ohne Längenangabe an, wird standardmäßig die Länge der Host-Anlage gewählt. Geben Sie bei dem Kommando s oder a eine numerische Längenspezifikation an, werden dementsprechend viele Zeichen ausgegeben. Anderenfalls erfolgt die Ausgabe fortlaufend aufeinanderfolgender Zeichen, bis entweder ein Null-Byte gefunden wird oder 128 Zeichen ausgegeben sind. Die letzte Variable läßt sich mit dem Kommando ./ noch einmal anzeigen.

Sie können die Shell-Metazeichen * und ? in Prozedur- und Variablennamen verwenden, so daß Sie diese Namen in verkürzter, generalisierter Form angeben können (eingeschränktes Pattern Matching). Ist kein Prozedurname angegeben, werden lokale Variablen der aktuellen Prozedur und globale Variablen durch die Metazeichen angesprochen; ist ein Prozedurname spezifiziert, gilt dies nur für lokale Variablen der Prozedur. Sollen nur globale Variablen angesprochen werden, geschieht dies durch :*Muster*.

*Zeilennummer*?*lm*
*Variable*:?*lm*

Den Wert der Adresse aus a.out oder dem Befehlsbereich ausgeben, die durch *Zeilennummer* bzw. *Variable* (Prozedurname) angegeben wird. Das Format wird von *lm* bestimmt. Standard ist 'i'.

## sdb

*Variable=lm*
*Zeilennummer=lm*
*Nummer=lm*

Ausgabe der Adresse von *Variable* oder *Zeilennummer*, bzw. Ausgabe des Wertes von *Nummer*. Das Format geben Sie durch *lm* vor. Ist kein Format angegeben, so wird lx verwendet. Mit der letzten Variante dieses Kommandos können Sie eine Konvertierung von dezimalen, oktalen und hexadezimalen Zahlen vornehmen.

*Variable Wert*

Der angegebenen *Variablen* wird *Wert* zugewiesen. *Wert* kann eine Zahl, eine Zeichenkonstante oder eine Variable sein. Sie müssen den *Wert* genau definieren. Ausdrücke, die mehrere Werte erzeugen, z. B. Strukturen, sind nicht zulässig. Zeichenkonstanten werden mit '*Zeichen* bezeichnet. Zahlen werden als Ganzzahlen betrachtet solange kein Dezimalpunkt oder Exponent vorhanden ist. In diesen Fällen werden sie als vom Typ „double" interpretiert. Register werder als Ganzzahlen betrachtet. Die Variable kann ein Ausdruck sein, der mehrere Variablen angibt, d. h. also beispielsweise ein Array- oder Strukturname. Geben Sie die Adresse einer Variablen an, so wird diese als Adresse einer Variablen vom Typ „int" betrachtet. Ist zur Durchführung der angegebenen Zuweisung eine Typkonvertierung erforderlich, werden die Konventionen der Sprache C angewendet.

x Ausgabe des Maschinenregisters.

X Ausgabe des aktuellen Befehls in Maschinensprache.

Folgende Kommandos stehen zur Untersuchung von Quelldateien zur Verfügung:

e *Prozedur*
e *Dateiname*
e *Verzeichnis/*
e *Verzeichnis Dateiname*

Die ersten beiden Formen machen die Datei, die die angegebene Prozedur enthält, bzw. die angegebene Datei zur aktuellen Datei. Als aktuelle Zeile gilt dann die erste Zeile in der angegebenen Prozedur bzw. Datei. Bei Quelldateien geht das System davon aus, daß diese sich in *Verzeichnis* befinden. Standard ist das aktuelle Verzeichnis. Bei den beiden letzten Formen wird der

## sdb

Wert des angegebenen Verzeichnisses geändert. Geben Sie keine Prozedur, keinen Dateinamen und kein Verzeichnis an, werden der aktuelle Prozedurname und Dateiname gemeldet.

*/Regulärer Ausdruck/*
Ab der aktuellen Zeile wird in Vorwärtsrichtung nach einer Zeile gesucht, die eine Zeichenkette enthält, die dem angegebenen regulären Ausdruck entspricht. Diese Suchfunktion gibt es auch in ed. Der abschließende / kann weggelassen werden.

*?Regulärer Ausdruck?*
Ab der aktuellen Zeile wird in Rückwärtsrichtung nach einer Zeile gesucht, die eine Zeichenkette enthält, die dem angegebenen regulären Ausdruck entspricht. Diese Suchfunktion gibt es auch in ed. Das abschließende ? kann weggelassen werden.

p Ausgabe der aktuellen Zeile.

z Ausgabe der aktuellen Zeile sowie der nächsten neun Zeilen. Die letzte ausgegebene Zeile wird zur aktuellen Zeile.

w Fenster. Insgesamt 10 Zeilen vor und hinter der aktuellen Zeile ausgeben.

*Nummer*
Die aktuelle Zeile wird die Zeile mit der angegebenen Zeilennummer. Anschließende Ausgabe der neuen aktuellen Zeile.

*Anzahl+*
Die aktuelle Zeile wird um die angegebene Anzahl Zeilen vorgerückt. Anschließende Ausgabe der neuen aktuellen Zeile.

*Anzahl-*
Die aktuelle Zeile wird um die angegebene Anzahl Zeilen zurückgesetzt. Anschließende Ausgabe der neuen aktuellen Zeile.

Folgende Kommandos stehen zur Steuerung der Ausführung des Quellprogramms zur Verfügung:

*Anzahl* r *Argumente*
*Anzahl* R
Das Programm wird mit den angegebenen Argumenten ausgeführt. Geben Sie r ohne Argumente an, so werden die vorher angegebenen Argumente für das Programm noch einmal verwendet. Das Kommando R hingegen führt das Programm ohne Argumente aus. Ein Argument, das mit < oder > beginnt, bewirkt eine

Umlenkung der Standardeingabe bzw. Standardausgabe. *Anzahl* ist die Anzahl der zu ignorierenden Prüfpunkte (Breakpoints).

*Zeilennummer* c *Anzahl*
*Zeilennummer* C *Anzahl*
Nach einem Breakpoint oder einer Unterbrechung fortfahren. *Anzahl* ist die Anzahl der zu ignorierenden Breakpoints. C fährt mit dem Signal, das zum Programmstopp geführt hat, fort, während c es ignoriert. Geben Sie eine Zeilennummer an, so wird in dieser Zeile ein temporärer Prüfpunkt gesetzt und mit der Ausführung fortgefahren. Der Breakpoint wird gelöscht, wenn das Kommando beendet ist.

*Zeilennummer* g *Anzahl*
Nach einem Breakpoint fortfahren und die Ausführung in der angegebenen Zeile wieder aufnehmen. *Anzahl* ist die Anzahl der zu ignorierenden Breakpoints.

s *Anzahl*
S *Anzahl*
Das Programm in Einzelschritten für *Anzahl* Zeilen ausführen. Geben Sie keine Anzahl an, so wird eine Programmzeile ausgeführt. S ist gleichbedeutend mit s, führt jedoch durch Unterprogramm-Aufrufe.

i
I Eine Instruktion in Maschinensprache in Einzelschritten ausführen. I bewirkt, daß das Signal, das zum Programmstopp geführt hat, reaktiviert wird, während i es ignoriert.

*Variable*$m *Anzahl*
*Adresse*:m *Anzahl*
Programm in Einzelschritten ausführen (wie bei s), bis die angegebene Adresse durch einen neuen Wert geändert wird. Geben Sie keine Anzahl an, bedeutet dies effektiv unendlich. Die Variable muß aus der aktuellen Prozedur ansprechbar sein. Da dieses Kommando durch Software ausgeführt wird, kann es unter Umständen sehr langsam ablaufen.

*Ebene* v
Den ausführlichen (verbose) Modus ausschalten, wenn in Einzelschritten mit S, s oder m ausgeführt wird. Geben Sie keine Ebene an, wird nur der Name der aktuellen Quelldatei und/oder des

aktuellen Unterprogramms ausgegeben, wenn eins dieser Elemente wechselt. Ist *Ebene* 1 oder größer, so werden alle C-Quellzeilen vor der Ausführung ausgegeben; ist *Ebene* 2 oder größer, so werden außerdem alle Assembler-Anweisungen ausgegeben.

k Abbruch des derzeit getesteten Programms.

*Prozedur*(*Argument1*,*Argument2*,...)
*Prozedur*(*Argument1*,*Argument2*,...)/*m*

Die genannte Prozedur wird mit den angegebenen Argumenten ausgeführt. Argumente können Ziffer-, Zeichen-, String-Konstanten oder Namen von Variablen sein, die von der aktuellen Prozedur angesprochen werden können. In der zweiten Form wird der von der Prozedur zurückgelieferte Wert entsprechend dem Format *m* ausgegeben. Standard ist d.

*Zeilennummer* b *Kommandos*

Setzen eines Breakpoints in der angegebenen Zeile. Geben Sie einen Prozedurnamen ohne Zeilennummer an (z. B. „progc:"), so wird ein Prüfpunkt in der ersten Zeile der Prozedur gesetzt, auch wenn diese nicht mit der Option -g kompiliert wurde. Ist keine Zeilennummer angegeben, so wird ein Prüfpunkt in die aktuelle Zeile gesetzt. Geben Sie keine Kommandos an, so wird die Ausführung unmittelbar vor dem Prüfpunkt gestoppt und die Kontrolle wieder sdb übergeben. Anderenfalls werden die Kommandos bei Erreichen des Breakpoints ausgeführt und dann die Ausführung fortgesetzt. Sie können mehrere Kommandos – durch Semikolon getrennt – angeben. Geben Sie k als Kommando zur Ausführung an einem Prüfpunkt an, wird die Ausführung nicht fortgesetzt und die Kontrolle an sdb zurückgegeben.

B Ausgabe einer Liste der derzeit aktiven Breakpoints.

*Zeilennummer* d

In der angegebenen Zeile wird der Breakpoint gelöscht. Geben Sie keine Zeilennummer an, erfolgt die Löschung der Prüfpunkte interaktiv. Die einzelnen Prüfpunktadressen werden ausgegeben. Anschließend wird jeweils eine Zeile von der Standardeingabe gelesen. Beginnt diese Zeile mit einem y oder d, wird der Prüfpunkt gelöscht.

D Löschen aller Breakpoints.

## sdb

l Ausgabe der letzten ausgeführten Zeile.

*Zeilennummer* a
Das a steht für announce, d. h. bekanntgeben. Hat *Zeilennummer* die Form *Prozedur:Nummer*, führt das Kommando effektiv eine *Zeilennummer* b l aus. Hat *Zeilennummer* die Form *Prozedur*:, führt das Kommando effektiv eine *Prozedur*: b T aus.

Diverse Kommandos:

!*Kommando*
Das angegebene Kommando wird von sh interpretiert.

Newline
Hat das vorangehende Kommando eine Quellzeile ausgegeben, wird die aktuelle Zeile um eins erhöht und die neue aktuelle Zeile ausgegeben. Hat das vorangehende Kommando eine Speicheradresse angezeigt, wird die nächste Speicheradresse ausgegeben.

Control-D
Die nächsten 10 Zeilen mit Instruktionen, Quellprogramm oder Daten ausgeben, je nachdem was zuletzt ausgegeben wurde.

<*Dateiname*
Kommandos werden aus der angegebenen Datei gelesen, bis das Dateiende erreicht ist. Wird sdb durch ein Kommando in einer solchen Datei veranlaßt, eine Variable anzuzeigen, wird der Variablenname zusammen mit dem Wert ausgegeben. Dieses Kommando kann nicht geschachtelt werden; < darf nicht als Kommando in der Datei stehen.

M Ausgabe der Adreß-Umrechnungstabellen (Maps).

M[?/][*] *b e f*
Ausgabe neuer Werte für die Adreß-Umrechnungstabelle. ? und / spezifizieren die Text- bzw. die Daten-Tabelle. Das erste Segment (*b1*, *e1*, *f1*) bleibt unverändert, außer wenn Sie '*' angeben. In diesem Fall wird das zweite Segment (*b1*, *e1*, *f1*) der Abbildung geändert. Geben Sie weniger als drei Werte an, bleiben die übrigen Parameter unverändert.

## sdb

"*Zeichenkette*
Ausgabe der angegebenen Zeichenkette. Die bei C üblichen Escape-Folgen der Form *Zeichen* werden erkannt.

q Verlassen des Debuggers.

Folgende Kommandos sind nur zum Testen des Debuggers vorgesehen:

V Ausgabe der Versionsnummer.

Q Ausgabe einer Liste der im Test befindlichen Prozeduren und Dateien.

Y Debug-Ausgabe umschalten.

DATEIEN:

a.out

core

WARNUNG:

In Textabschnitten gespeicherte Daten können von Funktionen nicht unterschieden werden.

HINWEISE:

Wird eine Prozedur aufgerufen, während das Programm an einem Prüfpunkt nicht stoppt (z. B. wenn ein Core-Image getestet wird), sind alle Variablen initialisiert, bevor die Prozedur gestartet wird. Daher ist es nicht möglich, eine Prozedur zu verwenden, die Daten aus einem Core-Image formatiert.

Die Standardausgabe von F77-Parametern ist nicht korrekt. Statt des Wertes wird die Adresse der Parameter ausgegeben.

Tracebacks, die F77-Unterprogramme mit mehreren Einsprungstellen enthalten, geben möglicherweise zu viele Argument in der falschen Reihenfolge aus. Die Werte sind jedoch korrekt.

## sdb

Der Bereich eines Array-Index bei F77 wird mit 1 bis $n$ angenommen, wobei $n$ die Dimension des Index ist. Dies ist nur dann von Bedeutung, wenn Sie den Index weglassen oder ein '*' für den vollen Bereich eingeben. Im allgemeinen gibt es keine Probleme mit Arrays, deren untere Grenze bei den Indizes nicht 1 ist.

SIEHE AUCH:

cc, f77, sh

## sdiff

sdiff – Gleichzeitige Anzeige zweier Dateien

SYNTAX:

sdiff [*Optionen*] *Datei1 Datei2*

BESCHREIBUNG:

Sdiff benutzt die Ausgabe von diff, um nebeneinander eine Liste von zwei Dateien anzuzeigen, wobei unterschiedliche Zeilen gekennzeichnet werden.
Sind die Zeilen identisch, werden sie durch einen Leerraum getrennt; ist eine Zeile nur in *Datei1* vorhanden, wird dieses durch ein '<' angezeigt. Existiert eine Zeile nur in *Datei2* wird ein '>' ausgegeben; '|' bezeichnet unterschiedliche Zeilen.

Beispiel:

```
x | y
a   a
b <
c <
d   d
  > c
```

Optionen:

-w*n* Die Zeilenlänge der Ausgabe beträgt *n* Zeichen. Standard ist 130 Zeichen pro Zeile.

-l Bei identischen Zeilen nur Anzeige von *Datei1*.

-s Identische Zeilen werden nicht angezeigt („Schweige-Modus").

-o*Ausgabedatei*
*Ausgabedatei* wird angelegt, um dem Benutzer ein kontrolliertes Mischen von *Datei1* und *Datei2* zu ermöglichen. Identische Zeilen werden automatisch in *Ausgabedatei* kopiert, unterschiedliche Zeilen werden – in der oben beschriebenen Form markiert – angezeigt.

## sdiff

Nach der Angabe jeder unterschiedlichen Zeile wartet sdiff mit einem '%'-Zeichen auf folgende Eingaben des Benutzers:

| | |
|---|---|
| l | Einfügen der linken Spalte in *Ausgabedatei*. |
| r | Einfügen der rechten Spalte in *Ausgabedatei*. |
| s | Identische Zeilen werden ab sofort nicht mehr angezeigt („Schweige-Modus"). |
| e l | Aufruf des Editors mit der linken Spalte. |
| e r | Aufruf des Editors mit der rechten Spalte. |
| e b | Aufruf des Editors mit der Verkettung der rechten und linken Spalte. |
| e | Aufruf des Editors mit einer leeren Datei. |
| v | „Schweige-Modus" wird ausgeschaltet. |
| q | Verlassen des Programms. |

Beim Verlassen des Editors wird die editierte Datei an die *Ausgabedatei* angehängt.

SIEHE AUCH:

diff, ed

# sed

## sed – Stream-Editor

SYNTAX:

sed [-n] [-e*script*] [-f*scriptdatei*] [*Datei* ...]

BESCHREIBUNG:

Die in dem Befehl sed angegebenen Dateien (falls keine angegeben: Standardeingabe) werden entsprechend der in *script* genannten Befehlsfolge bearbeitet und in die Standardausgabe kopiert.

Optionen:

-f Diese Option bewirkt, daß die Befehlsfolge der Datei *scriptdatei* entnommen wird.

-e Die Befehlsfolgen werden unmittelbar als *script* angegeben. Gibt es nur eine -e-Option und keine -f-Option, darf -e auch entfallen.

-n Die Standardausgabe wird unterdrückt.

Ein „script" besteht aus einer Folge von Ausgabebefehlen – pro Zeile ein Befehl – folgender Art:

[*Adresse* [,*Adresse*]] *Funktion* [*Argumente*]

Bei einer normalen Operation wird jede Eingabezeile einzeln in einen Puffer geschrieben, alle Befehle, die diese Zeile betreffen, sequentiell ausgeführt und der Inhalt des Puffers in die Standardausgabe übertragen.

Eine Adresse ist entweder eine Dezimalzahl, die die Zeilennummer angibt, das Zeichen '$', welches die letzte Eingabezeile adressiert oder eine Kontextadresse der Form '/*regulärer Ausdruck*/':

1. Die Zeichenfolge '\n' entspricht einem Zeilenende-Zeichen.
2. Eine Befehlszeile, die keine Adresse enthält, wirkt auf jede Zeile.
3. Eine Befehlszeile mit einer Adresse wirkt auf die entsprechende Zeile.

4. Eine Befehlszeile mit zwei Adressen wirkt auf den festgelegten Bereich, einschließlich der ersten und zweiten Adresse. (Ist die zweite Adresse kleiner oder gleich der ersten, wird nur eine Zeile bearbeitet). Danach wird der Vorgang wiederholt, indem die erste Adresse nochmals gesucht wird.

Befehle sprechen genau die nicht-selektierten Zeilen an, wenn Sie die Funktion '!' (Negation) verwenden (s. unten).

In der nachfolgenden Aufstellung ist für jede Funktion die maximal erlaubte Anzahl Adressen in Klammern angegeben.

Ein Argument mit der Bezeichnung *Text* besteht aus einer oder mehreren Zeilen. Die einzelnen Zeilen, außer der letzten, werden durch einen Backslash (\) abgeschlossen.
Das Zeichen '\' im Text wird behandelt wie ein '\' in einer Zeichenkette eines 's'-Befehls. Es wird eingesetzt, um Leerstellen und Tabulatoren am Anfang einer Zeile zu erhalten. Diese werden in *scriptdatei* somit automatisch vom Zeilenanfang entfernt.

Geben Sie bei einem 'r'- oder 'w'-Befehl eine Datei an, so muß diese am Ende einer Befehlszeile stehen, und es muß genau ein Leerzeichen vorausgehen.
Jede Ausgabedatei wird vor Beginn der Ausführung angelegt; höchstens zehn unterschiedliche Ausgabedatei-Argumente sind möglich.

(1) a\
*Text*
: Bringt den Text in die Ausgabe, bevor die nächste Eingabezeile gelesen wird (append).

(2) b [*Marke*]
: Verzweigt zum ':'-Befehl, der die angegebene Marke enthält (branch). Fehlt *Marke*, ist das Skriptende gemeint.

(2) c\
*Text*
: Löscht den Puffer. Bei einer Adresse von 0 oder 1 oder am Ende eines 2-Adressen-Bereichs wird *Text* in die Ausgabe gebracht (change). Anschließend beginnt der nächste Durchgang.

## sed

| | |
|---|---|
| (2) d | Löscht den Puffer (delete). Anschließend beginnt der nächste Durchgang. |
| (2) D | Löscht den Anfang des Puffers bis zum ersten Zeilenende-Zeichen. Der nächste Durchgang beginnt. |
| (2) g | Ersetzt den Inhalt des Puffers durch den Inhalt des Zusatzpuffers. |
| (2) G | Fügt den Inhalt des Zusatzpuffers an den Pufferinhalt an. |
| (2) h | Schreibt den Inhalt des Puffers in den Zusatzpuffer. |
| (2) H | Fügt den Pufferinhalt an den Zusatzpuffer an. |
| (1) i\<br>*Text* | Bringt den Text in die Standardausgabe (insert). |
| (2) l | Bringt den Pufferinhalt in die Standardausgabe. Dabei werden nichtdruckbare Zeichen im 2-Zeichen-ASCII-Modus ausgegeben. Zu lange Zeilen werden in mehreren Zeilen dargestellt. |
| (2) n | Kopiert den Pufferinhalt in die Standardausgabe. Ersetzt den Pufferinhalt durch die nächste Eingabezeile (next). |
| (2) N | Fügt die nächste Eingabezeile inklusive Zeilenende-Zeichen an den aktuellen Pufferinhalt an. (Die aktuelle Zeilennummer wird verändert). |
| (2) p | Kopiert den Pufferinhalt in die Standardausgabe (print). |
| (2) P | Kopiert den Anfang des Puffers bis zum ersten Zeilenende in die Standardausgabe. |
| (1) q | Verzweigt an das Ende der Befehlsfolge. Es wird kein neuer Durchgang gestartet (quit). |
| (2) r *Datei* | Liest *Datei* und schreibt sie in die Ausgabe, bevor eine neue Eingabezeile gelesen wird. |



S

## sed

(2) s/*regulärer Ausdruck*/*Zeichenkette*/[*Optionen*]
Setzt *Zeichenkette* anstelle des angegebenen regulären Ausdrucks im Puffer ein. Statt '/' kann jedes beliebige Zeichen verwendet werden.

Optionen:

*n* Ersetzt den angegebenen regulären Ausdruck *n* Mal (*n* = 1-512).

g Ersetzt den angegebenen regulären Ausdruck im gesamten Text (global).

p Der Inhalt des Puffers wird nach einer Substitution ausgegeben.

w *Datei* Fügt nach einer Substitution den Pufferinhalt an den Inhalt von *Datei* an.

(2) t *Marke* Verzweigt zum Befehl ':', dessen Argument *Marke* ist, wenn nach dem letzten Lesen einer Eingabezeile oder der letzten Ausführung eines 't'-Befehls eine Substitution erfolgt ist (test).

(2) w *Datei* Anfügen des Pufferinhalts an den Inhalt von *Datei* (write).

(2) x Tauscht die Inhalte von Puffer und Ersatzpuffer aus (exchange).

(2) y/*Zeichenkette1*/*Zeichenkette2*/
Ersetzt (transform) alle vorkommenden Zeichen von *Zeichenkette1* durch die entsprechenden Zeichen von *Zeichenkette2*. Die angegebenen Zeichenketten müssen gleich lang sein!

(2) ! *Funktion* Negation. Wendet die angegebene Funktion (oder Gruppe von Funktionen, wenn geschweifte Klammern benutzt werden) nur für Zeilen an, die nicht von Adressen angesprochen werden.

(0) : *Marke* Dieser Befehl führt nichts aus; *Marke* dient lediglich als Ansprungspunkt für 'b'- und 't'-Befehle.

(1) = Zeigt die aktuelle Zeilennummer als Zeile in der Standardausgabe an.

## sed

| | |
|---|---|
| (2) { | Führt die nachfolgenden Befehle bis '}' nur dann aus, wenn der Puffer angesprochen ist. |
| (0) | Ein leerer Befehl wird ignoriert. |
| (0) # | Erscheint das Zeichen # als erstes Zeichen in einer Scriptdatei, wird die entsprechende Zeile als Kommentarzeile interpretiert.<br>Die gemeinsame Angabe von #n bewirkt die zusätzliche Unterdrückung der Standardausgabe.<br>Eine Scriptdatei muß wenigstens aus einer Zeile bestehen, die keine Kommentarzeile ist. |

SIEHE AUCH:

awk, ed, grep

# sh, rsh

sh, rsh – Shell, restricted Shell, die normale/eingeschränkte Kommandosprache

SYNTAX:

sh [-ceiknrstuvx] [*Argumente*]

rsh [-ceiknrstuvx] [*Argumente*]

BESCHREIBUNG:

Sh ist eine Kommandosprache, die Kommandos ausführt, welche von einem Terminal oder aus einer Datei gelesen werden. Rsh ist eine eingeschränkte (restricted) Version des normalen Kommando-Interpreters sh; mit rsh werden Login-Namen und Ausführungsumgebungen eingerichtet, deren Funktionsmöglichkeiten eingeschränkter und kontrollierter sind als bei der normalen Shell. Die Bedeutung der Argumente für die Shell sind unter „Aufruf" weiter unten beschrieben.

## Definitionen

Als Blank gilt ein Tab oder ein Leerzeichen. Ein Name ist eine Folge von Buchstaben, Ziffern oder Unterstrichen, wobei das erste Zeichen ein Buchstabe oder ein Unterstrich sein muß. Ein Parameter ist ein Name, eine Ziffer oder eins der Zeichen *, @, #, ?, -, $ und !.

## Kommandos

Ein einfaches Kommando ist eine Folge nicht-leerer Wörter, die durch Blanks voneinander getrennt sind. Das erste Wort gibt den Namen des auszuführenden Kommandos an. Abgesehen von den weiter unten beschriebenen Ausnahmen werden die übrigen Wörter als Argumente an das aufgerufene Kommando übergeben. Der Kommandoname wird als Argument 0 übergeben (siehe Systemschnittstelle exec). Der Wert eines einfachen Kommandos ist sein Rückgabecode, wenn es normal endet, oder (oktal) 200+Rückgabecode, wenn es abnormal endet (Liste der Rückgabecodes siehe Systemschnittstelle signal).

## sh, rsh

Eine Pipeline ist eine Folge von ein oder mehr Kommandos, die durch | (oder aus Kompatibilitätsgründen durch ^) voneinander getrennt sind. Die Standardausgabe jedes Kommandos mit Ausnahme des letzten ist durch eine Pipe mit der Standardeingabe des nächsten Kommandos verbunden. Jedes Kommando wird als eigener Prozeß ausgeführt; die Shell wartet, bis das letzte Kommando beendet ist. Der Rückgabecode einer Pipeline ist der Rückgabecode des letzten Kommandos.

Eine Liste ist eine Folge von ein oder mehr Pipelines, die durch ;, &, && oder || voneinander getrennt und wahlweise durch ; oder & beendet wird. Von diesen vier Symbolen haben ; und & dieselbe Priorität, und zwar ist diese niedriger als bei && und ||. Die Symbole && und || haben ebenfalls gleiche Priorität. Ein Semikolon (;) bewirkt die sequentielle Ausführung der vorangehenden Pipeline; ein Und-Zeichen (&) bewirkt die asynchrone Ausführung der vorangehenden Pipeline (d. h. die Shell wartet nicht, bis diese Pipeline beendet ist). Das Symbol && (||) bewirkt, daß die sich daran anschließende Liste nur dann ausgeführt wird, wenn die vorangehende Pipeline einen Rückgabecode gleich Null (ungleich Null) zurückliefert. Zur Begrenzung von Kommandos können in einer Liste beliebig viele Neue-Zeile-Zeichen statt Semikolons angegeben werden.

Ein Kommando ist entweder ein einfaches Kommando oder eine der folgenden Kommandoformen. Wenn nichts anderes angegeben ist, ist der Rückgabewert eines Kommandos der Rückgabewert des letzten in dem Kommando ausgeführten einfachen Kommandos.

for *Name* [in *Wort* ...] do *Liste* done

Jedesmal wenn ein for-Kommando ausgeführt wird, wird *Name* der nächste Wert zugewiesen, der in der „in *Wort*-Liste" angegeben ist. Ist die „in *Wort*-Liste" weggelassen, führt das for-Kommando die do-Liste einmal für jeden gesetzten positionsgebundenen Parameter aus (siehe Parametersubstitution unten). Die Ausführung endet, wenn keine weiteren Wörter in der Liste vorhanden sind.

sh, rsh

case *Wort* in [*Muster* [| *Muster*] ...) *Liste* ;;]... esac
Ein case-Kommando führt die Liste aus, die zu dem ersten Muster gehört, das mit *Wort* übereinstimmt. Die Muster haben dasselbe Format wie Dateinamen (siehe den Abschnitt „Erzeugung von Dateinamen" weiter unten), wobei jedoch ein Schrägstrich, ein führender Punkt oder ein Punkt unmittelbar hinter einem Schrägstrich nicht explizit übereinstimmen muß.

if *Liste* then *Liste* [elif *Liste* then *Liste*]... [else *Liste*] fi
Die Liste nach if wird ausgeführt, und wenn sie einen Rückgabecode gleich Null zurückliefert, wird die Liste hinter dem ersten then ausgeführt. Anderenfalls wird die Liste hinter elif ausgeführt. Liefert sie den Wert Null zurück, wird die Liste hinter dem nächsten then ausgeführt. Ist auch dies nicht der Fall, wird die Liste hinter else ausgeführt. Wird keine else-Liste oder then-Liste ausgeführt, liefert das if-Kommando den Rückgabecode Null.

while *Liste* do *Liste* done
Ein while-Kommando führt die while-Liste wiederholt aus, und wenn der Rückgabecode des letzten Kommandos in der Liste gleich Null ist, führt es die do-Liste aus; anderenfalls endet die Schleife. Werden in der do-Liste keine Kommandos ausgeführt, liefert das while-Kommando den Rückgabecode Null; statt while können Sie until angeben, wenn die Abfrage für die Schleifenbeendigung negiert werden soll.

(*Liste*) Die angegebene Kommandoliste wird in einer Sub-Shell ausgeführt.

{*Liste*;} Ausführung von *Liste*.

Die folgenden Wörter werden nur als erstes Wort eines Kommandos erkannt und dürfen nicht in Anführungszeichen eingeschlossen sein:

if then else elif fi case esac for while until do done { }

Kommentare

Beginnt ein Wort mit #, werden dieses Wort und alle folgenden Zeichen bis zu einem Neue-Zeile-Zeichen ignoriert.

## sh, rsh

### Kommandosubstitution

Die Standardausgabe eines Kommandos, das in Accents Grave (`) eingeschlossen ist, kann als Teil eines Wortes oder vollständiges Wort verwendet werden; abschließende Neue-Zeile-Zeichen werden entfernt.

### Parametersubstitution

Mit dem Zeichen $ werden substituierbare Parameter eingeleitet. Es gibt zwei Arten von Parametern: positionsgebundene Parameter und Schlüsselwortparameter. Ist der Parameter eine Ziffer, handelt es sich um einen positionsgebundenen Parameter. Positionsgebundenen Parametern können mit set Werte zugewiesen werden. Bei Schlüsselwortparametern (auch als Variablen bezeichnet) wird die Wertezuweisung folgendermaßen vorgenommen:

*Name*=*Wert* [*Name*=*Wert*] ...

Bei *Wert* wird keine Mustererkennung (Pattern Matching) durchgeführt (es dürfen also keine Metazeichen verwendet werden). Eine Funktion und eine Variable mit demselben Namen sind nicht zulässig.

${*Parameter*}
: Der Wert (falls vorhanden) des Parameters wird substituiert. Die geschweiften Klammern sind nur dann notwendig, wenn auf den Parameter ein Buchstabe, eine Ziffer oder ein Unterstrich folgt, der nicht als Bestandteil des Parameternamens interpretiert werden soll. Ist *Parameter* ein * oder @, werden alle positionsgebundenen Parameter beginnend mit $1 substituiert (getrennt durch Leerzeichen). Der Parameter $0 wird vom Argument Null gesetzt, wenn die Shell aufgerufen wird.

${*Parameter*:-*Wort*}
: Ist der Parameter gesetzt und nicht leer, wird sein Wert substituiert; anderenfalls *Wort*.

${*Parameter*:=*Wort*}
: Ist der Parameter nicht gesetzt oder leer, den Parameter auf *Wort* setzen; anschließend wird der Wert des Parameters substituiert. Positionsgebundenen Parametern kann auf diese Weise kein Wert zugewiesen werden.

${*Parameter*:? *Wort*}
Ist der Parameter gesetzt und nicht leer, wird sein Wert substituiert; anderenfalls wird *Wort* ausgegeben und die Shell verlassen. Ist *Wort* nicht angegeben, wird die Meldung „parameter null or not set" ausgegeben.

${*Parameter*:+*Wort*}
Ist der Parameter gesetzt und nicht leer, wird *Wort* substituiert; anderenfalls erfolgt keine Substitution.

In den o. g. Formen wird *Wort* nur dann ausgewertet, wenn es als substituierter String verwendet werden soll. Daher wird in dem folgenden Beispiel pwd nur dann ausgeführt, wenn d nicht gesetzt oder leer ist:

```
echo ${d:-'pwd'}
```

Wird der Doppelpunkt (:) in den o. g. Ausdrücken weggelassen, prüft die Shell lediglich, ob der Parameter gesetzt ist oder nicht.

Die folgenden Parameter werden automatisch von der Shell gesetzt:

| | |
|---|---|
| # | Die Anzahl positionsgebundener Parameter in Dezimalschreibweise. |
| - | Von der Shell beim Aufruf oder vom set-Kommando gesetzte Optionen. |
| ? | Der vom letzten synchron ausgeführten Kommando zurückgelieferte Dezimalwert. |
| $ | Die Prozeßnummer der aktuellen Shell. |
| ! | Die Prozeßnummer des letzten aufgerufenen Hintergrundkommandos. |

Folgenden Parameter werden von der Shell benutzt:

| | |
|---|---|
| HOME | Das Standardargument (Home-Verzeichnis) für das Kommando cd. |
| PATH | Der Suchpfad für Kommandos (siehe „Ausführung" unten). Sie dürfen PATH nicht ändern, wenn Sie unter rsh arbeiten. |
| CDPATH | Der Suchpfad für das Kommando cd. |

## sh, rsh

MAIL Ist diesem Parameter der Name einer Postdatei zugewiesen, benachrichtigt die Shell den Benutzer, wenn Post in der angegebenen Datei eingetroffen ist.

PS1 Enthält das Zeichen, das die Shell bei interaktiver Benutzung ausgibt, sobald eine Eingabe erwartet wird; normalerweise $.

PS2 Falls im Dialog syntaktische Shell-Strukturen benutzt werden, die über mehrere Zeilen gehen, so gibt die Shell am Beginn jeder neuen Zeile den Wert dieses Parameters aus, bis die Struktur abgeschlossen ist; normalerweise >.

IFS Interne Feldtrennzeichen (internal field separators), d. h. normalerweise Leerzeichen, Tab und Neue-Zeile-Zeichen.

Die Shell weist PATH, PS1, PS2 und IFS Standardwerte zu; HOME sowie MAIL werden beim Anmelden gesetzt.

### Interpretation von Blanks

Nach der Parameter- und Kommandosubstitution werden die Ergebnisse der Substitution nach internen Feldtrennzeichen (die in IFS definiert sind) durchsucht und an den Stellen, an denen Trennzeichen erkannt werden, in einzelne Argumente unterteilt. Explizite Leerargumente ("" oder '') werden beibehalten. Implizite Leerargumente (die sich aus Parametern ohne Wert ergeben) werden entfernt.

### Erzeugung von Dateinamen

Nach der Substitution werden die einzelnen Kommandowörter nach den Zeichen *, ? und [ abgesucht. Wird eins dieser Zeichen gefunden, so wird das Wort als Muster (Pattern) betrachtet. Das Wort wird durch alphabetisch sortierte Dateinamen ersetzt, die mit dem Muster übereinstimmen. Wird kein Dateiname gefunden, der mit dem Muster übereinstimmt, bleibt das Wort unverändert. Das Zeichen . am Anfang eines Dateinamens oder unmittelbar nach einem / sowie das Zeichen / selbst müssen explizit übereinstimmen.

* Bezeichnet eine beliebig lange Zeichenfolge einschließlich der leeren Zeichenfolge.

? Steht für ein einzelnes beliebiges Zeichen.

[...] Steht für eins der in die eckigen Klammern eingeschlossenen Zeichen. Sie können einen Bereich von Zeichen angeben, indem der erste und der letzte Buchstabe getrennt durch ein Minuszeichen geschrieben wird. Ist das erste Zeichen nach der eckigen Klammer ein !, so sind alle Zeichen außer den angegebenen gemeint.

### Aufhebung der Spezialbedeutung

Die folgenden Zeichen haben für die Shell eine Spezialbedeutung und beenden ein Wort, außer wenn die Spezialbedeutung aufgehoben wird:

; & ( ) | ^ < > Neue-Zeile-Zeichen Leerzeichen Tab

Die Spezialbedeutung eines Zeichens können Sie aufheben, indem Sie ihm einen Backslash (\) voranstellen. Das Zeichenpaar \Neue-Zeile wird ignoriert. Alle Zeichen, die in ein Paar einfache Hochkommata (') eingeschlossen sind, haben mit Ausnahme eines einfachen Hochkommata keine Spezialbedeutung mehr. Innerhalb von Anführungszeichen ("") erfolgt eine Parameter- und Kommandosubstitution; hier wird die Spezialbedeutung der Zeichen \, ', " und $ durch ein vorangestelltes \ aufgehoben. $* ist gleichbedeutend mit „$1 $2 ...", wohingegen $@ gleichbedeutend mit „$1" ist.

### Bedienerführung

Bei der Arbeit im Dialog zeigt die Shell den Wert von PS1 als Aufforderungszeichen (Prompt) an, bevor ein Kommando gelesen wird. Geben Sie ein Neue-Zeile-Zeichen ein und sind zur Vervollständigung eines Kommandos noch weitere Eingaben erforderlich, so zeigt die Shell das sekundäre Prompt (d. h. den Wert von PS2) an.

### Ein-/Ausgabe

Bevor ein Kommando ausgeführt wird, kann seine Eingabe und Ausgabe durch eine spezielle Schreibweise, die von der Shell interpretiert wird, umgelenkt werden. Die folgenden Angaben können an beliebiger Stelle in einem einfachen Kommando angegeben werden oder können vor oder hinter einem Kommando stehen und werden nicht an das aufgerufene Kommando weitergegeben; eine Substitution erfolgt, bevor *Wort* oder *Ziffer* ausgewertet werden:

`<`*Wort* Die Datei *Wort* als Standardeingabe verwenden (Dateibeschreibungsnummer 0).

`>`*Wort* Die Datei *Wort* als Standardausgabe verwenden (Dateibeschreibungsnummer 1). Ist die Datei noch nicht vorhanden, wird sie angelegt; anderenfalls wird sie auf die Länge Null verkürzt.

`>>`*Wort* Die Datei *Wort* als Standardausgabe verwenden. Ist die Datei vorhanden, werden die Ausgabedaten an das Ende angefügt (dazu wird zunächst das Dateiende gesucht); anderenfalls wird die Datei angelegt.

`<<[-]`*Wort* Die Shell-Eingabe wird bis zu einer Zeile gelesen, die mit *Wort* identisch ist, oder bis zu einem Dateiende. Das entstehende Dokument wird zur Standardeingabe. Haben Sie die Spezialbedeutung von einem der Zeichen von *Wort* aufgehoben, so werden die Zeichen des Dokuments nicht interpretiert; anderenfalls erfolgt eine Parameter- und Kommandosubstitution. Das (nicht durch das Escape-Zeichen umgeschaltete) \Neue-Zeile wird ignoriert, und die Spezialbedeutung der Zeichen \, $, ' und des ersten Zeichens in *Wort* muß durch \ aufgehoben werden. Folgt hinter dem << ein -, so werden alle führenden Tabs aus *Wort* und aus dem Dokument entfernt.

`<&`*Ziffer* Die Datei mit der Dateibeschreibungsnummer *Ziffer* wird zur Standardeingabe. Analog für Standardausgabe mit `>&`*Ziffer*.

`<&-` Die Standardeingabe wird geschlossen. Analog für Standardausgabe mit `>&-`.

Wird einer der o. g. Angaben eine Ziffer vorangestellt, so wird diese Ziffer als Dateibeschreibungsnummer für die Datei genommen (statt der standardmäßigen 0 bzw. 1). Beispiel:

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

ordnet die Dateibeschreibungsnummer 2 der Datei zu, die derzeit die Dateibeschreibungsnummer 1 hat.

## sh, rsh

Die Reihenfolge, in der Umlenkungen angegeben werden, ist wesentlich. Die Shell wertet Umlenkungen von links nach rechts aus. Beispiel:

...1>*xxx*2>&1

ordnet zunächst der Datei *xxx* die Dateibeschreibungsnummer 1 zu. Anschließend wird die Dateibeschreibungsnummer 2 der Datei mit der Dateibeschreibungsnummer 1 (d. h. *xxx*) zugeordnet. Kehren Sie die Reihenfolge der Umlenkungen um, würde die Dateibeschreibungsnummer 2 dem Terminal zugeordnet (wenn dies vorher die Dateibeschreibungsnummer 1 hatte), und die Dateibeschreibungsnummer 1 würde der Datei *xxx* zugeordnet.

Steht hinter einem Kommando ein &, ist die Standardeingabe für das Kommando die leere Datei /dev/null. Normalerweise enthält die Umgebung für die Ausführung eines Kommandos die Dateibeschreibungsnummern der aufrufenden Shell, die durch Ein-/Ausgabespezifikationen modifiziert sind.
In der eingeschränkten Shell (rsh) ist die Umlenkung der Ausgabe nicht zulässig.

### Umgebung

Die Umgebung ist eine Liste von Name-Wert-Paaren, die genauso wie eine normale Argumentliste an ein ausgeführtes Programm übergeben wird. Die Shell kommuniziert auf verschiedene Art und Weise mit der Umgebung. Beim Aufruf sucht die Shell die Umgebung ab und erzeugt für jeden gefundenen Namen einen Parameter und gibt ihm den entsprechenden Wert. Wenn der Benutzer den Wert einiger dieser Parameter ändert oder neue Parameter anlegt, wird davon die Umgebung nicht beeinflußt, außer wenn der Parameter der Shell mit dem export-Kommando an die Umgebung gebunden wird. Die Umgebung, wie sie von einem ausgeführten Kommando gesehen wird, setzt sich also zusammen aus den unveränderten Name-Wert-Paaren, die ursprünglich von der Shell geerbt wurden, zuzüglich eventueller Änderungen oder Hinzufügungen, die alle in export-Kommandos angegeben sein müssen.

## sh, rsh

Sie können die Umgebung eines einfachen Kommandos erweitern, indem Sie ihm eine oder mehrere Zuweisungen an Parameter voranstellen:

TERM=450 *Kommando Argumente*

und

(export TERM; TERM=450; *Kommando Argumente*)

sind gleichwertig bezüglich der Kommandoausführung.

Haben Sie die Option -k gesetzt, werden alle Schlüsselwortargumente in der Umgebung plaziert, auch wenn sie nach dem Kommandonamen auftreten. Die folgenden echo-Befehle geben zuerst „a=b c" und beim zweiten Aufruf lediglich „c" aus:

```
echo a=b c
set -k
echo a=b c
```

### Signale

Die Signale INTERRUPT und QUIT für ein aufgerufenes Kommando werden ignoriert, wenn hinter dem Kommando ein & steht; anderenfalls haben Signale die Werte, die die Shell von ihrem Vaterprozeß geerbt hat. Eine Ausnahme bildet das Signal 11 (s. Kommando trap).

### Ausführung

Jedesmal wenn ein Kommando ausgeführt wird, werden die obengenannten Substitutionen durchgeführt. Außer beim Aufruf von Spezialkommandos (s. unten) wird ein neuer Prozeß kreiert und der Versuch unternommen, das Kommando durch exec (Systemschnittstelle) auszuführen.

Der Shell-Parameter PATH definiert den Suchpfad zu dem Verzeichnis, das das aufgerufene Kommando enthält. Alternative Verzeichnisnamen werden durch einen Doppelpunkt (:) voneinander getrennt. Standardsuchpfad ist :/bin:/usr/bin (aktuelles Verzeichnis, /bin und /usr/bin).

Das aktuelle Verzeichnis wird durch einen leeren Pfadnamen angegeben, der unmittelbar hinter dem Gleichheitszeichen oder zwischen den Doppelpunkten an beliebiger anderer Stelle in der Pfadliste erscheinen kann. Enthält der Kommandoname einen /, wird der Suchpfad nicht verfolgt; Kommandos dieser Art werden von der eingeschränkten Shell (rsh) nicht ausgeführt. Anderenfalls werden alle Verzeichnisse in dem Suchpfad nach einer ausführbaren Datei durchsucht. Ist die Datei ausführbar aber keine Objektdatei, wird angenommen, daß sie Shell-Kommandos enthält. Daraufhin wird eine Sub-Shell erzeugt, um die Datei zu lesen. Ein in runde Klammern eingeschlossenes Kommando wird ebenfalls in einer Sub-Shell ausgeführt.

### Spezialkommandos

Die folgenden Kommandos werden vom Interpreter direkt ausgeführt. Die Umlenkung der Ein-/Ausgabe ist für diese Kommandos nicht zulässig (außer wenn besonders darauf hingewiesen wird).

: Das Kommando hat keinerlei Wirkung. Rückgabecode = 0.

.*Datei* Kommandos aus der angegebenen Datei lesen, ausführen und dann zurückkehren. Das Verzeichnis, das die Datei enthält, wird in dem von PATH angegebenen Suchpfad gesucht.

break[$n$] Die umschließende for- oder while-Schleife verlassen. Geben Sie $n$ an, werden $n$ Stufen verlassen.

continue[$n$]
Die Ausführung mit dem nächsten Durchlauf der umschließenden for- oder while-Schleife fortsetzen. Geben Sie $n$ an, wird die Ausführung bei der $n$-ten umschließenden Schleife fortgesetzt.

cd[*Verzeichnis*]
Das angegebene Verzeichnis wird das aktuelle Verzeichnis. (Standard = HOME). CDPATH definiert den Suchpfad zu dem Verzeichnis, das das angegebene Verzeichnis enthält. Alternative Verzeichnisnamen werden durch Doppelpunkte (:) voneinander getrennt. Der standardmäßige Pfad ist leer (und gibt das aktuelle Verzeichnis an). Das aktuelle Verzeichnis wird von einem leeren Pfadnamen angegeben, der unmittelbar hinter dem Gleichheitszeichen oder an beliebiger anderer Stelle in der Pfadliste zwischen den Doppel-

punkten erscheinen kann. Beginnt das angegebene Verzeichnis mit einem /, wird der Suchpfad ignoriert. Anderenfalls werden alle Verzeichnisse des Suchpfads nach dem angegebenen Verzeichnis durchsucht. Das Kommando cd kann unter rsh nicht ausgeführt werden.

eval[*Argument* ...]
Die angegebenen Argumente werden als Shell-Eingabe gelesen, und das/die entstehende(n) Kommando(s) wird/werden ausgeführt.

exec[*Argument* ...]
Die in den Argumenten angegebenen Kommandos werden anstelle dieser Shell ausgeführt, ohne daß ein neuer Prozeß kreiert wird. Sie können Eingabe/Ausgabe-Argumente verwenden, und wenn keine anderen Argumente angegeben sind, bewirken sie die Modifikation der Shell-Eingabe/Ausgabe.

exit[*n*] Die Shell liefert den angegebenen Wert als Rückgabecode. Wird *n* nicht spezifiziert, wird der Rückgabecode des letzten ausgeführten Kommandos zurückgeliefert (EOF bewirkt ebenfalls ein Verlassen der Shell).

export[*Name* ...]
Die angegebenen Namen werden für den automatischen Export in die Umgebung anschließend ausgeführter Kommandos bestimmt. Geben Sie keine Argumente an, wird eine Liste aller Namen, die in der Umgebung der aktuellen Shell sind, ausgegeben. Funktionsnamen können Sie nicht exportieren.

newgrp[*Argument* ...]
Gleichbedeutend mit exec newgrp *Argument*...

read[*Name* ...]
Es wird eine Zeile aus der Standardeingabe gelesen, und das erste Wort wird dem ersten Namen, das zweite Wort dem zweiten Namen usw. zugeordnet. Übrigbleibende Wörter werden dem letzten Namen zugeordnet. Der Rückgabecode ist 0, es sei denn, ein Dateiende wird gefunden.

sh, rsh

readonly[*Name* ...]
Die angegebenen Namen werden als änderungsgeschützt (readonly) markiert, und die Werte dieser Namen können nicht durch spätere Zuweisungen geändert werden. Geben Sie keine Argumente an, so wird eine Liste aller änderungsgeschützten Namen ausgegeben.

set[--ekntuvx [*Argument* ...]]

-e Sofort zurückverzweigen, wenn ein Kommando mit einem Rückgabecode ungleich Null beendet wird.

-k Alle Schlüsselwortargumente werden in die Umgebung eines Kommandos plaziert und nicht nur diejenigen vor dem Kommandonamen.

-n Kommandos lesen aber nicht ausführen.

-t Nach dem Lesen und Ausführen eines Kommandos zurückverzweigen.

-u Nicht gesetzte Variablen werden beim Substituieren als Fehler behandeln.

-v Gelesene Shell-Eingabezeilen anzeigen.

-x Kommandos und die zugehörigen Argumente – wie ausgeführt – anzeigen.

-- Keine der Flags ändern; nützlich wenn $1 auf - gesetzt wird.

Geben Sie ein + statt - an, so werden diese Flags ausgeschaltet. Die Flags können beim Aufrufen der Shell angegeben werden. Der aktuelle Flag-Satz ist in $- zu finden. Die übrigen Argumente sind positionsgebundene Parameter und werden in der Reihenfolge $1, $2, ... zugewiesen. Geben Sie keine Argumente an, werden die Werte aller Namen ausgegeben.

shift[$n$] Die positionsgebundenen Parameter ab $\$n+1$ ... werden in $1... umbenannt. Fehlt $n$, so wird dafür standardmäßig der Wert 1 angenommen.

test Bedingte Ausdrücke auswerten. Gebrauch und Beschreibung siehe Benutzerkommando test.

times Die kumulierten Benutzer- und Systemzeiten für die aus der Shell ausgeführten Prozesse ausgeben.

trap[*Argument*] [*n*] ...
*Argument* ist ein Kommando, das gelesen und ausgeführt werden soll, wenn die Shell die Signale *n* erhält. Beachten Sie, daß *Argument* einmal gelesen wird, wenn trap gesetzt wird und einmal, wenn trap wirksam wird. Trap-Kommandos werden in der Reihenfolge der Signalnummern ausgeführt. Der Versuch, trap für ein Signal zu setzen, das beim Einsprung in die aktuelle Shell ignoriert wurde, bleibt unwirksam. Das Signal 11 (Speicherfehler) mit trap abzufangen, führt zu einem Fehler. Wenn *Argument* fehlt, werden alle traps *n* auf ihre ursprünglichen Werte zurückgesetzt. Ist *Argument* ein leerer String, wird dieses Signal von der Shell und von den von ihr aufgerufenen Kommandos ignoriert. Hat *n* den Wert 0, wird das Kommandoargument beim Verlassen der Shell ausgeführt. Ein trap-Kommando ohne Argumente gibt eine Liste der den einzelnen Signalnummern zugeordneten Kommandos aus.

ulimit[-fp] [*n*]
Begrenzt die Größe auf *n*.

-f Begrenzt die Größe von Dateien, die von Kindprozessen ausgegeben werden, auf *n* Blöcke (hingegen können Dateien beliebiger Größe gelesen werden). Geben Sie kein Argument an, wird die aktuelle Maximalgröße angezeigt.

-p Ändert die Größe der Pipe auf *n*.

Bei fehlender Option wird -f angenommen.

umask[*nnn*]
Die Dateierstellungsmaske (Standardwerte für Zugriffsrechte) für den Benutzer wird auf *nnn* gesetzt (siehe Benutzerkommando und Systemschnittstelle umask). Lassen Sie *nnn* weg, gibt das System den aktuellen Wert der Maske aus.

wait[$n$] Auf den angegebenen Prozeß warten und seinen Rückgabecode melden. Ist $n$ nicht angegeben, so wird auf alle derzeit aktiven Kindprozesse gewartet, und der Rückgabecode ist Null.

Aufruf

Wird die Shell über die Systemschnittstelle exec aufgerufen und das erste Zeichen von Argument 0 ist -, werden Kommandos aus /etc/profile und $HOME/.profile gelesen, falls diese Dateien existieren. Anschließend werden die angegebenen Kommandos in der unten beschriebenen Weise gelesen. Dies gilt auch, wenn die Shell durch /bin/sh aufgerufen wird. Die unten angeführten Optionen werden von der Shell nur beim Aufruf interpretiert. Hier sei darauf hingewiesen, daß immer dann, wenn die Optionen -c oder -s nicht angegeben werden, das erste Argument als Name einer Shell-Kommandodatei interpretiert wird und die übrigen Argumente als positionsgebundene Parameter an diese Kommandodatei übergeben werden:

-c*String* Kommandos werden aus *String* gelesen.

-s Sind keine weiteren Argumente mehr vorhanden, werden Kommandos von der Standardeingabe gelesen. Weitere gegebenenfalls vorhandene Argumente geben die positionsgebundenen Parameter an. Die Shell-Ausgabedaten (außer bei Spezialkommandos) werden in die Dateibeschreibungsnummer 2 geschrieben.

-i Bei Angabe dieser Option oder wenn die Eingabe und Ausgabe der Shell mit einem Terminal verbunden sind, ist diese Shell interaktiv.

-r Eingeschränkte Shell.

Die übrigen Optionen und Argumente sind beim Kommando set weiter oben beschrieben.

## sh, rsh

Hinweise nur für rsh

Mit rsh werden Login-Namen und Ausführungsumgebungen eingerichtet, deren Möglichkeiten eingeschränkter und kontrollierter sind als bei der normalen Shell. Die Möglichkeiten bei rsh sind dieselben wie bei sh, wobei allerdings folgende Kommandos nicht ausgeführt werden:

1. Wechseln des Verzeichnisses (siehe cd).
2. Setzen des Wertes von $PATH.
3. Angabe von Pfad- oder Kommandonamen mit /.
4. Umlenken der Ausgabe (> und >>).

Die genannten Einschränkungen sind nach dem Lesen der Datei .profile automatisch in Kraft.

Ist ein auszuführendes Kommando eine Shell-Prozedur, ruft rsh sh auf, um sie auszuführen. Somit ist es möglich, dem Endbenutzer Shell-Prozeduren zur Verfügung zu stellen, die den vollen Leistungsumfang der standardmäßigen Shell ausnutzen können, obwohl die Auswahl an Kommandos gleichzeitig eingeschränkt ist. Dieses Konzept geht davon aus, daß der Endbenutzer keine Schreib- und Ausführungserlaubnis in demselben Verzeichnis hat.

Insgesamt bewirkt dieses Konzept also, daß der Ersteller von .profile die vollständige Kontrolle über die Aktivitäten des Benutzers hat, indem er beim Einschalten und Anmelden bestimmte Aktionen vorgibt und den Benutzer in ein bestimmtes Verzeichnis, (bei dem es sich möglicherweise nicht um das Login-Verzeichnis handelt) führt.

Der Systemadministrator wird häufig ein Verzeichnis mit Kommandos (d. h. /usr/rbin) einrichten, das mit der eingeschränkten Shell aufgerufen werden kann.

## sh, rsh

RÜCKGABECODE:

Erkennt die Shell Fehler -- beispielsweise Syntaxfehler --, liefert sie einen Rückgabecode ungleich 0. Arbeiten Sie mit der Shell nicht interaktiv, wird die Ausführung der Shell-Datei in solch einem Fall abgebrochen. Anderenfalls liefert die Shell den Rückgabecode des letzten ausgeführten Kommandos (siehe auch das Kommando exit oben).

DATEIEN:

/etc/profile

$HOME/.profile

/tmp/sh*

/dev/null

SIEHE AUCH:

cd, env, login, newgrp, test, umask
Systemliteratur TARGON /35: „Shell"

# size

size – Ermitteln des Speicherplatzes einer Objektdatei

SYNTAX:

size [-o] [-x] [-V] [*Objektdatei* ...]

BESCHREIBUNG:

Mit diesem Befehl können Sie sich die Anzahl Bytes (als Dezimalzahl) eines Objektprogramms anzeigen lassen; geben Sie keine Dateien an, nimmt das Programm die Datei a.out.

Folgende Ausgaben erfolgen von links nach rechts:

- Größe des Programm-Codes (dezimal).
- Größe der initialisierten Variablen (dezimal).
- Größe der nicht-initialisierten Variablen (dezimal).
- Tabellengröße der globalen Symbole (dezimal).
- Gesamtgröße des Objektprogramms (dezimal, oktal, hexadezimal).

Optionen:

-o Die Ausgabe erfolgt oktal.

-x Die Ausgabe erfolgt hexadezimal.

-V Die Ausgabe enthält Angaben über die Version von *Objektdatei*.

SIEHE AUCH:

as, cc, ld

# sleep

## sleep – Verzögerung der Ausführung eines Prozesses

SYNTAX:

sleep *Zeit*

BESCHREIBUNG:

Sleep verzögert die Ausführung eines Prozesses um die in *Zeit* angegebenen Sekunden.

## sno

### sno – SNOBOL-Interpreter

SYNTAX:

sno [*Dateien*]

BESCHREIBUNG:

Sno ist ein SNOBOL-Compiler und -Interpreter (mit geringfügigen Unterschieden). Sno erhält seine Eingabe durch Verkettung der angegebenen Dateien und der Standardeingabe. Alle Eingaben bis einschließlich zu der Anweisung, die die Marke end enthält, werden als Programm betrachtet und kompiliert. Der Rest steht syspit zur Verfügung.

Sno unterscheidet sich in folgender Hinsicht von SNOBOL:

- Es gibt keine unverankerte Suche (unanchored search). Derselbe Effekt wird aber wie folgt erzielt:

  a ** b  Unverankerte Suche nach b

  a *x* x = x c  Unverankerte Zuweisung

- Es gibt keine Rückverweise (back referencing).

  x = "abc"
  a *x* x  ist eine unverankerte Suche nach abc.

- Die Funktionsdeklaration erfolgt zur Kompilierungszeit mit Hilfe des (nicht-eindeutigen) Labels define. Die Ausführung eines Funktionsaufrufs beginnt bei der Anweisung hinter define. Es ist nicht möglich, Funktionen zur Laufzeit zu definieren, und der Gebrauch des Namens define ist reserviert. Es gibt keine Möglichkeiten für automatische Variablen außer Parametern. Beispiele:

  define f()
  define f(a, b, c)

- Zu allen Marken außer define (auch zu end) muß eine nicht-leere Anweisung gehören.

- Marken, Funktionen und Variablen müssen unterschiedliche Namen haben. Insbesondere kann die nicht-leere Anweisung bei end nicht nur eine Marke benennen.

sno

- Wenn start eine Marke im Programm ist, beginnt die Programmausführung an dieser Stelle. Wenn nicht, beginnt die Ausführung bei der ersten ausführbaren Anweisung; define ist keine ausführbare Anweisung.
- Es gibt keine Built-in-Funktionen.
- Bei arithmetischen Ausdrücken sind keine Klammern erforderlich. Es gelten die normalen Prioritätsregeln. Deshalb müssen die arithmetischen Operatoren / und * durch Leerzeichen abgesetzt werden.
- Die rechte Seite von Zuweisungen darf nicht leer sein.
- Als Anführungszeichen für Literale kann " oder ' verwendet werden.
- Die Pseudo-Variable sysppt ist nicht verfügbar.

SIEHE AUCH:

awk

## sort

sort – Sortieren oder Mischen von Dateien

SYNTAX:

sort [-cmubdfinrt*Zeichen*] [+*Pos1* [-*Pos2*]] [-o*Ausgabedatei*] [*Datei*] ...

BESCHREIBUNG:

Sort sortiert die angegebenen Dateien und sendet das Ergebnis nach Standardausgabe. Wird kein Dateiname angegeben, wird die Standardeingabe gelesen.

Optionen:

| | |
|---|---|
| -b | Führende Blanks und Tabs werden ignoriert. |
| -d | Nur Buchstaben, Ziffern und Blanks werden beim Sortieren beachtet. |
| -f | Großbuchstaben werden als Kleinbuchstaben sortiert. |
| -i | Nichtdruckbare Zeichen beim nichtnumerischen Sortieren werden ignoriert. |
| -n | Numerischer Vergleich. Die Zahlen dürfen führende Blanks, ein Vorzeichen und einen Dezimalpunkt enthalten (schließt -b ein). |
| -r | Rückwärtssortierung. |
| -t*Zeichen* | Sortierfelder werden durch *Zeichen* getrennt (Standard ist Blank). |

Die Angabe von *Pos1* und *Pos2* verkürzt den Sortierschlüssel (normalerweise eine Zeile) auf ein Feld, welches bei *Pos1* beginnt und vor *Pos2* endet. Die Numerierung der Felder einer Zeile beginnt bei 0.

Ein Feld ist jeweils eine Zeichenfolge, die durch Blanks, Tabs oder Zeilenvorschub begrenzt ist. Mit der Option -t*Zeichen* wird die Begrenzung der Felder durch das angegebene Zeichen bzw. Zeilenvorschub festgelegt.

sort

*Pos1* und *Pos2* können auch in der Form *m.n* angegeben werden, wobei *m* die Zahl der Felder angibt, die vom Anfang der Zeile ab zu überspringen sind, und *n* die Zahl der zusätzlich zu überspringenden Zeichen beinhaltet. Der Defaultwert für *n* ist 0. Fehlt die Angabe von *Pos2* wird das Ende der Zeile angenommen.
Feldangaben können auch mit einer oder einer Kombination von Optionen verbunden werden. Diese Optionen gelten dann nur für das entsprechende Sortierfeld und überlagern die global angegebenen.

Beinhaltet das Sortierkriterium mehrere Felder, werden die nachfolgenden Felder nur sortiert, wenn die vorangegangenen Felder gleich sind.

Weitere Optionen:

| | |
|---|---|
| -c | Überprüft, ob die Datei bereits sortiert ist. |
| -m | Mischen der angegebenen sortierten Dateien. |
| -o*Ausgabedatei* | Speicherung des Sortierergebnisses in die angegebene Datei. |
| -u | Löscht doppelt vorkommende Zeilen in der sortierten Datei. Zeichen außerhalb der angegebenen Schlüssel werden bei diesem Vergleich ignoriert. |
| -y*KB* | Der Platz im Hauptspeicher, der von sort belegt wird, hat einen großen Einfluß auf die Ausführungszeit. Es wäre Verschwendung, einen unverhältnismäßig großen Platz zum Sortieren einer kleineren Datei zu belegen. Geben Sie diese Option nicht an, belegt sort den Speicherplatz, der ihm standardmäßig vom System zugewiesen wird. Diese Größe wird in den meisten Fällen nicht in vollem Umfang beansprucht. Durch die Angabe von *KB* können Sie die von Ihnen gewünschte Speicherplatzmenge reservieren, die sort solange belegt, bis das administrative Minimum oder Maximum übertreten wird. In diesem Fall wird dann das entsprechende Extrem benutzt. Z. B. wird bei Angabe von -y0 die Ausführung von sort mit einem Minimum an Speicherplatz gestartet. Analog dazu bewirkt die Angabe von -y ohne Argument die Belegung des Speicherplatzmaximums. |

sort

-z*Satzlänge* Die Größe der längsten gelesenen Zeile wird während der Sortierphase protokolliert, so daß die Puffer während der Mischphase allokiert werden können. Haben Sie durch Angabe der -c- oder -m-Option die Sortierphase unterdrückt, wird eine Standardlänge vorausgesetzt. Aus diesem Grund führen Zeilen, die länger als die Puffergröße sind zu einem Programmabbruch. Davor können Sie sich schützen, in dem Sie die maximale Satzlänge (oder auch mehr) mit dieser Option angeben.

Beispiele:

Sortiere den Inhalt von *Datei* und benutze dazu das zweite Feld als Sortierschlüssel:

sort +1 -2 *Datei*

Sortiere in umgekehrter Reihenfolge den Inhalt von *Datei1* und *Datei2*. Der Sortierschlüssel ist das erste Zeichen des zweiten Feldes; das Ergebnis wird in *Ausgabedatei* abgestellt:

sort -r -o *Ausgabedatei* +1.0 -1.2 *Datei1 Datei2*

HINWEIS:

Sehr lange Zeilen können verstümmelt werden.
Fehlt in der letzten Zeile einer Eingabedatei das Zeilenende-Zeichen, fügt sort eins an, gibt eine entsprechende Meldung aus und setzt die Verarbeitung fort.

SIEHE AUCH:

comm, join, uniq

## spell, hashmake, spellin, hashcheck

spell, hashmake, spellin, hashcheck – Feststellen von Rechtschreibfehlern

SNYTAX:

spell [-v] [-b] [-x] [-l] [+*lokale_Datei*] [*Dateien*]

/usr/lib/spell/hashmake

/usr/lib/spell/spellin *n*

/usr/lib/spell/hashcheck *Wortliste*

BESCHREIBUNG:

Spell vergleicht die Wörter in den angegebenen Dateien mit einer Wortliste. Wörter, die weder in der Wortliste vorkommen noch (durch bestimmte Beugungen, Präfixe und/oder Suffixe) von den Wörtern in der Wortliste abgeleitet werden können, werden auf der Standardausgabe ausgegeben. Geben Sie keine Dateien an, so werden Wörter von der Standardeingabe übernommen.

Spell ignoriert die meisten nroff-, tbl- und eqn-Steuerkommandos.

Optionen:

-v Ausgabe aller Wörter, die nicht in genau dieser Form in der Wortliste enthalten sind. Plausible Ableitungen von den in der Wortliste enthaltenen Wörtern werden angezeigt.

-b Prüfung auf britische Rechtschreibung. In diesem Fall erkennt das Programm beispielsweise nur die Formen centre, colour, programme, speciality, travelled usw. an, und Wörter wie standardise müssen mit -ise enden, gleichgültig welche Rechtschreibungsform in den maßgebenden englischen Wörterbüchern Oxford English Dictionary (OED) und Fowler angegeben ist.

-x Bei jedem Wort wird jeder plausible Stamm mit = ausgegeben.

-l Standardmäßig durchläuft spell (wie droff) Ketten verknüpfter Dateien (nroff-Kommandos .so und .nx), außer wenn die Namen der verknüpften Dateien mit /usr/lib beginnen. Bei der Option -l durchläuft spell die Ketten aller verknüpften Dateien.

*+lokale_Datei*
Die in dieser Datei enthaltenen Wörter erscheinen nicht in der Ausgabe von spell. *Lokale_Datei* ist der Name einer vom Benutzer bereitgestellten Datei, die eine Liste sortierter Wörter mit einem Wort pro Zeile enthält. Sie haben so die Möglichkeit, eine zusätzliche Liste korrekt geschriebener Wörter einzugeben, die die Wortliste von spell ergänzt.

Die Wortliste basiert auf zahlreichen verschiedenen Quellen. Die Auswahl der Wörter ist zwar willkürlicher als in einem gewöhnlichen Wörterbuch, enthält gleichzeitig aber mehr Wörter aus dem Bereich der Eigennamen und des technischen Vokabulars. Der Fachwortschatz aus den Bereichen Biologie, Medizin und Chemie wird allerdings nur in geringem Umfang abgedeckt.

Relevante Hilfsdateien können durch Namensargumente spezifiziert werden, die weiter unten mit ihren Standardeinstellungen angegeben sind (siehe DATEIEN). Kopien aller Ausgabedaten werden in der Protokolldatei (history file) gesammelt. Mit Hilfe der Kontroll-Stoppliste werden orthografische Fehler (z. B. thier=thy-y+ier) herausgefiltert, die ansonsten durchgehen würden.

Es gibt drei Unterprogramme, die beim Unterhalten und Prüfen der von spell verwendeten Kontrollisten behilflich sind:

hashmake Liest eine Liste von Wörtern aus der Standardeingabe und schreibt den entsprechenden neunstelligen Kontroll-Code in die Standardausgabe.

spellin *n* Liest *n* Kontroll-Codes aus der Standardeingabe und schreibt eine komprimierte Wortliste in die Standardausgabe.

hashcheck Liest eine komprimierte Wortliste und erstellt die neunstelligen Hash-Codes für alle darin enthaltenen Wörter neu. Diese Codes erscheinen in der Standardausgabe.

## spell, hashmake, spellin, hashcheck

BEISPIELE:

Das folgende Programmbeispiel erzeugt die aufbereitete Wortliste hlist und prüft das Ergebnis, indem es die beiden temporären Dateien vergleicht. Die beiden Dateien müssen gleich sein.

```
cat Datei | /usr/lib/spell/hashmake | sort -u >tmp 1
cat tmp 1 | /usr/lib/spell/spellin 'cat tmp1 | wc -l' >hlist
cat hlist | /usr/lib/spell/hashcheck >tmp2
diff tmp1 tmp2
```

DATEIEN:

| | |
|---|---|
| DSPELL=/usr/lib/spell/hlist[ab] | aufbereitete Wortlisten, amerikanisches und britisches Englisch |
| SSPELL=/usr/lib/spell/hstop | aufbereitete Kontroll-Stoppliste |
| HSPELL=/usr/lib/spell/spellhist | Protokolldatei |
| /usr/lib/spell/spellprog | Programm |

HINWEIS:

Die Trefferquote der Wortliste ist je nach Fachgebiet unterschiedlich. Bei neuen Anlagen ist es sinnvoll, die Ausgabedaten über längere Zeit zu überwachen und lokale Zusatzlisten anzulegen. Diese werden in einer separaten lokalen Datei gespeichert, die der aufbereiteten Wortliste über spellin hinzugefügt wird.

SIEHE AUCH:

deroff, eqn, sed, sort, tbl, tee

## spline

spline – Interpolation geglätteter Kurven

SYNTAX:

spline [*Optionen*]

BESCHREIBUNG:

Spline interpretiert Zahlenpaare aus der Standardeingabe als Abszissen und Ordinaten einer Funktion. Es errechnet eine ähnliche Menge von Zahlenpaaren, die ungefähr gleiche Abstände haben, wobei die neue Menge die Eingabemenge enthält. Die neue Menge wird auf der Standardausgabe ausgegeben. Die Ausgabe der kubischen Glättungsfunktionen (R. W. Hamming, Numerical Methods for Scientists and Engineers, 2. Ausgabe, Seite 349ff) hat zwei stetige Ableitungen und genügend viele Punkte, um bei der graphischen Darstellung glatt auszusehen.

Die folgenden Optionen werden jede für sich als separates Argument erkannt:

-a Abszissen automatisch übergeben (sie sind in der Eingabe nicht vorhanden); das Intervall wird vom nächsten Argument angegeben oder wird mit 1 angenommen, wenn das nächste Argument keine Zahl ist.

-*k* Die Konstante k, die bei der Berechnung des Randwertes verwendet wird:

(2. Ableitung bei Ende) = k * (2. Ableitung vor Ende) ????

wird vom nächsten Argument gesetzt (Standardwert von *k* = 0).

-*n* Die Ausgabepunkte in solchen Abständen anordnen, daß zwischen der unteren und oberen Grenze von x eine Anzahl von *n* Intervallen liegt (Standardwert von *n* = 100).

-p Die Ausgabe periodisch machen, d. h. die Ableitungen an den Enden übereinstimmend machen. Der erste und der letzte Eingabewert sollten normalerweise übereinstimmen.

## spline

-x Das nächste Argument (bzw. die nächsten zwei Argumente) sind die untere (und die obere) Grenze von x. Normalerweise werden diese Grenzwerte aus den Daten errechnet. Die automatischen Abszissen beginnen bei der unteren Grenze (Standardwert 0).

DIAGNOSE:

Wenn die Daten nicht streng monoton in x sind, gibt spline die Eingabe wieder, ohne zusätzliche Punkte zu interpolieren.

HINWEIS:

Es sind maximal 1000 Eingabepunkte möglich. Diese Grenze wird automatisch kommentarlos eingehalten.

# split

split – Aufteilen einer Datei auf mehrere Ausgabedateien

SYNTAX:

split [-*n*] [*Datei* [*Name*]]

BESCHREIBUNG:

Split liest die angegebene Datei und verteilt den Inhalt zeilenweise (*n* = Anzahl Zeilen; Standardwert = 1000) auf verschiedene Ausgabedateien. Die Bezeichnung der ersten Ausgabedatei ist *Name*, an die die Endung 'aa' angehängt wird. Dies erfolgt fortlaufend in alphabetischer Reihenfolge, bis hin zur Endung 'zz'. Maximal 676 Ausgabedateien sind möglich.

*Name* darf höchstens 12 Zeichen enthalten. Geben Sie keine Ausgabedatei an, wird automatisch 'x' genommen. Geben Sie keine Eingabedatei an oder verwenden stattdessen '-', so wird die Standardeingabedatei gelesen.

SIEHE AUCH:

bfs, csplit

# strip

strip – Löschen von Symboltabellen und Zeilennummer-Informationen in Objektdateien

SYNTAX:

strip [*Optionen*] *Datei* ...

BESCHREIBUNG:

Strip wird verwendet, um die Objektdatei-Größe zu verringern.
Durch strip löschen Sie in Objektdateien – Archive eingeschlossen – Symboltabelle und Zeilennummer-Informationen. Wenn Sie eine Datei mit strip bearbeitet haben, können Sie für diese Datei kein symbolisches Debugging mehr ausführen; daher ist es wichtig, daß die Datei vorher einem gründlichen Fehlertest unterzogen wird.
Die Anzahl der Informationen, die beim Aufruf des Befehls gelöscht werden, können Sie mit folgenden Optionen kontrollieren:

-l Es werden nur die Zeilennummer-Informationen gelöscht; die Symboltabelle bleibt unberücksichtigt.

-x Es werden keine Informationen über statische oder externe Symbole gelöscht.

-r Die Relativ-Adressen auf die Symboltabelle werden neu gesetzt.

-s Die Zeilennummern werden neu gesetzt.

-V Version eines strip-Befehls.

Wenn in der Objektdatei Einträge für Relativ-Adressen vorhanden sind und Symboltabellen-Informationen gelöscht werden sollen, müssen Sie die Option -r verwenden; anderenfalls erhalten Sie eine Fehlermeldung, und der strip-Befehl wird beendet.

Wird strip für eine Archivdatei aufgerufen, so wird die Symboltabelle gelöscht. Sie muß mit Hilfe des Befehls ar mit der Option -s wieder erstellt werden, damit die Archivdatei mit ld gelinkt werden kann. Vor dieser Situation werden Sie rechtzeitig durch eine Meldung gewarnt.

## strip

DATEIEN:

/usr/tmp/strp??????

MELDUNGEN:

| | |
|---|---|
| cannot open | Die Datei kann nicht gelesen werden. |
| bad magic | Die Datei ist keine Objektdatei. |
| relocation entries present; cannot strip | Enthält die Datei relative Adressen und Sie benutzen nicht die Option -r, können die Symboltabellen nicht gelöscht werden. |

SIEHE AUCH:

as, cc, ld

# stty

## stty – Einrichtung der Terminaloptionen

SYNTAX:

stty [-a] [-g] [*Option* ...]

BESCHREIBUNG:

Stty legt die Ein- und Ausgabeparameter für die aktuelle Standardeingabe fest. Ohne Argument angegeben, zeigt es bestimmte Parameter an. Die -a-Option bewirkt die Anzeige aller gesetzten Parameter. Geben Sie -g an, werden die aktuellen Parameter in der Form angezeigt, die auch als Argument eines anderen stty-Befehls benutzt werden kann.
Beachten Sie, daß einige Kombinationen der Optionen keinen Sinn ergeben. Eine Überprüfung in dieser Richtung findet nicht statt.

Bei den Optionen handelt es sich um folgende:

Kontroll-Modi:

| | |
|---|---|
| parenb (-parenb) | Betrieb mit (ohne) Paritäts-Generierung und -Überprüfung. |
| parodd (-parodd) | Ungerade (gerade) Parität. |
| cs5 cs6 cs7 cs8 | Auswahl der Zeichengröße. |
| 0 | Sofortiges Trennen von Telefonleitungen. |
| 50 75 110 134 150 200 300 600 1200 1800 2400 4800 9600 exta extb | Einstellen der Baud-Rate auf die angegebene Zahl. (Diese Geschwindigkeiten werden nicht ausnahmslos von allen Hardware-Interfaces unterstützt.) |
| hupcl (-hupcl) | Automatisches (kein automatisches) Trennen von DFÜ-Leitungen. |
| hup (-hup) | Identisch mit hupcl (-hupcl). |
| cstopb (-cstopb) | 2 Stop-Bits (1 Stop-Bit) pro Zeichen. |

## stty

| | |
|---|---|
| cread (-cread) | Einschalten (Abschalten) des Empfängers. |
| clocal (-clocal) | Einrichten einer Leitung ohne (mit) Modemkontrolle. |
| **Eingabe-Modi:** | |
| ignbrk (-ignbrk) | Eingabeunterbrechung (nicht) ignorieren. |
| brkint (-brkint) | Signalisiert (kein) Interrupt bei Unterbrechungen. |
| ignpar (-ignpar) | Paritätsfehler werden (nicht) ignoriert. |
| parmrk (-parmrk) | Paritätsfehler werden (nicht) markiert. |
| inpck (-inpck) | (Keine) Prüfung der Eingabe auf Parität. |
| istrip (-istrip) | (Keine) Umwandlung der Eingabe in 7-Bit-Code. |
| inlcr (-inlcr) | <NL> wird (nicht) in <CR> umgewandelt. |
| igncr (-igncr) | <CR> wird (nicht) ignoriert. |
| icrnl (-icrnl) | <CR> wird (nicht) in <NL> umgewandelt. |
| iuclc (-iuclc) | (Keine) Umwandlung von Großbuchstaben in Kleinbuchstaben. |
| ixon (-ixon) | Aktivieren (deaktivieren) der Start/Stop-Ausgabekontrolle. Die Ausgabe wird durch Senden von ASCII DC3 gestoppt und durch Senden von DC1 wieder gestartet. |
| ixany (-ixany) | Jedes Zeichen (nur DC1) kann die Ausgabe wieder starten. |
| ixoff (-ixoff) | (Kein) Senden von Start/Stop-Zeichen durch das System, wenn die Eingabe-Queue fast leer/voll ist. |
| **Ausgabe-Modi:** | |
| opost (-opost) | Post-Prozeß-Ausgabe. (Keine Post-Prozeß-Ausgabe. Alle anderen Ausgabe-Modi werden ignoriert.) |
| olcuc (-olcuc) | (Keine) Umwandlung von Kleinbuchstaben in Großbuchstaben. |
| onlcr (-onlcr) | (Keine) Umwandlung von <NL> in <CR>. |

## stty

| | |
|---|---|
| ocrnl (-ocrnl) | (Keine) Umwandlung von <CR> in <NL>. |
| onocr (-onocr) | <CR> wird nicht (wird) in Spalte 0 ausgegeben. |
| onlret (-onlret) | <NL> erfüllt (nicht) die CR-Funktion. |
| ofill (-ofill) | Benutzung von Füllzeichen (einer Zeitsteuerung) für Verzögerungen. |
| ofdel (-ofdel) | DEL (NUL) wird als Füllzeichen benutzt. |
| cr0 cr1 cr2 cr3 | Verzögerung von <CR>. |
| nl0 nl1 | Verzögerung von <NL>. |
| tab0 tab1 tab2 tab3 | Verzögerung der horizontalen Tabulatoren. |
| bs0 bs1 | Verzögerung von <Back Space>. |
| ff0 ff1 | Verzögerung von <Form Feed>. |
| vt0 vt1 | Verzögerung der vertikalen Tabulatoren. |

Lokale-Modi:

| | |
|---|---|
| isig (-isig) | (Keine) Prüfung der Zeichen auf die speziellen Steuerzeichen INTR, QUIT und SWTCH. |
| icanon (-icanon) | Ermöglicht (keine) vorschriftsmäßige Eingabe. (ERASE- und KILL-Ausführung.) |
| xcase (-xcase) | (Keine) veränderte Darstellung von Groß- und Kleinbuchstaben. |
| echo (-echo) | Eingegebene Zeichen werden (nicht) angezeigt. |
| echoe (-echoe) | ERASE-Zeichen werden (nicht) als Backspace-Space-Backspace-Zeichenkette angezeigt. |
| echok (-echok) | (Keine) Ausgabe von NL hinter einem KILL-Zeichen. |
| lfkc (-lfkc) | Identisch mit echok (-echok). |
| echonl (-echonl) | NL wird (nicht) angezeigt. |
| noflsh (-noflsh) | Keine Übertragung (Übertragung) nach INTR, KILL oder SWTCH. |

## stty

| | |
|---|---|
| stwrap (-stwrap) | Zeilen auf einer synchronen Leitung, die länger als 79 Zeichen sind, werden nicht (werden) getrennt. |
| stflush (-stflush) | Auf einer synchronen Leitung wird (nicht) nach jedem write übertragen. |
| stappl (-stappl) | Auf einer synchronen Leitung wird der Anwendungs-Modus (Zeilen-Modus) benutzt. |

Steueranweisungen:

| | |
|---|---|
| CTRL *c* | Das angegebene Zeichen wird in Verbindung mit <CTRL> als ERASE, KILL, INTR usw. interpretiert. |
| line *i* | Die Zeilenlänge wird auf *i* gesetzt (0 < *i* < 127). |

Kombinierte Modi:

| | |
|---|---|
| evenp oder parity | Einstellung von parenb und cs7. |
| oddp | Einstellung von parenb, cs7 und parodd. |
| -parity, -evenp oder -oddp | Ausschaltung von parenb und Einstellung von cs8. |
| raw (-raw oder cooked) | Einstellung (Ausschaltung) der raw-Betriebsart. |
| lcase (-lcase) | Einschalten (Ausschalten) von xcase, iuclc und olcuc. |
| LCASE (-LCASE) | Identisch mit lcase (-lcase). |
| tabs (-tabs oder tab3) | Erhalten der Tabulatoren (Tabulatoren werden durch Leerstellen ersetzt). |
| ek | Ersetzt das ERASE- und KILL-Zeichen durch # und @. |
| sane | Rücksetzen aller Modi auf einige passende Werte. |
| *Terminal* | Alle Modi werden passend für den angegebenen Terminaltyp gesetzt. Der Terminaltyp kann tty33, tty37, vt05, tn300, ti700 oder tek sein. |

## stty

SIEHE AUCH:

tabs

# su

## su – Zeitweilige Änderung der Benutzer-ID

SYNTAX:

su [-] [*Benutzername* [*Argument* ...]]

BESCHREIBUNG:

Su erlaubt es Ihnen, vorübergehend unter einer anderen Benutzer-ID zu arbeiten, ohne Ihre aktuelle Umgebung zu wechseln. Geben Sie su ohne *Benutzernamen* an, erhalten Sie standardmäßig die Benutzer-ID des Superusers (root).

Beim Aufruf von su wird das Paßwort des angegebenen Benutzers abgefragt, es sei denn, Sie sind bereits Superuser. Geben Sie das Paßwort korrekt an, ruft su eine neue Shell mit der ID des angegebenen Benutzers auf. Um Ihre eigene ID wiederherzustellen, geben Sie EOF ein.

Werden beim Aufruf von su zusätzliche Argumente angegeben, wird immer /bin/sh benutzt. Geben Sie kein Argument an, wird die – in der Datei /etc/passwd spezifizierte – Shell des angegebenen Benutzers aufgerufen.

Die zusätzliche Option '-' bewirkt die Änderung Ihrer aktuellen Umgebung in das Login-Verzeichnis – sowie die Ausführung der Datei .profile – des angegebenen Benutzers.

DATEIEN:

| | |
|---|---|
| /etc/passwd | Paßwortdatei |
| /etc/profile | Systemprofil |
| $HOME/.profile | Benutzerprofile |
| /usr/adm/sulog | Logdatei |

SIEHE AUCH:

env, login, sh

# sum

sum – Anzeige der Blockanzahl einer Datei sowie Berechnung einer Prüfsumme

SYNTAX:

sum [-r] *Datei*

BESCHREIBUNG:

Sum berechnet eine 16-Bit-Prüfsumme für die angegebene Datei. Die Anzahl der in *Datei* enthaltenen Blöcke wird ebenfalls ausgegeben. Dieses Kommando können Sie benutzen, um fehlerhafte Stellen in Dateien zu ermitteln oder um Dateien zu überprüfen, die über Datenfernübertragungsleitungen übermittelt wurden.

Die Option -r ruft einen alternativen Algorithmus zur Berechnung der Prüfsumme auf.

SIEHE AUCH:

wc

# sync

sync – Synchronisation der Systempuffer mit der Platte

SYNTAX:

sync

BESCHREIBUNG:

Sync ruft die Synchronisationsprozedur des Systems auf. Diese stellt sicher, daß alle noch nicht übertragenen Blöcke aus den E/A-Puffern auf die Platte geschrieben werden. Sync muß vor einem Systemstopp gestartet werden, um die Integrität des Dateisystems zu gewährleisten. Im Mehrbenutzer-Modus wird sync alle dreißig Sekunden vom System durchgeführt.

S

# tabs

tabs – Setzen von Tabs an einem Terminal

SYNTAX:

tabs [*Tabspezifikation*] [+m*n*] [-T*typ*]

BESCHREIBUNG:

Tabs setzt auf dem Benutzerterminal die Tabstopps entsprechend der *Tabspezifikation*, nachdem zuvor die bisherigen Tabstopps gelöscht wurden. Das Benutzerterminal muß mit remote-setzbaren Hardware-Tabs ausgestattet sein.

Für die Benutzer von GE TermiNet-Terminals sei speziell darauf hingewiesen, daß diese sich bei bestimmten Tabeinstellungen anders verhalten, als die meisten anderen Terminals. Die erste Nummer in der Liste der Tabpositionen wird auf einem TermiNet-Terminal der linke Rand. Ist das erste Element in der Liste der Tabnummern ungleich 1, so bleibt auf einem TermiNet-Terminal links ein Rand frei, auf anderen Terminals aber nicht. Eine Tabliste, die mit 1 beginnt, hat auf allen Terminaltypen dieselbe Wirkung. Auf einigen anderen Terminals kann ebenfalls ein linker Rand gesetzt werden, wenn auch auf andere Weise (siehe unten).

Bei der Tabspezifikation werden vier Arten unterschieden: vordefinierte Tabs, regelmäßiges Tabgitter, willkürliche Tabs und dateimäßig gespeicherte Tabs. Geben Sie keine Tabspezifikation an, wird standardmäßig -8 angenommen, d. h. die „Standardtabs" auf UNIX-Systemen. Die kleinste Spaltennummer ist 1. Bei den Tabs bezieht sich Spalte 1 immer auf die am weitesten links stehende Spalte auf dem Terminal, auch wenn auf dem Terminal die Spaltenkennzeichnungen mit 0 beginnen, z. B. DASI 300, DASI 300s und DASI 450.

-*code* Gibt den Namen einer der verfügbaren vordefinierten Tabstrukturen an. Folgende Codes stehen zur Verfügung:

-a 1,10,16,36,72
Assembler, IBM S/370, erstes Format

-a2 1,10,16,40,72
Assembler, IBM S/370, zweites Format

## tabs

-c 1,8,12,16,20,55
COBOL, normales Format

-c2 1,6,10,14,49
Kompaktes Format bei COBOL (Spalte 1-6 weggelassen). Bei diesem Code steht das erste eingegebene Zeichen in Kartenspalte 7, mit einem Leerschritt kommt man zu Spalte 8 und mit einem Tab zu Spalte 12. Dateien mit dieser Tabeinstellung sollten die folgende Formatspezifikation enthalten:

<:t -c2 m6 s66 d:>

-c3 1,6,10,14,18,22,26,30,34,38,42,46,50,54,58,62,67
Kompaktes Format bei COBOL (Spalte 1-6 weggelassen), jedoch mit mehr Tabs als bei -c2. Dieses Format wird bei COBOL empfohlen. Die entsprechende Formatspezifikation lautet wie folgt:

<: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/1

-s 1,10,55
SNOBOL

-u 1,12,20,44
UNIVAC 1100 Assembler

Neben diesen vordefinierten Formaten gibt es drei weitere Arten:

-*n* Mit dieser Spezifikation wird ein regelmäßiges Tabgitter auf den Positionen 1+*n*, 1+2***n* usw. gesetzt. Hier ist zu beachten, daß nur bei den TermiNet-Terminals ein linker Rand von *n* Spalten entsteht. Besondere Bedeutung hat der Wert -8: Er stellt die „Standard-Tabeinstellung" für UNIX-Systeme dar und bildet das Tabgitter, daß man am häufigsten auf den Terminals findet. Dieses Tabgitter ist für die nroff-Option -h für die schnelle Ausgabe unbedingt erforderlich. Ein anderer besonderer Wert ist -0, d. h. überhaupt keine Tabs.

## tabs

*n1,n2,...* Bei dem willkürlichen Format können Sie beliebige, durch Kommata getrennte, Zahlen in aufsteigender Reihenfolge eingeben. Bis zu 40 Zahlen sind zulässig. Steht vor einer Zahl (mit Ausnahme der ersten) ein Pluszeichen, so wird dieser Wert zu dem vorangehenden Wert addiert. Die Tablisten 1,10,20,30 und 1,10+10,+10 sind daher gleichbedeutend.

-- *Datei* Wenn der Name einer Datei angegeben wird, liest tabs die erste Zeile der Datei und sucht nach einer Formatspezifikation. Findet das Kommando dort eine Tabspezifikation, so setzt es die Tabstopps dementsprechend; anderenfalls setzt es sie wie bei -8. Mit dieser Spezifikationsart wird gewährleistet, daß eine mit Tabs versehene Datei mit den korrekten Tabeinstellungen ausgegeben wird. Diese Spezifikation wird mit dem Kommando pr verwendet:

tabs -- *Datei*; pr *Datei*

Die folgenden Optionen können ebenfalls angegeben werden; tritt eine Option mehrfach auf, gilt der letzte angegebene Wert:

-T*type* Tabs muß normalerweise den Terminaltyp kennen, um die Tabs zu setzen, und für das Setzen der Ränder ist die Kenntnis des Terminaltyps immer unbedingte Voraussetzung. Geben Sie keine Option -T an, sucht tabs den Wert $TERM in der Umgebung. Wird kein Typ gefunden, versucht tabs eine Einstellung, die bei vielen Terminals sinnvoll ist.

+m*n* Bei einigen Terminals kann dieses Argument für den Rand (margin) angegeben werden. Es bewirkt, daß alle Tabs um *n* Spalten verschoben werden, indem die Spalte *n*+1 zum linken Rand gemacht wird. Geben Sie +m ohne einen Wert für *n* an, wird der Wert 10 angenommen. Bei einem TermiNet-Terminal sollte der erste Wert in der Tabliste eine 1 sein, anderenfalls rutscht der Rand noch weiter nach rechts. Bei den meisten Terminals erhält man den normalen (linksbündigen) Rand durch die Angabe +m0. Bei vielen Terminals wird der Rand nur dann zurückgesetzt, wenn die Option +m explizit angegeben wird.

Das Setzen von Tabstopps und Rändern erfolgt über die Standardausgabe.

## tabs

DIAGNOSE:

| | |
|---|---|
| illegal tabs | Unzulässige Tabs. Willkürliche Tabs sind in falscher Reihenfolge angegeben. |
| illegal increment | Unzulässige Erhöhung. In einer willkürlichen Tabspezifikation ist eine Erhöhung von 0 oder keine Erhöhung angegeben. |
| unknown tab code | Unbekannter Tab-Code. Ein vordefinierter Code wird nicht gefunden. |
| can't open | Eröffnen nicht möglich. Die Option -- *Datei* wurde verwendet, und die Datei konnte nicht eröffnet werden. |
| file indirection | Weiterführender Dateiverweis. Die Option -- *Datei* wurde verwendet, und die Spezifikation in dieser Datei verweist auf eine andere Datei. Ein Dateiverweis dieser Form ist nicht zulässig. |

HINWEISE:

Die Art und Weise, wie Tabs gelöscht und der linke Rand gesetzt werden, ist von einem Terminaltyp zum nächsten unterschiedlich.
Im allgemeinen ist es nicht möglich, den linken Rand sinnvoll zu ändern, ohne gleichzeitig auch Tabs zu setzen.
Tabs löscht nur 20 Tabs (auf Terminals, die eine lange Sequenz erfordern), kann aber 40 Tabs setzen.

SIEHE AUCH:

pr

# tail

tail – Anzeigen des letzten Teils einer Datei

SYNTAX:

tail [+-[ *Zahl*] [lcb [f]]] [*Datei*]

BESCHREIBUNG:

Mit diesem Befehl haben Sie die Möglichkeit, den letzten Teil einer Datei am Bildschirm direkt anzeigen zu lassen, ohne daß die Datei vollständig ausgegeben wird. Sie können dabei exakt die Stelle festlegen, ab der die Anzeige erfolgen soll, indem Sie alternativ die Anzahl

- Zeilen [l]
- Zeichen [c]
- Blöcke [b]

angeben, die vom Anfang der Datei [+] oder vom Ende [-] gerechnet, den Beginn der Anzeige bestimmt. Wird keine *Zahl* angegeben, ist der Standard 10. Geben Sie keine Einheit an, rechnet das Programm zeilenweise.

Die Option -f (follow) bewirkt bei Eingabedateien (keine Pipes), daß das Programm nach Kopieren der angegebenen Zeilen nicht beendet wird. Stattdessen wird eine Endlos-Schleife gestartet, in der tail jeweils eine Sekunde wartet und dann versucht, neu hinzugekommene Zeilen zu lesen und zu kopieren. Dies ist sinnvoll, um das Wachstum einer Datei (z. B. einer Log-Datei) zu überwachen, die von einem anderen Prozeß beschrieben wird.

Beispiele:

tail -f *Datei*

Tail gibt die letzten 10 Zeilen der angegebenen Datei aus, gefolgt von allen Zeilen, die danach angefügt wurden.

tail -15cf *Datei*

## tail

Tail gibt die letzten 15 Zeichen der angegebenen Datei aus, gefolgt von allen Zeilen, die zwischen Start und Beendigung von tail angefügt wurden.

HINWEIS:

Die Bearbeitung von zeichenorientierten Gerätedateien mit tail ist noch nicht ausgereift und kann zu Fehlbehandlungen führen.

SIEHE AUCH:

dd

# tar

## tar – Dateienarchivierung auf Magnetband

SYNTAX:

tar [*Schlüssel*] [*Datei* ...]

BESCHREIBUNG:

Mit tar können Sie eine Anzahl von Dateien in eine einzige Datei (normalerweise auf Magnetband) sichern und rekonstruieren.

Die Befehlsausführung wird durch das Schlüssel-Argument kontrolliert, das aus einem die Funktion beschreibenden Buchstaben besteht und gegebenenfalls durch einen Änderungsfaktor ergänzt wird. Weitere Argumente sind die Dateien (oder Verzeichnisse), die mit diesem Befehl archiviert oder rekonstruiert werden sollen. Wird der Name eines Verzeichnisses als Argument eingegeben, so bezieht sich der Befehl auf sämtliche darin enthaltenen Dateien und Unterverzeichnisse.
Nachfolgend sind die als *Schlüssel* anzugebenden Funktionen beschrieben:

r Die angegebenen Dateien werden an das Ende des Magnetbandes geschrieben. Die Funktion c impliziert diese Funktion.

x Die genannten Dateien werden vom Band kopiert. Handelt es sich bei einem eingegebenen Namen um ein Verzeichnis, dessen Inhalt auf dem Magnetband gespeichert ist, so wird das vollständige Verzeichnis kopiert. Eigentümer, Änderungszeit und Modus werden – falls möglich – rekonstruiert.
Geben Sie keine Datei an, wird der gesamte Inhalt des Magnetbandes kopiert.

Sie müssen beachten, daß gleichnamige Dateien von der jüngsten Datei dieses Namens überschrieben werden.

t Die angegebenen Dateinamen werden immer dann, wenn sie auf dem Band vorkommen, aufgelistet. Haben Sie keine Datei angegeben, werden alle auf dem Magnetband gespeicherten Dateinamen aufgeführt.

u Die angegebenen Dateien werden auf dem Band gespeichert, sofern sie noch nicht vorhanden sind oder nach der letzten Ausgabe auf das Band geändert wurden.

## tar

c Ein neues Band wird angelegt. Das Sichern beginnt in diesem Fall am Anfang des Magnetbandes und nicht im Anschluß an die letzte gespeicherte Datei.

o Bei der Ausgabe speichert tar normalerweise Informationen über Eigentümer und Zugriffsrechte der Verzeichnisse im Archiv. Diese Option unterdrückt die Informationen über die Verzeichnisse.

p Unter dieser Option werden die Dateien mit ihren ursprünglichen Zugriffsrechten – ohne Berücksichtigung des aktuellen Systemaufrufs umask – rekonstruiert.

Mit folgenden Optionen können Sie die ausgewählte Funktion modifizieren:

0-9 Bestimmung des Laufwerks, auf dem das Band installiert ist. Der vorgegebene Wert ist Laufwerk 0 mit 1600 bpi (normalerweise /dev/rmt8).

v Der Name jeder bearbeiteten Datei wird unter Angabe der Funktion aufgeführt.
Zusammen mit der Funktion t gibt Ihnen v neben dem Dateinamen zusätzliche Informationen.

w Die Funktion, die ausgeführt werden soll, wird – gefolgt von dem Dateinamen – angezeigt. Diese Angaben müssen Sie durch Eingabe eines Wortes mit dem Anfangsbuchstaben y bestätigen. Jede beliebige andere Eingabe bedeutet, daß die Funktion nicht ausgeführt werden soll.

f Anstelle des Archivdateinamens /dev/mt? wird das nächste Argument als Archivname eingesetzt. Ist dieser Name '-', so schreibt das Programm in die Standardausgabe oder liest aus der Standardeingabe; Sie können tar auf diese Weise an den Anfang oder an das Ende einer Pipe stellen.
Darüber hinaus können Sie mit tar auch Hierarchien verändern:

cd fromdir; tar cf -. | (cd todir; tar xf -)

b Das nächste Argument wird als Blockungsfaktor für Magnetbandsätze genommen. 20 ist der Standardwert und gleichzeitig auch das Maximum.
Beim Lesen von Magnetbändern wird die Blockgröße automatisch festgestellt (Funktionen x und t).

## tar

l Durch diese Option veranlassen Sie, daß eine Fehlermeldung erfolgt, wenn Probleme bei den Datei-Verknüpfungen auftreten. Geben Sie l nicht an, wird im Falle eines Fehlers keine Meldung ausgegeben.

m Beim Zurückschreiben der Dateien wird das Datum der letzten Sicherung als aktuelles Datum eingesetzt.

h Unter dieser Option verfolgt tar die symbolischen Verknüpfungen, als ob sie Datendateien oder Verzeichnisse wären. Standardmäßig wird den symbolischen Links nicht gefolgt.

B Der Blockungsfaktor für Eingabe und Ausgabe ist 20 Blocks pro Satz.

Geben Sie vor einem Dateinamen -C an, führt tar den Systemaufruf chdir für diesen Dateinamen aus. Dies erlaubt das Archivieren verschiedener Verzeichnisse unter Angabe verkürzter Pfadnamen. Wenn Sie z. B. Dateien aus /usr/include und aus /etc archivieren wollen, geben sie folgenden Befehl an:

```
tar c -C /usr include -C / etc
```

DATEIEN:

/dev/rmt?

/tmp/tar*

HINWEISE:

Fehlermeldungen erfolgen für:

Fehlerhafte Angabe von *Schlüssel*, Lese-/Schreibfehler und bei zuwenig Speicherplatz um die Linktabellen zu erhalten.
Es gibt keine Möglichkeit, das *n*te Vorhandensein einer Datei abzufragen.
Die Länge für Dateinamen ist zur Zeit auf 100 Zeichen begrenzt.
Es gibt keine Möglichkeit, symbolischen Links selektiv zu folgen.

# tbl

## tbl – Preprozessor für nroff

SYNTAX:

tbl [-TX] [*Dateien*]

BESCHREIBUNG:

Tbl ist ein Preprozessor für den Textformatierer nroff. Es durchsucht den Quelltext nach Abschnitten, die als Tabellen gekennzeichnet sind. Tabellen sind in die Steuerkommandos .TS und .TE eingebettet. Tbl generiert aus dem Eingabetext, der in einer Tabellen-Beschreibungssprache formuliert ist, nroff-Eingabetext mit den entsprechenden Steuerkommandos für die korrekte Gestaltung der Tabellen.

Durch die Option -TX veranlassen Sie tbl, nur vollständige, vertikale Zeilenbewegungen zu benutzen. Dies ist sinnvoll, wenn Sie Ausgabegeräte ansprechen, auf denen keine Rückwärtspositionierung möglich ist.

SIEHE AUCH:

eqn, nroff
Systemliteratur TARGON /35: „Textformatierer nroff"

## tee

tee – Duplizieren der Standardeingabe

SYNTAX:

tee [-ia] [*Datei*] ...

BESCHREIBUNG:

Tee überträgt die Eingaben der Standardeingabe in die Standardausgabe und schreibt die Daten gleichzeitig in die angegebenen Dateien.

Optionen:

-i Ignorieren des Unterbrechungssignals.

-a Anhängen der Daten an bestehende Dateiinhalte.

## test

test – Überprüfen von Datei-Stati und Parameterwerten

SYNTAX:

test *Ausdruck*

[*Ausdruck*]

BESCHREIBUNG:

Jedes Programm und jeder Befehl gibt bei seiner Beendigung einen sogenannten Rückgabe-Code in Form einer Dezimalzahl an die Shell zurück.
Ein Code gleich Null entspricht normaler Beendigung, ein Code ungleich Null fehlerhafter Beendigung oder Abbruch.

Mit test haben Sie die Möglichkeit, einen Datei-Status oder den Wert eines Parameters oder einer Variablen zu prüfen. Test vergleicht den in *Ausdruck* vorgegebenen Prüfwert mit der angegebenen Datei und liefert den Rückgabe-Code 0, wenn dieser Wert übereinstimmt bzw. ungleich 0, wenn die Überprüfung falsch ergibt. Der Rückgabe-Code ist auch dann ungleich Null, wenn keine Argumente angegeben wurden.

Der Befehl kann folgendermaßen verwendet werden:

-r*Datei* Wahr, wenn die Datei vorhanden ist und gelesen werden kann.

-w*Datei* Wahr, wenn die Datei vorhanden ist und beschrieben werden kann.

-x*Datei* Wahr, wenn die Datei vorhanden ist und ausgeführt werden kann.

-f*Datei* Wahr, wenn die Datei vorhanden und eine Datendatei ist.

-d*Datei* Wahr, wenn die Datei vorhanden und ein Verzeichnis ist.

-c*Datei* Wahr, wenn die Datei vorhanden ist, und es sich um eine zeichenorientierte Datei handelt.

-b*Datei* Wahr, wenn die Datei vorhanden ist, und es sich um eine blockorientierte Datei handelt.

## test

-p*Datei* Wahr, wenn die Datei vorhanden ist, und es sich um eine benannte Pipe handelt.

-u*Datei* Wahr, wenn die Datei vorhanden und 'set user ID' gesetzt ist.

-g*Datei* Wahr, wenn die Datei vorhanden und 'set group ID' gesetzt ist.

-k*Datei* Wahr, wenn die Datei vorhanden und das Textsegment-Bit gesetzt ist.

-s*Datei* Wahr, wenn die Datei vorhanden und nicht leer ist.

-t[*Dateibeschreibungsnummer*]
Wahr, wenn die Datei mit der angegebenen Dateibeschreibungsnummer (Standard = 1) mit einem Terminal verbunden ist.

-z*Zeichenkette*
Wahr, wenn die Länge der angegebenen Zeichenkette 0 ist.

-n*Zeichenkette*
Wahr, wenn die Länge der angegebenen Zeichenkette nicht 0 ist.

*Zeichenkette1* = *Zeichenkette2*
Wahr, wenn die angegebenen Zeichenketten identisch sind.

*Zeichenkette1* != *Zeichenkette2*
Wahr, wenn die angegebenen Zeichenketten nicht identisch sind.

*Zeichenkette*
Wahr, wenn die angegebene Zeichenkette nicht leer ist.

*Zeichenkette1* -eq *Zeichenkette2*
Wahr, wenn die angegebenen Zeichenketten Zahlen enthalten und sich beim algebraischen Vergleich als identisch erweisen. Für den Vergleich können anstelle von -eq auch

-ne not equal = ungleich,

-gt greater than = größer als,

-ge greater equal = größer gleich,

## test

-lt less than = kleiner als und

-le less equal = kleiner gleich verwendet werden.

Alle Ausdrücke können durch die nachfolgenden Möglichkeiten miteinander verknüpft werden:

! Logische Verneinung.

-a Logisches 'Und'.

-o Logisches 'Oder'.

(*Ausdruck*) Bei Gruppenbildung müssen Klammern gesetzt werden. Dabei ist die Bedeutung der Klammern für die Shell zu beachten!

HINWEIS:

Wenn Sie die zweite Kommandoform benutzen – eckige Klammern statt test –, müssen Sie vor und hinter jeder Klammer jeweils ein Leerzeichen angeben.

SIEHE AUCH:

find, sh

# time

time – Angabe über die Laufzeit eines Programms

SYNTAX:

time *Kommando*

BESCHREIBUNG:

Das angegebene Kommando wird ausgeführt. Im Anschluß daran wird die real-gebrauchte Zeit für die gesamte Kommandoausführung, die Systemzeit und die Ausführungszeit – jeweils in Sekunden – angezeigt.

SIEHE AUCH:

timex

## timex

timex – Generierung eines Berichts über die Systemaktivitäten

SYNTAX:

timex [*Optionen*] *Befehl*

BESCHREIBUNG:

Mit diesem Befehl erhalten Sie nach der Ausführung eines Kommandos folgende Informationen – jeweils in Sekunden – über:

- die Zeitspanne zwischen Eingabe des Befehls und Ausführung,
- die Dauer der Ausführung,
- die Dauer der Inanspruchnahme des Systems.

Darüber hinaus werden Sie über Systemaktivitäten, die während der Befehlsausführung erfolgten, unterrichtet, wie

- die Dauer der Beanspruchung der CPU,
- Ein- und Ausgabeaktivitäten,
- Zugriffe auf Dateisysteme.

Es werden also sämtliche Systemaktivitäten gemeldet, nicht nur diejenigen, die auf den Befehl zurückzuführen sind.

Diese Ausgaben erfolgen in der Standard-Fehlerausgabe.

Optionen:

-p Auflistung aller Prozeßaktivitäten von *Kommando* sowie der zugehörigen Kindprozesse.

-o Angabe der von *Kommando* selbst und seiner Kindprozesse gelesenen und geschriebenen Anzahl Blöcke sowie der transferierten Zeichen.

-s Angabe **aller** Systemaktivitäten während der Ausführung von *Kommando*.

SIEHE AUCH:

acctcom, sar

# touch

touch – Aktualisierung der Zugriffs- und Änderungsdaten von Dateien

SYNTAX:

touch [-amc] [*mmddhhmm*[*yy*]] *Datei* ...

BESCHREIBUNG:

Touch aktualisiert das Modifikations- bzw. Zugriffsdatum jeder angesprochenen Datei. Geben Sie keine spezielle Zeit an, wird die aktuelle Uhrzeit eingetragen. Ist eine angesprochene Datei nicht vorhanden, wird sie angelegt.

Optionen:

-a Aktualisierung des Zugriffsdatums.

-m Aktualisierung des Änderungsdatums.

-c Nicht-existierende Dateien werden nicht angelegt.

Touch liefert die Anzahl Dateien als Rückgabe-Code, deren Modifikations- bzw. Zugriffsdatum nicht geändert werden konnte. In der Anzahl enthalten sind ebenfalls die Dateien, die nicht vorhanden waren und auch nicht angelegt wurden.

SIEHE AUCH:

date

tr

## tr – Umwandlung von Zeichen

SYNTAX:

tr [-cds] [*Zeichenkette 1* [*Zeichenkette2*]]

BESCHREIBUNG:

Dieser Befehl überträgt die Eingabe von der Standardeingabe in die Standardausgabe. Dabei werden die Zeichen der *Zeichenkette 1* in die entsprechenden Zeichen der *Zeichenkette2* umgewandelt.

Zeichenfolgen können auch verkürzt angegeben werden; so interpretiert das Programm z. B. [a-z] als die Gesamtheit aller kleinen Buchstaben des Alphabets oder [0-9] als die entsprechenden Ziffern in aufsteigender Reihenfolge.

Das Escapesymbol '\' muß – wie in der Shell – benutzt werden, um die Interpretation von Sonderzeichen zu unterdrücken.
Ein Backslash (\), gefolgt von einer ein-, zwei- oder dreistelligen Oktalzahl entspricht dem Zeichen, dessen ASCII-Code durch diese Ziffern dargestellt wird.

Optionen:

-c Alle Zeichen werden – mit Ausnahme der in *Zeichenkette 1* angegebenen – umgewandelt (Komplement). Das Komplement bezieht sich auf die Menge aller ASCII-Zeichen im Bereich 001 bis 377 oktal.

-d Alle Zeichen der *Zeichenkette 1* werden bei der Umwandlung gelöscht. Es muß keine zweite Zeichenkette angegeben werden; das Programm würde sie ignorieren.

-s Mit dieser Option werden alle sich wiederholenden Zeichen auf der Standardausgabe, die in der *Zeichenkette2* enthalten sind, auf ein Zeichen reduziert.

Die beschriebenen Optionen können beliebig miteinander kombiniert werden.

tr

Beispiel:

Alle Ziffern sollen gelöscht und die Kleinbuchstaben in Großbuchstaben umgewandelt werden:

```
tr -d [0-9] tr [a-z] [A-Z]
```

HINWEIS:

ASCII NUL wird nicht verarbeitet, sondern gleich bei der Eingabe gelöscht.

SIEHE AUCH:

ed, sh

# true

## true – Lieferung des Rückgabe-Codes 0 (wahr)

SYNTAX:

```
true
```

BESCHREIBUNG:

Der true-Befehl hat keine andere Aufgabe, als den Rückgabe-Code 0 zu liefern. True wird in Shell-Kommandoprozeduren benutzt.

Beispiel:

```
while true
do
    Kommando
done
```

SIEHE AUCH:

false, sh

# tsort

tsort – Sortieren partiell sortierter Dateien

SYNTAX:

tsort [*Datei*]

BESCHREIBUNG:

Mit tsort können Sie die mit lorder erstellte Referenzierungsordnung von Objektdateien sortieren.

Die Eingabedatei enthält Zeichenkettenpaare, die durch Leerzeichen voneinander getrennt sind. Unterschiedliche Zeichenketten-Paare zeigen die Referenz an. Identische Zeichenketten-Paare zeigen das Vorhandensein an, weisen jedoch nicht auf eine Referenz hin.

HINWEIS:

Odd data: Die Eingabedatei beinhaltet eine ungerade Anzahl von Feldern.

SIEHE AUCH:

lorder

tty

## tty – Anzeigen Terminalname

SYNTAX:

tty [-l] [-s]

BESCHREIBUNG:

Der tty-Befehl zeigt den Pfadnamen Ihres Terminals an.

Optionen:

-l Anzeige der Leitungsnummer, mit der das Terminal verbunden ist, wenn es sich um eine aktive, synchrone Leitung handelt.

-s Unterdrückung der Angabe des Pfadnamens. Es wird lediglich ein Code zurückgeliefert:

2 = Ungültige Optionen wurden angegeben.
0 = Standardeingabe ist das Terminal.
1 = Standardeingabe ist nicht das Terminal.

HINWEISE:

Geben Sie die Option -l an, wird eine Fehlermeldung ausgegeben, wenn die Standardeingabe kein synchrones Terminal ist. Ebenso erfolgt ein Hinweis, wenn die Standardeingabe kein Terminal ist und Sie die -s-Option nicht angegeben haben.

# umask

## umask – Einstellen der Standardwerte für Zugriffsrechte

SYNTAX:

umask [*OktalzifferOktalzifferOktalziffer*]

BESCHREIBUNG:

Die Standard-Werte für Zugriffsrechte beim Anlegen von Benutzerdateien werden umgestellt. Die drei Oktalziffern stehen für die Lese-, Schreib- und Ausführungsrechte für den Eigentümer, die Gruppe und andere.
Der Wert jeder angegebenen Ziffer wird von der korrespondierenden Ziffer – die vom System für das Anlegen von Dateien spezifiziert ist – subtrahiert.

Beispiel:

umask 022

Schreiberlaubnis wird der Gruppe und dem „Rest der Welt" entzogen. Dateien, die gewöhnlich mit dem Modus 777 angelegt wurden, bekommen dann den Wert 755. Dateien, die mit dem Wert 666 angelegt wurden, werden auf den Status 644 gesetzt.

Wird kein Parameter angegeben, wird der derzeitige Wert der Maske angezeigt.

SIEHE AUCH:

chmod, sh

## uname

uname – Anzeige aktueller Systemdaten

SYNTAX:

uname [-snrvma]

BESCHREIBUNG:

Uname informiert auf der Standardausgabe über aktuelle Daten des Systems. Durch Eingabe der Optionen erhalten Sie spezielle Informationen:

-s Anzeige des Systemnamens (Standard).

-n Anzeige des Knotennamens in einem Netzwerk.

-r Anzeige des Betriebssystem-Releasestandes.

-v Anzeige der UNIX-Version.

-m Anzeige des Hardware-Namen.

-a Anzeige aller o. g. Informationen.

Geben Sie nicht erlaubte Argumente an, wird standardmäßig die -s-Option angenommen.

# unget

unget – Rückgängigmachen eines vorausgegangenen get-Befehls

SYNTAX:

unget [-r*SID*] [-s] [-n] *Datei* ...

BESCHREIBUNG:

Mit unget können Sie das Ergebnis eines ausgeführten get-Befehls wieder aufheben.

Geben Sie bei diesem Befehl ein Verzeichnis an, so gilt er für jede in diesem Verzeichnis enthaltene Datei, ausgenommen Nicht-SCCS-Dateien und nicht-lesbare Dateien.
Wenn Sie '-' als Datei angeben, liest das Programm die Standardeingabe. Jede Zeile wird dann als Name einer zu bearbeitenden SCCS-Datei interpretiert.

Aus einzelnen Buchstaben bestehende Kennungen werden für jede angegebene Datei als eigenständige Argumente verarbeitet.

-r*SID* Angabe des aufzuhebenden Deltas. Diese Kennung müssen Sie nur dann angeben, wenn ein oder zwei ausstehende get-Befehle auf dieselbe SCCS-Datei unter derselben Benutzerkennung (login-Name) erfolgt sind. Sie erhalten eine Fehlermeldung, wenn diese Kennung nicht eindeutig ist oder – obwohl erforderlich – weggelassen wurde.

-s Die Ausgabe der ursprünglich geplanten Delta-Identifikationszeichenkette auf der Standardausgabe wird unterdrückt.

-n Die Datei, deren Eintrag normalerweise aus dem aktuellen Verzeichnis entfernt würde, bleibt dort bestehen.

SIEHE AUCH:

delta, get, help, sact
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# uniq

uniq – Ermittlung doppelter Zeilen einer Datei

SYNTAX:

uniq [-udc] [+*n*] [-*n*] [*Eingabedatei*] [*Ausgabedatei*]

BESCHREIBUNG:

Uniq liest die *Eingabedatei* und überprüft benachbarte Zeilen auf Identität. Die zweite der doppelt vorkommenden Zeilen wird gelöscht, die verbleibende Zeile wird in die *Ausgabedatei* geschrieben. Wird keine Ein- oder Ausgabedatei angegeben, wird Standardeingabe bzw. -ausgabe angesprochen. Damit doppelte Zeilen gefunden werden können, muß die Datei sortiert sein, da nur benachbarte Zeilen überprüft werden.

Optionen:

-u Anzeige der Zeilen, die sich nicht wiederholen.

-d Anzeige der Zeilen, die sich wiederholen.

-c Stellt jeder Zeile eine Zahl voran, aus der hervorgeht, wie häufig sie in der Datei vorhanden ist.

-*n* Ignorierung der ersten *n* Felder sowie der ihnen vorangehenden Leerstellen oder Tabs. Felder sind Folgen von Zeichen, die durch Leerstellen oder Tabs getrennt sind.

+*n* Ignorierung der ersten *n* Zeichen sowie der vorangehenden Leerstellen.

SIEHE AUCH:

comm, sort

# units

## units – Umwandeln von Maßeinheiten

SYNTAX:

units

BESCHREIBUNG:

Mit units können Sie Mengenangaben einer Maßeinheit in die entsprechenden Mengen einer anderen Maßeinheit konvertieren. Das Programm arbeitet interaktiv und zwar folgendermaßen:

Beispiel:

Konvertieren von 'inch' in 'cm'

```
* 2.540000e+00
/ 3.937008e-01
```

Eine Menge ist ein Vielfaches einer bestimmten Einheit, die durch einen vorangestellten Multiplikator ausgedrückt wird.
Potenzen werden durch eine angehängte positive oder negative ganze Zahl dargestellt, Divisionen durch das übliche Zeichen.

Beispiel:

Konvertieren von 15 lbs force/in2 in atm

```
* 1.020689e+00
/ 9.7299e-01
```

Mit units können Sie ausschließlich Maßeinheitskonvertierungen multiplikativer Art ausführen, d. h. daß Sie mit diesem Befehl Stunden in Minuten umwandeln können, jedoch nicht Celsius in Fahrenheit. Fast alle gängigen Maßeinheiten und Abkürzungen werden von dem Programm erkannt, darüber hinaus aber auch einige weniger gebräuchliche sowie mathematische Konstanten, z. B.:

## units

| | |
|---|---|
| pi | Verhältnis von Kreisumfang zu Kreisdurchmesser (3,1415) |
| c | Lichtgeschwindigkeit |
| e | Ladung eines Elektron |
| g | Konstante der Erdbeschleunigung |
| force | Analog zu g |
| mole | Zahl des Avogadro |
| water | Wasserdruck |
| au | Astronomische Einheit |

Pound wird als Maßeinheit für Gewicht nicht erkannt, jedoch lb. Englische Maßeinheiten werden durch Voranstellen der Abkürzung 'br' wie folgt von den entsprechenden amerikanischen Einheiten unterschieden: brgallon.

Mit Hilfe des Befehls:

```
cat /usr/lib/unittab
```

können Sie sich sämtliche verwendbare Maßeinheiten anzeigen lassen.

## universe, att, ucb

universe, att, ucb – Setzen bzw. Ändern der Umgebung

SYNTAX:

universe

att [-t] [*Kommando*]

ucb [-t] [*Kommando*]

BESCHREIBUNG:

Die Kommandos att und ucb erlauben Ihnen das Universum, in welchem Sie arbeiten wollen, zu setzen oder zu ändern. Unter Universum wird eine spezielle UNIX-Umgebung verstanden, entweder UNIX System V von AT+T (att) oder das 4.2BSD der University of California (ucb). Diese beiden Umgebungen existieren nebeneinander in der gleichen Dateistruktur und teilen sich einen Kernel im System TARGON /35.

Die Umgebungsstruktur erlaubt den Benutzern beider Umgebungen (att und ucb), Kommandos auszuführen und Programme zu entwickeln und ist folgendermaßen implementiert:

1. Beide Verzeichnis-Strukturen – System V und 4.2BSD – sind vorhanden.
2. Kommandos, Bibliotheken oder Header-Dateien, die normalerweise unter System V oder 4.2BSD nicht vorhanden sind, sind auch in den entsprechenden Umgebungen nicht verfügbar.
3. Kommandos, die unter beiden Versionen verfügbar sind, behalten dennoch ihre spezielle Funktionalität. Z. B. können Sie im att-Universum das Kommando find mit der Option -c*Zeit* aufrufen, während dies im ucb-Universum nicht möglich ist.

## universe, att, ucb

Das Universum, in welchem Sie sich nach dem Anmelden befinden, ist durch einen Eintrag in der Datei /etc/u_universe festgelegt. Diese Umgebung können Sie jederzeit durch Eingabe der Kommandos:

att

oder

ucb

ändern. Um zu erfahren, in welchem Universum Sie gerade arbeiten, geben Sie einfach das Kommando

universe

an.

Rufen Sie att oder ucb mit einem Kommando auf, wird das Kommando in der entsprechenden Umgebung ausgeführt. Anschließend befinden Sie sich wieder in der ursprünglichen Umgebung.

Geben Sie beim Aufruf von att oder ucb die Option -t an, wird der Terminalstatus gesichert, bevor die Umgebung gewechselt wird. Nach der Kommandoausführung oder Rückkehr aus einer Sub-Shell wird der Terminalstatus wieder hergestellt.

SIEHE AUCH:

ln

## unpack

unpack – Dekomprimieren von Dateien

SYNTAX:

unpack *Datei* ...

BESCHREIBUNG:

Unpack dekomprimiert die angegebenen Dateien. Die zu bearbeitenden Dateien müssen mit dem Suffix '.z' enden. Beim Aufruf von unpack darf der Suffix fehlen. Unpack beseitigt die '.z'-Dateien und legt ihren dekomprimierten Inhalt in Dateien ab, deren Namen durch Weglassen von '.z' gebildet werden.

Unpack liefert als Rückgabe-Code die Anzahl der Dateien, die aus einem der folgenden Gründe nicht dekomprimiert werden konnten:

- Eine Datei mit dem generierten Namen existiert bereits.
- Der angegebene Dateiname hat mehr als 12 Zeichen, so daß '.z' nicht angehängt werden kann.
- Die Datei wurde nicht von pack erzeugt.
- Die dekomprimierte Datei kann nicht angelegt werden.

SIEHE AUCH:

cat, pack, pcat

## uucp, uulog, uuname

uucp, uulog, uuname – Kopieren von UNIX- zu UNIX-System

SYNTAX:

uucp [*Optionen*] *Quelldatei ... Ziel*

uulog [*Optionen*]

uuname [-l] [-v]

BESCHREIBUNG:

### uucp

Uucp kopiert Quelldateien in das Ziel. Ein Dateiname kann hierbei ein Pfadname auf dem lokalen System sein oder folgende Form haben:

*System-Name!Pfadname*

wobei *System-Name* in einer Namenliste enthalten sein muß, auf die uucp zugreift (/usr/lib/uucp/L.sys). *System-Name* kann auch eine Liste von Systemnamen sein, die den Weg der Datei über mehrere Systeme beschreibt.

*System-Name!System-Name!....!System-Name!Pfadname*

In diesem Fall wird die Datei anhand der System-Liste von System zu System weitergereicht, wobei jedesmal nur das Verzeichnis '/usr/spool/uucppublic' das Zwischenziel ist. Im letzten System darf dann auch nur dieses Verzeichnis als Ziel angegeben werden.

Für diese Art der Übertragung muß sichergestellt sein, daß alle Zwischensysteme

1. in der Lage sind, das jeweils nächste Ziel zu erreichen, und
2. bereit sind, empfangene Dateien an das nächste System weiterzuleiten.

## uucp, uulog, uuname

In Pfadnamen der Quelldateien enthaltene Shell-Metazeichen werden auf die entsprechenden Systeme ausgedehnt. Folgende Pfadnamen sind möglich:

1. Vollständige Pfadnamen.
2. Ein Pfadname, angeführt von '~*Benutzer*', wobei *Benutzer* ein Login-Name des spezifizierten Systems ist. Dem Pfadnamen wird dann der Pfadname des LOGIN-Verzeichnisses des entsprechenden Benutzers vorangestellt.
3. Ein Pfadname, angeführt von '~/*Benutzer*', wobei *Benutzer* ein Login-Name des spezifizierten Systems ist. Dem Pfadnamen wird das Verzeichnis /usr/spool/uucppublic, erweitert um ein benutzerspezifisches Verzeichnis, vorangestellt (/usr/spool/uucppublic/Benutzer/Pfadname).
4. In allen anderen Fällen wird das aktuelle Verzeichnis vorangestellt.

Geben Sie einen falschen Pfadnamen für das Remote-System an, scheitert der Kopiervorgang.

Ist das angegebene Ziel ein Verzeichnis, wird der letzte Teil des Quelldatei-Pfadnamens als Dateiname genommen. Sind mehrere Quelldateien angegeben, muß das Ziel ein Verzeichnis sein.

Uucp erhält während der Übertragung die Programmausführungsrechte der Dateien und erteilt Schreib- und Leseerlaubnis für alle (0666).

Optionen:

-d Anlegen aller für den Kopiervorgang benötigten Verzeichnisse im Zielsystem.

-f Es werden keine Zwischenverzeichnisse für den Kopiervorgang erzeugt.

-c Kopiert die Quelldatei direkt, ohne daß diese zuerst in ein lokales Spoolverzeichnis kopiert wird.

-C Die Quelldatei wird zuerst in ein lokales Spoolverzeichnis kopiert. Aus diesem Verzeichnis wird sie dann zum nächsten System kopiert.

## uucp, uulog, uuname

-m[*Datei*] Der Status des Kopiervorgangs wird in die angegebene Datei eingetragen. Geben Sie keinen Dateinamen an, werden Sie nach Beendigung des Kopiervorgangs durch mail benachrichtigt.

-n*Benutzer* Der genannte Benutzer des Zielsystems wird durch mail vom Eintreffen einer für ihn bestimmten Datei benachrichtigt.

-e*System* Das uucp-Kommando wird mit allen Parametern an das angegebene System gesendet, damit es dort ausgeführt wird. Dies ist jedoch nur erfolgreich, wenn das entsprechende System die Erlaubnis hat, das uucp-Kommando durch das Programm /usr/lib/uucp/uuxqt ausführen zu lassen.

-r Der Job wird lediglich in die Warteschlange gestellt; der Transferprozeß wird nicht gestartet.

-j Ausgabe der uucp-Jobnummer auf der Standardausgabe.

Die uucp-Jobnummer – die Sie durch Eingabe der -j-Option erhalten – können Sie benutzen, um mit dem Kommando uustat den Status des Auftrags anzusehen oder diesen abzubrechen.

### uulog

Durch das uucp-Programmpaket werden an verschiedenen Stellen LOG-Einträge in die Dateien /usr/spool/uucp/LOGFILE und /usr/spool/uucp/SYSLOG vorgenommen.

Mit uulog können Sie sich diese Log-Einträge – ggf. gefiltert – anzeigen lassen.

Optionen:

-s*System* Alle LOG-Informationen, die das angegebene System betreffen, werden gelistet.

-u*Benutzer* Alle LOG-Informationen, die den angegebenen Benutzer betreffen, werden gelistet.

## uucp, uulog, uuname

### uuname

Uuname listet die System-Namen der uucp-bekannten Fremdsysteme. Mit der Option -l wird der Name des lokalen Systems angezeigt; durch -v erhalten Sie Zusatzinformationen über jedes System. Diese Informationen erstrecken sich auf jedes System, für das ein Eintrag in der Datei /usr/spool/uucp/ADMIN existiert. Folgendes Format ist für diese Einträge vorgeschrieben:

*Systemname* Tabulator *Beschreibung*

DATEIEN:

| | |
|---|---|
| /usr/spool/uucp | Spool-Verzeichnis |
| /usr/spool/uucppublic | Allgemeines Verzeichnis zum Senden und Empfangen (PUBDIR) |
| /usr/lib/uucp/* | Daten- und Programmdateien |

SIEHE AUCH:

mail, uux
Systemliteratur TARGON /35: „UUCP – UNIX to UNIX copy"

## uustat

uustat – Statusabfrage und Job Control für uucp

SYNTAX:

uustat [*Optionen*]

BESCHREIBUNG:

Uustat zeigt den Status von vorher eingegebenen uucp-Kommandos an, bricht diese ab oder liefert allgemeine Statusinformationen über uucp-Verbindungen zu anderen Systemen.

Optionen:

-j[*Jobnr*] Der Status des uucp-Auftrags mit der angegebenen Jobnummer wird angezeigt. Geben Sie anstelle der Jobnummer 'all' an, wird der Status aller uucp-Aufträge gelistet. Wird keine spezielle Jobnummer angegeben, werden alle uucp-Aufträge des aufrufenden Benutzers ausgegeben.

-k*Jobnr* Der uucp-Auftrag mit der angegebenen Jobnummer wird abgebrochen. Ein uucp-Auftrag kann nur vom Auftraggeber selbst oder vom Superuser abgebrochen werden.

-r*Jobnr* Der uucp-Auftrag mit der angegebenen Jobnummer wird als erneut eingegeben gekennzeichnet. Dadurch wird verhindert, daß durch den zyklischen Aufruf von uuclean dieser als zu alt erkannt und gelöscht wird.

-c*Stunden* Statuseinträge, die älter als die angegebene Anzahl Stunden sind, werden entfernt. Diese Verwaltungsoption kann nur vom Benutzer uucp oder vom Superuser aufgerufen werden.

-u*Benutzer* Der Status aller uucp-Aufträge, die vom angegebenen Benutzer abgesetzt wurden, werden gelistet.

-s*System* Der Status aller uucp-Aufträge, die mit dem angegebenen Fremdsystem in Verbindung stehen, werden gelistet.

-o*Stunden* Statuseinträge, die älter als die angegebene Anzahl Stunden sind, werden gelistet.

## uustat

-y*Stunden* Statuseinträge, die jünger als die angegebene Anzahl Stunden sind, werden gelistet.

-m*System* Der aktuelle Kommunikationsstatus zum angegebenen System wird angezeigt. Geben Sie 'all' an, wird der Status aller Systeme ausgegeben, die dem lokalen uucp bekannt sind.

-M*System* Die Bedeutung dieser Option entspricht der Option -m, nur daß zusätzlich zum aktuellen Status die Zeit des zuletzt erfolgreichen Systemzugangs gelistet wird.

-O Die Ausgabe der Statusinformationen erfolgt in oktaler Verschlüsselung.

-q Für jede dem lokalen uucp bekannte Maschine wird die Anzahl der Jobs, bestimmte Kontrolldateien und der Zeitpunkt des ältesten und des jüngsten uucp-Auftrags der jeweiligen Warteschlange gelistet. Falls eine Sperrdatei zur Synchronisierung verschiedener Aufträge existiert, wird deren Erzeugungszeitpunkt angezeigt.

Wird keine Option angegeben, werden die Stati aller uucp-Aufträge gelistet, die vom aufrufenden Benutzer abgesetzt worden sind. Beachten Sie, daß nur eine der Optionen -j, -m, -k, -c bzw. -r in Verbindung mit den anderen Optionen angegeben werden darf.

Beispiel:

```
uustat -utax -sTARGON /351 -y72
```

listet den Status aller uucp-Aufträge, die vom Benutzer 'tax' innerhalb der letzten 72 Stunden abgesetzt wurden, um mit dem System 'TARGON /351' Verbindung aufzunehmen.

## uustat

Der Job-Status setzt sich wie folgt zusammen:

Job-Nummer
Benutzer
Fremdsystem
Job-Start-Zeit
Zeitpunkt der letzten Statusänderung
Status

Der Status wird entweder im Klartext oder als Oktalzahl (Option -O) angezeigt.

Der Oktalcode entspricht den folgenden Statusmeldungen:

| Oktalcode | Status |
|---|---|
| 000001 | Kopiervorgang nicht erfolgreich; die Ursache kann nicht spezifiziert werden. |
| 000002 | Erlaubnis, auf die lokale Datei zuzugreifen, wird verweigert. |
| 000004 | Erlaubnis, auf die Fremddatei zuzugreifen, wird verweigert. |
| 000010 | Unzulässiges uucp-Kommando wurde erzeugt. |
| 000020 | Fremdsystem kann keine temporäre Datei einrichten. |
| 000040 | Es kann nicht in Fremdverzeichnis kopiert werden. |
| 000100 | Es kann nicht in lokales Verzeichnis kopiert werden. |
| 000200 | Lokales System kann keine temporäre Datei erzeugen. |
| 000400 | Uucp-Kommando kann nicht ausgeführt werden. |
| 001000 | Kopiervorgang (teilweise) erfolgreich. |
| 002000 | Kopiervorgang beendet, Job gelöscht. |
| 004000 | Job ist in Warteschlange eingereiht. |
| 010000 | Job gelöscht (Job unvollständig). |
| 020000 | Job gelöscht (Job vollständig). |

## uustat

Der Kommunikationsstatus eines Fremdsystem sieht folgendermaßen aus:

Fremdsystemname Zeitpunkt Status

wobei 'Zeitpunkt' der Zeitpunkt der letzten Statusänderung und 'Status' eine selbsterklärende verbale Beschreibung ist.

DATEIEN:

| | |
|---|---|
| /usr/spool/uucp | Spool-Verzeichnis |
| /usr/lib/uucp/L_stat | System-Status Datei |
| /usr/lib/uucp/R_stat | Auftrags-Status Datei |

SIEHE AUCH:

uucp
Systemliteratur TARGON /35: „UUCP – UNIX to UNIX copy"

## uuto, uupick

uuto, uupick – UNIX-zu-UNIX Datei-Transfer mit vereinfachter Dateizugriffsstruktur

SYNTAX:

uuto [*Optionen*] *Quelldatei* ... *Ziel*

uupick [-s*System*]

BESCHREIBUNG:

### uuto

Uuto sendet Quelldateien des lokalen Systems mit Hilfe des uucp-Programmpakets in das Ziel eines Fremdsystems. Dabei wird die Möglichkeit der lokalen Zugriffskontrolle durch uucp genutzt. Quelldateien sind hierbei Dateien im Sinne von uucp. Das Ziel wird in folgender Form angegeben:

*System!Benutzer*

wobei *System* ein dem uucp-Programm bekanntes System ist. *Benutzer* ist der Login-Name eines dem Fremdsystem bekannten Benutzers.

Optionen:

-p Die Quelldateien werden vor der Übertragung ins Fremdsystem zunächst in ein lokales Spoolverzeichnis kopiert.

-m Der Absender der Datei erhält eine Nachricht, wenn der Kopiervorgang beendet ist.

Die Dateien (oder Teilbäume, falls Verzeichnisse angegeben sind) werden im Zielsystem in das Verzeichnis PUBDIR, genauer PUBDIR/receive/*Benutzer*/*System*, gesendet.

Der Empfänger im Fremdsystem wird durch mail über die Ankunft von Dateien informiert.

## uuto, uupick

### uupick

Nach der Benachrichtigung kann der Benutzer durch uupick die gesendeten Dateien bearbeiten. Hierbei durchsucht uupick das Verzeichnis PUBDIR nach Dateien, die für den Benutzer bestimmt sind. Für jeden gefundenen Eintrag (Datei oder Verzeichnis) wird die folgende Nachricht auf der Standardausgabe angezeigt:

from system: [file *Dateiname*] [dir *Verzeichnis*] ?

Nach der Anzeige erwartet uupick eine Eingabezeile von der Standardeingabe um festzustellen, wie mit der Datei verfahren werden soll:

| | |
|---|---|
| <CR> | Gehe zum nächsten Eintrag. |
| d | Lösche den Eintrag. |
| m[*Verzeichnis*] | Bringe den Eintrag in das angegebene Verzeichnis (Standard ist das Arbeitsverzeichnis). |
| a[*Verzeichnis*] | Gleiche Funktion wie m, außer daß **alle** Dateien, die gesendet wurden, in das angegebene Verzeichnis kopiert werden. |
| p | Drucke den Dateiinhalt. |
| q | Beende die Funktion. |
| EOT | Wie q. |
| !*Kommando* | Das angegebene Kommando wird ausgeführt. |
| * | Ausgabe einer Kommandozusammenfassung. |

Rufen Sie uupick mit der Option -s*System* auf, wird das Verzeichnis PUBDIR nur nach Einträgen des angegebenen Systems durchsucht.

SIEHE AUCH:

mail, uucp, uustat, uux
Systemliteratur TARGON /35: „UUCP – UNIX to UNIX copy"

## uux

uux – Ausführung von Kommandos auf Fremdsystemen

SYNTAX:

uux [*Optionen*] *Kommando*

BESCHREIBUNG:

Mit dem Kommando uux können mehrere Dateien auf unterschiedlichen Systemen angesprochen, ein Kommando auf einem bestimmten System ausgeführt und die Standardausgabe des Kommandos wiederum an ein anderes System gesendet werden.

Aus Sicherheitsgründen kann bei der Installation eines Systems die Durchführbarkeit von Kommandos dieser Art eingeschränkt werden.

Das aufzurufende Kommando kann – wie ein lokales Kommando – aus mehreren Argumenten zusammengesetzt werden, wobei hier jedoch vor dem eigentlichen Kommandonamen und den ggf. folgenden Dateinamen ein Systemname, gefolgt von dem Zeichen '!', angegeben werden kann. Geben Sie keinen Systemnamen an, wird das Kommando lokal ausgeführt bzw. die Dateien werden als lokal angenommen.

Dateinamen können sein:

- Vollständige Pfadnamen.
- Ein Pfadname, der durch die Zeichenfolge '~*xxx*' eingeleitet wird, wobei '*xxx*' ein Login-Name auf dem spezifizierten System ist. In diesem Fall wird die Zeichenfolge durch das entsprechende Login-Verzeichnis ersetzt.
- Jeder anderen Zeichenkette wird das aktuelle Verzeichnis vorangestellt.

## uux

Beispiel:

Durch die Eingabe

```
uux "!diff usg!/usr/dan/f1 pwba!/a4/dan/f2 > !f.diff"
```

werden zunächst die Dateien f1 bzw. f2 von den Systemen usg bzw. pwba auf das lokale System kopiert. Anschließend werden diese Dateien auf dem lokalen System mit dem Kommando diff verglichen und das Ergebnis wird im lokalen System in die Datei f.diff des aktuellen Verzeichnisses eingetragen.

Zeichen, die für den Kommando-Interpreter Shell eine besondere Bedeutung haben (z. B.: <>|;), sind besonders zu beachten. Sie können die Bedeutung dieser Zeichen entweder jeweils durch individuelle Behandlung (z. B. '>') ausschalten oder dadurch, daß Sie das gesamte Kommando durch doppelte Hochkommata einrahmen (s. o. Beispiel).

Uux versucht alle benötigten Dateien auf das System zu kopieren, in dem das Kommando ausgeführt werden soll. Ausgabedateien sind durch Einklammern zu kennzeichnen.

Beispiel:

```
uux TARGON /35/1!uucp TARGON /35/2!/usr/file
    \(TARGON /35/3!/usr/file2)
```

Das uucp-Kommando wird auf dem System TARGON /35/1 ausgeführt. Dabei wird die Datei /usr/file des Systems TARGON /35/2 zunächst zum System TARGON /35/1 kopiert, um von dort aus in die Datei /usr/file2 des Systems TARGON /35/3 weitergeleitet zu werden.

Optionen:

- Die Standardeingabe von uux wird Standardeingabe des Kommandos.

-n Der Benutzer wird nicht benachrichtigt, daß das Kommando ausgeführt werden konnte. Standardmäßig erhält der Benutzer nach der Ausführung des Kommandos durch mail eine Nachricht.

-m*Datei* Der Status der Übertragung wird in die angegebene Datei eingetragen.

Uux gibt auf der Standardausgabe eine Nummer zurück. Diese Nummer ist die Job-Nummer, mit der Sie durch uustat den jeweiligen Status des Auftrags abfragen bzw. den Auftrag abbrechen können.

DATEIEN:

| | |
|---|---|
| /usr/spool/uucp | Spool-Verzeichnis |
| /usr/spool/uucppublic | Allgemeines Verzeichnis zum Senden und Empfangen (PUBDIR) |
| /usr/lib/uucp/* | Daten- und Programmdateien |

SIEHE AUCH:

uucp, uustat
Systemliteratur TARGON /35: „UUCP – UNIX to UNIX copy"

# val

val – Ermitteln von SCCS-Dateien

SYNTAX:

val -

val [-s] [-r*SID*] [-m*Name*] [-y*Typ*] *Datei* ...

BESCHREIBUNG:

Val ermittelt, ob die angegebene Datei eine SCCS-Datei ist und fügt die in der optionalen Argumentenliste spezifizierten Merkmale ein.

Argumente zu val können beliebig eingegeben werden. Die Argumente setzen sich aus Schlüsselbuchstaben (beginnend mit '-') und Dateinamen zusammen.

Geben Sie '-' an, liest val solange aus der Standardeingabe, bis das EOF-Zeichen erkannt wird. Jede einzelne gelesene Zeile wird als Kommandozeile interpretiert.

Val gibt für jede ausgeführte Kommandozeile und Datei auf der Standardausgabe eine Meldung aus. Nach Beendigung der Befehlsausführung wird ein 8-Bit-Code zurückgegeben.

Jeder Schlüsselbuchstabe wird, unabhängig von der Reihenfolge seiner Eingabe, auf jede Datei angewandt, die in der Kommandozeile angegeben ist. Die Optionen sind folgendermaßen definiert:

-s Unterdrückt die Fehlermeldungen, die normalerweise für jeden Fehler ausgegeben werden.

-r*SID* Das Argument *SID* (Identifikationszeichenkette) ist eine SCCS-Deltanummer. Sie wird auf Mehrdeutigkeit bzw. Richtigkeit überprüft. Ist die angegebene SID weder ungültig noch mehrdeutig, wird überprüft, ob sie z. Z. vorhanden ist.

-m*Name* Der Wert von *Name* wird mit dem SCCS-%M%-Schlüsselwort in der angegebenen Datei verglichen.

-y*Typ* Der Wert von *Typ* wird mit dem SCCS-%Y%-Schlüsselwort in der angegebenen Datei verglichen.

## val

HINWEIS:

Val kann bis zu fünfzig Dateien in einer einzelnen Kommandozeile bearbeiten. Die Angabe von mehr als fünfzig Dateien führt zu einem Programmabbruch.

SIEHE AUCH:

admin, delta, get, help, prs
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# vc

## vc – Versionskontrolle

SYNTAX:

vc [-a] [-t] [-c*Zeichen*] [-s] [*Schlüsselwort*=*Wert*... *Schlüsselwort*=*Wert*]

BESCHREIBUNG:

Das Kommando vc kopiert, gesteuert von seinen Argumenten und den in der Standardeingabe auftretenden Steueranweisungen, Zeilen von der Standardeingabe zur Standardausgabe. Während des Kopiervorgangs können die vom Benutzer deklarierten Schlüsselwörter durch ihren Wert ersetzt werden, wenn sie in normalem Text und/oder Steueranweisungen auftreten.

Das Kopieren von Zeilen von der Standardeingabe zur Standardausgabe erfolgt bedingt und wird durch die Abfrage (in Steueranweisungen) von Schlüsselwortwerten gesteuert, die in Steueranweisungen oder als Argumente des Kommandos vc angegeben sind.

Eine Steueranweisung ist eine einzelne Zeile, die mit einem Steuerzeichen beginnt, außer wenn eine Modifikation durch -t erfolgt (siehe unten). Das Standard-Steuerzeichen ist der Doppelpunkt (:), wenn Sie nicht mit Hilfe der Option -c ein anderes Steuerzeichen definieren. Eingabezeilen, die mit einem umgekehrten Schrägstrich (\) und einem nachfolgenden Steuerzeichen beginnen, sind keine Steueranweisungen und werden – ohne den Backslash – in die Standardausgabe kopiert. Zeilen, die mit einem Backslash und einem Zeichen beginnen, das kein Steuerzeichen ist, werden vollständig kopiert.

Ein Schlüsselwort besteht aus neun oder weniger alphanumerischen Zeichen. Das erste Zeichen muß ein Buchstabe sein. Ein Wert ist ein beliebiger ASCII-String, der mit ed erzeugt werden kann. Ein numerischer Wert ist eine vorzeichenlose Folge von Ziffern. Schlüsselwortwerte dürfen keine Leerzeichen oder Tabs enthalten.

vc

Die Ersetzung von Schlüsselwörtern durch ihre Werte erfolgt immer dann, wenn ein von Steuerzeichen umgebenes Schlüsselwort in einer Versionskontrollanweisung gefunden wird. Die Option -a fordert zwingend die Ersetzung von Schlüsselwörtern in allen Textzeilen. Ein nicht-interpretiertes Steuerzeichen kann in einem Wert angegeben werden, wenn ihm ein \ vorangestellt wird. Wünschen Sie ein \ als Textzeichen, muß ihm ebenfalls ein \ vorangestellt werden.

Optionen:

-a Schlüsselwörter, die von Steuerzeichen umgeben sind, werden in allen Textzeilen und nicht nur in vc-Anweisungen durch den ihnen zugeordneten Wert ersetzt.

-t Zum Erkennen einer Steueranweisung werden alle Zeichen vom Zeilenanfang bis einschließlich des ersten Tabs ignoriert. Wird eine Steueranweisung gefunden, so werden alle Zeichen bis einschließlich des Tabs unterdrückt.

-c*Zeichen*
Das angegebene Zeichen ersetzt den Doppelpunkt als Steuerzeichen.

-s Unterdrückung von Warnmeldungen (keine Fehlermeldungen), die normalerweise in der Diagnoseausgabe erscheinen.

Versionskontrollanweisungen

:dcl *Schlüsselwort*[,..., *Schlüsselwort*]
Deklaration von Schlüsselwörtern. Alle Schlüsselwörter müssen deklariert werden.

:asg *Schlüsselwort*=*Wert*
Zuweisung von Werten für Schlüsselwörter. Eine asg-Anweisung setzt die Zuweisung für das entsprechende Schlüsselwort in der vc-Kommandozeile und alle vorangehenden Zuweisungen mit asg für dieses Schlüsselwort außer Kraft. Schlüsselwörter, die deklariert sind, denen aber keine Werte zugewiesen sind, haben Null-Wert.

## VC

:if *Bedingung*

.

.

.

:end

Hiermit werden Zeilen in der Standardeingabe übersprungen. Ist die Bedingung wahr, werden alle Zeilen zwischen der if-Anweisung und der entsprechenden end-Anweisung auf die Standardausgabe kopiert. Ist die Bedingung falsch, werden alle dazwischenliegenden Zeilen einschließlich der Steueranweisungen ignoriert. Dazwischenliegende if-Anweisungen und entsprechende end-Anweisungen werden ausschließlich zum Zweck der einwandfreien if-end-Zuordnung erkannt. Eine Bedingung hat folgende Syntax:

<Bedingung>::= ["not"] <oder>
<oder> ::= <und> | <und> "|" <oder>
<und> ::= <Ausdruck> | <Ausdruck> "&" <und>
<Ausdruck> ::= "(" <oder> ")" | <Wert> <Operator> <Wert>
<Operator> ::= "=" | "!=" | "<" |"2"
<Wert> ::= <beliebiger ASCII-String> | <numerischer String>

Die folgenden Operatoren stehen Ihnen zur Verfügung:

= gleich

!= ungleich

& und

| oder

> größer

< kleiner

( ) Klammerung zur logischen Gruppierung

not kann nur unmittelbar nach dem if stehen und kehrt den Wert der gesamten Bedingung um

Die Operatoren > und < gelten nur für vorzeichenlose Ganzzahlen (012 > 12 ist beispielsweise falsch). Bei allen anderen Operatoren können Strings als Argumente angegeben werden (012 != 12 ist beispielsweise wahr). Die Operatoren haben die folgenden Prioritäten in absteigender Reihenfolge:

VC

= != > < haben dieselbe Priorität
&
|

Die Priorität der Operatoren kann durch Klammerung verändert werden. Zwischen Werten und Operatoren oder Klammern muß mindestens ein Leerzeichen oder Tab stehen.

::*Text* Dient zur Ersetzung von Schlüsselwörtern in Zeilen, die in die Standardausgabe kopiert werden. Die beiden Steuerzeichen am Anfang werden entfernt, und von Steuerzeichen umgebene Schlüsselwörter im Text werden durch ihren Wert ersetzt, bevor die Zeile in die Ausgabedatei kopiert wird. Diese Wirkung ist unabhängig von der Option -a.

:on
:off Die Ersetzung von Schlüsselwörtern in allen Zeilen aktivieren bzw. deaktivieren.

:ctl *Zeichen*
Das Steuerzeichen in *Zeichen* ändern.

:msg *Meldung*
Gibt die angegebene Meldung in der Diagnoseausgabe aus.

:err *Meldung*
Gibt die angegebene Meldung und anschließend

ERROR: err statement on line ...

in der Diagnoseausgabe aus. Vc terminiert die Ausführung und liefert den Rückgabecode 1.

RÜCKGABE-CODES:

0 – Fehlerfrei

1 – Fehler

SIEHE AUCH:

ed, help

# vi

## vi – Bildschirmorientierter Editor

SYNTAX:

vi *Datei* ...

BESCHREIBUNG:

Vi (visual) ist ein bildschirmorientierter Editor, der auf dem Zeileneditor ex basiert.
Im Editor vi können Sie z. B. einzelne ex-Befehle direkt eingeben und ausführen. Sie können aber auch in den ex-Modus umschalten, wenn mehrere ex-Kommandos hintereinander ausgeführt werden sollen.

SIEHE AUCH:

Systemliteratur TARGON /35: „Einführung in den Editor vi"

# wait

wait – Warten auf Beendigung aller Hintergrundprozesse

SYNTAX:

wait

BESCHREIBUNG:

Wait wartet die Beendigung aller Prozesse ab, die mit '&' im Hintergrund gestartet wurden. Ein eventueller Abbruch von Prozessen wird gemeldet.

Da der wait-Systemaufruf im Vaterprozeß ausgeführt werden muß, führt die Shell selbst den Befehl aus, ohne einen neuen Prozeß zu kreieren.

SIEHE AUCH:

sh

wc

## wc – Durchzählen von Dateiinhalten

SYNTAX:

wc [-lwc] [*Datei ...*]

BESCHREIBUNG:

Wc zählt Zeilen, Wörter und Zeichen der angegebenen Dateien. Geben Sie keine Datei an, wird die Standardeingabe gelesen.

Zeilen werden durch <CR> getrennt, Wörter durch Leerstellen, Tabs oder <CR>.

Optionen:

-l Zählen der Zeilen.

-w Zählen der Wörter.

-c Zählen der Zeichen.

Die Optionen können miteinander kombiniert werden. Geben Sie keine Option an, wird der Standard -lwc ausgeführt.

# what

what – Anzeige der Versionen von SCCS-Dateien

SYNTAX:

what *Datei* ...

BESCHREIBUNG:

What liest die Eingabedateien und sucht nach Sequenzen in der Form '@(#)', die von SCCS eingefügt wurden. Anschließend wird der Rest der Zeichenkette hinter diesem Merker ausgegeben bis zu einem Null-Zeichen, neue Zeile, Doppelpunkt oder '>'-Zeichen.

What wird normalerweise in Verbindung mit dem SCCS-Kommando get benutzt, das automatisch Identifikationsinformationen in Dateien einfügt. Sie können jedoch what auch zur Bearbeitung von Dateien verwenden, in die diese Informationen manuell eingefügt wurden.

SIEHE AUCH:

get, help
Systemliteratur TARGON: „Programmentwicklungs-Tools"

# who

## who – Wer arbeitet mit dem System?

SYNTAX:

who [-uTlpdbrtas] [*Datei*]

who am i

BESCHREIBUNG:

Who kann folgende Informationen auflisten: Benutzername, Terminalleitung, Login-Zeit, verstrichene Zeit seit der letzten Aktivität auf der Leitung und die Prozeß-ID der Shell für jeden z. Z. aktiven Systembenutzer. Um seine Informationen zu bekommen, untersucht der Befehl die Datei /etc/utmp. Geben Sie eine spezielle Datei an, wird diese untersucht. Normalerweise sollte dies /etc/wtmp sein, die ein Protokoll aller Anmeldungen seit dem Zeitpunkt enthält, an dem sie zuletzt erzeugt wurde.

Who mit dem Zusatz „am i" identifiziert den aufrufenden Benutzer.

Außer bei der Standard-Option -s sieht das allgemeine Format für die Ausgabezeilen folgendermaßen aus:

Name [Status] Leitung [Zeit] Aktivität PID [Kommentar] [Ausgang]

Mit Optionen kann who Anmeldungen, Abmeldungen, Systemstarts und Veränderungen der Systemuhr sowie andere Prozesse listen, die Ableger des init-Prozesses sind.

Optionen:

-u Diese Option listet Informationen über die Benutzer auf, die momentan im System angemeldet sind. 'Name' ist der Login-Name des Benutzers. 'Leitung' ist die Leitungsbezeichnung aus dem Verzeichnis /dev. 'Zeit' ist die seit dem Anmelden des Benutzers vergangene Zeit. 'Aktivität' enthält die Anzahl von Stunden und Minuten, die seit der <letzten Tätigkeit auf diesem Gerät vergangen ist. Ein Punkt deutet an, daß das Terminal während der letzten Minute aktiv war und daher aktuell ist. Wenn mehr als 24

## who

Stunden vergangen sind, oder die Leitung seit dem Systemstart nicht benutzt wurde, wird der Eintrag als alt markiert. Dieses Feld ist nützlich, wenn man feststellen möchte, ob jemand am Terminal arbeitet oder nicht. 'PID' ist die Prozeß-ID des Benutzerprogramms. 'Kommentar' ist das mit dieser Zeile verbundene Kommentarfeld laut /etc/inittab (s. inittab). Es kann Informationen darüber enthalten, wo das Terminal steht, die Telefonnummer der Datenendstation, den Terminaltyp usw.

-T Diese Option bewirkt, daß der Status der Terminalleitung ausgegeben wird, d. h. ob ein anderer Benutzer auf dieses Gerät schreiben kann. Ein Pluszeichen erscheint, wenn das Terminal von jedermann beschrieben werden kann; ein Minuszeichen, falls dies nicht zutrifft. Der Superuser kann auf alle Terminals schreiben, die '+' oder '-' im Statusfeld haben. Ist eine Leitung nicht in Ordnung, wird ein Fragezeichen ausgegeben.

-l Es werden nur diejenigen Leitungen (Terminals) aufgelistet, bei denen das System auf eine Anmeldung wartet. Das Feld 'Name' lautet in solchen Fällen 'LOGIN'. Die anderen Felder sind die gleichen wie bei Benutzereinträgen. Das Statusfeld wird nicht angezeigt.

-p Es wird jeder Prozeß aufgelistet, der im Moment aktiv ist und von init erzeugt wurde. Das Feld 'Name' enthält den Namen des von init ausgeführten Programms. Der Name ist in der Datei /etc/inittab enthalten und wird von dort entnommen. Die Felder 'Status', 'Leitung', und 'Aktivität' haben keine Bedeutung. Das Kommentarfeld enthält das ID-Feld (aus /etc/inittab) der Leitung (des Terminals), die diesen Prozeß erzeugte.

-d Es werden alle Prozesse aufgelistet, die beendet und von init nicht neu erzeugt wurden. Das Feld 'Ausgang' wird bei terminierten Prozessen ausgegeben und enthält die Rückgabe-Codes (wie von wait zurückgegeben). Diese Option dient zur Bestimmung der Gründe, warum ein Prozeß beendet wurde.

-b Datum und Zeit des letzten Systemstarts werden angezeigt.

## who

-r Diese Option zeigt den momentanen „run-level" des init-Prozesses an. Nach der „run-level"- und Datumsangabe kommen drei Felder, die den augenblicklichen Status, die Häufigkeit, mit der dieser Status bereits durchlaufen wurde sowie den vorherigen Status anzeigen.

-t Die letzte Änderung der Systemuhr durch den Superuser wird angezeigt (s. date).

-a Diese Option verarbeitet standardmäßig /etc/utmp oder die angegebene Datei. Alle Optionen sind dabei in Kraft.

-s Standard-Option. Sie listet nur die Felder 'Name', 'Leitung' und 'Zeit' auf.

DATEIEN:

/etc/utmp

/etc/wtmp

/etc/inittab

SIEHE AUCH:

date, login, mesg, su

# write

write – Senden von Nachrichten an einen anderen Benutzer

SYNTAX:

write *Benutzername* [*ttynr*]

BESCHREIBUNG:

Write kopiert zeilenweise die Eingaben von Ihrem Terminal auf das des angesprochenen Benutzers. Nach dem Aufruf erscheint auf dem entsprechenden Bildschirm die Benachrichtigung, daß und von wem eine Sendeleitung aufgebaut wurde. Gleichzeitig ertönt an Ihrem Terminal ein akustisches Signal, um anzuzeigen, daß Sie zum Senden bereit sind.

Der Empfänger seinerseits kann nun ebenfalls Nachrichten an Sie senden. Die Verbindung zwischen den beiden Terminals bleibt solange bestehen, bis ein Abbruchsignal oder EOF gegeben wird. Die Beendigung der Verbindung wird durch EOT (Ende der Übertragung) auf dem Terminal des anderen Benutzers angezeigt.

Wollen Sie sich mit einem Benutzer in Verbindung setzen, der an mehreren Terminals gleichzeitig angemeldet ist, können Sie durch Angabe der Terminalnummer (z. B. tty10) den Aufbau der Verbindung gezielt steuern. Geben Sie keine Terminalnummer an, wird anhand der Datei /etc/utmp überprüft, an welchem Terminal sich der Adressat zuerst angemeldet hat. Mit diesem Terminal wird die Leitung hergestellt. Anschließend erscheint an Ihrem Bildschirm eine Benachrichtigung, mit welchem Arbeitsplatz Sie in Verbindung stehen, unter Angabe der Terminalnummern, an denen sich der von Ihnen angeschriebene Benutzer noch angemeldet hat.

Die Erlaubnis, angeschrieben zu werden, kann mit dem Befehl 'mesg y' erteilt oder durch Eingabe von 'mesg n' entzogen werden. Standardwert ist die Empfangsbereitschaft des Terminals. Beim Aufruf bestimmter Kommandos, insbesondere nroff und pr, wird die Empfangsbereitschaft automatisch unterdrückt, um die Ausgabe dieser Befehle nicht zu stören. Nur der Superuser kann Nachrichten an ein sonst schreibgeschütztes Terminal senden.

## write

Geben Sie am Anfang einer Zeile ein Ausrufungszeichen (!) ein, wird der Rest der Zeile von der Shell als Kommandozeile interpretiert und ausgeführt.

Die folgende Vorgehensweise wird beim Gebrauch von write empfohlen:
Wenn Sie einen anderen Benutzer anschreiben, sollten Sie warten bis er sich ebenfalls meldet, bevor Sie anfangen zu senden. Die Beteiligten sollten jede Nachricht mit einem bestimmten Signal (z. B. o für over) beenden, so daß der Empfänger weiß, daß er nun seinerseits senden kann. Soll der Austausch von Nachrichten beendet werden, sollten Sie dies durch ein anderes Kürzel (z. B. oo für over und out) bekanntmachen.

DATEIEN:

| | |
|---|---|
| /etc/utmp | Zur Auffindung der Benutzer |
| /bin/sh | Zur Ausführung von '!' |

SIEHE AUCH:

mail, mesg, who

## xargs

xargs – Erstellen von Argumentlisten und Ausführen von Befehlen

SYNTAX:

xargs [*Optionen*] [ *Befehl* [ *Initial-Argumente*]]

BESCHREIBUNG:

Xargs kombiniert die angegebenen *Initial-Argumente* mit den von der Standardeingabe gelesenen Argumenten und führt den genannten *Befehl* einmal oder mehrmals aus. Die spezifizierten Optionen bestimmen dabei die Anzahl Argumente, die bei jedem Aufruf des Befehls gelesen werden sowie die Art und Weise, wie sie miteinander kombiniert werden.

Der Befehl kann eine Shell-Datei sein; das Programm sucht ihn mit Hilfe von $PATH.

Geben Sie keinen Befehl an, wird /bin/echo eingesetzt.

Die von der Standardeingabe eingelesenen Argumente sind zusammenhängende Zeichenketten, die durch ein oder mehrere Leerzeichen, Tabulatoren oder Zeilenende-Zeichen voneinander getrennt werden. Leere Zeilen werden gelöscht.
Leerzeichen oder Tabulatoren können als Teil eines Arguments angegeben werden, wenn dieses Argument in Hochkommata eingeschlossen wird. Geben Sie einzelne Zeichen in Hochkommata an, werden diese „wörtlich" genommen; die begrenzenden Hochkommata werden entfernt.
Außerhalb einer in Hochkommata eingeschlossenen Zeichenkette dient der Backslash (\) dazu, das nächste Zeichen außer Kraft zu setzen.

Jede Argumentliste beginnt mit den „Initial-Argumenten", gefolgt von den Argumenten, die das Programm von der Standardeingabe liest (Ausnahme: die Option -i).

Die Optionen -i, -I und -n regeln, wie die Argumente für den Befehlsaufruf ausgewählt werden. Geben Sie keine dieser Optionen an, folgen auf die „Initial-Argumente" die von der Standardeingabe fortlaufend gele-

## xargs

senen Argumente, bis die Speicherkapazität des internen Puffers erschöpft ist. Dann führt das Programm den angegebenen Befehl mit allen Argumenten aus. Dieser Vorgang wird solange wiederholt, bis keine Argumente mehr vorhanden sind. Geben Sie Optionen an, die einander ausschließen – z. B. -l und -n –, hat die letzte Option Vorrang.

-l[*Anzahl*] Der Befehl wird für die angegebene Anzahl nicht-leerer Argumentzeilen, die aus der Standardeingabe gelesen werden, ausgeführt.
Das Ende einer Zeile erkennt das Programm an dem ersten Zeilenende-Zeichen, falls das letzte Zeichen kein Leerzeichen oder ein Tabulator ist; dieses signalisiert die Fortsetzung in der nächsten beschriebenen Zeile.
Geben Sie *Anzahl* nicht an, wird standardmäßig 1 ausgeführt. Die Option -x tritt in Kraft.

-i[*Einfügung*]
Einfügen. Der angegebene Befehl wird für jede Zeile der Standardeingabe ausgeführt, wobei eine Zeile einem einzigen Argument entspricht, das jedesmal, wenn *Einfügung* vorkommt, in die Initial-Argumente eingefügt wird. Maximal 5 Argumente in den Initial-Argumenten dürfen ein oder mehrere *Einfügungen* enthalten.
Leerzeichen und Tabulatoren am Anfang einer Zeile werden gelöscht.
Die erstellten Argumente dürfen maximal 255 Zeichen lang sein; die Option -x tritt in Kraft. Geben Sie *Einfügung* nicht an, wird {} als Standard angenommen.

-n[*Anzahl*] Dieses Argument bewirkt, daß bei der Befehlsausführung so viele Argumente wie möglich – maximal bis zu der angegebenen *Anzahl* – von der Standardeingabe angewendet werden. Weniger Argumente werden dann verwendet, wenn ihre Gesamtgröße die Zeichengröße (s. Option -s*Größe*) übersteigt und wenn beim letzten Aufruf des Befehls weniger Argumente übrigbleiben als in *Anzahl* angegeben ist.

Verwenden Sie zusätzlich die Option -x, muß jede Anzahl von Argumenten in die festgelegte Größenordnung passen, anderenfalls wird die Ausführung von xargs beendet.

## xargs

-t Trace-Modus: Der Befehl und jede erstellte Argumentliste werden unmittelbar vor der Ausführung in die Standard-Fehlerausgabe geschrieben.

-p Interaktiver Modus. Sie werden gefragt, ob der Befehl bei jedem Aufruf ausgeführt werden soll.
Der Trace-Modus -t wird eingeschaltet, und der Befehl, der ausgeführt werden soll, wird – mit einem Fragezeichen versehen – angezeigt.
Antworten Sie mit 'y' (yes), wird der Befehl ausgeführt. Eine beliebige andere Eingabe oder Auslösen der CR-Taste bewirkt, daß der Befehl übergangen wird.

-x Diese Option führt zu einem Programmabbruch, wenn eine der Argumentlisten die Zeichengröße (s. Option -s*Größe*) übersteigt. Diese Option wird durch -i und -l aufgerufen. Verwenden Sie keine der Optionen -i, -l oder -n, muß die Gesamtlänge aller Argumente innerhalb der angegebenen Zeichengröße liegen.

-s*Größe* Die Maximalgröße jeder Argumentliste ist abhängig von der in dieser Option angegebenen Anzahl Zeichen. *Größe* muß eine positive Zahl <= 470 sein. Verwenden Sie diese Option nicht, wird ein Größe von 470 Zeichen angenommen.

Sie müssen beachten, daß im Zähler von *Größe* ein zusätzliches Zeichen für jede Argumentliste und die Anzahl Zeichen des Befehlsnamens enthalten ist.

-e[*Dateiende*]
*Dateiende* ist die Zeichenkette, die das logische Dateiende (end of file) kennzeichnet.
Geben Sie -e nicht an, wird der Unterstrich (_) als Dateiende-Kennung angenommen. Verwenden Sie -e ohne *Dateiende*, verliert der Unterstrich diese Bedeutung und wird als Unterstreichung interpretiert.
Xargs liest solange von der Standardeingabe, bis entweder das Dateiende erreicht ist oder eine Kennung für das logische Dateiende angetroffen wird.

## xargs

Das Programm wird beendet, wenn es entweder eine Rücksprunganweisung (-1) erhält oder der Befehl nicht ausgeführt werden kann. Ist der Befehl eine Shell-Prozedur, sollte diese mit einem zugeordneten Wert beendet werden (s. sh), um einen zufälligen Rückgabe-Code von -1 zu vermeiden.

Beispiele:

1. Im folgenden Beispiel werden alle Dateien mit Hilfe des move-Befehls vom Verzeichnis $1 in das Verzeichnis $2 verschoben. Dabei wird vor der Ausführung jedes move eine Meldung ausgegeben:

   ls $1 | xargs -i -t mv $1/{} $2/{}

2. Im folgenden Beispiel erfolgt die Ausgabe der eingeklammerten Befehle in einer einzigen Zeile, die dann an das Ende der Datei mit Namen log angehängt wird:

   (logname; date; echo $0 $*) | xargs >> log

3. In diesem Beispiel wird der Benutzer gefragt, welche der im aktuellen Verzeichnis enthaltenen Dateien in der Datei mit Namen Archiv archiviert werden sollen:

   - Die Dateien werden nacheinander kopiert:

     ls | xargs -p -l ar r Archiv

   - Mehrere Dateien werden gleichzeitig kopiert:

     ls | xargs -p -l xargs ar r Archiv

4. Im nachstehenden Beispiel wird der Befehl diff mit nacheinander eingegebenen Argumentenpaaren ausgeführt, die wie Shell-Argumente angegeben sind:

   echo $* | xargs -n2 diff

SIEHE AUCH:

sh

# yacc

yacc – Programm für Syntaxanalyse (yet another compiler-compiler)

SYNTAX:

yacc [-vdlt] *Grammatik*

BESCHREIBUNG:

Yacc konvertiert eine kontextfreie Grammatik in eine Gruppe von Tabellen für einen einfachen Automaten, der einen LR(1) Syntaxanalyse-Algorithmus ausführt. Die Grammatik darf mehrdeutig sein. Mehrdeutigkeiten werden durch angegebene Prioritätsregeln aufgehoben.

Die Ausgabedatei y.tab.c muß vom C-Compiler kompiliert werden, so daß das Programm yyparse entsteht. Dieses Programm muß zusammen mit dem lexikalischen Analyseprogramm yylex sowie main und yyerror (einer Fehlerbehandlungsroutine) geladen werden. Diese Routinen müssen von Ihnen bereitgestellt werden. Lex bietet sich für die Erstellung lexikalischer Analyseprogramme an, die von yacc verarbeitet werden können.

Geben Sie die Option -v an, wird die Datei y.output erstellt, die eine Beschreibung der Syntaxanalyse-Tabellen und eine Liste der durch Mehrdeutigkeiten in der Grammatik hervorgerufenen Konflikte enthält.

Bei Angabe von -d, wird die Datei y.tab.h mit den #define-Anweisungen erzeugt, die die von yacc zugeordneten „Token-Codes" den vom Benutzer deklarierten „Token-Namen" zuordnen. Auf diese Weise ist es möglich, aus Quelldateien, bei denen es sich nicht um y.tab.c handelt, „Token-Codes" anzusprechen.

Wenn die Option -l angegeben ist, enthält der von y.tab.c erzeugte Code keine #line-Konstrukte. Dies ist nur ratsam, nachdem die Grammatik und die zugehörigen Optionen vollständig ausgetestet wurden.

Programmcode für das Debugging zur Laufzeit wird immer in y.tab.c erzeugt. Die Kompilierung dieses Programmcodes kann von Bedingungen abhängig gemacht werden. Standardmäßig gilt, daß dieser Code beim Kompilieren von y.tab.c nicht mit verarbeitet wird. Geben Sie jedoch die Option -t an, wird der Debugging-Code standardmäßig kom-

yacc

piliert. Gleichgültig ob die Option -t angegeben wurde oder nicht, wird der Code für das Laufzeit-Debugging von dem Preprozessor-Symbol YYDEBUG gesteuert. Hat YYDEBUG einen Wert ungleich Null, wird der Debugging-Code eingebunden. Hat YYDEBUG den Wert Null, so wird der Code nicht mit eingebunden. Das Programm ohne Laufzeit-Debugging-Code ist kleiner und läuft etwas schneller.

MELDUNGEN:

Die Anzahl der reduce-reduce- und shift-reduce-Konflikte wird auf der Standardfehlerausgabe gemeldet; eine ausführlichere Auswertung steht in der Datei y.output. Sind einige Regeln vom Startsymbol aus nicht erreichbar, wird dies ebenfalls gemeldet.

HINWEIS:

Da die Dateinamen fest vorgegeben sind, kann in einem Verzeichnis immer nur ein yacc-Prozeß aktiv sein.

SIEHE AUCH:

lex
Systemliteratur TARGON: „Textmustererkennung und Verarbeitung"