robotron

Programmtechnische Beschreibung
für Bürocomputer A 5120.16

Systemhandbuch MUTOS 8000 TEIL I

Abschnitt 1 - Kommandos

VEB Robotron Buchungsmaschinenwerk
Karl-Marx-Stadt
Stand: 12/85

INTRO(1)

NAME

intro - Einfuehrung

BESCHREIBUNG

Diesers Abschnitt des Systemhandbuchs Teil I beschreibt die allgemein zur Verfuegung stehenden Kommandos. Sie sind in alphabetischer Reihenfolge beschrieben und werden unterschieden in:

(1) Allgemeine Kommandos
(1M) Kommandos fuer die Systempflege und -wartung

DIAGNOSTIK

Von jedem Kommando werden zwei Statusbytes zurueckgegeben.
Das erste wird vom System erzeugt. Es gibt die Ursache des Kommandoabbruchs an. Bei normaler Beendigung ist es Null. Das zweite niederwertige ist ein Rueckkehrwert bei normalem Programmende. Es wird von exit(2) erzeugt. Bei erfolgreicher Ausfuehrung ist es gewoehnlich ebenfalls Null.

exit(2) erzeugt. Bei erfolgreicher Ausfuehrung ist es gewoehnlich ebenfalls NULL.

Ein Wert ungleich 0 zeigt in der Regel Stoerungen an, wie z.B. fehlerhafte Parameter, fehlerhafte oder fehlende Daten und andere Defekte.
Der Rueckkehrcode wird nur dort beschrieben, wo spezielle Endebedingungen moeglich sind.

In diesem Zusammenhang muss erwaehnt werden, dass Kommandos auch selbsterklaerendg Fehlertexte fuer den Bediener ausgeben.

---

*

NAME

adb - Testprogramm

UEBERSICHT

adb [-w][-m][objfil [corfil]]

BESCHREIBUNG

Adb ist ein vielseitig verwendbares Testprogramm. Sein Hauptanwendungsgebiet ist die Programmabarbeitung in einer Testumgebung sowie das Auswerten von Core-Files (Speicherabzug, der von MUTOS 8000 bei Programmabstuerzen erzeugt wird).
Objfil ist normalerweise ein File, welches das ausfuehrbares Programm, moeglichst mit Symboltabelle, enthaelt. Ist keine Symboltabelle vorhanden, koennen die Moeglichkeiten der symbolischen Arbeit von adb nicht genutzt werden. Standardannahme fuer objfil ist **a.out**. Corfil muss ein Core-File sein, das beim Abarbeiten von objfil entstanden ist. Standard fuer corfil ist **core**.

Adb erwartet Kommandos ueber Standard-Input und gibt ueber Standard-Output aus. Ist das **-w** Flag gesetzt, sind objfil und corfil zum Lesen und Schreiben geoeffnet und werden, falls nicht vorhanden, angelegt. Soll ein beliebiges File mit adb betrachtet oder veraendert werden, ohne dass der Fileheader speziell behandelt wird, so muss beim Aufruf von abd das **-m** Flag angegeben werden. Dadurch werden die Auswertungen des Fileheaders ausgeschaltet, keine gueltige Map erzeugt und entsprechende Variable nicht berechnet. Unter diesen Umstaenden sind nur bestimmte adb-Kommandos sinnvoll. Adb ignoriert QUIT; INTERRUPT fuehrt zum Abbruch des gerade ausgefuehrten und zum Aufruf des naechsten Kommandos.

Das allgemeine Format von adb - Kommandos ist:

[address] [, count] [command] [;]

Ist address (Adresse) vorhanden, so wird Punkt (.) auf diesen Wert eingestellt. Der Anfangswert fuer Punkt ist 0. Fuer die meisten Kommandos gibt count (Anzahl) die Anzahl der Wiederholungen fuer dieses Kommando an; Standardwert ist 1. Address und count koennen Ausdruecke sein.

Wie eine Adresse interpretiert wird, haengt vom Kontext ab, in dem diese Adresse verwendet wird. Wird ein Programm unter adb (als Unterprozess) abgearbeitet, werden Adressen normalerweise als Adressen im Adressraum

des Unterprozesses interpretiert. Detaillierte Ausfuehrungen zur Adresszuordnung siehe unter ADRESSEN.

AUSDRUECKE

`.` Der Wert von Punkt.

`+` Der Wert von Punkt erhoeht um die aktuelle Schrittweite.

`^` Der Wert von Punkt verringert um die aktuelle Schrittweite.

`"` Der zuletzt eingegebene Wert fuer address.

integer
: Eine Oktalzahl, wenn integer mit 0 beginnt; eine Hexadezimalzahl, wenn 0x vorangestellt wird; sonst eine Dezimalzahl.

integer.fraction
: Eine 32-Bit Gleitkommazahl.

'cccc' Der ISO-Wert von bis zu 4 Zeichen. \ wird verwendet, um ' zu kennzeichnen.

(name Der Wert von name, der entweder Register- oder Variablenname ist. Adb verwendet und verwaltet eine Menge von Variablen (siehe VARIABLEN), die durch einzelne Buchstaben oder Ziffern bezeichnet werden. Ist name ein Registername, wird der Wert des Registers dem System-Header von corfil entnommen . Die Registernamen sind **r0 ... r5 sp pc ps**.

symbol Ein symbol ist eine Folge von Gross- und Kleinbuchstaben, Unterstreichungszeichen oder Ziffern, die nicht mit einer Ziffer beginnt. Der Wert von symbol wird der Symboltabelle von objfil entnommen. Wenn noetig wird symbol _ oder _ vorangestellt.

_symbol
: In C beginnt der tatsaechliche Name (wie er im Assemblerprogramm auftritt) einer externen Variablen mit _. Dies ist notwendig, um die Namen der externen Variablen von denen der internen und Hilfsvariablen unterscheiden zu koennen.

routine.name
: Die Adresse der angegebenen Variablen name in der genannten C-Routine. Beides: routine und name sind vom Typ symbol. Ist name nicht angegeben, so erfolgt der Bezug auf die zuletzt aktivierte Routine (entsprechend Stack-Inhalt).

(exp) Der Wert des Ausdrucks exp.

Monadische Operatoren

*exp Der Inhalt des Speicherplatzes, der durch exp in corfil adressiert wird.

@exp Der Inhalt des Speicherplatzes, der durch exp in objfil adressiert wird.

-exp Ganzzahlige Negation.

~exp Bitweises Komplement (Einerkomplement).

Zweistellige Operatoren werden von links nach rechts abgearbeitet und haben eine geringere Prioritaet als monadische 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 e1 wird zum naechsten Vielfachen von e2 aufgerundet.

KOMMANDOS

Ein Kommando besteht aus einem Verb dem ein Modifikator oder eine Liste von Modifikatoren folgt. Folgende Verben stehen zur Verfuegung (Den Kommandos '?' und '/' kann '*' folgen; siehe dazu ADRESSEN):

?f Ab Speicherplatz address in objfil wird im durch f angegebenen Format ausgegeben.

/f Ab Speicherplatz address in corfil wird im durch f angegebenen Format ausgegeben.

=f Der Wert von address selbst wird im durch f angegebenen Format ausgegeben (Beim Format i wird fuer die Teile des Befehls, die sich auf nachfolgende Worte beziehen, '?' ausgegeben).

Die Formatangabe format besteht aus einem oder mehreren Zeichen, die die Form der Ausgabe bestimmen. Jedem Formatzeichen kann eine ganze Dezimalzahl vorangestellt werden, die die Anzahl der Wiederholungen fuer das Formatkennzeichen angibt. Bei der Ausgabe entsprechend dem angegebenen Format wird Punkt jeweils um die Formatlaenge

erhoeht. Ist kein Format angegeben wird entsprechend dem zuletzt verwendeten ausgegeben. Es gibt folgende Formatkennzeichen (dahinter ist die jeweilige Formatlaenge angegeben):

o 2 Ausgabe von zwei Bytes als Oktalzahl. Alle Oktalzahlen sind bei adb durch eine vorangestellte 0 gekennzeichnet.
O 4 Ausgabe von 4 Byte als Oktalzahl.
q 2 Ausgabe als vorzeichenbehaftete Hexadezimalzahl
Q 4 Ausgabe als lange vorzeichenbehaftete Hexadezimalzahl
d 2 Ausgabe als vorzeichenbehaftete Dezimalzahl.
D 4 Ausgabe als lange vorzeichenbehaftete Dezimalzahl.
x 2 Ausgabe von 2 Bytes als Hexadezimalzahl.
X 4 Ausgabe von 4 Bytes als Hexadezimalzahl.
u 2 Ausgabe als vorzeichenlose Dezimalzahl.
U 4 Ausgabe als lange vorzeichenlose Dezimalzahl.
f 4 Ausgabe der 32 Bit als Gleitkommazahl.
F 8 Ausgabe als doppelt genaue Gleitkommazahl.
b 1 Ausgabe des adressierten Bytes als Oktalzahl.
c 1 Ausgabe des adressierten Bytes als ASCII-Zeichen.
C 1 Ausgabe des adressierten Bytes als ISO-Zeichen unter Benutzung folgender Schreibweisen fuer nicht-graphische Zeichen: Die Werte 0x00 bis 0x20 werden als @, dem das dazugehoerige Zeichen im Bereich 0x40 bis 0x60 folgt, ausgegeben. Das Zeichen @ wird als @@ ausgegeben.
s n Ausgabe der adressierten Zeichenkette, bis ein Zeichen mit dem Wert Null erreicht wird.
S n Ausgabe einer Zeichenkette unter Benutzung der oben angegebenen Regeln fuer nicht-graphische Zeichen. n hat als Wert die Laenge der Zeichenkette, einschliesslich Null.
Y 4 Ausgabe von 4 Byte in Datumformat ctime (3).
i n Ausgabe als PLZ/ASM-Befehl. n ist dabei die Anzahl von Bytes, die der Befehl belegt. Diese Art der Ausgabe benutzt die Variablen 1 und 2, in denen die Verschiebung von Quell- bzw. Zieloperand (bei Adressrechnung) des jeweils zuletzt ausgegebenen Befehls gespeichert wird.
a 0 Ausgabe von Punkt in symbolischer Form. Dabei wird gesichert, dass die Symbole in den nachfolgenden Anweisungen immer folgenden Typ haben:

/ lokales oder globales Datensymbol
? lokales oder globales Textsymbol
= lokales oder globales absolutes Symbol.

p 2 Ausgabe des adressierten Wertes in symbolischer Form unter Verwendung derselben Regeln wie unter **a**.

t 0 Ist eine ganze Zahl vorangestellt, so erfolgt ein Tabulatorsprung zur entsprechenden Tabulatormarke. Zum Beispiel bewirkt **8t** einen Tabulatorsprung zur naechsten 8-Leerzeichen-Tabulatormarke.

r 0 Ausgabe eines Leerzeichens.

n 0 Ausgabe eines Zeilenwechsels.

"..." 0
     Ausgabe der eingeklammerten Zeichenkette.

^   Punkt wird um die aktuelle Schrittweite verringert. Nichts wird ausgegeben.

+   Punkt wird um 1 erhoeht. Keine Ausgabe.

-    Punkt wird um 1 verringert. Keine Ausgabe.

neue Zeile
Hat das vorangegangene Kommando Punkt temporaer veraendert, wird dieser Wert Punkt permanent zugewiesen. Das vorangegangene Kommando wird mit count gleich 1 wiederholt.

[?/]lvalue mask
Alle Worte ab Punkt werden mit mask maskiert und mit value verglichen, bis es zu einer Uebereinstimmung kommt.
Wird L verwendet, werden immer 4 Byte zusammen untersucht. Kann keine Uebereinstimmung festgestellt werden, bleibt Punkt unveraendert; sonst wird Punkt auf den Wert der Adresse gestellt, bei der Uebereinstimmung festgestellt wurde. Wird mask weggelassen, ist -1 Standard.

[?/]wvalue ...
Schreiben von value (2 Byte) in den angegebenen Speicherplatz. Bei W Schreiben von 4 Byte. Ein Schreiben in den Adressraum eines Unterprozesses (beim dynamischen Test) ist nur ab geraden Aderessen (Wortgrenzen) moeglich.

[?/]mb1 e1 f1[?/]
Den Map-Variablen (b1, e1, f1) werden neue Werte zugeordnet. Sind weniger als drei Ausdruecke angegeben, so bleiben die restlichen Werte unveraendert. Folgt '?' oder '/' ein '*', dann wird im zweiten Segment der Map geaendert. Wird die Liste durch '?' oder '/' abgeschlossen, kann in weiteren Kommandos auch durch '/' auf objfil oder durch '?' auf corfil zugegriffen werden. (So bewirkt zum Beispiel '/m?', dass '/' sich auf objfil bezieht.)

>name
Punkt wird der angegebenen Variablen bzw. dem angegebenen Register als Wert zugewiesen.

! Der Rest der Zeile, der '!' folgt, wird von Shell als Kommando abgearbeitet.

*modifier

Weitere Kommandos. Fuer modifier koennen folgende Kommandos angegeben werden:

<f	Liest die naechsten Kommandos vom File f.
>f	Gibt die Ausgabe auf File f aus und legt dieses File an, wenn es noch nicht existiert.
r,R	Ausgabe der Inhalte der Register r0,r1, ... ,sp,pc,ps und des durch pc adressierten Befehls. Punkt erhaelt den Wert von pc.
f	Ausgabe der Gleitkommaregisterinhalte in einfacher oder doppelter Genauigkeit (je nach Stand des Prozessorstatusworts).
b	Ausgabe aller Unterbrechungspunkte und des
c	C-Stack-Backtrace. Ist address angegeben, gilt diese Adresse im weiteren als die Adresse des gueltigen Stackinhaltes (anstelle von r5). Wird C benutzt, werden auch die Namen und Werte aller automatic und static Variablen fuer jede aktive Funktion ausgegeben. Ist count angegeben, werden diese Informationen nur fuer die count ersten (zuletzt aufgerufenen) Funktionen ausgegeben.
e	Die Namen und Werte aller externen Variablen werden ausgegeben.
w	Setzen der Seitenbreite fuer die Ausgabe auf den Wert von address (Standard ist 80).
s	Setzen der Hoechstgrenze fuer das Verwenden eines Symbols zum Bezeichnung von Adressen als dieses Symbol plus Verschiebung (Standard ist 255).
o	Alle Dezimalzahlen bei der Eingabe werden im weiteren als Oktalzahlen gewertet.
x	Alle Dezimalzahlen und alle mit einer Dezimalziffer beginnenden Worte (ausser Oktalzahlen) werden im weiteren auch ohne fuehrendes 0 oder 0x als Hexadezimalzahlen gewertet. Die Wirkung von o wird ausgeschaltet.
d	Die Wirkung der modifier o und x wird aufgehoben, Dezimalzahlen erhalten ihre urspruengliche Bedeutung.
q,Q	Verlassen von adb.
v	Ausgabe aller Variablen mit Werten ungleich Null (oktal).
m	Ausgabe aller Adress-Maps.

:modifier

Arbeit mit Unterprozessen (dynamischer Test); moegliche Modifikatoren sind:

bc Setzen eines Unterbrechungspunktes auf adresss. Der Unterbrechungspunkt wird count -1 mal uebergangen, ehe es zum Programmstop kommt. Bei jedem Passieren des Unterbrechungspunktes wird das Kommando c ausgefuehrt. Setzt dieses Kommando Punkt auf Null, kommt es immer zum Programmstop (.=0 ist Standardkommando).

d Loeschen eines Unterbrechungspunktes.

r Abarbeiten von obifil als Unterprozess. Ist address explizit angegeben, wird die Programmabarbeitung an dieser Stelle begonnen; sonst wird das Programm vom Standardeintrittspunkt an abgearbeitet. Count gibt an, wieviele Unterbrechungspunkte vor dem ersten Programmstop uebergangen werden sollen. Argumente des zu startenden Programms koennen in derselben Zeile mit angegeben werden. Beginnt ein Argument mit < oder >, wird fuer den Unterprozess Standard-Input oder Standard-Output festgelegt. Alle Signale werden an den Unterprozess weitergegeben

cs Der Unterprozess wird mit dem Signal s fortgesetzt. Siehe auch signal (2). Ist address angegeben, erfolgt die Fortsetzung des Unterprozesses ab dieser Adresse. Wird kein Signal angegeben, wird das Signal gesendet, das den Prozess unterbrochen hat. Die Unterbrechungspunktbehandlung erfolgt wie bei r.

ss Es gilt dasselbe wie bei **c** ausser, dass im Einzelschrittbetrieb bzw. in Schritten von count Befehle abgearbeitet werden. Ist kein Unterprozess vorhanden, wird wie bei **r** obifil als solcher abgearbeitet. In diesem Fall koennen keine Signale uebergeben werden; der Rest der Zeile kann aber Argumente fuer den Unterprozess enthalten.

k Der aktuelle Unterprozess wird, wenn vorhanden, beendet.

VARIABLEN

Adb stellt eine Reihe von Variablen zur Verfuegung. Alle Variablen werden am Anfang initialisiert, obwohl nicht alle im weiteren von adb selbst benoetigt werden. Folgende Variablen sind fuer die Kommunikation wie folgt reserviert:

0 Der letzte Wert, der ausgegeben wurde.
1 Der letzte Offset (Verschiebung) in einem Befehl.
2 Der vorangegangene Wert von Variable 1.

Zu Beginn werden die folgenden Variablen entsprechend dem Systen-Header von corfil gesetzt. Ist corfil kein Core-File werden diese Werte objfil entnommen. Wurde adb mit -m aufgerufen, so werden die folgenden Variablen nicht belegt:

b Die Basisadresse des Datensegments.
d Die Laenge des Datensegments.
e Der Eintrittspunkt.
m Die Magic-Number (0xE807).
s Die Groesse des Stack-Segments.
t Die Groesse des Textsegments.

ADRESSEN

Die Berechnung der Adresse im File aus der Adress im Kommando geschieht unter Verwendung der Map (Speicherzuordnung) fuer dieses File. Jede Map besteht aus zwei Tripeln (b1, e1, f1) und (b2, e2, f2). Das Berechnen der Fileadresse aus einer angegebenen Adresse geschieht folgendermassen:

b1<=address<e1 ====> file address=address+f1-b1 oder

b2<=address<e2 ====> file address=address+f2-b2

, andernfalls ist address nicht zulaessig. Folgt nach ? oder / ein *, wird nur das zweite Tripel benutzt.

Das Setzen der beiden Maps zu Beginn der Arbeit von adb mit den Werten von **a.out** und **core** ist im Normalfall wuenschenswert. Ist aber ein File nicht vom erwarteten Typ, so wird b1 0, e1 auf die maximale Fileaenge und f1 ebenfalls 0 gesetzt. Auf diese Art und Weise kann auf den gesamten Fileinhalt ohne Adressumrechnung zugegriffen werden.

Damit adb auch mit sehr grossen Files arbeiten kann, werden alle Werte als ganzzahlige 32-Bit-Zahlen mit Vorzeichen behandelt.

FILES

/dev/mem
/dev/swap
a.out
core

SIEHE AUCH

ptrace(2), a.out(5), core(5)

FEHLERBEHANDLUNG

'adb' wird ausgegeben, wenn ein aktuelles Format oder Kommando erwartet wurde, aber nicht angegeben ist. Ausserdem erfolgen Meldungen ueber nicht zugreifbare Files, Syntaxfehler, anormales Ende von Kommandos usw. Exit-Status ist immer 0.

FEHLERQUELLEN

Das Setzen eines Unterbrechungspunktes auf den Anfang eines Programms sollte vermieden werden.
Im Einzelschrittbetrieb zaehlen Systemrufe nicht als Format ausgefuehrte Befehle.
Lokale Variablen, die denselben Namen wie eine externe Variable haben, erschweren den Zugriff zu externen Variablen.

NAME

ar - Archiv- bzw. Bibliothekspflege

UEBERSICHT

ar key [posname] afile name ...

BESCHREIBUNG

Ar behandelt Gruppen von Files, die zu einem Archivfile zusammengefasst sind. Das Kommando wird hauptsaechlich zum Anlegen und Aktualisieren von Bibliotheken verwendet, die vom Lader genutzt werden. Es kann natuerlich auch fuer andere, aehnliche Aufgaben verwendet werden.

Key ist ein Zeichen aus der Menge **drqtpmx**, moeglicherweise in Verbindung mit einem oder mehreren Zeichen aus der Menge **vuaibcl**. Afile ist das Archivfile. Name bezeichnet im Archivfile enthaltene Files.
Die Schluesselzeichen key haben folgende Bedeutung:

d Loeschen des genannten Files aus dem Archivfile.

r Ersetzen des genannten Files im Archivfile. Bei Verwendung von **r** zusammen mit **u** werden nur die Files ersetzt, deren Daten nach dem Anlegen des Archivfiles noch modifiziert wurden. Das Benutzen des Positionszeichens **abi** verlangt die Angabe des posname-Arguments.

Es gibt an, dass neue Files hinter (a) oder vor (b oder i) posname zu plazieren sind. Anderenfalls werden die neuen Files am Ende angefuegt.

q Rasches Anfuegen der genannten Files an das Ende des Archivfiles. Eventuelle Positionszeichen sind ungueltig. Das Kommando prueft nicht, ob sich die anzufuegenden Files schon im Archivfile befinden. Allerdings ist es sinnvoll, diesen Mehraufwand zu vermeiden, indem man ein Archiv schrittweise erzeugt.

t Ausgeben des Inhaltsverzeichnisses eines Archivfiles. Sind keine namen genannt, werden alle Files des Archivs aufgelistet; sind namen angegeben, werden nur diese Files aufgelistet.

p Der Inhalt der genannten Files wird ausgegeben (Standardausgabe).

m Umsortieren der genannten Files entsprechend dem posname Argument (siehe key r). Ist kein Positionsparameter angegeben, werden die Files an das Archivende gebracht.

x Kopieren des genannten Files ins Directory. Falls keine namen angegeben sind, werden alle Files kopiert. (Sie stehen dann einzeln im aktuellen Directory und koennen dort beispielsweise korrigiert werden). Das Archivfile wird durch x nicht veraendert.

v Diese Option wird nur in Verbindung mit Schluesselparametern verwendet. Sie gibt noch einmal explizit aus, welche Files in welcher Form von der aktuellen Archivoption betroffen sind (z.B. d - file1, falls file1 geloescht wurde). Wird sie zusammen mit t verwendet, liefert sie eine ausfuehrliche Liste saemtlicher Fileinformationen, wie Erstellungsdatum, Zugriffsrecht u.s.w. . Wird sie zusammen mit p verwendet, wird bei der Ausgabe jedem File sein Name vorangestellt.

c Normalerweise eroeffnet ar selbst ein Archivfile, wenn es noch nicht vorhanden ist und benoetigt wird. Darueber erfolgt eine Information (creating afile). Bei Verwendung von c wird diese Nachricht unterdrueckt.

l Normalerweise bringt ar seine temporaeren Files in das Directory /tmp. Diese Option ermoeglicht es, sie in das lokale Directory zu bringen.

FILES

/tmp/v* temporaere Files

SIEHE AUCH

ld(1), ar(5)

FEHLERQUELLEN

Tritt das gleiche File mehrfach in der Argumentliste auf kann es passieren, dass es mehrfach in das Archivfile gebracht wird.

AS(1) **AS(1)**

NAME

as - PLZ/ASM-Assembler

UEBERSICHT

as [-][-u][-o objektfile] file

BESCHREIBUNG

AS assembliert das genannte File. Wird '-' oder '-u' als Option verwendet, werden alle undefinierten Symbole als EXTERNAL vereinbart.
Wird nicht mittels der Option '-o' ein Objektfilename vereinbart, wird asm.out verwendet.
Sind keine externen Referenzen vorhanden, ist dieses File ausfuehrbar.

FILES

/usr/bin/as	Assembler
asm.out	Objekt-File

SIEHE AUCH

ld(1),nm(1),adb(1),a.out(5)

FEHLERBEHANDLUNG

Beim Auftreten von Fehlern wird ein Objektmodul der Laenge Null angelegt.
Fehlerausschriften im Langformat erfolgen ueber "stderr mit Angabe der Zeilennummer im Assemblerprogramm.
Zeilennummern bei Fehlerausschriften koennen um ± 1 abweichen.

NAME

cat - Verketten und Ausgeben von Files

UEBERSICHT

cat [-u] file ...

BESCHREIBUNG

Cat liest die Files file ... in der angegebenen Reihenfolge und gibt deren Inhalt nach Standard-Output aus. Somit bewirkt

'cat file1'

die Ausgabe von file1 auf Standard-Output und

'cat file1 file2 >file3'

fuegt dem Inhalt von file1 den von file2 an und legt das Ergebnis in file3 ab.

Wurde bei Kommandoaufruf kein file oder als Argument nur '-' angegeben, so liest cat vom Standard-Input.
Die Ausgaben durch cat erfolgen gepuffert in 512-Byte-Bloecken. Dies gilt jedoch nur dann, wenn der Standard-Output nicht einem Terminal zugeordnet oder die Option -u angegeben ist.

SIEHE AUCH

pr(1), cp(1)

FEHLERQUELLEN

Kommandos der Art 'cat a b >a' und 'cat a b >b' sollten vermieden werden. Sie fuehren zum Zerstoeren von Eingabefiles, bevor diese gelesen werden.

NAME

cc - C-Compiler

UEBERSICHT

cc [option] ... file ...

BESCHREIBUNG

CC ist der MUTOS 8000-C-Compiler. Er akzeptiert verschiedene Typen von Argumenten:

Files, deren Namen mit '.c' enden, werden als C-Quellprogramme betrachtet. Falls nicht durch eine der unten angegebenen Optionen anderweitig festgelegt, werden sie standardmaessig compiliert, vom Assembler as (1) uebersetzt und durch den Lader ld (1) zu einem ausfuehrbaren Programm **a.out** geladen.

Von Files, deren Namen auf '.s' enden, wird angenommen, dass es Assembler-Quellprogramme sind, die von as (1) uebersetzt und in der oben beschriebenen Weise weiterverarbeitet werden.

Von anderen Argumenten und anderen als den unten beschriebenen Optionen wird vorausgesetzt, dass es sich um solche des Laders ld (1) handelt bzw. dass es einzelne C-kompatible Objektprogramme oder Bibliotheken C-kompatibler Routinen sind, die in frueheren cc- bzw. as-Laeufen erzeugt wurden. Die Standardbibliothek "/usr/lib/libc.a" wird durch cc automatisch fuer ld (1) aufgerufen.

Die Komponenten werden zusammen mit den Ergebnissen der im cc-Kommandoaufruf spezifizierten Compilationen in der angegebenen Reihenfolge zu einem ausfuehrbaren Programm **a.out** geladen.

Das Filesystem (Diskette), in dem die Komponenten zur Programmentwicklung gespeichert sind, ist unter dem Namen '/usr' einzugliedern.

Die Optionen sind:

-c Der Aufruf des Laders wird unterdrueckt. Es wird ein Objektfile erzeugt.

-O Der Objekt-Code-Optimierer wird bei der Compilation aufgerufen.

-S Das angegebene C-Programm wird compiliert und die Assembler-Sprach- Ausgabe wird einem entsprechenden File mit dem Suffix '.s' uebergeben.

-P Nur der Macro-Praeprozessor wird abgearbeitet Das Ergebnis befindet sich in einem File, das anstatt mit '.c' mit '.i' endet. In diesem File gibt es keine '#'-Zeilen.

-E Nur der Macro-Praeprozessor wird abgearbeitet. Das Ergebnis wird der Standardausgabe uebergeben. Diese Ausgabe ist fuer Compiler-Pruefungen vorgesehen und wird durch cc nicht als Eingabe akzeptiert.

-o output
Das Ausgabefile wird output genannt. Ist das Argument output nicht angegeben, wird der Name des Ausgabefiles aus dem Namen des ersten nach der Option angegebenen Eingabefiles ohne die Suffixe ".c", ".s" oder ".o" gebildet.

-Dname=def
-Dname Der name wird fuer den Praeprozessor wie durch `#define' definiert. Wird keine Definition angegeben, so wird der Name als 1 definiert.

-Idir '#include'-Files werden unter dem Directory dir gesucht. Werden sie dort nicht gefunden bzw. ist die -I-Option nicht angegeben, wird unter einem Standarddirectory gesucht ('/usr/include/').

FILES

file.c	input file
file.o	object file
a.out	geladener Objektmodul
/tmp/ctm?	temporaere Dateien fuer cc
/lib/cpp	Praeprozessor
/lib/c[01]	Compiler fuer cc
/lib/c2	wahlweiser Optimierer
/lib/crt0.o	Laufzeitstartroutine
/lib/libc.a	Standardbibliothek, siehe intro (3)
/usr/include	Standard-Directory fuer `#include'-Files

SIEHE AUCH

monitor (3), adb (1), as (1), ld (1)

FEHLERQUELLEN

Die vom C-Compiler gelieferten Fehlermeldungen sind selbsterklaerend. Durch Assembler und Lader sind Meldungen moeglich.

NAME

cd - Aendern des Arbeits-Directorys

UEBERSICHT

cd directory

BESCHREIBUNG

Durch cd wird directory zum aktuellen Arbeitsdirectory. Der Prozess muss das Zugriffsrecht "Ausfuehren" (bzw. "Suchen") in diesem Directory besitzen.

cd wird von Shell direkt ausgefuehrt, ohne einen neuen Prozess zu schaffen.

SIEHE AUCH

sh(1), pwd(1), chdir(2)

NAME

chmod - Aendern der Zugriffsrechte

UEBERSICHT

chmod mode file ...

BESCHREIBUNG

Die Zugriffsrechte fuer jedes angegebene file werden gemaess mode geaendert, wobei sie absolut oder symbolisch angegeben werden koennen.
Ein asolut angegebener mode ist eine Oktalzahl, die durch die ODER-Verknuepfung der folgenden Zugriffsrechte entsteht:

4000 Setzt die Eigentuemeridentifikation bei Ausfuehrung
2000 Setzt die Gruppenidentifikation bei Ausfuehrung
1000 Sticky Bit, Retten Textabbild

0400 Lesen durch den Eigentuemer
0200 Schreiben durch den Eigentuemer
0100 Ausfuehrung (Suchen in einem Directory) durch den Eigentuemer

0070 Lesen, Schreiben, Ausfuehren (Suchen) durch die Gruppe
0007 Lesen, Schreiben, Ausfuehren (Suchen) durch alle anderen Nutzer

Ein symbolischer mode hat die Form:

[who] op permission [op permission] ...

Der who-Teil ist eine Kombination der Buchstaben

u fuer Nutzer,
g fuer die Gruppe,
o fuer alle anderen Nutzer und
a steht fuer ugo.

Wurde who nicht angegeben, wird 'a' zum Standard, gleichzeitig werden die Zugriffsrechte mit der File-Erstellung-Maske (vgl. umaske(2)) durch logisches UND verknuepft. (bei MUTOS 8000 wird 0002 verwendet, d.h. nicht schreibbar fuer andere Nutzer)

Op gibt an, wie die unter permission angegebenen Zugriffsrechte (fuer who) verwendet werden:

+ Zugriffsrechte erteilen (zu den vorhandenen addieren)
- Zugriffsrechte streichen (von den vorhandenen subtrahieren)
= nur die angegebenen Zugriffsrechte erteilen, alle anderen entziehen.

Permission ist eine beliebige Kombination der Zugriffsrechte

r Lesen,
w Schreiben,
x Ausfuehren,
s Setzen von Eigentuemer oder Gruppen-ID bei Ausfuehrung
t Retten Textabbild nach Ausfuehrung (sticky) x.
ugo Die Zugriffsrechte werden aus den vorhandenen Zugriffsrechten von Nutzer, Gruppe oder anderen Nutzern uebernommen (nur sinnvoll in Verbindung mit u g).

Das Weglassen von permission ist nur sinnvoll mit = als op, um alle Zugriffsrechte zu entziehen.

Beispiele:

- Schreibrechte fuer alle anderen Nutzer entziehen
  chmod o-w file

- ein File soll fuer alle ausfuehrbar werden
  chmod +x file

- die Gruppe soll auch alle Zugriffsrechte des Nutzers erhalten
  chmod g+u file

Mehrere symbolische Zugriffsrechte werden durch Komma getrennt.

- der Gruppe und den anderen Nutzern die Schreiberlaubnis entziehen
  chmod g-w, o-w file

Nur der Eigentuemer eines Files (oder der Super-User) kann Zugriffsrechte aendern.

Beachten Sie bitte, dass in Directories, die nicht ausfuehrbar sind, kein Suchen von Dateien, Pfadnamen usw. erlaubt ist.

SIEHE AUCH

ls(1), chmod(2), chown (1), stat(2), umask(2)

CHOWN(1) **CHOWN(1)**

NAME

chown, chgrp - Aenderung von Eigentuemer und Gruppe

UEBERSICHT

chown owner file ...

chgrp group file ...

BESCHREIBUNG

Chown aendert den Eigentuemer der files zu owner. Der Eigentuemer kann entweder ein dezimaler UID oder ein Login - Name sein, der im Password File zu finden ist.

Chgrp aendert den Gruppen-ID der files zu group. Die Gruppe kann entweder ein dezimaler GID oder ein Gruppenname sein, der im group-File enthalten ist.

Nur der Super-Nutzer kann den Eigentuemer oder die Gruppe aendern.

FILES

/etc/passwd : Password File
/etc/group : Gruppen-ID File

SIEHE AUCH

chown(2), passwd(5), group(5)

CLRI(1M) **CLRI(1M)**

NAME

clri - Loeschen eines i-node

UEBERSICHT

clri filesystem i-number ...

BESCHREIBUNG

Clri schreibt auf die i-nodes von filesystem Nullen, die i-number wird dezimal angegeben. Nach clri erscheint jeder Block der betroffenen Datei bei einem icheck(1) des Dateisystems als 'missing'.

Lese- und Schreibrechte sind fuer das angegebene Filesystem erforderlich. Der i-node wird wieder frei.

Der Hauptzweck dieses Programms ist das Loeschen einer Datei, die aus irgendwelchen Gruenden in keinem Directory enthalten ist. Wird es auf einen i-node angewendet, der in einem Directory enthalten ist, muss diese Directoryeintragung bis zum Ende verfolgt und entfernt werden.

Andernfalls zeigt die alte Directory-Eintragung bei Neuvergabe des i-nodes auf das neue File. Ein Loeschversuch fuer das alte File fuehrt dann zum Loeschen des neuen Files. Die Directoryeintragung des neuen Files aber weist wiederum auf einen unbelegten i-node. Damit kann sich der beschriebene Effekt staendig wiederholen.

SIEHE AUCH

icheck(1)

FEHLERQUELLEN

Auf geoeffnete Files sollte clri nicht angewendet werden.

NAME

cmp - Vergleich zweier Files

UEBERSICHT

cmp [-l] [-s] file1 file2

BESCHREIBUNG

Die beiden angegebenen Files werden verglichen. Ist als file1 nur'-' angegeben, so wird Standard-Input mit file2 verglichen. Werden keine Unterschiede zwischen den verglichenen Files festgestellt, so wird von cmp keine Meldung ausgegeben. Wird ein Unterschied bemerkt, so wird die Byte- und die Zeilennummer (dezimal) des ersten festgestellten Unterschiedes ausgegeben und der Vergleich abgebrochen. Stellt ein File eine vom Beginn an mit dem anderen File uebereinstimmende echte Teilfolge von Zeichen dar, so wird dies angezeigt.

Eine modifizierte Arbeitsweise laesst sich durch Angabe der folgenden Optionen erreichen:

-l Ausgabe der Byte-Adresse (dezimal) und des Inhaltes der differierenden Bytes (oktal) fuer jeden festgestellten Unterschied.

-s Keine Ausgabe der Unterschiede, d.h., nur ueber den Rueckkehrcode kann das Ergebnis dieses Vergleiches ausgewertet werden.

SIEHE AUCH

sh(1)

FEHLERMELDUNGEN

Fuer identische Files ist der Rueckkehrcode gleich 0, fuer unterschiedliche Files gleich 1. Bei einem Zugriffsfehler oder einer fehlerhaften Anzahl von Argumenten wird 2 als Ergebnis zurueckgegeben.

NAME

convert - Wandlung Fileformat

UEBERSICHT

convert filesystem [from][to]

BESCHREIBUNG

Convert wandelt das Filesystem der Diskette.
Durch die Spezifikation to wird das Filesystem einer MUTOS 8000-Diskette in das Filesystem von MUTOS-1630 gewandelt.
From leitet das Wandeln des Filesystems von MUTOS-1630 zu MUTOS8000 ein.
Das Kommando darf nur vom Super-User ausgefuehrt werden.

FEHLERBEHANDLUNG

Ein irrtuemlich gewandeltes Filesystem kann mi convert wieder zurueckgewandelt werden.

CP(1) **CP(1)**

NAME

cp -Kopieren von Files

UEBERSICHT

cp file1 file2

cp file ... directory

BESCHREIBUNG

File1 wird nach file2 kopiert. Modus und Eigentuemer von file2 werden, wenn es bereits existierte, erhalten. Ist das nicht der Fall, wird der Modus des Quellfiles uebernommen.
In der zweiten Form werden ein oder mehrere files mit ihren Original-Filenamen in das directory kopiert.

Cp erlaubt nicht, ein File auf sich selbst zu kopieren.

SIEHE AUCH

cat(1), pr(1), mv(1)

NAME

date - Ausgabe und Setzen des Datums

UEBERSICHT

date [yymmddhhmm [.ss]]

BESCHREIBUNG

Durch date wird Datum und Uhrzeit ausgegeben oder gesetzt.
Ist kein Argument angegeben, werden das augenblickliche Datum und die Zeit ausgegeben.

Ist ein Argument spezifiziert, wird das angegebene Datum gesetzt. Die Zeichen yy bedeuten die letzten zwei Einheiten des Jahres. Das erste mm ist die Nummer des Monats; dd ist die Nummer des Tages im jeweiligen Monat; hh sind die Stunden (24-Stunden-System); das zweite mm sind die Minuten; .ss kann wahlweise fuer Sekunden angegeben werden.
Z.B. setzt:

```
date 10081245
```

das Datum auf den 8.Oktober, 12.45 Uhr. Jahr, Monat und Tag koennen auch weggelassen werden, in diesem Fall werden die momentanen Werte beibehalten.

FILES

/usr/adm/wtmp zum Abspeichern der Zeit, falls vorhanden.

SIEHE AUCH

utmp(5)

FEHLER -DIAGNOSTICS

Falls jemand, der nicht als Super-User angemeldet ist, versucht, das Datum zu aendern, erfolgt die Ausgabe "no permission" ("keine Erlaubnis").
Ist die Eingabe syntaktisch fehlerhaft, wird die Meldung "bad conversion" ausgegeben.

NAME

dd - Konvertieren und Kopieren eines Files

UEBERSICHT

dd [option=value] ...

BESCHREIBUNG

Das Dienstprogramm dd kopiert das angegebene Input-File auf das spezifizierte Output-File, wobei Konvertierungen moeglich sind. Sind keine Parameter angegeben, werden stattdessen Standard-Input und Standard-Output angenommen.
Die Groesse des Ein- bzw. Ausgabeblockes kann variiert und somit koennen die Vorteile der speziellen physischen Ein- bzw. Ausgaberoutinen ('raw'-Version) genutzt werden.

Option	Werte
if=	Bezeichnung des Input-Files; Standard ist Standard-Input
of=	Bezeichnung des Output-Files; Standard ist Standard-Output
ibs=n	Input-Blocklaenge (input block size) n in Byte (Standardwert 512)
obs=n	Output-Blocklaenge (output block size - Standardwert 512 Byte)
bs=n	Setzen von Input- und Output-Blocklaenge; die Angaben von ibs und obs werden ueberschrieben.Das ist insbesondere dann effizient, wenn keine Konvertierung spezifiziert ist, weil in diesem Falle kein Kopieren erforderlich ist.
cbs=n	Groesse des Konvertierungspuffer (conversion buffer size)
skip=n	Ueberspringen (skip) von n Saetzen (Bloecken) des Input-Files, bevor mit dem Kopieren begonnen wird
seek=n	Positionieren auf den n-ten Satz des Output-Files.
count=n	Es sollen nur n Saetze vom Input kopiert werden
conv=ascii	Konvertieren von DKOI-Code in ISO-Code
ebcdic	Konvertieren von ISO-Code in DKOI-Code
ibm	modifiziertes Konvertieren von ISO-Code in DKOI-Code
lcase	alphabetische Zeichen werden in Kleinbuchstaben (lower case) umgewandelt
ucase	alphabetische Zeichen werden in Grossbuchstaben (upper case) umgewandelt
swab	jedes Bytepaar wird vertauscht (L-H)
noerror	bei Auftreten eines Fehlers erfolgt kein Halt bei der Programmabarbeitung.

sync jeder Input-Satz wird bis zur Input-Blocklaenge ibs aufgefuellt.
... , ... verschiedene Konvertierungsformen sind durch Kommata zu trennen

Ueberall dort, wo Laengen spezifiziert sind, wird eine Zahlenangabe in Byte (dezimal) erwartet. Eine Zahlenangabe kann dabei mit **k** , **b** oder **w** abgeschlossen werden, um eine Multiplikation mit 1024, 512 bzw. 2, anzuweisen. Ein Zahlenpaar kann durch 'x' separiert werden, um ein Produkt anzuzeigen.

Die Option cbs wird nur benützt, wenn ISO- oder DKOI-Konvertierung spezifiziert ist. Im ersten Fall werden soviel Zeichen, wie in cbs angegeben sind, in den Konvertierungspuffer geschafft und in ISO konvertiert. Abschliessende Leerzeichen und ein Zeilenwechselzeichen (oktal 12) werden hinzugefuegt, bevor die Zeile zum Output-File transportiert wird. Im zweiten Fall werden ISO- Zeichen in den Konvertierungspuffer gelesen, in DKOI konvertiert und Leerzeichen hinzugefuegt, um einen Output-Satz von der in cbs spezifizierten Laenge zu erzielen.

Nach Abschluss der Ausfuehrung gibt dd die Anzahl der vollstaendigen und der angefangenen Input- und Output-Blocke an.

Um z.B. ein ISO-File test in ein ISO-File test1 zu ueberfuehren, wobei alphabetische Zeichen in Grossbuchstaben umgewandelt werden sollen, ist zu schreiben:

"dd if=test of=test1 conv=ucase"

SIEHE AUCH

cp(1),

AUSGABEN

f+p records in(out): Anzahl der vollstaendigen [full f] und angefangenen [partial p], gelesenen [in] bzw. geschriebenen [out] Saetze [records].

FEHLERQUELLEN

Die ISO/DKOI-Konvertierungstabellen wurden dem 256-Zeichen-Standard aus der "Communication of the ACM", November 1968 entnommen. Die verwendete Konvertierung ist kein echter Standard. Eine universelle Loesung ist an dieser Stelle nicht moeglich.

Zeilenwechselzeichen (oktal 12) werden nur beim Konvertieren in ISO eingefuegt. Ein Auffuellen wird nur bei der Konvertierung in DKOI vorgenommen. Beide sollten als getrennte Optionen verstanden werden.

NAME

df - freier Speicherplatz auf Filesystemen

UEBERSICHT

df [filesystem] ...

BESCHREIBUNG

Df listet die Anzahl der freien Bloecke auf dem Filesystem aus.
Ist kein Filesystem spezifiziert, wird der freie Platz aller eingegliederten (mounted) Filesysteme ausgeschrieben.

SIEHE AUCH

icheck(1), fitesys(5)

NAME

dcheck - Pruefen der Folgerichtigkeit der Directories eines Filesystems

UEBERSICHT

dcheck [-i numbers] [filesystem]

BESCHREIBUNG

Dcheck liest die Directories eines Filesystems und vergleicht die Link-Zaehler in jedem i-node mit der Anzahl der Directoryeintragungen, die auf den i-node Bezug nehmen.

Die -i Option liefert fuer alle nachfolgend angegebenen i-node-Nummern, die i-node-Nummer der zugehoerigen Directory und den Namen der Eintragung.

Das Programm arbeitet am schnellsten, wenn die 'Raw'-Variante (rrk) des Geraetes benutzt wird. In diesem Fall wird die i-Liste in grossen Abschnitten gelesen.

SIEHE AUCH

icheck(1), filsys(5), clri(1), ncheck(1)

DIAGNOSTIK

Es werden alle Files angezeigt, bei denen der Link-Zaehlernicht mit der Anzahl der Directory-Eintragungen uebereinstimmt oder beide Werte 0 sind. Dabei ist nur der Fall kritisch, bei dem die Anzahl von Eintragungen groesser ist als der Link-Zaehler; erreicht der Link-Zaehler beim Loeschen von Eintragungen den Wert 0, fehlt den verbleibenden Eintragungen der Bezug auf den Datenbereich. Derartige Zustaende sollten durch Loeschen des i-nodes und der Directory-Eintragungen beseitigt werden. In den beiden anderen Faellen kann Speicherbereich verlorengehen, jedoch kein kritischer Zustand entstehen.

FEHLERQUELLEN

Dcheck arbeitet in zwei Durchlaeufen. Treten zwischen den beiden Phasen Veraenderungen im zu pruefenden Filesystem auf, kann das zu Fehlermitteilungen fuehren. Deshalb ist dcheck nicht auf aktive Filesysteme anzuwenden.

ECHO(1) **ECHO(1)**

NAME

echo - Ausgabe der Argumente

UEBERSICHT

echo [-n] [arg]

BESCHREIBUNG

Echo schreibt seine Argumente nach Standard-Output. Die Argumente werden bei der Ausgabe durch Leerzeichen getrennt und mit Newline abgeschlossen. Ist die Option -n angegeben, wird kein Newline ausgegeben.

Echo kann zum Erzeugen von Fehlermeldungen in Shell-Programmen und zum Uebergeben definierter Daten an Pipes verwendet werden. Um Fehlermeldungen an das Standard-Error-File zu uebergeben, ist die Angabe 'echo ... 1>&2' notwendig.

NAME

ed - Texteditor

UEBERSICHT

ed [-] [-x] [name]

BESCHREIBUNG

Ed ist der Standardeditor.
Die folgende Liste beinhaltet die Kommandos des ed.

Wurde das Argument name angegeben, simuliert der Editor ed ein e Kommando fuer das bezeichnete File. Das bedeutet, das File wird in den Editorpuffer gelesen und kann dort bearbeitet werden. Wurde -x angegeben, so wird ein x Kommando simuliert, um primaer ein verschluesseltes File zu behandeln. Die Option - unterdrueckt das Drucken der Zeichenzahlangabe bei e , r , und z Kommandos.

Ed bewirkt das Editieren der Kopie eines beliebigen Files; die in der Kopie ausgefuehrten Aenderungen haben keine Auswirkung, solange bezueglich des Files kein w Kommando (write) gegeben wurde. Die Textkopie, die editiert wird, befindet sich in einem temporaeren File, dem Puffer.
Die Kommandos fuer ed haben eine einfache und feststehende Stuktur : Null oder mehrere Adressen , denen ein einzelnes Zeichen, das Kommando, folgt; eventuell durch Parameter fuer dieses Kommando ergaenzt. Die Adressen bestimmen eine oder mehrere Zeilen im Puffer. Fehlende Adressen werden durch Standardwerte ersetzt. Es kann jeweils nur ein Kommando auf einer Zeile erscheinen.

Verschiedene Kommandos erlauben eine Eingabe von Text in den Puffer. Nimmt ed Text auf, sagt man, der Editor ist im Eingabemodus input mode. In diesem Modus werden Kommandos nicht als solche erkannt; alles Eingegebene wird angefuegt. Der Eingabemodus wird mit einem einzelnen Punkt '.' am Zeilenanfang verlassen.

Ed unterstuetzt eine begrenzte Form einer regulaeren Ausdrucksnotation. Ein regulaerer Ausdruck weist auf eine Gruppe von Zeichenfolgen. Ein Bestandteil dieser Gruppe von Folgen kann als regulaerer Ausdruck von Folgen betrachtet werden. In der folgenden Beschreibung der regulaeren Ausdruecke bedeutet das Wort 'Zeichen' ein beliebiges Zeichen, ausser "newline".

1. Gewoehnliche Zeichen, ausser Sonderzeichen, entsprechen sich selbst. Sonderzeichen sind Begrenzer fuer regulaere Ausdruecke sowie \{. und manchmal ^*$.

2. Ein . entspricht einem beliebigen Zeichen.

3. Ein '\' , dem ein beliebiges Zeichen, ausser einer Ziffer oder () folgt, entspricht diesem Zeichen.

4. Eine nichtleere Folge s , die in eckigen Klammern eingeschlossen ist, [s] (oder [^s]), entspricht irgendeinem Zeichen aus (oder nicht aus) s. Innerhalb von s hat '\' keine besondere Bedeutung und kann nur an erster Stelle erscheinen. Eine Teilfolge a-b, mit a und b in steigender ASCII-Ordnung, steht fuer die eingeschlossene Folge von ASCII-Zeichen.

5. Ein regulaerer Ausdruck gemaess Pkt. 1-4, dem ein * folgt, entspricht dem 0 oder mehrmaligen Vorkommen des regulaeren Ausdrucks.

6. Ein regulaerer Ausdruck x, gemaess Pkt. 1-8 , der eingeklammert ist \(x\), entspricht der Bedeutung von x.

7. Ein \, dem eine Ziffer n folgt, entspricht der Kopie einer Folge eines eingeklammerten regulaeren Ausdrucks, der beim n-ten Vorkommen von \(beginnt.

8. Ein regulaerer Ausdruck x gemaess Pkt. 1-8, dem ein regulaerer Ausdruck y gemaess Pkt. 1-7 gefolgt wird, entspricht der Existenz von x, gefolgt durch die Existenz von y, wobei die Existenz von x so lang wie moeglich ist und trotzdem die Existenz von y zulaesst.

9. Ein regulaerer Ausdruck gemaess Pkt. 1-8 mit vorangestelltem ^ (oder nachgestelltem $) wird gezwungen, am Beginn (oder am rechten Ende) einer Zeile zu stehen.

10. Ein regulaerer Ausdruck gemaess PKT. 1-9 sucht den laengsten unter den am weitesten links in einer Zele vorkommenden.

11. Ein leerer regulaerer Ausdruck steht fuer eine Kopie des letzten vorhandenen regulaeren Ausdrucks.

Regulaere Ausdruecke werden bei der Adressierung benutzt, Zeilen zu bestimmen und in Substitutionskommandos, um den zu ersetzenden Teil einer Zeile zu finden. Moechte man in einem regulaeren Ausdruck Sonderzeichen als gewoehnliche Zeichen nutzen, so kann dies durch Vorsetzen eines '\' erreicht werden. Dies bezieht sich auch auf das Zeichen fuer die Begrenzung eines regulaeren Ausdrucks (im allgemeinen '/') und auf '\' selbst.

Um die Adressierung bei ed zu verstehen, ist es wichtig zu wissen, dass es immer eine aktuelle Zeilenadresse gibt. Allgemein betrachtet man die letzte Zeile, die durch ein Kommando beeinflusst wurde, als aktuelle Zeile; trotzdem wird die genaue Lage der aktuellen Zeile bei der Beschreibung des jeweiligen Kommandos erlaeutert. Die Adressen werden gemaess folgender Regeln erzeugt:

1. Das Zeichen '.' (dot) adressiert die aktuelle (laufende) Zeile.

2. Das Zeichen '$' adressiert die letzte Zeile des Puffers.

3. Eine Dezimalzahl n adressiert die n-te Zeile im Puffer.

4. ''x' adressiert die Zeile, die mit dem Namen x markiert wurde, welcher ein Kleinbuchstabe sein muss. Die Zeilen werden mit dem k Kommando markiert.

5. Ein regulaerer Ausdruck, der in Schraegstrichen '/' eingeschlossen ist, adressiert die Zeile, die durch Vorwaertssuchen von der aktuellen Zeilenadresse +1 bis zur ersten Zeile, die die entsprechende Folge des regulaeren Ausdrucks enthaelt, gefunden wurde. Wenn noetig, kann das Suchen ueber den Pufferbeginn umlaufen.

6. Ein regulaerer Ausdruck, der in Fragezeichen '?' eingeschlossen ist, adressiert die Zeile, die durch Rueckwaertssuchen von der aktuellen Zeilenadresse -1 bis zur ersten Zeile, die die entsprechende Folge des regulaeren Ausdrucks enthaelt, gefunden wurde. Wenn noetig, kann das Suchen ueber das Pufferende umlaufen.

7. Eine Adresse, der ein Pluszeichen '+' oder Minuszeichen '-' folgt und ausserdem eine Dezimalzahl, bedeutet : Adresse plus (bzw. minus) der angegebenen Zeilenzahl. Das Pluszeichen kann auch weggelassen werden.

8. Beginnt eine Adresse mit '+' oder '-', so wird die Addition oder Subtraktion in Bezug auf die aktuelle Zeile realisiert, d.h. '-5' bedeutet '.-5'.

9. Endet eine Adresse mit '+' (oder '-') so wird 1 addiert bzw. substrahiert. Als eine Folge dieser Regel und der Regel 8 bezieht sich '-' auf die Zeile vor der aktuellen. Ferner hat ein nachfolgendes '+' und '-' eine additive Wirkung. So bezieht sich '--' auf die aktuelle Zeile minus 2.

10. Um die Kompatibilitaet mit frueheren Versionen des Editors zu erhalten, ist das Zeichen '^' in Adressen aequivalent zu '-' .

Kommandos koennen null, ein oder zwei Adressen erfordern. Kommandos, die keine Adressen benoetigen, betrachten das Vorhandensein einer Adresse als einen Fehler. Kommandos, welche ein oder zwei Adressen akzeptieren, bilden Adressen, wenn solche fehlen. Wurden mehr Adressen angegeben, als ein entsprechendes Kommando akzeptiert, dann werden die letzte oder die letzten beiden Adressen benutzt.

Adressen werden durch Komma voneinander getrennt. Sie koennen aber auch durch ein Semikolon ';' getrennt sein. In diesem Falle wird die aktuelle Zeilenadresse '.' auf die zuletzt ausgewertete Zeile gesetzt, bevor die naechste Adresse interpretiert wird. Diese Eigenschaft kann zum Bestimmen der Anfangszeile fuer Vorwaerts- und Rueckwaertssuchen ('/' oder '?') benutzt werden. Die zweite Adresse irgendeiner Zweiadressfolge muss eine Zeile bestimmen, die der zuerst adressierten folgt.

In der nachstehenden Liste der ed Kommandos sind die Standardadressen in runden Klammern angegeben. Die Klammern sind nicht Teil der Adresse.

Es ist im allgemeinen nicht zulaessig, dass mehr als ein Kommando auf einer Zeile erscheint. Es kann jedoch an die meisten Kommandos ein 'p' oder 'l' angefuegt werden, wodurch die aktuelle Zeile entweder gedruckt oder gelistet wird.

```
(.)a
(text)
.
```

Das Anfuegekommando (append) liest den vorliegenden Text und fuegt ihn nach der adressierten Zeile an. Die aktuelle Zeilenadresse wird, wenn Text eingegeben wurde, auf die letzte Eingabezeile gelegt. Anderenfalls liegt sie auf der adressierten Zeile. Die Adresse '0' ist fuer dieses Kommando gueltig; der Text wird in diesem Fall am Anfang des Puffers untergebracht.

```
(., .)c
(text)
.
```

Das Aenderungskommando (change) loescht die adressierten Zeilen, nimmt den eingegebenen Text auf und ersetzt damit die geloeschten Zeilen. Die aktuelle Zeilenadresse liegt auf der letzten Eingabezeile. Wenn nichts eingegeben wurde, liegt diese auf der Zeile, die der geloeschten vorangeht.

(.,.)d

Das Kommando 'entferne Zeile' (delete) entfernt die addressierten Zeilen aus dem Puffer. Die Zeile, die urspruenglich nach der letzten zu entfernenden lag, wird die aktuelle Zeile; waren die entfernten Zeilen urspruenglich am Ende, so wird die neue letzte Zeile die aktuelle.

e filename

Dieses Kommando 'editieren' veranlasst, den vollstaendigen Inhalt des Puffers zu entfernen und danach das angegebene File einzulesen. Die aktuelle Zeilenadresse wird auf die letzte Zeile des Puffers gesetzt. Die Anzahl der gelesenen Zeichen wird ausgegeben. Der filename wird fuer eine moegliche Nutzung als Standardfilename fuer ein nachfolgendes r oder w Kommando gemerkt. Fehlt filename, wird der gemerkte Name benutzt.

E filename

Dieses Kommando entspricht dem e Kommando; es erfolgt jedoch keine Warnung, wenn nach der letzten Pufferaenderung kein w gegeben wird.

f filename

Dieses Kommando 'Filename' gibt den zuletzt gemerkten Filenamen aus. Wurde ein 'filename' angegeben, so wird der zuletzt gemerkte aktuelle Filename durch 'filename' ersetzt.

(1,$)g/regulaerer Ausdruck/Kommandofolge

Innerhalb des global Kommandos ist der erste Schritt, jede Zeile zu markieren, die den angegebenen regulaeren Ausdruck enthaelt. Danach wird jede markierte Zeile entsprechend der gegebenen Kommandofolge bearbeitet und jedes Mal zur aktuellen Zeile gemacht. Ein einzelnes Kommando oder das erste von mehreren Kommandos erscheint auf der gleichen Zeile wie das globale Kommando. Alle Zeilen einer mehrzeiligen Kommandofolge, ausser der letzten Zeile, muessen mit '\' abgeschlossen werden. a, i, und c Kommandos und gemeinsame Eingabe sind erlaubt. Der '.', der den Eingabemodus beendet, kann weggelassen werden, wenn er auf der letzten Zeile der Kommandofolge ist. Die Kommandos g und v sind in der Kommandofolge nicht zugelassen.

(.)i
(text)

Dieses Kommando fuegt (insert) den vorliegenden Text vor die adressierte Zeile ein. Die aktuelle Zeilenadresse liegt auf der letzten Eingabezeile oder, wenn nichts eingegeben wurde, auf der Zeile, die der adressierten vorangeht. Dieses Kommando

unterscheidet sich von dem g Kommando nur durch die Plazierung des Textes.

(., .+1)j
Dieses Kommando verbindet (join) die adressierten Zeilen zur einer Zeile; dazwischenliegende "newline" verschwinden. Die resultierende Zeile stellt die aktuelle Zeile dar.

(.)kx
Das Markierungskommando (mark) markiert die adressierte Zeile mit dem Namen x, der ein Kleinbuchstabe sein muss. Die Adressform ''x' adressiert diese Zeile.

(., .)l
Das List-Kommando druckt die adressierten Zeilen in eindeutiger Form: Nichtgrafikzeichen werden zweistellig oktal gedruckt, lange Zeilen werden geteilt. Das l Kommando kann auf der gleichen Zeile nach einem beliebigen Nicht-E/A-Kommando angefuegt werden.

(., .)ma
Das Transportkommando (move) verlagert die adressierten Zeilen hinter die durch a adressierte Zeile. Die letzte transportierte Zeile wird die aktuelle Zeile.

(., .)p
Das Druckkommando (print) druckt die adressierten Zeilen. Die aktuelle Zeilenadresse ist die der letzten gedruckten Zeile. Das p Kommando kann auf der gleichen Zeile nach einem Nicht-E/A- Kommando angefuegt werden.

(., .)P
Diese Kommando ist identisch mit p.

q Das Quitkommando veranlasst das Verlassen des Editors. Es erfolgt kein automatisches Schreiben des Files.

Q Dieses Kommando entspricht q; es erfolgt keine Warnung, wenn nach der letzten Pufferaenderung kein w gegeben wird.

(*)r filename
Das Filelesekommando (read) liest das angegebene File hinter die adressierte Zeile (d.h., es schreibt es!). Wurde kein Filename angegeben, so wird der gemerkte Filename benutzt (siehe e und f Kommandos). Der Filename wird gemerkt, wenn es noch keinen gemerkten Filenamen gibt. Die Adresse '0' ist fuer r

gueltig und bestimmt, dass das File an den Anfang des Puffers gelesen wird. War das Lesen erfolgreich, wird die Anzahl der gelesenen Zeichen ausgegeben. Die letzte von dem File gelesene Zeile ist die Zeile mit der aktuellen Adresse.

(., .)s/regulaerer Ausdruck/Ersatz/ oder
(., .)s/regulaerer Ausdruck/Ersatz/g

Das Substitutionskommando durchsucht jede adressierte Zeile nach Vorhandensein des angegebenen regulaeren Ausdrucks. Auf jeder Zeile, auf der dieser gefunden wurde, werden alle entsprechenden Zeichenfolgen durch den angegebenen Ersatz ausgetauscht, wenn das Kennzeichen fuer ein globales Ersetzen g nach dem Kommando erscheint. Wurde das global-Kennzeichen nicht gesetzt, so wird nur die jeweils erste Zeichenfolge einer Zeile ausgetauscht. Es ist ein Fehler fuer eine Substitution, wenn sie auf keiner Zeile ausgefuehrt werden kann. Als Begrenzung des regulaeren Ausdrucks und des Ersatzes kann an Stelle von '/' jedes beliebige Zeichen, ausser Leerzeichen und "newline", verwendet werden. Die aktuelle Zeilenadresse wird auf die letzte substituierte Zeile gelegt.
Ein Ampersand '&', das im Ersatz erscheint, wird durch eine Zeichenfolge ersetzt, die dem regulaeren Ausdruck entspricht. Die spezielle Bedeutung von '&' in diesem Textteil kann durch das Vorstellen eines '\' unterdrueckt werden. Die Zeichen '*n*' , wobei *n* Ziffern sind, werden durch den Text ersetzt, der dem *n*-ten in '\(' und '\)' eingeschlossenen regulaeren Unterausdruck entspricht. Sind die eingeklammerten Unterausdruecke ineinandergeschachtelt, wird *n* durch Zaehlen der vorhandenen '\(', von links beginnend, ermittelt.

Die Zeilen koennen auch durch Einfuegen des "newline"- Zeichen geteilt werden. Dem Zeichen "newline" muss in der herzustellenden Folge ein '\' vorangestellt werden.

(., .)t*a*

Dieses Kommando wirkt wie das *m* Kommando, wobei jedoch die Kopie der adressierten Zeilen nach der Adresse *a* plaziert wird, die auch 0 sein kann. Die letzte Zeilenkopie wird die aktuelle Zeile.

(., .)u
Das Kommando "rueckgaengigmachen" (undo) gibt den ursprueglichen Inhalt der aktuellen Zeile zurueck, wobei in dieser Zeile zuletzt eine Substitution ausgefuehrt worden sein muss.

(1, $)v/regulaerer Ausdruck/Kommandofolge
Dieses Kommando entspricht dem globalen Kommando g, jedoch wird die Kommandofolge fuer jede aktuelle Zeile ausgefuehrt, die den regulaeren Ausdruck nicht enthaelt.

(1, $)w filename
Das Fileschreibkommando (write) schreibt die adressierten Zeilen in das angegebene File. Existierte das File nicht, wird der Modus 666 (lesbar und beschreibbar durch jeden) hergestellt. Der Filename wird gemerkt, wenn es noch keinen gemerkten Filenamen gibt. Wurde kein Filename angegeben, so wird irgendein gemerkter Filename benutzt (siehe e und f Kommandos). Die aktuelle Zeilenadresse bleibt unveraendert. War das Kommando erfolgreich, wird die Anzahl der rueckgeschriebenen Zeichen angezeigt.

(1,$)W filename
Dieses Kommando entspricht w; die adressierten Zeilen werden jedoch an das File angefuegt.

x Es wird eine Verschluesselungsfolge von der Standardeingabe verlangt. Anschliessend verschluesseln bzw. entschluesseln r, e und w Kommandos den Text mit diesem Schluessel durch den crypt(1) Algorithmus. Ein spezieller leerer Schluessel schaltet das Verschluesseln aus.

($)= Die Zeilennummer wird durch dieses Kommando ausgegeben, wobei die aktuelle Zeilenadresse nicht veraendert wird.

!(shell command)
Das dem '!' auf dieser Zeile folgende wird nach sh(1) gesendet, um es als ein Kommando zu interpretieren. Die aktuelle Zeilenadresse wird nicht veraendert.

(.+1)(newline)
Eine Adresse allein auf einer Zeile veranlasst, dass die adressierte Zeile gedruckt wird. Eine Leerzeile ist aequivalent mit .+1. Damit ist ein schrittweises Anzeigen dieses Textes moeglich.

Wird ein Interruptsignal (CONTROL C) gesendet, druckt ed ein '?' und kehrt zurueck zu seiner Kommandoebene.
Einige Groessen fuer die Grenzen sind 512 Zeichen pro Zeile, 256 Zeichen pro globaler Kommandofolge, 64 Zeichen pro Filenamen und 128 K Zeichen im temporaeren File. Die Grenze der Zeilenzahl ist von der Groesse des Speichers abhaengig: jede Zeile nimmt ein Wort in Anspruch.

Wird ein File gelesen wird, entfernt der Editor ISO-NUL-Zeichen und alle Zeichen nach dem letzten "newline". Er liest keine Files, die Nicht-ISO Zeichen enthalten.

FILES

/tmp/e*
ed.hup: das Ergebnis wird gerettet, wenn das Terminal abgehaengt wird.

DIAGNOSTIK

'?name' fuer ein nichtauffindbares File; '?' fuer einen Kommandofehler; '?TMP' fuer Ueberlauf des temporaeren Files.

Ein g oder e Kommando war fehlerhaft, sofern nicht nach der letzten Pufferaenderung ein w gegeben wurde. Ein zweites g oder e wird immer befolgt.

FEHLERQUELLEN

Das l Kommando veraendert DEL.
Ein ! Kommando kann einem g Kommando nicht unterworfen werden.
Da 0 eine ungueltige Adresse fuer ein w Kommando ist, ist es nicht moeglich, ein leeres File mit ed herzustellen.

HEX(1) **HEX(1)**

NAME

hex - Hexadezimale Ausgabe von Files

UEBERSICHT

hex file ...

BESCHREIBUNG

Hex listet den Inhalt von file in hexadezimaler Darstellung und byteweise interpretiert als ISO-Zeichen aus. Jeweils 16 Byte werden auf einer Zeile dargestellt. File wird vollstaendig auf Standard-Output ausgegeben.

SIEHE AUCH

od(1)

NAME

icheck - Pruefen der Folgerichtigkeit der Speicherbelegung

UEBERSICHT

icheck [-s] [-b numbers] [filesystem]

BESCHREIBUNG

Icheck ermittelt die belegten Bloecke eines Filesystems und vergleicht diese mit der Liste freier Bloecke, die im Filesystem gefuehrt wird.

Das Programm liefert folgende Angaben:

- Anzahl der Files insgesamt sowie die Anzahl normaler Files, Directories und Spezial-Files fuer blockorientierte und nicht-blockorientierte Geraete,
- Gesamtzahl belegter Bloecke und die Anzahl einfach-, doppelt- und dreifach-indirekter Bloecke sowie die Anzahl von Directories,
- Anzahl freier Bloecke,
- Anzahl nicht erfasster Bloecke (weder in einem File noch in der Liste freier Bloecke enthalten).

Die -s Option bewirkt, dass die aktuelle Liste freier Bloecke ignoriert wird. Icheck legt dann eine neue Freiliste entsprechend der ermittelten Speicherbelegung an und schreibt den Superblock (der diese Liste enthaelt) neu. Dabei ist darauf zu achten, dass mit dem behandelten Filesystem keine anderen Aktivitaeten laufen. Es sollte ausgegliedert sein. Ist das nicht moeglich, wie z.B. beim root-Filesystem selbst, darf waehrend icheck keine andere Filearbeit erfolgen. Unmittelbar nach Abschluss von icheck auf das root-Filesystem ist dieses neu zu booten, damit fuer die Weiterarbeit der neue Superblock (und nicht der noch im Hauptspeicher befindliche alte) benutzt wird. Die im Superblock enthaltenen Angaben fuer die Groesse der Liste freier Bloecke und der i-Liste werden von icheck nicht veraendert. Bei Benutzung der -s Option werden die normalen Ausschriften unterdrueckt.

Die -b Option bewirkt, dass jedes Auftreten der hinter -b angegebenen Blocknummern in einem File angezeigt wird.

Icheck arbeitet am schnellsten, wenn die 'Raw'- Version (rrk) des Geraetes benutzt wird. In diesem Fall wird die i-Liste in grossen Abschnitten gelesen.

SIEHE AUCH

dcheck(1), ncheck(1), filsys(5), clri(1)

DIAGNOSTIK

Gemeldet werden doppelt vergebene Bloecke (dup) und Bloecke, deren Adressen ausserhalb des Filesystems liegen (bad), mit ihrer i-Nummer und der Art ihrer Benutzung. Treten (Hardware-)Lesefehler auf, wird die Nummer des fehlerhaften Blockes angezeigt und sein Inhalt fuer die Weiterarbeit mit 0 angenommen. Ist eine Blocknummer ausserhalb des zur Verfuegung stehenden Bereiches in der Liste freier Bloecke enthalten, wird 'Bad freeblock' angezeigt. 'n dups in free' bedeutet, dass n Bloecke in der Liste freier Bloecke gefunden wurden, die auch in einem File oder nochmals in der Liste freier Bloecke enthalten sind.

FEHLERQUELLEN

Icheck arbeitet in zwei Durchlaeufen. Treten zwischen den beiden Phasen Veraenderungen im zu pruefenden Filesystem auf, kann das zu Fehlermeldungen fuehren. Deshalb ist icheck nicht auf aktive Filesysteme anzuwenden.

NAME

kill - Beenden eines Prozesses

UEBERSICHT

kill [-signo] processid ...

BESCHREIBUNG

Das Kommando kill sendet Signal 15 (beenden) zu den durch processid (Prozessidentifikator) spezifizierten Prozessen. Falls als erstes Argument eine Signalnummer signo mit vorangestelltem - spezifiziert ist, wird dieses Signal anstelle von "Beenden" gesendet. (Siehe signal(2)). Dadurch koennen auch Prozesse abgebrochen werden, die das Signal sonst nicht empfangen, insbesondere ist kill -9 ... ein sicherer Abbruch.

Ist der Prozessidentifikator 0 spezifiziert, wird an alle Mitglieder der Prozessgruppe (d.h. Prozesse, die vom momentanen "login" herruehren,) das Signal gesendet.

Die abgebrochenen Prozesse muessen dem jeweiligen Nutzer gehoeren, falls er nicht der Super-User ist.

Der Prozessidentifikator eines Hintergrundprozesses, (mit '&' gestartet), wird durch Shell ausgegeben. Der Prozessidentifikator kann auch mit ps(1) ermittelt werden.

SIEHE AUCH

ps(1), kill(2), signal(2)

LD(1) **LD(1)**

NAME

ld - Lader

UEBERSICHT

ld [option] file ...

BESCHREIBUNG

Ld verbindet mehrere Objektprogramme zu einem, loest externe Referenzen auf und durchsucht Bibliotheken. Im einfachsten Fall sind mehrere Objekt- files gegeben und ld verbindet sie zu einem Objektmodul. Dieser ist entweder der ausfuehrbar oder Eingabemodul fuer einen weiteren ld- Lauf. (Im zweiten Fall muss die -r Option angegeben werden, um die Relocations-Bits zu retten.) Ld gibt auf das File **a.out** aus. Dieses File wird nur dann ausfuehrbar gemacht, wenn keine Fehler waehrend des Ladens auftraten.

Die als Argumente angegebenen Routinen werden in der angegebenen Reihenfolge verknuepft. Der Eintrittspunkt fuer den Objektmodul ist der Beginn der ersten Routine.

Ist ein Argument eine Bibliothek, so wird sie genau einmal an der Stelle gesucht, an der sie in der Argumentliste aufgetreten ist. Nur jene Routinen, die eine unaufgeloeste externe Referenz definieren, werden geladen.

Ruft eine Bibliotheksroutine eine andere Routine aus derselben Bibliothek auf, so muss die aufgerufene hinter der aufrufenden Routine stehen.

Die Symbole '_etext', '_edata' und '_end' ('etext', 'edata' und 'end' in C) sind reserviert. Wird auf sie Bezug genommen, werden sie vor das Programm eingefuegt.

Ld versteht mehrere Optionen. Ausser -l sollten sie vor den Filenamen erscheinen.

-s Reduziert den Objektmodul, indem die Symboltabelle und die Relocation-Bits geloescht werden. (Dies verschlechtert aber das Handhaben des Debuggers.) Diese Informationen koennen auch durch strip(1) geloescht werden.

-u Das folgende Argument wird als Symbol verwendet und undefiniert in die Symboltabelle eingetragen. Dies ist ausschliesslich fuer das Laden einer Bibliothek nuetzlich, da anfangs die Symboltabelle in solch einem Fall leer ist. Mit Hilfe der unaufgeloesten Referenz wird das Laden der ersten Bibliotheksroutine realisiert.

-lx Diese Option ist eine Abkuerzung fuer 'lib/libx.a', wobei x eine Zeichenkette darstellt. Eine Bibliothek wird durchsucht, wenn ihr Name auftritt. Daher ist die Stelle, an der -l auftritt, entscheidend.

-x Loescht lokale (nicht-globale) Symbole in der Symboltabelle des Objektmoduls; nur externe Symbole werden eingetragen. Diese Option spart etwas Speicherplatz ein.

-X Loescht alle lokalen Symbole, die mit 'L' beginnen. Diese Option wird von cc(1) verwendet, um intern generierte Marken zu beseitigen, wobei lokale Symbole der Routinen beibehalten werden.

-r Generiert Relocation-Bits in den Objektmodul, so dass dieser von einem anderen ld -Lauf verwendet werden kann. Diese Option schuetzt auch davor, dass gemeinsamen Symbolen endgueltige Definitionen gegeben werden und unterdrueckt die 'undefined symbol'-Fehlerausschrift.

-d Fordert die Definition von gemeinsamem Speicher, auch wenn die -r -Option angegeben wurde.

-o Das Argument nach -o wird vom ld als Ausgabedatei anstelle von a.out verwendet.

-e Der folgende Name ist der Eintrittspunkt des geladenen Programms. Standard ist 0.

-D Das folgende Argument ist eine Dezimalzahl, die die Groesse des Datensegmentes festlegt.

FILES

/lib/lib*.a	Bibliotheken
/usr/lib/lib*.a	weitere Bibliotheken
a.out	Ausgabefile

SIEHE AUCH

as(1), ar(1), cc(1), ranlib(1)

FEHLERQUELLEN

Text oder Daten, die 64K Byte uebersteigen, werden nicht als Fehler gemeldet.

LN(1) **LN(1)**

NAME

ln - Erzeugen eines Links

UEBERSICHT

ln name1 [name2]

BESCHREIBUNG

Ein Link ist eine Directoryeintragung, die sich auf ein File bezieht. Ein File, dessen Eigeneschaften im i-node beschrieben sind, kann mehrere Links aus verschiedenen Directories haben. Es kann nicht unterschieden werden, welcher Link zu diesem File von seinem urspruenglichen Directory stammt.

Ln erzeugt ein Link zu einem existierenden File name1. Ist name2 angegeben, erhaelt der Link diesen Namen. Sonst wird er in das aktuelle Arbeitsdirectory eingetragen, der Name wird die letzte Komponente von name1.
Der Linkzaehler im i-node des Files wird incrementiert.

SIEHE AUCH

rm(1), Link(2)

FEHLERQUELLEN

Links zu einem Directory oder ueber Filesysteme hinaus sind nicht erlaubt.

NAME

login - Eintragen in das System

UEBERSICHT

login [Nutzername]

BESCHREIBUNG

Das login-Kommando wird benutzt, wenn sich ein Nutzer zu Beginn seiner Arbeit in das System eintragen will oder wenn sich waehrend der Arbeit ein anderer Nutzer anmelden moechte.

Wird login ohne Argument aufgerufen, so wird nachfolgend die Eingabe eines Nutzernamens und, wenn erforderlich, eines Passwortes gefordert. Waehrend der Eingabe des Passwortes wird der Echobetrieb ausgeschaltet, so dass die notwendige Geheimhaltung gewaehrleistet ist.

Entsprechend den Angaben im Passwort-File werden mit login: Nutzer und Gruppen-ID und das aktuelle (HOME) Arbeitsdirectory initialisiert sowie der eingetragene Kommandointerpreter (i.a. sh(1)) gestartet.

Nach einem erfolgreichen Login werden die Files zur Buchhaltung aktualisiert. Der Nutzer wird ueber in seinem Nachrichten-File eingegangene Nachrichten informiert. Ausserdem werden die Meldungen aus dem 'Nachricht-des-Tages'-File (/etc/motd) ausgegeben. Diese Leistungen werden nur dann ausgefuehrt, wenn die entsprechenden Files vorhanden sind.

Login wird durch sh(1) direkt ausgefuehrt, d.h. ohne einen neuen Prozess zu schaffen.

Die Abmeldung erfolgt mit (CNTRL)("Z").

FILES

/etc/utmp	Buchhaltung-Login
/usr/adm/wtmp	Buchhaltung-Login, Logout
/usr/mail/*	Nachrichten-Files
/etc/motd	Nachricht-des-Tages
/etc/passwd	Password-File

SIEHE AUCH

init(8), newgrp(1), getty(8), mail(1), passwd(1), passwd(5)

FEHLERMELDUNGEN

'login:', falls der eingegebene Name oder das Passwort falsch waren.

'No Shell', 'cannot open password file', 'no directory': Ein verantwortlicher Systemprogrammierer sollte konsultiert werden.

NAME

ls - Listen Directories

UEBERSICHT

ls [- ltasdrucifg] file ...

BESCHREIBUNG

Ls listet den Inhalt aller angegebenen Directories aus. Wurde kein File angegeben, so wird das aktuelle Arbeits-Directory verwendet. Ist file kein Directory, liefert ls die gleichen Informationen wie fuer ein Directory. Werden Files und Directories gemischt angegeben, werden Files vor Directories gelistet.
Die Ausgabe erfolgt standardmaessig als Liste der Filenamen, die in den Directories enthalten sind.

Es gibt folgende Optionen:

-l Listet im langen Format und gibt die Zugriffsrechte, die Anzahl der "links", den Eigentuemer, die Laenge in Byts und den Zeitpunkt der letzten Aenderung fuer jedes File aus. Fuer ein Special File wird statt der Laenge die Major bzw. Minor Geraetenummer angegeben.

-t Sortiert nach dem Zeitpunkt der Aenderung (die Letzte zuerst), anstatt standardmaessig nach Namen.

-a Listet alle Eintragungen, auch '.' und '..' (sonst standardmaessig unterdrueckt).

-s Gibt fuer jede Eintragung die belegte Blockanzahl an, einschliesslich der indirekten Bloecke.

-d Listet das Directory selbst, nicht dessen Inhalt (wird meist zusammen mit -l benutzt, um den Status eines Directory anzuzeigen).

-r Kehrt die Sortierreihenfolge um, alphabetisch rueckwaerts oder die aelteste zuerst.

-u Benutzt den Zeitpunkt des letzten Zugriffs, anstatt der letzten Aenderung fuer das Sortieren (-t) oder Drucken (-l).

-m multiple files in a column

-R Recursive (**) Kataloge weiter durchsuchen

-c Benutzt den Zeitpunkt der letzten Modifikation des i-node (Zugriffsrechte, usw.), anstelle der letzten Aenderung des Files fuer das Sortieren (-t) oder Drucken (-l).

-i Die i-node-Nummer fuer jedes gelistete File wird in der ersten Spalte des Protokolles gedruckt.

-f Erzwingt, dass jedes File als ein Directory interpretiert wird und listet die gefundenen Namen. Diese Option schaltet -l, -t, -s sowie -r aus und -a ein.
Die Reihenfolge entspricht den Eintragungen eines Directory.

-g Gibt den Gruppen-ID anstelle des Eigentuemer-ID beim Listen im langen Format aus.

Die Option -l liefert Informationen in folgender Struktur:

```
-rw-ru-r--    1   root   64   NOV   27   13:22   file
drwxrux---    2   root   34   NOV   16   13:44   dir
```

Das erste Zeichen kennzeichnet

d ein Directory
b ein blockorientiertes Special File
c ein zeichenorientiertes Spezial File
- ein normales File.

Die naechsten 9 Zeichen zeigen in 3 Teilen die Zugriffsrechte fuer den Eigentuemer, die Gruppe und fuer alle anderen Nutzer an.
Die Zugriffsrechte sind:
r Lesen erlaubt
w Schreiben erlaubt
x Ausfuehrung erlaubt

Fuer ein Directory wird die Ausfuehrungserlaubnis als Erlaubnis zum Suchen in der Directory interpretiert.

Wird das Ausfuehrungsrecht des Nutzers mit s angegeben, so ist fuer das File der Modus "Setzen Eigentuemeridentifikation bei Ausfuerung des Files" gesetzt, analog gilt das auch fuer den Gruppen-ID.
Der Modus 1000 (Retten Textbild) wird mit dem Zeichen t auf der letzten Position x angegeben.

FILES

/etc/passwd, um den Nutzer-ID fuer -l bereitzustellen
/etc/group, um den Gruppen-ID fuer -g bereitzustellen

SIEHE AUCH

chmod(1),mkdir(1),umask(2),filesys(5)

NAME

mkdir - Erzeugen eines Directory

UEBERSICHT

mkdir dirname ...

BESCHREIBUNG

Mkdir erzeugt die angegebenen Directories mit dem Modus 777. Die Standardeintrittspunkte '.' fuer das Directory selbst und '..' fuer sein Parent-Directory werden automatisch erzeugt.

Mkdir verlangt Schreiberlaubnis im Parent-Directory.

SIEHE AUCH

rm(1)

DIAGNOSTIK

Mkdir gibt den Rueckkehrcode 0 zurueck, wenn alle Directories erfolgreich erzeugt wurden. Anderenfalls druckt es einen Fehlerhinweis und gibt einen Wert ungleich 0 zurueck.

NAME

mkfs - Aufbau eines Filesystems

UEBERSICHT

/etc/mkfs special proto [m n] [x]

BESCHREIBUNG

Mkfs erstellt ein Filesystem auf einem Special File special, entsprechend den Spezifikationen im Prototyp File proto oder nach Standard. Die Angaben im Prototyp File werden durch Leerzeichen oder 'Newlines' getrennt.
Die erste Angabe ist der Name des Files, welches als Bootblock auf den ersten Block kopiert wird (nicht unterstuetzt vom MUTOS 8000).
Die zweite Angabe ist die Blockzahl des zu erstellenden Filesystems. Normalerweise ist das die Anzahl der Bloecke auf dem Geraet, moeglicherweise vermindert um den Platz fuer den swap-Bereich.
Die naechste Angabe ist die Anzahl der i-nodes in der i-node-Liste, vermindert um eins.

Der naechste Satz von Angaben enthaelt die Spezifikation fuer das root File und weitere zu erzeugende Files.

Die Angaben bestehen aus:

- Filenamen,
- den Zugriffsrechten,
- Nutzer-ID und Gruppen-ID und
- Anfangsinhalt des Files.

Als erstes wird das root File ohne Namen spezifiziert.
Die Angabe der Zugriffsrechte fuer ein File besteht aus einer 6 Zeichen langen Zeichenkette.
Das erste Zeichen spezifiziert den Typ des Files. (Die Zeichen -bcd spezifizieren regulaere, blockorentierte, zeichenorentierte Files bzw. Directories).
Das zweite Zeichen ist entweder u oder -, um das Set-User-ID Zugriffsrecht zu spezifizieren oder nicht.
Das Dritte ist g oder -, fuer das Set-Group-ID Zugriffsrecht.
Der Rest der Zugriffsrechte ist eine dreistellige Oktalzahl, die fuer Eigentuemer, Gruppe und andere Nutzer die Lese-, Schreib- und Ausfuehrungsrechte festlegt. Siehe auch chmod(1).

Zwei dezimale Zahlen folgen nach der Angabe der Zugriffsrechte. Sie spezifizieren den Nutzer- bzw. Gruppen ID fuer den Eigentuemer des Files.
Ist das File ein regulaeres File, dann ist die naechste Angabe ein Pfadname, von dem Inhalt und Groesse kopiert werden.

Ist das File ein block- bzw. zeichenorentiertes Special File, folgen zwei dezimale Zahlen als Angaben fuer die Major bzw. Minor Geraetenummer.
Ist das File ein Directory, dann erzeugt mkfs die Eintragungen '.' und '..' sowie die Eintragungen fuer weitere, nachfolgend spezifizierte Files. Die Liste der Filespezifikationen jedes Directory wird mit "*" abgeschlossen.

An Stelle des Prototyp-Files, kann auch eine Dezimalzahl angegeben werden, die der Blockzahl auf dem Filesystem entspricht. Dann legt mkfs ein Filesystem mit einer einfachen leeren Directory an.

Die Anzahl der i-nodes wird als Funktion der Groesse des Filesystems berechnet. Der Bootblock bleibt leer.

Beispiel fuer die Prototyp Specifikation:

```
/rkboot
4872 55
d--777 3 1
usr   d--777 3 1
      sh    ---755 3 1 /bin/sh
      ken   d--755 6 1
            $
      b0    b--644 3 1 0 0
      c0    c--644 3 1 0 0
      $
$
```

Die optionalen Argumente m und n spezifizieren den "interleave" Faktor (siehe filsys(5)); Standard ist m = 3 und n = 500.

Ist die Gesamtanzahl der Argumente 6, dann wird auf jeden Block des Filesystems ein Leseversuch unternommen, bevor er in die "Frei"-Liste eingetragen wird. Dadurch wird Mkfs sehr langsam.
Bloecke mit Lesefehler werden in den i-node 1 geschrieben.

SIEHE AUCH
filsys(5), dir(5), boot(8)

MKNOD(1M) **MKNOD(1M)**

NAME
mknod - Anlegen eines Special Files

UEBERSICHT
/etc/mknod name [c] [b] major minor

BESCHREIBUNG
Mknod erstellt ein Special File. Das erste Argument ist der Name name des Files. Die Option b kennzeichnet ein blockorientiertes Geraet (z.B. Diskette, Band); c ein zeichenorientiertes Geraet (z.B. Terminal).
Die letzten beiden Argumente sind Zahlen, welche die major (Geraetetyp) und die minor (Laufwerksnummer) beinhalten. Die Reihenfolge der Major Geraetenummern ist fuer jedes System spezifisch. Die Beschreibung erfolgt im Abschnitt 4 des Systemhandbuches Teil I.

SIEHE AUCH
mknod(2)

NAME

mount, umount - Eingliedern oder Entfernen eines Filesystems

UEBERSICHT

/etc/mount [special name [-r]]

/etc/umount special

BESCHREIBUNG

Mount wird zum Erweitern des Wurzel-Filesystems root benutzt.

Mount meldet dem System, dass ein wechselbares Filesystem auf dem Geraet special anliegt. Das File name muss immer existieren und ein Directory sein.
Die Directory name bezieht sich jetzt auf das Root-File des neu eingegliederten Filesystems. Der urspruengliche Inhalt des Directory ist nicht mehr zugaenglich, solange die mount-Verbindung besteht.
Das letzte wahlweise Argument bewirkt, dass das Filesystem als "nur lesbar" montiert wird.

Umount meldet dem System, dass ein wechselbares Filesystem, das vorher an das Geraet special angegliedert war, entfernt werden soll. Der urspruengliche Inhalt der Directory name wird wieder zugaenglich.

Diese Kommandos erzeugen eine Tabelle der montierten Geraete.
Wird mount ohne Argumente aufgerufen, so wird diese Tabelle ausgelistet.

Physisch schreibgeschuetzte und Magnetband-Filesysteme muessen als "nur lesbar" angegliedert werden.

FILES

/etc/mtab: Mount Tabelle

SIEHE AUCH

mount(2), mtab(5)

FEHLERQUELLEN

Angegliederte Filesysteme, die voellig defekt sind, koennen zum Absturz des Systems fuehren.
Montiert man an ein Nicht-Directory an, so werden einige scheinbar richtige Pfadnamen zerstoert.

NAME

mv - Uebertragen oder Umbenennen von Files und Directories

UEBERSICHT

```
mv file1 file2
mv directory1 directory2
mv file ... directory
```

BESCHREIBUNG

Mv uebertraegt file1 nach file2. Beide Files koennen auch Directories sein. Existiert file2 schon, wird es zurueckgesetzt, bevor file1 uebertragen wird.
Hat file2 keine Schreiberlaubnis, druckt mv den Modus und fordert eine Eingabe.
Bei 'y' findet die Uebertragung statt, andernfalls endet mv. Durch die Option-f wird die Frage unterdrueckt.

File1 wird in file2 umbenannt, wenn sich beide im gleichen Directory befinden.

Befinden sich file1 und file2 auf verschiedenen Filesystemen, wird nach dem Kopieren file1 geloescht (auch alle Links darauf). File2 erhaelt den Eigentuemernamen des rufenden Prozesses.

Beide Files koennen auch Directories sein, dabei wird der Inhalt des Directory1 mit gleichen Filenamen uebernommen.

Im dritten Fall werden ein oder mehrere files nach directory mit ihren urspruenglichen Filenamen uebertragen.

SIEHE AUCH

cp(1), chmod(1)

FEHLERQUELLEN

Mv auf das gleiche File ist nicht erlaubt.

NAME

ncheck - Ermitteln von Pfadnamen aus i-node-Nummern

UEBERSICHT

ncheck [-i numbers] [-a] [-s] [filesystem]

BESCHREIBUNG

ncheck ohne Optionen gibt fuer alle i-nodes eines Filesystems die Liste der zugehoerigen Pfadnamen aus. Directories sind durch den Anhang '/.' gekennzeichnet. Die -i Option begrenzt die Liste auf die angegebenen i-Nummern.
Die -a Option bewirkt, dass auch die Namen '.' und '..' , die normalerweise unterdrueckt werden, in der Liste erscheinen.
Die -s Option beschraenkt die Liste auf Spezial-Files und Files mit "set-user-ID"-Modus. Sie dient dazu, verborgene Verletzungen der Sicherheitsregeln zu ermitteln.(Sind solche Files nicht enthalten, wirkt ncheck wie ohne Option -s).

Die Liste erscheint ungeordnet. Fuer eine sinnvolle Nutzung sollte sie geordnet werden.

SIEHE AUCH

dcheck(1), icheck(1), sort(1)

DIAGNOSTIK

Ist die Struktur des Filesystems nicht in Ordnung, koennen folgende Meldungen auftreten:

'??' bedeutet, dass ein File kein uebergeordnetes Directory hat.

'...' am Anfang eines Pfadnamens weist auf eine Schleife hin.

NAME

nm - Ausgabe Symboltabelle

UEBERSICHT

nm [-gnopru] [file ...]

BESCHREIBUNG

Nm druckt die Symboltabelle von jedem Objekt file in der Argumentliste. Ist ein Argument ein Archiv, so wird eine Liste fuer jedes Objektfile erzeugt. Wird kein file angegeben, so wird die Symboletabelle von 'a.out' ausgegeben.

Jeder Symbolname wird durch seinen Wert (Leerzeichen, falls undefiniert) und einen der Buchstaben **U** (undefiniert), **A** (absolut), **T** (Textsegmentsymbol), **D** (Datensegmentsymbol), **B** (Bss-Segmentsymbol) oder **C** (Common Symbol) dargestellt. Objektfilenamen sind durch **F** gekennzeichnet.
Ist das Symbol lokal (nicht extern), wird ein kleiner Buchstabe ausgegeben. Die Ausgabe wird alphabetisch sortiert.

Die Options sind:

-g Gibt nur globale (externe) Symbole aus.

-n Sortiert numerisch anstatt alphabetisch.

-o Schreibt vor jede Ausgabezeile den File- oder Archivelementnamen.

-p Es wird nicht sortiert, sondern in Symbol-Tabellen-Reihenfolge ausgegeben.

-r Sortiert in umgekehrter Reihenfolge.

-u Gibt nur undefinierte Symbole aus.

SIEHE AUCH

ar(1), ar(5), a.out(5)

NAME

nroff-Textformatierer fuer Drucker

UEBERSICHT

nroff [options][files]

optionen:

-olist	Drucken eines Ausschnittes der entsprechend angegebenen Liste (list) list = N-M N = Anfangsseite, M = Endeseite -N Vom Anfang der Datei bis Seite N N- Ab Seite N bis Ende der Datei N,N,N Nur die Seiten N,N,N ausgeben
-nN	Die Seitennummer wird auf N gesetzt (1. Seite beginnt mit N) Standard ist 1
-sN	Stop des Druckes nach jeweils N Seiten (Standard =1), Fortsetzung mit ENTER-Taste
-mname	Aufruf des unter name verwendeten Makropaketes unter /usr/lib/tmac.name
-raN	Setzen des numerischen Registers a auf den Wert von N
-i	Lesen von Standard-Input, nachdem alle Dateien gedruckt sind
-q	Simultanes Lesen und Schreiben bei einem .rd-Kommando
-Tname	Verwenden der unter name zugeordneten Druckertabelle fuer die Ausgabe (1152, 1157)
-e	Ausgabe mit feinstmoeglicher Aufloesung ausgleichen (stufenlose Aufteilung der Zwischenraeume zwischen den Worten)

-h Ausgabe von Tabulatorzeichen zur Platzersparnis anstelle einer Folge von Space

BESCHREIBUNG

Der Aufruf eines Formatierers kann z.B. folgende Gestalt haben:

```
nroff -o4,8-10 -T1152 -mabc file1 file2
```

Dieser Aufruf bedeutet:

Formatieren und Ausgabe der Seiten 4,8,9 und 10 der Dokumente mit den Namen file1 und file2 unter Verwendung der Druckertabelle 1152 (Drucker SD 1152 SIO-Interface) und dem Makropaket abc auf Standard-Output.

KOMMANDOLISTE FUER NROFF

Aufruf	Standard	Bezug	Basis	Erlaeuterung
Schriftart und Zeichenabstand				
.ft F	Roman	vorhg.	E	Aendern Schriftart F = x, xx, oder 1-4 [\fx, \f(xx, \fN]
.uf F	I	I		Setzen der Schriftart unterstreichen fuer die Schriftart F, F=2,3.

Seitenformat

.pl +/- N	11 inch	11 inch	v	Papierlaenge in Zeilen
.bp +/- N	N = 1	-	B,v	Seitenvorschub und Setzen Seitennummer auf den Wert von N
.pn +/- N	N = 1	ignor.	-	Naechste Seitennummer auf den Wert N setzen
.po +/- N	0;26/27inch	vorhg.	v	Zu addierende Seitenverschiebung fuer den linken Rand einer Seite
.ne N	-	N = 1V	D,v	Einstellen benoetigter Zeilenabstand (V = Zeilenabstand)
.mk R	keine	intern	D	Markieren der aktuellen vertikalen Position im Register R
.rt +/- N	keine	intern	D,v	Zurueck (nur hochwaerts) zur markierten vertikalen Position (fuer Spaltenarbeit)

Textgestaltung, Randausgleich und Zentrieren

.br	-	-	B	Break, neue Zeile
.fi	fill	-	B,E	Fuellen der Ausgabezeile bis zum rechten Rand aus den Eingabezeilen ist eingeschaltet
.nf	fill	-	B,E	Kein Fuellen oder Randausgleich der Ausgabezeile (Eingabezeile entspricht Ausgabezeile)

.ad c	Rand	Rand	E	Randausgleich der Ausgabezeile nach dem mode c c=f nur linker Rand r nur rechter " c Zentrieren b oder n fuer Spaltenausrichten
.na	Rand	-	E	Kein Randausgleich der Ausgabezeile
.ce N	aus	N = 1	B,E	Zentrieren der folgenden N Texteingabezeilen

Zeilenabstaende

.vs N	1/6inch 12points	vorherg.	E,p	Basiszeilenabstand in (V)
.ls N	N = 1	vorherg.	E	Ausgabe N-1 V's nach jeder Ausgabezeile
.sp N	-	N = 1V	B,v	Ausgabe von N Leerzeilen
.sv N	-	N = 1V	v	Speichern Anzahl Leerzeilen (N)
.os	-	-	-	Ausgabe gespeicherter Leerzeilen
.ns	space	-	D	Einschalten "kein Zwischenraummodus"
.rs	-	-	D	Ausschalten "kein Zwischenraummodus"

Zeilenlaenge und Einruecken

.ll +/- N	6,5inch	vorherg.	E,m	Definition der Zeilenlaenge (Anzahl Zeichen pro Zeile)

.in +/- N	N = 0	vorherg.	B,E,m	Einruecktiefe des linken Randes
.ti +/- N	-	ignor.	B,E,m	Temporaere Einrucktiefe des linken Randes

Makros, Zeichenketten, Diversionen und IAB-Positionieren

.de xx yy	-	yy = ..	-	Definition oder Redefinition des Makros xx; Ende des Aufrufes von yy
.am xx yy	-	yy = ..	-	Anfuegen des Makros xx an den Makro yy
.ds xx string	-	ignor.	-	Definition einer Zeichenkette, die in xx abgelegt ist
.as xx string	-	ignor.	-	Anfuegen Zeichenkette an Zeichenkette, die in xx abgelegt ist
.rm xx	-	ignor.	-	Loeschen Makro oder Zeichenkette unter dem Namen xx
.rn xx yy	-	ignor.	-	Umbenennen Makro oder Zeichenkette xx in yy
.di xx	-	ende	D	Diversion nach den Makro xx beginnen
.da xx	-	ende	D	Anfuegen der Ausgabe an bestehende Diversion xx
.wh N xx	-	-	v	Aufruf des Makros xx bei Erreichen einer bestimmten Zeile auf der Seite (N) N vom Beginn der Seite -N vom Ende der Seite
.ch xx N	-	-	v	Aendern der Zeilennummer fuer Aufruf von .wh

.dt N xx	-	aus	D,v	Setzen einer Diversion bei Erreichen einer bestimmten Zeilenanzahl auf der Seite (N)
.it N xx	-	aus	E	Aufruf des Makros xx bei der erreichten Anzahl von Eingabezeilen (N)
.em xx	keine	keine	-	Ende des Makros xx
..	-	-	-	Ende einer Makrodefinition

Numerische Register

.nr R +/- N M	-		u	Definition und Setzen des Registers R auf den Wert von N, bei Angabe M automatisches Inkrement des Registers beim Aufruf
.af R c	arabisch	-	-	Zuweisen des Formates des Registers R c = 1 | 001 | i |s.Tabl. I |TAB 1 a | A |
.rr R	-	-	-	Loeschen des Registers R

Tabulatoren, Einfuerungszeichen und Felder

.ta N t...	0,8;Q,5inch	keine	E,m	Setzen von Tabulatoren des Types links, wenn t nicht angegeben ist. t = R rechter Typ t = C zentrierter Typ
.tc c	keine	keine	E	Tabulatorzeichen ist c
.lc c		keine	E	Kommandoeinfueh rungszeichen; Standard ist der Punkt

.fc a b	aus	aus	-	Setzen Feldbegrenzer a und Pufferzeichen b

Ein- und Ausgabeverschriften und Zeichenwandlungen

.ec c	\	\	-	Setzen ESC - Zeichen auf das Zeichen c
.eo	ein	-	-	Umschalten der Wirkung der ESC-Mechanismen
.ul N	aus	N = 1	E	Unterstreichen ein Wort
.cu N	aus	N = 1	E	Kontinuierliches Unterstreichen
.uf F	italic	italic	-	Schriftart Unterstreichen wird auf F gesetzt (ist zu setzen bei .ul)
.cc c	.	.	E	Setzen Zeilenumbruchzeichen auf c (Kommandoeinfuehrungszeichen)
.c2 c	'	'	E	Setzen Nichtzeilenumbruchzeichen auf c (sekundaeres Kommandoeinfuehrungszeichen)
.tr abcd...	keine	-	0	Codewandel a zu b; c zu d usw. und Ausgabe

Abteilung

.nh	abteilen	-	E	Kein automatisches Abteilen
.hy N	abteilen	abteilen	E	Abteilen im N-Modus
.hc c	\%	\%	E	Abteilungszeichen auf c einstellen und Ablegen im Register \%

.hw word1.	–	ignor.	–	word1. = Ausnahmewortliste fuer Abteilung

Dreiteilige Titel

.tl 'links'mitte'rechts'	–		–	Titel bestehend aus drei selbstaendigen Teilen, die in "'" eingeschlossen sind. Die "'" muessen immer komplett sein
.pc c	%	aus	–	Seitennummernzeichen fuer den Aufruf der Seiten-Nr. im Kopf bzw. Fuss
.lt +/- N	6,5inch	vorherg.	E,m	Einstellen der Titellaenge entsprechend N

Ausgabezeilennumerierung

.nm +/ N M S I		aus	E	Setzen des Numerierungsmodus ein / aus, Beginn mit den Anfangswert N
.nn N		N = 1	E	Kein Numerieren der naechsten N Zeilen

Bedingte Ausfuehrung von Anweisungen

.if c	Anweis.	–	–	Ist die Bedingung erfuellt, dann Ausfuehrung der Anweisung; fuer mehrzeilige Bedingungen: \{ Bedingung \}
.if !c	Anweis.	–	–	Ist die Bedingung nicht erfuellt, dann Anweisung ausfuehren
.if N	Anweis.	–	u	Ist das Ergebnis N > 0, dann Ausfuehrung

.if !N	Anweis.	-	u	Ist das Ergebnis N </= 0, dann Ausfuehrung
.if 'string1'string2'Anweis.			-	Ist string1 enthalten in string2, dann Ausfuehrung
.if !'string1'string2'Anweis.			-	Ist string1 nicht in string2 enthalten, dann Ausfuehrung
.ie c	Anweis.	-	-	Ausfuehren des if-Teiles einer if-else Anweisung, wenn die Bedingung c erfuellt ist

Kommandoliste fuer nroff

Aufruf	Standard	Bezug	Basis	Erlaeuterung
.el	Anweis.	-	-	Ausfuehren des else-Teiles einer if-else Anweisung, wenn die Bedingung nicht erfuellt ist.
Umgebungsumschaltung				
.ev N	N = 0	vorherg.	-	Umschalten auf eine andere Umgebung (abwaerts kellern)
Einfuegungen von Standard-input				
.rd Aufforderungsz.		Auff.Z.=Bell		Eingabe des Textes von der Tastatur
.ex	-	-	-	Abbruch von nroff

Ein-/Ausgabeaenderung

.so name	-	-	-	Aendern der Quelldatei (abwaerts kellern)
.nx name	-	Ende	-	Naechste Datei aufrufen
.pi Progr.	-	-	-	Aufruf einer Pipeline Ausgabe wird zur Eingabe des angegebenen Programms

Verschiedenes/Sonstiges

.mc c N	-	aus	E,m	Setzen Randzeichen c im Abstand von N Zeilen
.tm string	-	CR	-	Ausgabe der Zeichenkette string auf den Bildschirm
.ig yy	-	yy = ..	-	Ignorieren aller Aufrufe von yy
.pm t	-	alle	-	Drucken aller bis zu diesem Zeitpunkt aufgerufenen Makronamen und deren Groesse. Ist t angegeben, dann nur die Groesse
.fl	-	-	B	Ausgabe des augenblicklichen Standes des Ausgabepuffers

Die Zeichen unter Basis haben folgende Bedeutung:

B Verursacht normalerweise einen Abbruch der Zeile (Break).
D Modus oder aequivalenter Parameter sind mit der jeweiligen Diversionsebene verbunden.
E Die Parameter sind Teil der aktuellen Umgebung.
O Muss sich auf die logische Ausgabe stuetzen.
P Der Modus muss sich ruhig, oder ferner im Effekt waehrend einer phyischen Ausgabe verhalten.
v,p,m,u Fest zugeordnete Parameter. Wenn nicht spezifiziert, werden sie ignoriert und Standardwerte angenommen.

CROSS - Referenz alphabtisch geordnet und ihre Standardwerte

ad	4	ce	4	ec	10	fp	2	it	7	na	4	os	5	rm	7	ta	9		
af	8	ch	7	el	16	ft	2	lc	9	ne	3	pc	14	rn	7	tc	9		
am	7	cs	2	em	7	hc	13	lg	10	nf	4	pi	19	rr	8	ti	6		
as	7	cu	10	eo	10	hw	13	li	10	nh	13	pl	3	rs	5	tl	14		
bd	2	da	7	ev	17	hy	13	ll	6	nm	15	pm	20	rt	3	tm	20		
bp	3	de	7	ex	18	ie	16	ls	5	nn	15	pn	3	so	19	tr	10		
br	4	di	7	fc	9	if	16	lt	14	nr	8	po	3	sp	5	uf	10		
c2	10	ds	7	fi	4	ig	20	mc	20	ns	5	ps	2	ss	2	ul	10		
cc	10	dt	7	fl	20	in	6	mk	3	nx	19	rd	18	sv	5	vs	5		
																wh	7		

Fest zugeordnete numerische Register

%	Aktuelle Seitennummer
dl	width-Funktion Maximum der Zeichen der letzten Diversion
dn	Groesste (vertikale Groesse) der letzten Diversion (in Zeilen)
dw	Aktueller Tag in der Woche
dy	Aktueller Tag im Monat
hp	Aktueller horizontaler Platz in der Eingabezeile
ln	Aktuelle Ausgabezeilen-Nr.
mo	Aktueller Monat
nl	Aktuelle vertikale Position nach dem letzten Textdruck in Zeilen
sb	Tiefe der Zeichenfolge unter der Basiszeile (Generiert mit der width-Funktion)
st	Abstand der Zeichenfolge von der Basiszeile (Generiert mit der width-Funktion)
yr	Letzten zwei Ziffern des aktuellen Jahres

Fest zugeordnete nur lesbare numerische Register

.A	Ist in TROFF 1 gesetzt, wenn eine -a Option angegeben wurde. Immer 1 in NROFF.
.H	Horizontaleinstellung in Basisgroessen
.T	Immer 0 in TROFF
.V	Vertikaleinstellung in Basisgroessen
.a	Anzahl Leerzeilen beim Aufruf von \x'N'
.c	Zeilennummer beim Lesen des aktuellen Eingabefiles
.d	Aktueller vertikaler Platz bei einer Ablenkung. Entspricht dem nl, wenn keine Ablenkung
.h	Textbasiszeile bei einer Flut von Markierungen auf der aktuellen Seite oder Ablenkung
.i	Aktuelle Zeilenlaenge
.n	Laenge des Textes der veraenderten Ausgabezeile
.o	Beinhaltet die aktuelle Verschiebung des Seiteneinrueckens
.p	Beinhaltet die aktuelle Seitenlaenge
.t	Distanz bis zum naechsten Trap
.u	Ist 1 gesetzt, wenn im Fuellmodus gearbeitet wird und ist 0, wenn im Nicht-Fuellmodus gearbeitet wird
.v	Aktueller Zeilenabstand
.w	Hoehe der geaenderten Zeichen

nroff hat eine Aufloesung von 240 units/inch
Horizontalaufloesung 1/2400 inch
Vertikalaufloesung 1/144 inch

nroff akzeptiert numerische Vorgaben. Ist s spezifiziert, so erfolgt die Angabe der Zeichengroessen in points und der Vertikalabstand in basis-units.

Es gilt folgende Masseneinheitentabelle

Scale Indikator	Masseneinheit	
i	inch (Zoll)	240
c	Zentimeter	240 x 50/127
P	Pica =1/6 inch	240/6
m	Em = S points	c
n	En = Em/2	c, Summe aus Em
p	point = 1/72 inch	240/72
u	Basis units	1
v	Zeilenabstand	v

Die angenommenen Werte sind 1/10 und 1/12 inch. Die aktuellen Zeichenhoehen sind bei nroff immer gleich, da es auch konstruierte Zeichen wie -) ((-) usw. gibt, die aber extra definiert sind.

Die Standardgroesse ist Ems fuer die horiziontal wirkenden Funktionen wie ll, in, ti, ta, lt, po, mc, \h und \I. Vs fuer die Vertikalgroesse der Funktionen pl, wh, ch, dt, sp, sv, ne, rt, \v, \x und \L.
p fuer vs.
u fuer nr, if und ie.
Alle anderen Funktionen ignorieren eine solche Typzuordnung.
Die Nummer N als Dezimalausdruck wird gerundet als Integer Zahl in Basis-points gewandelt.

Beispiel:

```
.sp  |3.2c        laesst beispielweise 3,2 cm Platz,
                  ehe die naechste Zeile ausgegeben
                  wird.
.ll  (4.25i+\nxP+3)/2u
                  Setzen der Zeilenlaenge auf I/2
                  aus der Summe von 4,24 inches + 2
                  Picas + 30 points.
```

Die Standardausgabeschriftart ist (R) Roman. Es gibt noch (I) Italic und (B) Bold. Physisch sind diese als 1, 2, 3 und 4 zugeordnet. Sie sind alle in einem Dokument anwendbar. Geaendert werden sie mit dem .ft-Kommando durch die Eingabe von \fx, \f(xx oder \fN .
x und xx repraesentieren Registernamen, die die entsprechende Schriftart enthalten. Der Parameter N stellt den physischen numerischen Wert der Schriftart dar. Die Standardeinstellung ist im Register .f (nur lesbar) eingestellt.
Der Zeichenabstand ist setzbar im Bereich 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 28, und 36. Das entspricht 1/12 inch bis 1/2 inch.
Eingestellt durch \sN oder \s+/-N (1 (/= N (/= 9), mit \s0 zurueck zur Standardeinstellung.

Tabelle fuer Registeraufrufe

Aufruffolge	Effekt	Einstellung
\nx	keiner	N
\n(xx	keiner	N
\n+x	x incrementiert bei M	N + M
\n-x	x decrementiert bei M	N - M
\n+(xx	xx incrementiert bei M	N + M
\n-(xx	xx decrementiert bei M	N - M

Tabelle 1 Zuweisung Format c im Register R

Format	Benummerungsfolge
1	0,1,2,3,4,5...
001	000,001,002,003,004,005...
i	0,i,ii,iii,iv,v...
I	0,I,II,III,IV,V...
a	0,a,b,c...z,aa,ab...zz,aaa...
A	0,A,B,C...Z,AA,AB...ZZ,AAA...

Die with-Funktion wird aufgerufen durch \w'string' und bestimmt die Laenge des Strings und setzt diesen Wert als numerischen Parameter in das entsprechende Kommando ein.

Beispiel: .i -\w'1'u bedeutet, ein relatives Einruecken um 2 Stellen weiter nach links zu setzen.

Fuer die bedingte Ausgabe gibt es noch zusaetlich folgende Bedingungen:

o Aktuelle Seiten-Nr. ist ungerade
e Aktuelle Seiten-Nr. ist gerade
n nroff wird als Formatierer verwendet

Groessenzuordnungen

vertikale lokale Einstellung	Effekt
\v'N'	verwendete Distanz N
\u	1/2 Zeile hoch
\d	1/2 Zeile tief

horizontale lokale Einstellung	Effekt
\h'N'	verwendete Distanz N
\(space)	Sonderspacegroesse als Abstand
\0	Ziffer als Abstand
\\|	ignoriert
\^	ignoriert

Zur Beachtung!

nroff gibt automatisch bei Ausgabe von . , ; zusaetzlich ein Leerzeichen aus.

NAME

od - Oktalausgabe von Files

UEBERSICHT

od [-bcdox] [file] [[+]offset[.][b]]

BESCHREIBUNG

Od gibt file in Formaten aus, die durch die angegebenen Argumente bestimmt werden koennen. Fehlt das erste Argument, gilt als Standard **-o**. Die Formatargumente sind:

b Interpretiere byteweise oktal.

c Interpretiere byteweise als ISO-Zeichen. Dabei erscheinen alle nicht darstellbaren Zeichen entweder in C-Schreibweise (null=\0, backspace=\b, formfeed=\f, newline=\n, return=\r, tab=\t) oder, wenn in C keine solche Schreibweise vorgesehen ist, als dreistellige Oktalzahl.

d Interpretiere Worte als Dezimalzahlen.

o Interpretiere Worte als Oktalzahlen.

x Interpretiere Worte als Hexadezimalzahlen.

Das Argument file gibt an, welches File in der angegebenen Form ausgegeben werden soll. Ist kein File angegeben, wird vom Standard-Input gelesen.

Das Argument offset gibt an, ab welcher Position im File mit der Ausgabe begonnen werden soll.
Normalerweise ist das eine Oktalzahl (Anzahl der zu ueberspringenden Bytes).
Ist die Zahl mit einem '.' abgeschlossen, wird sie als Dezimalzahl interpretiert.
Ist die angegebene Zahl durch **b** abgeschlossen, gibt sie die Anzahl der zu ueberspringenden 512-Byteblocke an.
Fehlt die Angabe offset, so wird als Standard 0 angenommen.
Wurde kein File als Argument angegeben, muss der Verschiebung ein '+' vorangestellt werden.

Die Ausgabe erfolgt bis zum Fileende.

SIEHE AUCH

hex(1),adb(1)

PASSWD(1)

NAME

passwd - Aendern des Nutzer-Passwortes

UEBERSICHT

passwd [Name]

BESCHREIBUNG

Mit diesem Kommando kann der Nutzer sein Passwort eintragen oder aendern. Ist Name nicht angegeben, wird der 'Login-Name' des aufrufenden Nutzers benutzt.

Das Programm fordert die Eingabe des alten Passwortes (falls vorhanden) und danach des neuen. Beide Eingaben muessen erfolgen. Das neue Passwort ist sogar zweimal einzugeben, um eventuelle Schreibfehler auszuschliessen.

Passworte muessen mindestens vier Zeichen lang sein, vorausgesetzt, sie enthalten einem genuegend umfangreichen Zeichenvorrat. Wird nur ein einfacher Zeichenvorrat benutzt (zum Beispiel nur Kleinbuchstaben), so muss die Laenge mindestens sechs Zeichen betragen. Der Nutzer kann diese Regeln jedoch umgehen, wenn er hartnaeckig genug dagegen verstoesst.

Nur der Nutzer oder der Superuser koennen das Passwort aendern. Die Kenntnis des alten Passwortes ist dazu notwendig.

FILES

/etc/passwd

SIEHE AUCH

login(1), passwd(5), crypt(3)

NAME

pr - Ausgabe eines Files

UEBERSICHT

pr [option] ... [file] ...

BESCHREIBUNG

Pr gibt ein oder mehrere files formatiert aus. Die Ausgabe erfolgt seitenweise. In der Kopfzeile jeder Seite erscheinen das Datum, der Name des Files oder eine extra spezifizierte Ueberschrift und die Seitennummer.
Ist kein File-Argument angegeben, so gibt pr die von Standard-Input gelesenen Eingaben aus.

Die Optionen werden auf alle nachfolgenden Files angewendet. Optionale Angaben zwischen den file-Argumenten sind moeglich.

-n Erzeugt eine n-spaltige Ausgabe.

+n Beginnt die Ausgabe mit der Seitennummer n.

-h Nimmt das naechste Argument als Seitenueberschrift.

-wn Fuer das Erzeugen einer mehrspaltigen Ausgabe wird eine Blattbreite von n Zeichen angenommen (standardmaessig 72).

-ln Nimmt als Blattlaenge n Zeilen statt der standardmaessigen 72 an.

-t Unterdrueckt die Ausgabe der 5-zeiligen Ueberschrift und des 5-zeiligen Nachsatzes, die sonst auf jeder Seite geliefert werden.

-sc Trennt die einzelnen Spalten durch das Zeichen c anstelle der standardmaessigen Leerzeichen. Fehlt die Angabe eines Zeichens c, wird stattdessen ein Tabulator verwendet.

-m Gibt alle files aus, jedes in eine Spalte.

SIEHE AUCH

cat(1)

FEHLERBEHANDLUNG

Es erfolgt keinerlei Warnung, falls pr auf ein Terminal ausgibt.

NAME

ps - Prozessor-Status

UEBERSICHT

ps [-x|Nutzername ...]

BESCHREIBUNG

Ps liefert Informationen ueber aktive Prozesse. Wird die -x -Option verwendet, so werden nur Informationen ueber solche Prozesse ausgegeben, die dem Terminal /dev/ttyx zugeordnet sind. Fuer das Geraet /dev/console ist -C zu spezifizieren. Alternativ zur -x -Option koennen ein oder mehrere Login-Namen angegeben werden, um Informationen von den Prozessen zu erhalten, die unter den Login-Namen zugeordneten, Nutzeridentifikatoren laufen. Ein Login-Name, dessen zugeordneter Nutzeridentifikator Null ist, also ein Login-Name fuer den Superuser, wird zurueckgewiesen. Wird ps ohne Argumente aufgerufen, so werden Informationen zu allen aktiven Prozessen ausgegeben.
Fuer jeden aktiven Prozess wird eine Ausgabezeile erzeugt. Die Ausgaben sind nach den die Prozesse steuernden Terminals sortiert.
Zu jedem Prozess werden folgende Informationen ausgegeben:

tty — Das steuernde Terminal fuer den Prozess bzw. % fuer den Scheduler-Prozess (Prozess Null).

owner — Der dem Terminal zugeordnete Login-Name. Diese Angabe wird fuer den Prozess Null und fuer Prozesse, die dem Superuser zugeornet sind, weggelassen.
Die Angabe des Login-Namens erfolgt nur fuer den Kopf-Prozess (process group leader) einer einem Terminal zugeordneten Prozess-Hierarchie. Weitere Prozesse einer solchen Prozessor-Gruppe werden durch ein oder mehrere) - Zeichen identifiziert. Die Zahl der) - Zeichen charakterisiert die Stellung des Prozesses in der Hierarchie.

pid — Der Prozessoridentifikator; dieser muss bekannt sein, um einen Prozess abbrechen zu koennen

status — Der Prozess-Status; RUN: laufend; idl: dazwischenliegend; zmb: beendet; stp: gestopt; slp:"schlafend" mit Prioritaet groesser Null; SLP: schlafend mit Prioritaet gleich Null (je groesser die Zahl, desto kleiner die Prioritaet).
Ein an den Status angefuegtes *-Zeichen zeigt an, dass der Prozess im Speicher resident ist. (Prozessor-Flag SLOAD).

Bei "schlafenden" Prozessen (Status slp oder SLP) wird das Ergebnis (die Adresse) hinzugefuegt, auf das der Prozess wartet:

proc[n]	Das n-te Element der "proc"-Tabelle (Das Zaehlen beginnt mit Ø).
kl11	Terminal - E/A.
trace	Beenden einer Debugger-Aktion.
I/O	Verfuegbarkeit einer "buf"-Struktur.
filesys	Beenden einer Block-E/A.
freebuf	Freiwerden eines 512-Byte-Puffers.
swap	Freiwerden eines Swap-"buf".
runin	Prozess, der eingelagert werden kann.
runout	Prozess, der ausgelagert werden kann.
pause	Eintreffen eines Signals.
clock	Ablauf einer bestimmten Zeitschleife.
raw I/O	Beenden von ungepufferter E/A.
inode	Verfuegbarkeit einer "inode"-Struktur.

Ist eine qualifizierte Angabe zum erwarteten Ergebnis nicht moeglich, wird die zum Ergebnis gehoerende Adresse in hexadezimaler Form ausgegeben.

size — Ø fuer den Schedulen Prozess, 64K fuer alle anderen Prozesse. Gibt nur an, ob der Prozess ein 64K- Byte-Segment des Speichers beansprucht.

cptime — Die vom Prozess schon in Anspruch genommene Prozessor-Zeit in Minuten und Sekunden. Die akkumulierte Prozessor-Zeit seit dem letzten Einlagern wird in der folgende Spalte mit den Zeiten anderer Prozesse verglichen (Angaben in Prozent).

command — Das zweite Argument des "excente"-Systemrufes, der zum Aufruf des aktuellen Programmes den Prozess gefuehrt hat, wird hier rekonstruiert. Zur Rekonstruktion wird der Nutzer-Stack des Prozessors (im Hauptspeicher oder auf den Swap-Bereich) herangezogen. Den Angaben ist nicht immer zu trauen.

Die Erlaeuterungen zum Status sind englisch, um die Bezuege zu den Generierungsquellen nicht zu verwischen.

FILES

/dev	Zum Bestimmen der Terminalnamen
/dev/mem	Hauptspeicher des Systems
/dev/swap	Zum Ermitteln des Kommandos
/etc/passwd	Zum Ermitteln der Login-Namen

SIEHE AUCH

kill(1)
pstat(1)

FEHLERQUELLEN

Waehrend ps abgearbeitet wird, koennen sich die Verhaeltnisse aendern; die Ausgabe ist eine Annaeherung an den tatsaechlichen Zustand.

NAME

pstat - Ausgabe von Systeminformationen

UEBERSICHT

pstat [-afiptu] [suboption]

BESCHREIBUNG

Pstat interpretiert den Inhalt verschiedener Systemtabellen. Die moeglichen Optionen sind:

-a In Verbindung mit -p, anzugeben. Es wird jeder Prozesstabellen-Eintrag ausgegeben, unabhaengig davon, ob er einen aktiven Prozess beschreibt oder nicht.

-f Ausgabe der Anzahl der eroeffneten Files und einer Tabelle der eroeffneten Files mit folgendem Kopf

LOC Die Speicheradresse des Eintrags (hexadezimal).

FLG Verschiedene Statusinformationen mit folgender Bedeutung:
R eroeffnet zum Lesen
W eroeffnet zum Schreiben
P Pipe

CNT Anzahl der 'Opens' fuer diesen File.

INOD Die Hauptspeicheradresse des i-node-Tabellen-Eintrags fuer dieses File.

OFFS Der File-Offset (siehe lseek(2)).

-i Ausgabe der Anzahl aktiver i-ndos und einer Tabelle dieser i-nodes mit folgendem Kopf:

LOC Die Adresse des Eintrags (hexadezimal)
FLAGS Verschiedene Statusvariable mit folgender Bedeutung:
L darf nicht ausgelagert werden
U Aktualisierungszeit filesys(5), die auf Diskette aktualisiert werden muss
A Zugriffszeit, die auf Diskette aktualisert werden muss
M hier ist ein Filesystem eingegliedert
W i-node wird von einem anderen Prozess benoetigt
T i-node enthaelt ein Text-File
CNT Anzahl der Eintraege in der Open-File-Tabelle fuer diesen i-node
DEVICE Geraetetyp und Laufwerksnummer des Filesystems, auf dem sich der i-node befindet
INO i-Nummer innerhalb des Geraetes
MODE Modusbits (hexadezimal), siehe chmod(2).
NLK Anzahl der Links zu diesem i-node.
UID Nutzeridentifikation des Fileeigentuemers
SIZ/DEV Anzahl der Bytes bei normalen Files oder Geraetetyp und Laufwerksnummer des Special-Files

-p Ausgabe der Anzahl der aktiven Prozesse und einer Tabelle der Eintraege fuer aktive/alle Prozesse mit folgendem Kopf:

LOC Die Adresse des Eintrags (hexadezimal)

S Prozess-Status. Folgende Werte sind moeglich:

0 Eintrag nicht benutzt
1 wartet auf ein Ereignis
3 abarbeitbar
4 im Stadium der Erzeugung
5 gerade beendet
6 gestoppt unter Testabarbeitung

F Verschiedene Statusvariable, von denen mehrere gleichzeitig gueltig sein koennen (hexadezimal)

01 im Hauptspeicher
02 Scheduler-Prozess
04 darf nicht ausgelagert werden
08 wird gerade ausgelagert (bei fork)
10 wird verfolgt (traced)
20 gerade beim Test
40 gesperrt durch lock(2)

PY Scheduler-Prioritaet, siehe nice(2)

SIG Empfangene Signale (die Signale 1-16 sind codiert in den Bit 0-15; Ausgabe hexadezimal)

UI Nutzer-ID

TI Zeit in Sekunden (hexadezimal) seit dem letzten Ein/Auslagern des Prozesses. Zeilen groesser 127 (0x7F) werden durch 127 angegeben.

CP Gewichte Summe der CPU-Zeit (hexadezimal)

NI Nice-Niveau (hexadezimal), siehe nice(2)

PRG Prozessidentifikator des Wurzelprozzesses (des eroeffnenden Prozesses am steuernden Terminal) dieses Prozesses

PI Der eindeutige Prozessidentifikator

PPI Der Prozessidentifikator des Parent-Prozesses

ADD Wenn hauptspeicherexident, die Adresse des Nutzersegments dividiert durch 256.
Wenn ausgelagert, die Start-Blocknummer im Swap - Bereich. Die Angaben erfolgen hexadezimal.

SIZE 0 beim Scheduler-Prozess, sonst 100 (Groesse des Prozessabbildes dividiert durch 64).

WCH Die dem Ereignis, auf das gewartet wird, zugeordnete Adresse (hexadezimal).

LINK Link-Zeiger in der Liste abarbeitbarer Prozesse bzw. in einer der Ereignisgruppe zugeordneten Liste

TEXT Immer Null.

CLKT Restzeit (hexadezimal) bis zum Eintreffen von alarm(2) in Sekunden

-t Ausgabe der Anzahl der Eintraege in der Terminal-Tabelle und einer Tabelle ausgewaehlter Informationen aus diesen Eintragungen mit folgenden Kopf:

\# Nummer des Eintrags in der Tabelle

RAW Anzahl der Zeichen in der Eingabeschlange

OUT Anzahl der Zeichen in der Ausgabeschlange

FLGS siehe tty(4)

ADDR Adresse (hexadezimal) des Terminals Ausgabe-Puffers

DEL Anzahl der Newlines in der Eingabeschlange des Terminals

COL Berechnete Spaltenposition des Terminals

STATE Verschiedene Statusvariable mit folgender Bedeutung :

W wartend auf Open
O Open
S hat eine spezielle (output) Startroutine
C Ueberbringer ist ein.
B beschaeftigt mit Ausgabe
A Prozess erwartet eine Ausgabe
X geoeffnet fuer die exklusive Benutzung
H Beendigung der Arbeit

PRG Prozess-Nummer des Wurzelprozesses der Prozess-Gruppe, fuer die dies ein steuerndes Terminal ist.

-u Ausgabe von Informationen zu auslagerbaren Systemdaten eines hauptspeicher-residenten Prozesses (Informationen aus der "user"-Struktur, siehe dort). Die Adresse der auslagerbaren Systemdaten ist als naechstes Argumment fuer pstat anzugeben (hexadezimal). Zulaessige Werte sind:

EC00 fuer den Scheduler-Prozess
F000 fuer den Prozess mit ADD-Wert 100 (im Prozess-Eintrag)
F400 fuer den Prozess mit ADD-Wert 200 (im Prozess-Eintrag)
F800 fuer den Prozess mit ADD-Wert 300 (im Prozess-Eintrag)
FC00 fuer den Prozess, in dem pstat laeuft.

FILES

/dev/mem fuer den Hauptspeicher

SIEHE AUCH

ps(1),stat(2),filsys(5)

NAME

pwd - Name des aktuellen Directory

UEBERSICHT

pwd

BESCHREIBUNG

Pwd schreibt den Pfadnamen des aktuellen Arbeitsdirectorys aus.

SIEHE AUCH

cd(1)

NAME

rm - Loeschen Files
rmdir - Loeschen Directories

UEBERSICHT

rm [-fri] file ...

rmdir dir ...

BESCHREIBUNG

Rm loescht die Directoryeintragung fuer ein oder mehrere Files in einem Directory. Gleichzeitig wird der Linkzaehler im i-node des Files decrementiert.
Wurde eine Eintragung der letzte Link zu einem File, wird das File zerstoert. Das Loeschen eines Files bedingt Schreiberlaubnis in seinem Directory, aber weder Lesenoch Schreiberlaubnis fuer das File selbst.

Hat ein File keine Schreiberlaubnis (und erfolgt der Standardeingang ueber Tastatur), werden seine Erlaubnisse gedruckt und eine Eingabe gefordert.
Bei 'y' wird das File geloescht, anderenfalls bleibt das File erhalten.
Wird die -f (force) Option angegeben, werden die Fragen unterdrueckt.
Ist das bezeichnete File ein Directory, wird ein Fehlerkommentar gedruckt, falls nicht das wahlweise Argument -r benutzt worden ist. In diesem Fall loescht rm rekursiv den gesamten Inhalt des angegebenen Directory und das Directory selbst.

Wird die -i (interactive) Option verwendet, fragt rm nochmals jedes File ab, ob es zu loeschen ist und ebenso jede Directory, wenn -r angegeben wurde.

Rmdir loescht (in den uebergeordneten Directories) die Eintragungen fuer die angegebenen Directories, die leer sein muessen.

SIEHE AUCH

unlink(2)

DIAGNOSTIK

Erklaert sich im allgemeinen selbst. Es ist verboten, das File '..' zu loeschen, nur um die Konsequenzen einer unbeabsichtigten Handlung, etwa wie 'rm -r .*', zu vermeiden.

NAME

sh, for, case, if, while, : , ., break, continue, cd, eval, exec, exit, export, login, newgrp, read, readonly, set, shift, times, trap, umask, wait - Kommandosprache

UEBERSICHT

sh [-ceiknrstuvx] [arg] ...

BESCHREIBUNG

Shell ist eine Kommando-Programmiersprache, mit deren Hilfe Kommandos ausgefuehrt werden koennen, die an einem Terminal eingegeben oder in einem File abgespeichert worden sind. Die Bedeutung der Argumente, die an Shell uebergeben werden koennen, wird unter der Ueberschrift **Aufruf** erlaeutert.

Kommandos

Ein einfaches Kommando ist eine Folge von Worten, die durch **Leerzeichen** oder **Tabulatoren** getrennt sind. Das erste dieser Worte legt den Namen des Kommandos, d.h. den Namen des Files, fest, das ausgefuehrt werden soll. Die verbleibenden Worte werden als Argumente an das aufgerufene Kommando (Programm) uebergeben. Der Kommandoname selbst wird als Argument 0 uebergeben (siehe exec(2)). Der Wert eines einfachen Kommandos ist sein Rueckkehrstatus, falls die Kommandoabarbeitung normal beendet wurde. Bei nichtnormalem Beendigen ist der Rueckkehrwert 200+Status (siehe signal(2) fuer eine Liste der Statuswerte).

Eine Pipeline ist eine Folge von einem oder mehreren Kommandos, die durch | getrennt sind. Der Standard-Output jedes der Kommandos (ausser des letzten) wird durch pipe(2) mit dem Standard-Input des nachfolgenden Kommandos verbunden. Jedes der Kommandos wird in einem separaten Prozess abgearbeitet. Shell wartet auf das Beendigen des letzten Kommandos, bevor es die Arbeit fortsetzt.

Eine Liste ist eine Folge aus einer oder mehreren Pipelines, die durch **;**, **&**, **&&** oder **||** getrennt sind und mit **;** oder **&** abgeschlossen werden koennen. Dabei haben **;** und **&** die gleiche Prioritaet, die niedriger ist als die Prioritaet von **&&** und **||**, die ebenfalls gleiche Prioritaet haben. Ein Semikolon bewirkt das sequentielle Ausfuehren der so getrennten Pipelines. Wird eine Pipeline mit "&" abgeschlossen, so wartet Shell nicht auf das Ende der Ausfuehrung dieser Pipeline, bevor Shell seine Arbeit fortsetzt.
Die Pipeline wird im 'Hintergrund' abgearbeitet. Das Symbol **&&** oder **||** bewirkt, dass die ihm folgende Liste nur abgearbeitet wird, falls die vorangegangene Pipeline einen Rueckkehrwert gleich Null bzw. ungleich Null lieferte. Ein Kommando kann in einer Liste statt mit einem Semikolon auch durch 'Newline', d.h. den Beginn einer neuen Zeile, begrenzt werden.

Ein Kommando ist entweder ein einfaches Kommando oder eine der folgenden Konstrukte. Der Rueckkehrwert eines Kommandos ist der des letzten einfachen Kommandos, das innerhalb des Kommandos ausgefuehrt wurde.

for Name [**in** Wort ...] **do** Liste **done**
Name wird nacheinander jedem der nach **in** angegebenen Worte gleichgesetzt. Fuer jede der Belegungen von Name wird einmal die Liste abgearbeitet. Die Ausfuehrung dieses Konstruktes ist beendet, wenn kein einzusetzendes Wort mehr vorhanden ist. Ist der Teil **in** Wort ... nicht angegeben, so wird statt dessen **in** "$@" angenommen.

case Wort **in** [Muster [| Muster] ...) Liste **;;**] ... **esac**
Durch dieses Kommando wird jene Liste abgearbeitet, die dem ersten Muster zugeordnet ist, das mit Wort uebereinstimmt. Die Form der Muster ist die gleiche, wie fuer die Filenamen-Generierung.

if Liste **then** Liste [**elif** Liste **then** Liste] ... [**else** Liste] **fi**
Die Liste nach **if** wird abgearbeitet. Falls sie den Rueckkehrwert Null liefert, wird die Liste nach **then** abgearbeitet. Andernfalls wird die Liste nach **elif** abgearbeitet, und falls diese einen Rueckkehrwert gleich Null hat, so erfolgt das Abarbeiten der zugehoerigen Liste nach **then**. Bei einem Rueckkehrwert ungleich Null wird die Liste nach **else** abgearbeitet.

while Liste [**do** Liste] **done**
Liefert das Abarbeiten der Liste nach **while** ein Rueckkehrwert gleich Null, so wird die Liste nach **do** abgearbeitet. Diese Schleife wird solange fortgesetzt, bis das Abarbeiten der **while**-Liste einen Rueckkehrwert ungleich Null liefert. Der Rueckkehrwert eines **while**-Kommandos ist der des zuletzt ausgefuehrten Kommandos der **do**-Liste. Anstelle von **while** kann **until** geschrieben werden, falls ein negierter Test zum Beenden der Schleife gewuenscht wird.

(Liste)
Die Liste wird in einem eigenen Unter-Shell-Prozess ausgefuehrt.

{ Liste }
Die Liste wird einfach ausgefuehrt.

Die folgenden Worte werden durch Shell nur dann erkannt, wenn sie als erstes Wort in einem Kommando auftreten und nicht apostrophiert sind.

if then else elif fi case in esac
for while until do done { }

Kommandosubstitution
Der Standard-Output eines Kommandos, das in Akzentzeichen eingeschlossen ist (`...`), kann als Wort oder als Teil eines Wortes benutzt werden. Die Ausgabe abschliessende 'Newlines' werden entfernt.

Parametersubstitution
Durch Voranstellen eines * vor den Namen eines Parameters wird an dieser Stelle der Wert des Parameters in das Kommando eingesetzt. Den Stellungsparametern koennen mit **set** Werte zugewiesen werden. Variablen koennen Werte auf folgende Arten zugewiesen werden:

Name=Wert [Name=Wert] ...

*{Parameter}
Ein Parameter ist eine Folge aus Buchstaben, Ziffern und Unterstrichen (ein Name), eine Ziffer oder eines der Zeichen * @ # ? - $!. Falls der Parameter einen Wert besitzt, so wird dieser substituiert. Die Klammern sind nur notwendig, wenn sich an den Parameter unmittelbar ein Buchstabe, eine Ziffer oder ein Unterstrich anschliessen, die nicht als Teil des Parameternamens interpretiert werden sollen. Ist Parameter eine Ziffer, dann handelt es sich um einen der Stellungsparameter. Ist fuer Parameter * oder @ angegeben, so werden dafuer alle Stellungsparameter, beginnend mit *1, substituiert, wobei zum Trennen zwischen den einzelnen Parametern Leerzeichen eingefuegt werden. *0 wird gleich dem nullten Argument beim Aufruf von Shell gesetzt.

*{Parameter-Wort}
Falls Parameter einen Wert hat, so wird dieser substituiert. Andernfalls wird Wort substituiert.

*{Parameter=Wort}
Falls Parameter noch kein Wert zugewiesen wurde, so wird ihm Wort als Wert zugewiesen. Danach wird der Wert des Parameters substituiert. Stellungsparameter koennen auf diese Weise nicht mit Werten belegt werden.

*{Parameter?Wort}
Falls Parameter einen Wert hat, so wird dieser substituiert. Andernfalls wird Wort ausgegeben und der Shell-Prozess beendet. Falls Wort nicht angegeben ist, wird statt dessen eine Standardmeldung ausgegeben.

*{Parameter+Wort}
Falls Parameter einen Wert hat, so wird Wort substituiert. Andernfalls wird nichts substituiert.

In allen angefuehrten Faellen wird Wort nur dann berechnet, wenn es als einzusetzende Zeichenkette gebraucht wird. (Das bedeutet zum Beispiel, dass in echo *{d-`pwd`} nur dann pwd ausgefuehrt wird, wenn die Variable d keinen Wert besitzt).

Die folgenden Parameter werden automatisch durch Shell gesetzt.

- **#** Die Anzahl der Stellungsparameter (dezimal).
- **-** Die Optionen, die bei Aufruf von Shell oder durch **set** eingestellt werden.
- **?** Der Rueckkehrwert des zuletzt ausgefuehrten Kommandos.
- **$** Der Prozessidentifikator des laufenden Shell-Prozesses.
- **!** Der Prozessidentifikator des zuletzt gestarteten Hintergrund-Kommandos.

Die folgenden Parameter werden von Shell benutzt, jedoch nicht gesetzt.

- **HOME** Das Standard-Argument (Home-Directory) des **cd**-Kommandos.
- **PATH** Der Suchpfad fuer Kommandos
- **MAIL** Falls diese Variable als Wert den Namen eines Files hat, so wird der Nutzer informiert, wenn eine Nachricht in diesem File abgelegt wurde.
- **PS1** Die primaere Eingabeanforderung (Promt-String); Standard ist '**$**'.
- **PS2** Die sekundaere Eingabeanforderung; Standard ist '> '.
- **IFS** Die internen Trennzeichen; Standard sind **Leerzeichen, Tabulator** und **Newline.**

Interpretation der Freiraeume
Nachdem die Parameter- und Kommandosubstitution ausgefuehrt sind, wird das entstandene Resultat nach eventuell vorhandenen internen Trennzeichen (wie sie in **$IFS** angegeben sind) durchsucht. Die Stellen, an denen solche Zeichen gefunden werden, markieren jeweils den Beginn eines neuen Argumentes.
Explizite Null-Argumente ("" oder '') bleiben erhalten, waehrend implizite Null-Argumente, die aus nichtgesetzten Parametern resultieren, geloescht werden.

Filenamen-Generierung
Nach den bisher geschilderten Ersetzungen wird jedes Wort des Kommandos nach den Zeichen *, ? und [durchsucht. Wird in einem Wort ein solches Zeichen gefunden, so wird dieses Wort als Muster aufgefasst und durch die alphabetisch sortierte Folge aller Filenamen ersetzt, die dem Muster entsprechen. Wird kein Filename gefunden, der dem vorgegebenen Muster entspricht, so wird dieses Muster unveraendert als Argument uebergeben. Das Zeichen . am Beginn eines Filenamens sowie unmittelbar nach einem / und das Zeichen / selbst muessen explizit angegeben werden.

* entspricht einer beliebigen Zeichenkette, schliesslich der Null-Zeichenkette.

? entspricht einem beliebigen einzelnen Zeichen.

[...]
entspricht jedem der in den eckigen Klammern angegebenen Zeichen. Ein Zeichenpaar, das durch - getrennt ist, entspricht allen Zeichen, die lexikalisch zwischen diesen beiden Zeichen liegen.

Apostrophieren

Die folgenden Zeichen haben eine besondere Bedeutung fuer Shell. Werden sie nicht apostrophiert, so bewirken sie das Beenden eines Wortes.

;	&	(	)	|	<	>
Newline		Leerzeichen			Tabulator	

Ein Zeichen kann durch Voranstellen von \ apostrophiert werden. Dadurch wird zum Beispiel **\Newline** von Shell ignoriert. Auch alle Zeichen, ausser einem Apostroph, die in ein Paar von Apostrophen (’’) eingeschlossen sind, gelten als apostrophiert und verlieren ihre besondere Bedeutung fuer Shell. Innerhalb von Anfuehrungsstrichen (“”) wird Parameter- und Kommandosubstitution durchgefuehrt. Durch \ koennen die Zeichen \ ` und ” apostrophiert werden.

“*” entspricht “*1 *2 ...” und
“*@” entspricht’ “*1” “*2” ... .

Eingabeanforderungen

Im interaktien Verkehr’ zeigt Shell seine Bereitschaft, Eingaben vom Nutzerterminal zu lesen, durch Ausgabe des Wertes von **PS1** auf dem Bildschirm an. Falls zu irgendeiner Zeit nach der Eingabe von ’Newline’ weitere Eingaben durch den Nutzer erforderlich sind, um ein Kommando korrekt zu beenden, so gibt Shell den Wert von **PS2** auf den Bildschirm aus.

Ein- und Ausgaben

Vor dem Ausfuehren eines Kommandos koennen dessen Ein- und Ausgaberichtung umgelagert werden. Dies wird durch eine spezielle Notation erreicht, die von Shell interpretiert und nicht als Argument an das zugehoerige Kommando weitergegeben wird. Die nachfolgend angegebenen Moeglichkeiten zur E/A-Umlagerung koennen an einer beliebigen Stelle in einem einfachen Kommando erscheinen oder einem Kommando voran- bzw. nachgestellt sein. Angegebene Substitutionen werden durchgefuehrt, bevor Wort oder Ziffer benutzt werden.

‹Wort
Das File Wort wird als Standard-Input (File-Deskriptor 0) benutzt.

›Wort
Das File Wort wird als Standard-Output (File-Deskriptor 1) benutzt. Falls dieses File noch nicht existiert, wird es angelegt. Andernfalls wird es auf die Laenge Null gebracht, d.h. der alte Inhalt ist verloren.

››Wort
Das File Wort wird als Standard-Output benutzt. Falls dieses File noch nicht existiert, so wird es erzeugt.
Andernfalls wird das File-Ende gesucht, so dass folgende Ausgaben an den alten Inhalt angefuegt werden.

‹‹Wort
Alle folgenden Eingaben bis zu einer Zeile, die gleich Wort ist, oder bis zum Erreichen von End-of-File, werden eingelesen. Das resultierende 'Paket' von Eingaben wird als Standard-Input benutzt. Falls ein Zeichen aus Wort apostrophiert ist, werden fuer das Eingabepaket keinerlei Interpretationen durchgefuehrt. Andernfalls werden Parameter- und Kommandosubstitutionen durchgefuehrt, **\Newline** wird ignoriert, und \ dient dem Apostrophieren von \ ‹ ' und dem ersten Zeichen von Wort.

‹&Ziffer
Der Standard-Input wird ein Duplikat des File-Deskriptors Ziffer (siehe dup(2)). Analog geschieht das Duplizieren fuer den Standard-Output beim Benutzen von ›.

‹&- Der Standard-Input wird geschlossen. Zum Schliessen des Standard-Output wird › benutzt.

Wird einer dieser Formen der E/A-Umlagerung eine Ziffer vorangestellt, so bezieht sich die beabsichtigte Umlagerung auf den durch die Ziffer angegebenen File-Deskriptor und nicht, wie im Standardfall, auf die Deskriptoren 0 und 1.
So wird, zum Beispiel, durch

```
... 2›&1
```

erreicht, dass File-Deskriptor 2 ein Duplikat des File-Deskriptors 1 wird.

Fuer Hintergrund-Kommandos (abgeschlossen mit &) ist das leere File /dev/null Standard-Input. Fuer andere Kommandos enthaelt die Ausfuehrungsumgebung die gleichen File-Deskriptoren, die der aufrufende Shell-Prozess hat, jedoch entsprechend der angegebenen E/A-Umlagerungen modifiziert.

Umgebung
Die Umgebung ist eine Liste von Paaren der Form Name=Wert, die wie die Liste der Argumente durch Shell an ein auszufuehrendes Programm uebergeben wird (siehe exec(2) und environ(5)). Shell kommuniziert mit der Umgebung auf verschiedene Arten. Beim Aufruf wird fuer jeden Namen, der in der Umgebung enthalten ist, ein Parameter erstellt und mit dem zugehoerigen Wert belegt. Auszufuehrende Kommandos 'erben' die Umgebung von Shell. Veraendert der Nutzer die Werte dieser Parameter oder fuehrt er neue Parameter ein, so hat dies keinen Einfluss auf die Umgebung. Nur mit Hilfe des **export**-Kommandos koennen Shell-Parameter an die Umgebung angebunden werden. Zur Umgebung eines auszufuehrenden Kommandos gehoeren daher alle nichtveraenderten Paare der Form Name=Wert, die bereits in der Umgebung von Shell enthalten waren sowie alle Modifikationen oder Ergaenzungen, fuer die **export**-Kommandos ausgefuehrt wurden.

Die Umgebung eines einfachen Kommandos kann erweitert werden, indem diesem beim Aufruf eine oder mehrere Parameter-Zuweisungen vorangestellt werden. Die folgenden zwei Kommandozeilen sind daher aequivalent.

```
WERT=478 cmd args
(export WERT; WERT=478; cmd args)
```

Ist die Option **-k** angegeben, werden alle Kennwort-Argumente in die Umgebung des zugehoerigen Kommandos eingebunden, auch wenn sie nach dem Kommandonamen angegeben sind. Durch

```
echo a=b c
set -k
echo a=b c
```

enstehen die Ausgaben 'a=b c' und 'c'.

Signale
Die Signale haben in einem Shell-Prozess normalerweise die Werte, die ihnen auch im jeweiligen Parent-Prozess zugeordnet waren. Eine Ausnahme bilden Hintergrund-Prozesse.
Diese ignorieren die Signale 2 (INTERRUPT) und 3 (QUIT). Den Signalen koennen mit dem **trap**-Kommando jedoch andere Werte zugewiesen werden.

Ausfuehrung
Vor jeder Kommandoausfuehrung werden die oben genannten Substitutionen durchgefuehrt. Ausser fuer die unten angefuehrten 'Speziellen Kommandos' wird zur Kommandoausfuehrung ein neuer Prozess geschaffen und versucht, das Kommando mit einem Systemruf (exec(2)) abzuarbeiten.

Der Shell-Parameter **%PATH** legt den Suchpfad fuer das Directory fest, welches das auszufuehrende Kommando enthaelt. Die verfuegbaren Alternativen sind durch Doppelpunkt (:) getrennt. Der Standardpfad ist **:/bin:/usr/bin**. Enthaelt der Kommandoname ein /, wird der Suchpfad nicht benutzt. Sonst wird in jedem im Suchpfad enthaltenen Directory nach einem ausfuehrbaren File mit dem im Kommando angegebenen Namen gesucht. Ist das gefundene File ausfuehrbar (executable), aber kein a.out-File, so wird angenommen, dass es Shell-Kommandos enthaelt. Ein Unter-Shell-Prozess wird geschaffen, um dieses File einzulesen.
Auch ein eingeklammertes Kommando wird in einem Unter-Shell-Prozess abgearbeitet.

Spezielle Kommandos
Zum Ausfuehren der folgenden Kommandos wird kein neuer Prozess eroeffnet. E/A-Umlagerungen koennen fuer diese Kommandos (ausser, wo es ausdruecklich erwaehnt ist) nicht durchgefuehrt werden.

: Dieses Kommando bewirkt nichts.

. file
Die in file enthaltenen Kommandos werden gelesen und ausgefuehrt. Der Suchpfad **%PATH** wird benutzt, um das file enthaltende Directory zu finden.

break [n]
Beenden der inneren **for**- oder **while**-Schleife, falls eine solche vorhanden ist. Ist n angegeben, so werden die n inneren Schleifen beendet.

continue [n]
Das Abarbeiten wird bei der naechsten Iteration der inneren **for**- oder **while**-Schleife fortgesetzt. Ist n angegeben, so wird mit der Iteration der n-ten inneren Schleife (n Schleifen-Niveaus nach aussen) fortgesetzt.

cd [Arg]
Arg wird das aktuelle Directory des Nutzers. Der Shell-Parameter **%HOME** ist Standard, wenn Arg nicht angegeben ist.

eval [Arg ...]
Die Argumente werden als Eingabe fuer Shell gelesen, die resultierenden Kommandos werden ausgefuehrt.

exec [Arg ...]
Die Argumente stellen ein Kommando dar, das anstelle des laufenden Shell-Prozesses (ohne einen neuen Prozess zu schaffen) abgearbeitet wird. E/A-Umlagerungen koennen angegeben werden. Diese beziehen sich auf die E/A-Richtungen des laufenden Shell-Prozesses, falls keine weiteren Argumente angegeben sind.

exit [n]
Beenden eines nichtinteraktiven Shell-Prozesses mit dem Rueckkehrstatus n. Ist n nicht angegeben, so ist der Rueckkehrstatus der des letzten ausgefuehrten Kommandos. (Die Eingabe von End-of-File fuehrt ebenfalls zum Beenden eines Shell-Prozesses).

export [Name ...]
Die angegebenen Namen werden der Umgebung von nachfolgend ausgefuehrten Kommandos hinzugefuegt. Sind keine Argumente angegeben sind, wird eine Liste aller exportierten Namen ausgegeben.

login [Arg ...]
Entspricht 'exec login Arg ...'.

newgrp [Arg ...]
Entspricht 'exec newgrp Arg ...'.

read Name ...
Von Standard-Input wird eine Zeile eingelesen. Die einzelnen Worte dieser Zeile werden in der angegebenen Reihenfolge den Variablen Name... als Werte zugewiesen. Ueberschuessige Worte werden der letzten Variablen zugewiesen. Der Rueckkehrwert dieses Kommandos ist 0, ausser wenn End-of-File eingelesen wurde.

readonly [Name ...]
Die angegebenen Namen werden als 'nur lesbar' festgelegt. Die diesen Namen zugeordneten Werte koennen durch spaetere Zuweisungen nicht veraendert werden. Falls keine Argumente angegeben sind, wird eine Liste aller **readonly**-Namen ausgegeben.

set [**-eknptuvx** [Arg ...]]
- **-e** Ein nichtinteraktiver Shell-Prozess wird sofort beendet, wenn bei einer Kommandoausfuehrung ein Fehler auftritt.
- **-k** Alle Kennwort-Argumente werden in die Umgebung eines Kommandos eingebunden; nicht nur diejenigen, die vor dem Kommandonamen angegeben sind.
- **-n** Die Kommandos werden nur gelesen, aber nicht ausgefuehrt.
- **-t** Beenden des Prozesses, nachdem ein Kommando gelesen und ausgefuehrt ist.
- **-u** Nicht mit Werten belegte Variablen werden als Fehler gewertet, wenn sie substituiert werden sollen.
- **-v** Die Shell-Eingabezeilen werden so, wie sie gelesen wurden, wieder ausgegeben.

-x Die Kommandos werden mit ihren Argumenten so ausgegeben, wie sie abgearbeitet werden.

- Aufheben der gesetzten Optionen -x und -v.

Diese Optionen koennen auch beim Aufruf von Shell gesetzt werden. Die aktuell gesetzten Optionen sind in $- enthalten.

Alle uebrigen Argumente sind Stellungsparameter und werden in der angegebenen Reihenfolge an $1, $2 usw. zugewiesen. Sind keine Argumente angegeben, so wird die aktuelle Belegung aller Namen ausgegeben.

shift

Die Stellungsparameter ab $2 ... werden umbenannt in $1 ...

times

Die bisher benoetigten Nutzer- und Systemzeiten fuer Prozesse, die von Shell gestartet wurden, werden ausgegeben.

trap [Arg [n] ...

Arg ist ein Kommando, das gelesen und ausgefuehrt werden soll, wenn der Shell-Prozess das Signal n empfaengt.
(Zu beachten ist, dass Arg zweimal von Shell bearbeitet wird - einmal, wenn das **trap**-Kommando eingelesen wird, und noch einmal, wenn das zugehoerige Signal empfangen wurde.) Die **trap**-Kommandos werden in der Reihenfolge der Signalnummern ausgefuehrt. Ist Arg nicht angegeben, so werden den Signalen n wieder die urspruenglichen Belegungen zugewiesen. Ist Arg die leere Zeichenkette (""), so wird das angegebene Signal n durch Shell und die von ihm aufgerufenen Kommandos ignoriert. Falls n gleich 0 ist, so wird das zugehoerige Kommando Arg ausgefuehrt, wenn der Shell Prozess beendet wurde. Die Nummern der Signale entsprechen den Angaben in signal(2). Wird trap ganz ohne Argumente aufgerufen, so wird eine Liste von auszufuehrenden Kommandos und zugeordneten Signalnummern ausgegeben.

umask [nnn]

Die Schutzmaske fuer zu erstellende Nutzer-Files wird gleich dem oktalen Wert von nnn gesetzt (siehe umask(2)). Ist nnn nicht angegeben ist, so wird der aktuelle Wert der Schutzmaske ausgegeben.

wait [n]

Auf das Beenden des durch den Prozessidentifikator n angegebenen Prozesses wird gewartet, und der Rueckkehrstatus dieses Prozesses wird ausgegeben. Ist n nicht angegeben, wird auf die Beendigung

aller gerade laufenden Child-Prozesse gewartet. Der Rueckkehrwert dieses Kommandos ist der des Prozesses, auf den gewartet wurde.

Aufruf
Beginnt das Argument Null beim Aufruf von Shell mit einem -, so werden zunaechst Kommandos aus dem File **$HOME/.profile** eingelesen und ausgefuehrt, falls dieses File existiert.

Danach erfolgt die weitere Arbeit von Shell wie bereits beschrieben. Die folgenden Optionen werden von Shell beim Aufruf interpretiert.

-c string Shell liest Kommandos aus string ein und fuehrt sie aus.

-s Shell liest Kommandos von Standard-Input ein. Shell-Ausgaben sind zum File-Deskriptor 2 gerichtet. Diese Festlegungen gelten auch, wenn beim Shell-Aufruf keine Argumente angegeben wurden.

-i Ein mit dieser Option aufgerufenes Shell ist interaktiv. Das gleiche gilt fuer Shell-Prozesse, deren Standard-Input und Standard-Output zu einem Terminal gerichtet sind. Derartige Shell-Prozesse ignorieren das Signal 15 (siehe signal(2)), sodass durch 'kill 0' ein interaktiver Shell-Prozess nicht abgebrochen werden kann. Das Signal 2 wird abgefangen, aber keine zugehoerige Aktion ausgefuehrt, so dass **wait** unterbrechbar ist. Das Signal 3 wird von Shell ignoriert.

Die uebrigen Optionen und Argumente sind beim **set**-Kommando beschrieben.

FILES

*HOME/.profile
/tmp/sh*
/dev/null

SIEHE AUCH

exec(2),

FEHLERMELDUNGEN

Werden durch Shell Fehler, wie Syntaxfehler, festgestellt, wird ein Rueckkehrstatus ungleich Null geliefert. Handelt es sich um einen nichtinteraktiven Shell-Prozess, so wird das Abarbeiten der entsprechenden Shell-Prozedur beendet.
Andernfalls liefert ein Shell-Prozess als Rueckkehrstatus den des zuletzt abgearbeiteten Kommandos (siehe auch **exit**).

FEHLERQUELLEN

Wenn << benutzt wird, um den Standard-Input fuer einen Hintergrund-Prozess (gestartet mit &) bereitzustellen, so bekommt Shell Schwierigkeiten beim Benennen des Eingabepaketes. Es wird versucht, auf ein nicht vorhandenes File /tmp/sh* zuzugreifen. Der Zugriffsfehler wird angezeigt.

SIZE(1) **SIZE(1)**

NAME

size - Groesse eines Objekt-Files

UEBERSICHT

size [object ...]

BESCHREIBUNG

Size druckt von jedem Objekt-File die (hexadezimale) Anzahl der Bytes, die vom Text-, Daten- und BSS-Bereich (uninitialisierte Daten) benoetigt werden und die Summe als oktalen und dezimalen Wert. Ist kein File angegeben, wird a.out verwendet.

SIEHE AUCH

a.out(5)

z.B. 4826 + 962 + 1064 + 0 = 6652 = 014774 = 0x19EC
Text Daten bss Stack

NAME

strip - Entfernt Symbole und Relocation-Bits

UEBERSICHT

strip name ...

BESCHREIBUNG

Strip entfernt die Symboltabelle und die Relocation-Bits, die gewoehnlich in die Ausgabefiles des Assemblers und Laders eingefuegt werden. Dies ist vernuenftig, um Speicherplatz zu sparen, nachdem ein Programm als fehlerfrei erkannt bzw. mit adb (1) erfolgreich korrigiert und getestet wurde.

Die Wirkung von strip ist dieselbe, wie das Verwenden der -s -Option von ld.

FILES

/tmp/stm? temporaeres File

SIEHE AUCH

ld(1)

NAME

stty - Setzen von Terminal-Optionen

UEBERSICHT

stty [option ...]

BESCHREIBUNG

Stty setzt verschiedene E/A-Optionen am vorliegenden Ausgabeterminal. Ohne Argumente zeigt es die aktuellen Werte der Optionen. Die Optionszeilen werden aus folgender Menge ausgewaehlt:

even geradzahlige Paritaet

-even keine geradzahlige Paritaet; nicht fuer MUTOS 8000

odd ungeradzahlige Paritaet; nicht fuer MUTOS 8000

-odd keine ungeradzahlige Paritaet

raw Eingabe im raw-Modus (kein erase, kill, interrupt, quit, EOT; Paritaetsbit bleibt erhalten)

-raw Negieren des raw Modus

cooked identisch zu `-raw'

cbreak jedes Zeichen wird sofort an read(2) uebergeben, wenn es empfangen wird; kein erase und kill

-cbreak Zeichen werden erst nach Empfang von new-line an read (2) uebergeben

crmod laesst carriage return fuer new-line zu und gibt CR-LF fuer carriage return oder new-line aus

-crmod akzeptiert nur new-line, um Zeilen zu beenden

echo jedes eingegebene Zeichen wird ueber "Echo" wieder ausgegeben

-echo kein Echo der Zeichen

lcase Grossbuchstaben werden in Kleinbuchstaben umgewandelt

-lcase kein Umwandeln von Grossbuchstaben

-tabs ersetzt Tabulatoren durch Leerzeichen bei der Ausgabe

tabs behaelt Tabulatoren bei

ek Zuruecksetzen der erase- und kill-Zeichen auf Standard DEL and `^U'

erase c setzt das erase-Zeichen auf c. C kann die Form `^X' haben, die als `control X' interpretiert wird.

kill c setzt das kill-Zeichen auf c. `^X' wirkt hier wie gehabt.

width n setzt die Seitenbreite auf n Zeichen fuer automatisches Einfuegen von Leerzeichen am Ende der Zeile (line folding).

engths n
setzt die Seitenlaenge auf n Zeilen, damit kann die Ausgabe seitenweise erfolgen. 0 schaltet die seitenweise Ausgabe voellig aus.

scope Zeichen verschwinden beim Loeschen.

-scope Abschalten des scope -Modus

indctl Echo der Steuerzeichen als '^(char+0141)'.

-indctl abschalten des indctl Modus

-sd   Negieren des sd Modus.

SIEHE AUCH

tty(4), ioctl(2), tabs(1)

SYNC(1M) **SYNC(1M)**

NAME

sync - Aktualisieren des Superblocks

UEBERSICHT

sync

BESCHREIBUNG

Durch Sync wird der Systemruf sync ausgefuehrt.
Sync schreibt alle zu aktualisierenden Informationen, die noch im Speicher stehen, auf Diskette.
Soll das System abgeschaltet werden, muss sync ausgefuehrt werden, um die Integritaet des Filesystems zu sichern.

SIEHE AUCH

sync(2), update(8)

WAIT(1) **WAIT(1)**

NAMEn

wait - Warten auf das Beenden von Prozessen

UEBERSICHT

wait

BESCHREIBUNG

Dieses Kommando wartet auf das Beenden aller mit & gestarteten Prozesse und informiert ueber eine nicht normales Beenden.

Da der Systemruf wait(2) im Parent-Prozess ausgefuehrt werden muss, wird wait durch Shell abgearbeitet, ohne einen neuen Prozess dafuer zu schaffen.

SIEHE AUCH
sh(1)

FEHLERQUELLEN
Nicht alle Prozesse einer 3- oder mehr-teiligen Pipline sind Child-Prozesse des Shell, sodass sie unter Umstaenden von wait nicht ausgewertet werden koennen.

WC(1) **WC(1)**

NAME
wc - Wortzaehlung

UEBERSICHT
wc [-lwc] [name ...]

BESCHREIBUNG
Das Dienstprogramm wc zaehlt Zeilen, Worte und Zeichen in den angegebenen Files bzw. im Standard-Input, falls kein name angegeben wurde. Das Ergebnis wird in dezimaler Form auf Standard-Output ausgegeben. Als Wort zaehlt ein String, der durch Leerzeichen, Tabulator bzw. Newline begrenzt ist.

Folgende Optionen koennen kombiniert werden:

- l Zeilenzaehler
- w Wortzaehler
- c Zeichenzaehler

Wurde eine der wahlfreien Optionen angegeben, wird nur der der Option entsprechende Zaehlerstand angezeigt.

Inhaltsverzeichnis

Teil 1 - Kommandos

		Seite
intro(1)	Einfuehrung	1
adb(1)	Testprogramm	2
ar(1)	Archiv- bzw. Bibliothekspflege	10
as(1)	PLZ/ASM-Assembler	12
cat(1)	Verketten und Ausgeben von Files	13
cc(1)	C-Compiler	14
cd(1)	Aendern des Arbeits-Directories	16
chmod(1)	Aendern der Zugriffsrechte	16
chown(1)	Aendern Gruppe	18
clri(1M)	Loeschen eines i-nodes	19
cmp(1)	Vergleich zweier Files	20
convert(1)	Wandeln Fileformat	21
cp(1)	Kopieren von Files	21
date(1)	Ausgabe und Setzen des Datums	22
dd(1)	Konvertieren und Kopieren eines Files	23
df(1M)	Freier Speicherplatz auf File-system	25
dcheck(1M)	Pruefen der Folgerichtigkeit eines Filesystems	25
echo(1)	Ausgabe der Argumente	26
ed(1)	Texteditor	27
hex(1)	Hexadezimale Ausgabe von Files	35
icheck(1M)	Pruefen der Folgerichtigkeit der Speicherbelegung	36
kill(1)	Beenden eines Prozesses	37
ld(1)	Lader	38

ln(1)	Erzeugen eines Links	40
login(1)	Eintragen in das System	41
ls(1)	Listen Directories	42
mkdir(1)	Erzeugen eines Directories	44
mkfs(1M)	Aufbau eines Filesystems	44
mknod(1M)	Anlegen eines Special Files	46
mount(1M)	Eingliedern oder Entfernen eines Filesystems	47
mv(1)	Uebertragen oder Umbenennen von Files und Directories	48
ncheck(1M)	Ermitteln von Pfadnamen aus i-node-Nummern	49
nm(1)	Ausgabe Symboltabelle	50
nroff(1)	Textformatierer fuer Drucker	51
od(1)	Oktalausgabe von Files	65
passwd(1)	Aendern des Nutzer-Passwortes	66
pr(1)	Ausgabe eines Files	67
ps(1)	Prozessor-Status	68
pstat(1M)	Ausgabe von Systeminformationen	70
pwd(1)	Name der aktuellen Directory	74
rm(1)	Loeschen Files	74
sh(1)	Kommandointerpreter	75
size(1)	Groesse eines Objekt-Files	87
strip(1)	Entfernen Symbole und Relocation-Bits	88
stty(1)	Setzen von Terminal-Optionen	88
sync(1M)	Aktualisieren des Superblockes	90
wait(1)	Warten auf das Beenden von Prozessen	90
wc(1)	Wortzaehlung	91

Kv 206/86 - V 3/15 - 446