robotron

SYSTEMHANDBUCH MUTOS 8000

Teil I
Abschnitt 4, 5, 7, 8

Specialfile
Fileformate
Makropakete Sprachkonventionen
Systemunterstützung

r o b o t r o n

Programmtechnische Beschreibung

fuer Buerocomputer A 5120.16

S Y S T E M H A N D B U C H

M U T O S 8 0 0 0

Teil I:

Abschnitt 4 - Specialfiles

Abschnitt 5 - Fileformate

Abschnitt 7 - Makropakete Sprachkonventionen

Abschnitt 8 - Systemunterstuetzung

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

10

Die vorliegende Dokumentation entspricht dem Stand 12/85.

Im Interesse einer staendigen Weiterentwicklung werden alle Leser gebeten, ihre Vorschlaege bzw. Hinweise dem

VEB Robotron-Buchungsmaschinenwerk
9010 Karl-Marx-Stadt
Annabergerstr. 93

mitzuteilen.

Fuer das Betriebssystem MUTOS 8000 wurden folgende Dokumentationen erarbeitet:

- Anwendungsbeschreibung

- Systemhandbuch MUTOS 8000 Teil I,
  - Abschnitt 1 Kommandos
  - Abschnitt 2, 3 Systemrufe, Bibliotheksfunktionen
  - Abschnitt 4, 5, 7, 8 Special Files, Fileformate, Makropakete, Systemunterstuetzung

- Systemhandbuch MUTOS 8000 Teil II,
  - Abschnitt 1 Kommandointerpreter Shell
  - Abschnitt 2 Editor
  - Abschnitt 3 Testhilfsprogramm adb

- Systemhandbuch MUTOS 8000 Teil III,
  - Abschnitt 1 Textverarbeitung nroff

- Sprachbeschreibungen
  - Assemblersprachbeschreibungen U8000
  - Sprachbeschreibung C-Sprache

Das Sachwortverzeichnis ist als Anlage in der Anwendungsbeschreibung enthalten.

Zum Programmentwickeln und -Testen fuer den Prozessor U8000 werden zusaetzlich folgende Dokumentationen angeboten:

- Sytemhandbuch MON8000
- CROSS-Software U8000 zum Betriebssystem UDOS
- Assembler U8000ASM
- Binder ZLINK
- Absolutbinder IMAGER
- Preprozessor INCLUDE

Inhaltsverzeichnis

Abschnitt 4 - Specialfiles

Abschnitt 5 - Fileformate

Abschnitt 7 - Makropakete Sprachkonventionen

Abschnitt 8

LP(4) LP(4)

NAME

lp - Drucker SD 1152, SD 1157

BESCHREIBUNG

lp ist das Special-File fuer den Drucker. Bei open(2) und close(2) wird ein Seitenvorschub (Form Feed) ausgegeben, sofern der Drucker nicht schon auf einer Blattgrenze (durch vorhergehendes Form Feed) steht.
Ist das Bit Øx1Ø der Minor-Geraetenummer gesetzt, wird angenommen, dass das Geraet nur einen 64 Zeichen umfassenden Zeichensatz hat (statt 96). Kleinbuchstaben werden deshalb in Grossbuchstaben gewandelt und einige Sonderzeichen entsprechend der folgenden Tabelle umschrieben:

{	←
}	→
`	↑
|	+
~	^

Der Driver interpretiert die Zeichen CR (Carrige return), BS (Backspace), HT (Tabulator) und FF (Form Feed) korrekt. Die Zeilenlaenge wird auf 132 Zeichen begrenzt, ist jedoch als Parameter im Driver veraenderbar. Weitere Zeichen einer Zeile werden verworfen. Die Seitenlaenge ist (ebenfalls als Driver-Parameter) auf 66 Zeilen eingestellt. Die Benutzung des Raw-Druckers rlp erfolgt analog. Alle Zeichen werden unveraendert an den Drucker weitergereicht. Ein Form Feed bei open(2) und close(2) wird unterbunden. Bei rlp wird das Bit Øx1Ø der Minor-Geraetenummer ignoriert. Ein Umwandeln von Klein-in Grossbuchstaben ist in diesem Fall nicht moeglich.
Die Unterscheidung zwischen lp und rlp erfolgt ueber das Bit Øx Ø4 der Minor-Geraetenummer (damit sind nur max. 4 Geraetenummern zulaessig).

FILES

/dev/lp ,/dev/rlp

NAME

mem, kmem - Hauptspeicher

BESCHREIBUNG

Mem ist ein Special-File und stellt ein Abbild des Hauptspeichers des Rechners dar. Es kann zum Beispiel benutzt werden, um im Betriebssystem-Kern Informationen zu lesen oder Veraenderungen vorzunehmen. Mem bezieht sich auf den physischen Speicher, kmem auf den virtuellen Systemadressraum. Byte-Adressen werden als Speicheradressen modulo 256 Kbyte (Mem) oder als Speicheradressen modulo 64 Kbyte (Kmem) interpretiert.
Der Systemadressraum beginnt auf der physischen Adresse 00000. Der zum laufenden Prozess gehoerende Datenbereich (Nutzerstruktur) beginnt auf der virtuellen Adresse 0xEC00 des Systemadressraumes.

FILES

/dev/mem, /dev/kmem

NAME

null - Datensenke

BESCHREIBUNG

Daten, die auf das Null-Special-File ausgegeben werden, werden verworfen.

Lesezugriffe bringen immer 0 Byte zurueck.

FILES

/dev/null

NAME

rk - Floppy-Disk-Laufwerk K 5602/K 5600.20

BESCHREIBUNG

rk? bezieht sich auf eine gesamte Diskette als ein einzelnes, sequentiell addressiertes File. Die Minor-Geraetenummer bezieht sich auf die Nummer des Disketten-laufwerkes. Es koennen max. 4 Laufwerke bedient werden. Die rk-Files greifen auf den Datentraeger Diskette ueber die normalen Puffermechanismen des Systems zu. Dabei werden logische Bloecke auf die Diskette geschrieben bzw. von dieser gelesen. Fuer den Zugriff auf das Floppy-Disk-Laufwerk existiert weiterhin ein 'Raw'-Interface. Bei Benutzung dieses Files werden Datenuebertragungen zwischen Floppy-Disk und Nutzerprogramm ohne Zwischenschaltung des Puffermechanismus direkt ausgefuehrt. Jeder einzelne Lese- oder Schreibzugriff fuehrt zu genau einer E/A-Operation. Deshalb sind die 'Raw'-Files bei Uebertragung grosser Datenmengen effektiver als die Puffervarianten. Die Namen der 'Raw'-Files ergeben sich durch Nachsetzen eines 'r' aus dem korrespondierenden rk-File. Bei Benutzung als 'Raw'-Files koennen in einer E/A-Operation max. 8 aufeinanderfolgende Bloecke (= 4 kB) geschrieben oder gelesen werden. Der E/A-Puffer im Nutzerprogramm muss bei der 'Raw'-Version an einer Wortgrenze beginnen. Die Uebertragungslaenge muss ein Vielfaches von 512 Byte sein. Bei seek-Rufen sollten ebenfalls 512 Byte angegeben werden.

Das Floppy-Disk ist ein kleiner Massenspeicher mit wahlfreiem Zugriff, der Daten auf einer formatierten Diskette speichert. Die Groesse eines Sektors betraegt 128/512 Bytes. Ein logischer Block besteht darausfolgend aus 4/1 Sektoren, die in einem bestimmten Standardversatz (siehe Tabelle) auf der Diskette abgelegt sind. Die logische Blockgroesse betraegt 512 Byte.

Floppy-Disk	phys. Satzlaenge	Blockanzahl	Standardversatz innerh. Spur	Standardversatz Spur-wechsel
K 5602/FM (8")	128 Byte 512 Byte	494 608	2 System:1 (root) andere:2	6 1 1
K 5600.20/FM (5,25")	512 Byte	632	System:1 (root) andere:2	1 1

FILES:

/dev/rk?,/dev/rrk?

FEHLERQUELLEN:

Bei der 'Raw'-Variante beschneiden read(2) und write(2) die File-Offsets auf 512-Byte Blockgrenzen. Dadurch werden evtl. vom Programmierer falsche Informationen gelesen oder geschrieben. Der Zugriff auf 'Raw'-Geraete erfolgt ohne Beachtung der systeminternen Puffer fuer die Block-E/A. Dadurch stehen Informationen, die durch den Puffermechanismus noch nicht auf die Diskette geschrieben wurden, den Zugriff ueber 'Raw'-Files nicht zur Verfuegung.

NAME

tty - Allgemeine Terminal-Schnittstelle

BESCHREIBUNG

In diesem Abschnitt werden sowohl die allgemeine Terminalschnittstelle als auch das Special - File fuer Terminals beschrieben. Die Beschreibung gilt sowohl fuer das Konsolterminal (BC A5120.16 mit Tastatur K7637) als auch fuer etwaige weitere Terminals.

Das File /dev/tty stellt fuer jeden Prozess das Synonym fuer das steuernde Terminal dar. Es ist sinnvoll, diese zu nutzen, wenn Ausgaben eines Programms unbeeinflusst von eventuellen Veraenderungen der Geraetezuweisungen immer auf dem Terminal erscheinen sollen. Als steuerndes Terminal fuer einen Prozess wird jenes Terminal betrachtet, auf das als erstes ein open(2)-Ruf gegeben wird. Praktisch geschieht das im Systemanlauf durch init(8).

Das "Steuerterminal" spielt bei der Behandlung von Quit- und Interrupt-Signalen eine spezielle Rolle (s.u.). Es wird waehrend eines fork(2)-Vorganges an den Kind-Prozess "vererbt" und bleibt sogar dann erhalten, wenn das Steuerterminal geschlossen wird (close(2)). Alle Prozesse, die auf diese Art ein gemeinsames Steuerterminal haben, werden als Prozessgruppe bezeichnet. Alle Mitglieder einer Prozessgruppe emfangen bestimmte Signale gemeinsam (siehe CTRL/C und kill(2)).

Die Terminals werden im Duplex - Betrieb bedient. Zu jedem Zeitpunkt koennen Zeichen eingegeben werden, auch bei gerade laufender Ausgabe. Datenverluste koennen nur in den Faellen auftreten, in denen saemtliche Eingabepuffer des Systems ausgelastet sind oder die max. Anzahl Zeichen erreicht wurde, die ein Nutzer eingeben kann, ohne dass sie von einem Programm uebernommen wurden. Gegenwaertig liegt diese Grenze bei 256 Zeichen. Alle weiteren Zeichen werden verworfen.

Normalerweise werden Terminaleingaben zeilenweise verarbeitet. Das bedeutet, dass ein Programm, das einen Leseruf gegeben hat, solange ruht, bis eine vollstaendige Zeile eingegeben wurde. Es wird immer hoechstens eine Zeile uebergeben, gleichgueltig, wieviele Zeichen angefordert wurden. Es ist jedoch moeglich, weniger als eine ganze Zeile zu lesen (z.B. einzelne Zeichen). Die restlichen Zeichen verbleiben im Eingabepuffer des Systems und gehen nicht verloren. Uber das Einstellen bestimmter Terminal-Modi ist es moeglich, jedes Zeichen sofort im Programm zur Verfuegung zu haben, ohne die Eingabe einer vollstaendigen Zeile abwarten zu muessen (s.u.).

Normalerweise werden waehrend der Eingabe Zeichen- und Zeilenirrung unterstuetzt (erase und kill). Standardmaessig loescht das Zeichen DEL (Taste DEL) das letzte eingegebene Zeichen, jedoch nicht ueber den Zeilenanfang oder ein Begrenzungszeichen ('^Z' (End of File)) oder 't_brkc' (s.u.)) hinweg. Mit '^U' (Standard) laesst sich eine unvollendete Zeile vollstaendig loeschen, jedoch nicht ueber ein '^Z' hinaus. Die Standardzeichen fuer Korrekturen koennen geaendert werden. Bei Benutzung druckbarer Zeichen an ihrer Stelle (z.B. '*' und '@') werden die zu loeschenden Zeichen aus dem Eingabepuffer entfernt, jedoch nicht auf dem Bildschirm geloescht. Das Erase-Zeichen wird selbst dargestellt, das Kill-Zeichen fuehrt auf eine neue Zeile.

Sollen Terminals bedient werden, die nur Grossbuchstaben darstellen koennen, kann der Terminal-Driver so eingestellt werden, dass ein Wandel von Gross- in Kleinbuchstaben bei Eingaben und umgekehrt bei Ausgaben erfolgt. Grossbuchstaben koennen dann durch Vorsetzen von '\' eingegeben werden. Ausserdem sind folgende Umschreibungen moeglich:

\' fuer '
\| fuer |
\^ fuer ~
\(fuer {
\) fuer }

Bei Ausgaben erfolgt die Darstellung ebenso.

Einige Steuerzeichen des KOI-7-Zeichensatzes haben eine besondere Bedeutung. Diese Zeichen werden im Normalmodus (nicht jedoch im 'Raw'-Modus) vom Terminal-Driver behandelt und nicht an das anfordernde Programm uebergeben. Diese speziellen Zeichen sind einstellbar. Standardzeichen sind:

^Z erzeugt ein 'End-of-File' von einem Terminal. Wird es empfangen, werden alle Zeichen im Eingabepuffer des Systems dem lesenden Programm uebergeben, ohne auf ein Zeilenende-Zeichen zu warten. '^Z' selbst wird verworfen. Erscheint '^Z' am Anfang einer Zeile (keine Zeichen im Eingabepuffer), werden Null Zeichen uebergeben. Das hat die Bedeutung von 'End-of-File'.

^C wird nicht an ein Programm uebergeben, sondern erzeugt ein Interrupt-Signal, das an alle Prozesse uebergeben wird, die als Steuerterminal das '^C'-ausloesende Terminal haben. Die Prozesse koennen fuer dieses Signal eine spezielle Behandlung vorsehen (siehe signal(2)). Ist dies nicht der Fall, wird der Prozess abgebrochen.

FS (CNTRL-\ oder CNTRL-SHIFT-L) erzeugt ein Quit-Signal. Seine Bedeutung ist identisch mit dem Interrupt-Signal bis auf die Tatsache, dass im Abbruch-Fall ein Speicherabzugsfile (core image file) des Prozesses erzeugt wird.

^P schaltet die seitenweise Ausgabe fuer den aktuellen Prozess ein/aus, wenn nicht der Wert sg_length in der ttiocb-Structure (s.u.) auf Null gesetzt wurde. Mit dieser Methode der seitenweisen Ausgabe sind beliebige Zeilenzahlen pro Seite moeglich.

^S stoppt die Ausgabe auf das Terminal bis zur Eingabe von '^Q'.

^Q setzt die mit '^S' gestoppte Ausgabe fort.

^R schreibt die aktuelle Zeile vom letzten Begrenzungszeichen an nochmals aus. Steuerzeichen werden wie im Modus INDCTL (s.u.) angezeigt.

^W loescht das letzte eingegebene 'Wort'.

Ausgaben auf ein Terminal gehen ueber einen Ausgabepuffer im System. Erzeugt ein Prozess schneller Ausgabezeichen als auf das Geraet uebertragen werden koennen, wird ab einer bestimmten Marke der Prozess zum Warten veranlasst. Ist der Ausgabepuffer bis unter eine zweite Marke geleert worden, wird der wartende Prozess fortgesetzt. Ausgabezeichen werden mit gerader Paritaet versehen.

Eingegebene Zeichen werden als Echo wieder ausgegeben, indem sie sofort nach Empfang vom Terminal-Driver in den Ausgabepuffer eingetragen werden. Ein- und Ausgaben koennen parallel erfolgen. Fuer die Terminals werden bei der Eingabe im Normalmodus alle Gross- in Kleinbuchstaben umgewandelt, sofern der Modus 'LCASE' eingeschaltet ist. Ist der Modus '-LCASE' eingeschaltet, so unterbleibt diese Umwandlung.

Es existieren einige ioctl(2)-Rufe, die sich auf Terminals beziehen. Die meisten benutzten die folgende, in (sgtty.h) definierte Structur:

```
struct sgttyb {
        char    sg_ispeed;
        char    sg_ospeed;
        char    sg_erase;
        char    sg_kill;
        short   sg_flags;
        char    sg_nldly;
        char    sg_crdly;
        char    sg_htdly;
        char    sg_vtdly;
        char    sg_width;
        char    sg_length;
};
```

Die Felder sg_ispeed und sg_ospeed legen die Uebertragungsgeschwindigkeit der jeweiligen Uebertragungsleitungen fest. Diese Parameter werden gegenwaertig nicht ausgewertet. Die Ubertragungsgeschwindigkeit betraegt immer 9600 Baud.

Die Felder sg_erase und sg_kill legen die Zeichen fuer Zeichenirrung (erase) und Zeilenirrung (kill) fest. Standard sind DEL und '^U'.

Im sg_flags-Feld legen verschiedene Bits die Art der Behandlung des Terminals durch den Driver fest:

XTABS	ØX0400	Auszugsebene TAB's sind durch Leerzeichen zu ersetzen
INDCTL	ØX0200	Eingegebene Steuerzeichen werden als ^(Zeichencode+0140) angezeigt
SCOPE	ØX0100	Zeichen- und Zeilenirrung werden durch Loeschen auf dem Bildschirm angezeigt
EVENP	ØX0080	Gerade Paritaet bei Eingaben erlaubt
ODDP	ØX0040	Ungerade Paritaet bei Eingaben erlaubt
RAW	ØX0020	Raw-Modus (s.u.)
CRMOD	ØX0010	Wandeln CR in LF; Echo von LF oder CR als CR-LF
ECHO	ØX0008	Echo-Arbeitsweise des Terminals (Voll-Duplex-Betrieb)

LCASE	ØX0004	Wandeln von Gross- in Kleinbuchstaben bei Eingaben
CBREAK	ØX0002	Sofortiges Uebergeben jedes Zeichens an das lesende Programm
TANDEM	ØX0001	Automatische Ablaufsteuerung (gegenwaertig nicht genutzt)

XTABS bewirkt, dass Tabulatoren durch die Ausgabe einer entsprechenden Anzahl von Leerzeichen realisiert werden.

INDCTL fuehrt zum Darstellen von Steuerzeichen beim Echo in der Form '^(Zeichen+0140)'. Normalerweise werden Steuerzeichen nicht als Echo ausgegeben, um unerwuenschte Reaktionen des Terminals zu verhindern. Mit INDCTL lassen sie sich sichtbar machen.

SCOPE legt fest, dass das Terminal ein Bildschirmgeraet ist. Zeichen- und Zeilenirrung werden durch Loeschen auf dem Bildschirm angezeigt.

Im RAW-Modus werden alle eingegebenen Zeichen sofort dem lesendem Programm uebergeben, ohne auf einen Zeilenabschluss zu warten. Zeichen- und Zeilenirrung werden nicht unterstuetzt, die End-of-File-Anzeige ('^Z'), das Interrupt-Zeichen ('^C') und das Quit-Zeichen (FS) werden nicht speziell behandelt. Die Wandlung von Gross- in Kleinbuchstaben ist ausser Kraft, ESC-Folgen werden nicht ausgewertet. Die Zeichen werden mit 8 Bit Breite in Ein- und Ausgaberichtung zwischen Programm und Geraet uebertragen (Paritaetsbit-Behandlung muss im Programm erfolgen).

Zusaetzlich zum RAW-Modus koennen

ECHO, CRMOD, LCASE, TANDEM und XTABS

gesetzt sein. Dabei ist zu beachten, dass die jeweilige Funktion durch das Paritaetsbit beeinflusst werden kann.

Ist CRMOD eingestellt, werden eingegebene CR's in LF's gewandelt. Die Eingabe von CR oder LF fuehrt in beiden Faellen zum Echo von LF-CR.

ECHO bewirkt die automatische Ausgabe eingegebener Zeichen seitens des Drivers. Dabei erfolgt die Ausgabe stets mit 7 Bit Breite, auch im 'Raw'-Modus.

CBREAK ist eine Zwischenstufe zwischen 'Raw'-Modus und Normalmodus. Jedes eingegebene Zeichen kann vom Programm sofort uebernommen werden, ohne auf eine vollstaendige Zeile warten zu muessen. Im Unterschied zum 'Raw'-Modus erfolgt jedoch die Behandlung von Interrupt- und Quit-Signalen, waehrend eine spezielle Behandlung von '^Z' sowie Zeichen- und Zeilenirrung nicht unterstuetzt werden. Gross-/Kleinwandlung, CRMOD, XTABS, ECHO und die Paritaetsbitbehandlung funktionieren wie im Normalmodus.

TANDEM ist fuer Geraete gedacht, die auf ein Stop-Zeichen hin (Standard '^S') ihre Eingabe bis zu einem folgenden Start-Zeichen (Standard '^Q') stornieren koennen. Diese Arbeitsweise verhindert Datenverluste beim Ueberlauf der Eingabepuffer im System.

Die Delay-Felder koennen Angaben ueber Verzoegerungszeiten enthalten, wenn angeschlossene Geraete diese bei den entsprechenden Funktionen benoetigen. Der Wert 0 bedeutet, dass keine Verzoegerungen erfoderlich sind.

Das Feld sg_width definiert das Volumen der Ausgabe (z.B. Anzahl Zeichen/Zeile). Ist der Wert ungleich 0, wird beim Erreichen dieses Wertes vom Terminal-Driver ein LF-Zeichen eingeschoben.

Ueber das Feld sg_length laesst sich die Laenge einer 'Seite' auf dem Geraet festlegen. Jeweils bei Erreichen der eingetragenen Zeilenzahl wird die Ausgabe gestoppt. Nach Eingabe eines beliebigen Zeichens ausser LF (oder CR), Interrupt oder Quit wird die Ausgabe fortgesetzt. Interrupt und Quit fuehren zu der ueblichen Reaktion, waehrend LF oder CR die Ausgabe der naechsten Zeile bewirken.

Die verschiedenen ioctl(2)-Rufe haben folgende Form:

```
#include <sgtty.h>

ioctl(fildes, code, arg)
struct sgttyb *arg;
```

Folgende Funktionen sind anwendbar:

TIOCGETA
: Abfragen der fuer das Terminal eingestellten Parameter und Abspeichern in der angegebenen Structure.

TIOCSETA
: Setzen der in der angegebenen Structure enthaltenen Parameter fuer das Terminal. Das Setzen erfolgt erst dann, wenn keine Ausgabe mehr laeuft. Bis zu diesem Zeitpunkt nicht abgeholte Eingabezeichen werden verworfen.

TIOCGETP und TIOCSETP
Diese Funktionen wirken wie TIOCGETA bzw. TIOCSETA, jedoch nur auf die ersten 6 Byte der Structure sgttyb (d.h. sg_ispeed bis sg_flags).

TIOCSETN
Setzen der Parameter ohne Verzoegerung oder Verwerfen von Eingabezeichen. Wirkt ebenfalls nur auf die ersten 6 Byte der Structure sgttyb. Das Ausschalten von RAW- oder CBREAK-Modus kann zu einer unbrauchbaren Eingabe fuehren.

Bei folgenden Funktionen wird arg ignoriert:

TIOCEXCL
Setzen des 'exclusive-use'-Modus.
Das File kann erst nach einem Schliessen erneut eroeffnet werden.

TIOCNXCL
Ausschalten des 'exclusive-use'-Modus.

TIOCHPCL
Beim letzten Schliessen des Files wird das Terminal abgemeldet.

TIOCFLUSH
Alle Zeichen, die noch im Eingabe- oder Ausgabepuffer warten, werden verworfen.

Die folgenden Funktionen beziehen sich auf Sonderzeichen der Terminal-Schnittstelle. Als Argument wird ein Pointer zur folgenden Structure uebergeben, die in (sgtty.h) definiert ist:

```
struct tchars {
      char t_intrc;     /* interrupt */
      char t_quitc;     /* quit */
      char t_startc;    /* start output */
      char t_stopc;     /* stop output */
      char t_eofc;      /* end-of-file */
      char t_brkc;      /* input delimiter (like nl) */
};
```

Standardwerte fuer diese Zeichen sind: '^C', FS, '^Q', '^S', '^Z' und -1. Der Zeichenwert -1 bedeutet, dass kein derartiges Zeichen wirken soll. Das Sonderzeichen t_brkc (Standard -1) wirkt wie ein LF-Zeichen, indem es eine 'Zeile' abschliesst, ueber Echo ausgegeben wird und eine Uebergabe an das Programm erfolgt. Die Sonderzeichen 'Stop' und 'Start' koennen identisch sein, wobei ein Wechseleffekt erzielt wird.

Folgende Funktionen existieren fuer die Sonderzeichen:

TIOCGETC
: Abfragen der Sonderzeichen, die fuer das betreffende Terminal eingestellt sind, und Abspeichern in der angegebenen Structure.

TIOCSETC
: Setzen der Sonderzeichen fuer das betreffende Terminal auf die in der Structure angegebenen Werte.

FILES

/dev/tty, /dev/tty*, /dev/console

SIEHE AUCH

getty(8), stty(1), signal(2), ioctl(2)

FEHLERQUELLEN

Halbduplexe Terminals werden nicht unterstuetzt. Bei Verwenden anderer Tastaturen wie z.B. K7636 muss infolge des Fehlens einer 'Control'-Taste anstelle 'Control-Buchstabe' zuerst ET2 und danach das entsprechende Zeichen gedrueckt werden.

ACCT(5) ACCT(5)

NAME

acct - Ausfuehrung Abrechnungsfile

UEBERSICHT

```
# include <sys/acct.h>
```

BESCHREIBUNG

Acct(2) bewirkt, dass fuer jeden beendeten Prozess Eintragungen in einem Abrechnungsfile vorgenommen werden. Das Abrechnungsfile ist eine Folge von Eintragungen, deren Struktur im include-File definiert ist:

```
/*
 * Abrechnungsstruktur
 */


typedef   unsigned  short   comp_t;
/*"Gleitkomma":3  bit Basis,8  bit Exponent */

struct acct
{
        char    ac_comm[10];/*Kommandoname                */
        comp_t  ac_utime;   /*Nutzerzeit                 */
        comp_t  ac_stime;   /*Systemzeit                 */
        comp_t  ac_etime;   /*verstrichene Zeit          */
        time_t  ac_btime;   /*Startzeit                  */
        short   ac_uid;     /*Nutzeridentifikator        */
        short   ac_gid;     /*Gruppenidentifikator       */
        short   ac_mem;     /*durchschnittliche          */
                            /*Speicherbenutzung          */
        comp_t  ac_io;      /*Anzahl der Disketten-I/O-  */
                            /*Bloecke                    */
        dev_t   ac_tty      /*steuerndes Terminal        */
        char    ac_flag     /*Flag                       */
};

extern struct acct    acctbuf,
extern struct inode   *acctp; */inode des  abrechnenden*/
                              */Files                    */

#define  AFORK 01     /*hat fork ausgefuehrt aber      */
                      /*nicht exec                     */

#define ASK    02     /*Superuserprivilegien verwendet */
```

Fuehrt der Prozess einen exec(2) aus, erscheinen die ersten 10 Zeichen des Filenamens in ac_comm. Das Flag enthaelt Bits, die anzeigen, ob exec(2) immer vollendet wurde und ob der Prozess immer Superuserprivilegien besass.

SIEHE AUCH

acct(2)

NAME

a.out - Assembler- und Link-Editor-Output

UEBERSICHT

```
#include <a.out.h>
```

BESCHREIBUNG

A.out ist das Output-File des Assemblers as(1) und des Link-Editors ld(1). Beide Programme machen a.out ausfuehrbar, falls es keine Fehler und unaufloesbare externe Symbolbezugnahmen gibt. Formatinformationen werden im entsprechenden Include-File gegeben:

```
struct  exec {                  /* a.out header                 */
        int     a_magic;        /* magic number                 */
        unsigned a_text;        /* size of text segment         */
        unsigned a_data;        /* size of initialized data     */
        unsigned a_bss;         /* size of uninitialized        */
                                /* data                         */
        unsigned a_syms;        /* size of symbol table         */
        unsigned a_entry;       /* entry point                  */
        unsigned a_unused;      /* not used                     */
        unsigned a_flag;        /* relocation info stripped     */
};

#define A_MAGIC1 0xE807         /* normal                       */
#define A_MAGIC3 0xE811         /* separated I&D                */
#define A_MAGIC4 0xE805         /* overlay                      */

#define AF_STRIP 0x0001         /* relocation info stripped     */

struct  nlist {                 /* symbol table entry           */
        char    n_name[8];      /* symbol name                  */
        char    n_type;         /* type flag                    */
        char    n_unused;       /* not used                     */
        unsigned n_value;       /* value                        */
};
```

```
                                /* values for type flag      */
#define N_UNDF    0x00          /* undefined                 */
#define N_ABS     0x01          /* absolute                  */
#define N_TEXT    0x02          /* text symbol               */
#define N_DATA    0x03          /* data symbol               */
#define N_BSS     0x04          /* bss symbol                */
#define N_TYPE    0x1F          /* mask for type             */
#define N_REG     0x14          /* register name             */
#define N_SN      0x1E          /* section name              */
#define N_FN      0x1F          /* file name symbol          */
#define N_EXT     0x20          /* external bit, or'ed in    */
#define FORMAT    "%06x"        /* to print a value          */
```

Das File hat 4 Sektionen: Einen Kopf, Programm und Daten (initialisiert), Verschieblichkeitsinformationen und eine Symboltabelle (in dieser Reihenfolge). Die letzten 2 Sektionen koennen leer sein, wenn das Programm mit der '-s' Option von ld geladen wurde, oder wenn die Symbole und Verschieblichkeitsinformationen durch strip(1) geloescht wurden.

Im Header wird die Groesse jeder Sektion (bzw. Segment) (s. Include-File) in Bytes angegeben (geradzahlig). Die Groesse des Headers ist in Keiner der anderen Groessen enthalten.

Wird ein a.out-File zur Ausfuehrung in den Speicher geladen, werden 3 Segmente aufgestellt: Das Textsegment, das Datensegment (enthaelt auch nichtinitialisierte Daten, gefuellt mit 0 fuer nachfolgende Initialisierung) und ein Stack. Das Textsegment beginnt auf Platz 0 im Speicherabbild; der Header wird nicht geladen.

Die Magic-Nummer 0xE807 im Header kennzeichnet, dass Text- und Datensegment kontinuierlich im Speicher stehen. Die zwei anderen im a.out-Include-File definierten Formate sind unter MUTOS 8000 (A 5120.16) nicht implementiert bzw. zulaessig.

Der Stack belegt die hoechstmoeglichen Speicherplaetze im Speicherabbild: von 0xFFFE abwaerts wachsend. Der Stack wird automatisch erweitert. Das Datensegment wird nur erweitert durch den brk(2)-Ruf.

Der Textbereich (entspricht Textsegment) beginnt im File auf Platz 0x10; der Bereich initialisierter Daten auf 0x10+St (St: Groesse des Textes). Der Anfang der Verschieblichkeitsinformationen liegt auf 0x10+St+Sd (Sd: Groesse des Datenbereiches). Die Symboltabelle beginnt auf Platz 0x10+2*(St+Sd), wenn Verschieblichkeitsinformationen vorhanden sind, andernfalls auf Platz 0x10+St+Sd.

Das Format der Symboltabellenplaetze und die prinzipiellen Flag-Werte der unterschiedlichen Symboltypen sind im Include-File angegeben.

Ist der Typ eines Symbols undefiniert extern und sein Wert nicht 0, interpretiert der Lader ld das Symbol als den Namen einer Common-Region, deren Groesse sich aus dem Wert des Symbols ergibt.
Der Wert eines Wortes in den Text- oder Datenbereichen, der nicht eine Bezugnahme zu einem undefinierten externen Symbol ist, ist genau der Wert, der im Speicher bei Ausfuehrung des Files erscheint. Schliesst ein Wort im Text- oder Datenbereich eine Bezugnahme zu einem undefinierten externen Symbol ein, die durch die Verschieblichkeitsinformation fuer dieses Wort angezeigt wird, dann ist der Wert des Wortes, wie er in das File gespeichert wird, ein Offset auf das bezuggenommene Symbol. Wird das File durch den Link-Editor verarbeitet und das externe Symbol erhaelt seinen Wert, wird dieser Wert zum Wort im File addiert.

Sind Verschieblichkeitsinformationen vorhanden, dann ist das "relocation info stripped" - Flag im Header gesetzt. Fuer jedes Wort des Programmtextes und der initialisierten Daten gibt es ein Wort mit Verschieblichkeitsinformationen.

Bit 3 eines Verschieblichkeitswortes zeigt an,

wenn 1, dass sich die Verschieblichkeitsinformation auf ein externes Symbol bezieht,

wenn 0, dass eine aufgeloeste Referenz vorliegt.

Die Bits 15 bis 4 enthalten im Fall einer externen Referenz eine Symbol-Nummer, andernfalls sind sie ungenutzt. Das erste Symbol hat die Nummer 0, das zweite 1 usw.

Bit 0 des Verschieblichkeitswortes zeigt an,

wenn 1, dass die Referenz relativ zum Befehlszaehler ist,

wenn 0, dass sie sich auf ein aktuelles Symbol bezieht.

Die Benutzung der Bits 3 bis 0 wird durch folgende Tabelle vollstaendig beschrieben:

```
Bit  3  2  1  0
-----------------  ----------------------
     1  0  0  0     16 bit offset
        0  1  1     12 bit PC relative
        0  0  1     16 bit PC relative

     0  0  0  0     absolute
        0  0  1     code reference  (12 bit PC relative)
        0  1  0                     (offset)
        0  1  1                     (16 bit PC relative)
        1  0  0     data reference  (offset)
        1  0  1                     (16 bit PC relative)
        1  1  0     bss reference   (offset)
        1  1  1                     (16 bit PC relative)
```

SIEHE AUCH

as(1), ld(1), nm(1), size(1), strip(1)

AR(5) **AR(5)**

NAME

ar - Archivfile-Format

UEBERSICHT

```
#include <ar.h>
```

BESCHREIBUNG

Das Kommando ar wird benutzt, um verschiedene Files in Archiven zusammenzufassen. Archive werden hauptsaechlich als Bibliotheken fuer den Link-Editor ld(1) benutzt.

Ein File, hergestellt mit ar, hat eine Magic-Nummer am Beginn, gefolgt von den enthaltenen Files. Jedem Einzelfile ist ein Header vorangestellt. Die Magic-Nummer und das Header-Format, wie im Include-File beschrieben, sind:

```
#define ARMAG 0177545
struct  ar_hdr {
        char    ar_name[14];
        long    ar_date;
        char    ar_uid;
        char    ar_gid;
        int     ar_mode;
        long    ar_size;
};
```

Der Name ist eine mit 0 beendete Zeichenkette; das Datum hat die Form von time(2); Der Nutzer-ID und der Gruppen-ID sind Zahlen; der Modus ist ein Bitmuster laut chmod(2); Size wird gezaehlt in Bytes.

Jedes File beginnt auf einer Wortgrenze. Ein 0-Byte wird, falls erforderlich, zwischen den Files eingefuegt. Size gibt die Groesse des Files ohne das eventuelle Auffuellungszeichens an.

Achtung! Es gibt keine Vorsorge fuer leere Felder in einem Archiv-File.

SIEHE AUCH

ar(1), ld(1), nm(1)

FEHLERQUELLEN

Kodierung der Nutzer- und Gruppen-ID's mit Zeichen fuehrt zu Fehlern.

CORE(5) **CORE(5)**

NAME

core - Format des Speicherabbild-Files

BESCHREIBUNG

MUTOS gibt ein Speicherabbild des abgebrochenen Prozesses aus, falls irgendein Fehler aufgetreten ist. Siehe signal(2), wo die Ursachen fuer die Ausgabe eines Speicherabzugs aufgelistet sind. Im allgemeinen sind es Speicherfehler, illegale Befehle, Busfehler und durch den Nutzer erzeugte Quittungssignale. Der Speicherabzug wird core genannt und in das Arbeitsdirectory des Prozesses geschrieben.

Die ersten 1024 Bytes des Speicherabbildes sind eine Kopie der nutzerspezifischen Daten des Prozesses innerhalb des Systems einschliesslich der Registerinhalte zum Zeitpunkt des Abbruches. Das Format dieses Feldes ist aus den Systemtabellen ersichtlich. Der Rest repraesentiert den aktuellen Inhalt des vom Nutzer belegten Speicherbereiches zum Zeitpunkt des Abbruchs.

Im allgemeinen reichen die Informationen des Speicherabbildes aus, um sie mit dem Debugger adb(1) zu analysieren.

SIEHE AUCH

adb(1), signal(2)

NAME

dir - Format der Directories

UEBERSICHT

```
#include <sys/dir.h>
```

BESCHREIBUNG

Ein Directory verhaelt sich wie ein gewoehnliches File, ist aber geschuetzt, so dass kein Nutzer in ein Directory scheiben kann. Die Kennzeichnung, dass es sich um ein Directory handelt, erfolgt durch ein Bit im Flag-Wort seiner I-Node-Eintragung (siehe filsys(5)). Die Struktur einer Directory-Eintragung, die im Include-File angegeben ist, lautet:

```
#ifndef DIRSIZ
#define DIRSIZ14
#endif
struct  direct
{
        ino_t d_ino;
        char  d_name[DIRSIZ];
};
```

Somit besteht jede Directoryeintragung aus der i-Nummer, die auf das i-node verweist (dort wird die Datei exakt beschrieben) und dem jeweiligen Dateinamen.

Die ersten Eintragungen in jedem Directory sind "." und "..". Die erste Eintragung bezieht sich auf das Directory selbst, die zweite auf das uebergordnete Directory. Geht man mit ".." den Pfad nach oben weiter, gelangt man letztlich zum Root-Directory (Ursprungsdirectory) des Filesystems. Da fuer eingegliederte (mount) Filesysteme keine uebergeordneten Bezugnamen zugelassen sind und auch das Wurzelfilesystem von MUTOS 8000 (root) kein uebergeordnetes Directory besitzt, hat im Root-Directory die Eintragung ".." die gleiche Bedeutung wie ".".

SIEHE AUCH

filsys(5)

NAME

environ - Umgebung

UEBERSICHT

```
extern char **environ;
```

BESCHREIBUNG

Zu einem Feld von Zeichenketten, das "Environment" (Umgebung) genannt wird, kann in einem Prozess zugegriffen werden. Die globale Variable environ ist ein Zeiger auf ein Feld von Zeichenkettenzeigern, die die Umgebung des Prozesses bilden. Entsprechend Vereinbarung haben diese Zeichenketten das Format "name = value" und werden durch ein Nullbyte abgeschlossen.
Die folgenden Namen werden durch verschiedene Kommandos benutzt und durch login(1) gesetzt:

PATH — Ist eine Folge von Directories, die sh(1) verwendet, um ausfuehrbare Files, die mit einem unvollstaendigen Pfadnamen benannt wurden, zu suchen. Die Directories werden durch ';' getrennt.

Login(1) setzt: PATH=:/bin:/usr/bin.

HOME — Ist der Pfadname des Home-Directory des Nutzers, das mit login(1) entsprechend Password-File gesetzt wird.

Durch das export-Kommando und 'name=value' - Argumente in sh(1) oder durch exec(2) koennen weitere Namen in die Environment-Liste eingebracht werden. Eine Aenderung der Zuordnung in niederen Prozessen hat keine Auswirkung auf die uebergeordneten Prozesse.
Man sollte beachten, dass keine Konflikte mit bestimmten Shell-Variablen entstehen, die haeufig durch 'profile' exportiert werden, wie MAIL, PS1, PS2, IFS.

SIEHE AUCH

exec(2), sh(1), login(1)

NAME

filsys, flblk, ino - Format der Filesystem-Speicherung

UEBERSICHT

```
#include <sys/types.h>
#include <sys/flbk.h>
#include <sys/filsys.h>
#include <sys/ino.h>
```

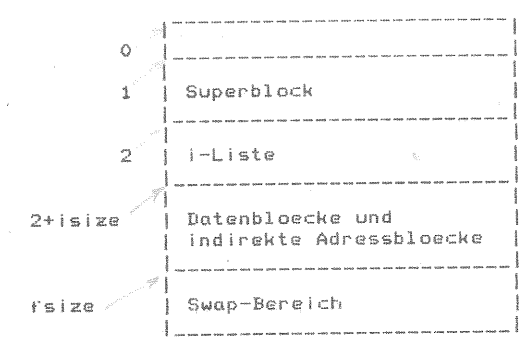
BESCHREIBUNG

Jedes Filesystem, wird auf dem Datentraeger Diskette in Bloecken von 512 Byte gespeichert.
Block 0 wird bei MUTOS 8000 nicht benutzt (sonst Urlader).
Block 1 ist der Superblock. Das Format des Superblocks, das im Include-File (sys/filsys.h) definiert ist, lautet:

```
/*
 * Structure of the super-block
 */
struct filsys {
        unsigned short s_isize;     /* size in blocks      */
                                    /* of i-list           */
        daddr_t  s_fsize;           /* size in blocks of   */
                                    /* entire volume       */
        short    s_nfree;           /* number of addresses */
                                    /* in s_free           */
        daddr_t  s_free[NICFREE];   /* free block list     */
        short    s_ninode;          /* number of i-nodes   */
                                    /* in s_inode          */
        ino_t    s_inode[NICINOD];  /* free i-node list    */

        char     s_flock;           /* lock during free    */
                                    /* list manipulation   */
        char     s_ilock;           /* lock during i-list  */
                                    /* manipulation        */
        char     s_fmod;            /* super block         */
                                    /* modified flag       */
        char     s_ronly;           /* mounted read-only   */
                                    /* flag                */
        time_t   s_time;            /* last super block    */
                                    /* update              */
        /* remainder not maintained by this version of     */
        /* the system                                      */
        daddr_t  s_tfree;           /* total free blocks   */
        ino_t    s_tinode;          /* total free inodes   */
        short    s_m;               /* interleave factor   */
        short    s_n;               /* " "                 */
        char     s_fname[6];        /* file system name    */
        char     s_fpack[6];        /* file system pack    */
                                    /* name                */
```

s_isize entspricht der Blockanzahl der i-Liste, die die i-nodes enthaelt und im Block 2 beginnt. Die Laenge von s_isize wird aus der absolut verfuegbaren Blockan-zahl ermittelt.
s_fsize entspricht der Anzahl der verfuegbaren Bloecke eines Filesystems. s_fsize stellt dann die Adresse eines Swap-Bereiches (auf der root) dar, wenn s_fsize kleiner als die Anzahl absolut verfuegbarer Bloecke auf der Diskette ist.



Die Freiliste ist wie folgt aufgebaut:

Das s_free Feld im Superblock enthaelt, in s_free[1],..., s_free[s_nfree-1], bis zu NICFREE freie Blocknummern. Die Konstante NICFREE ist konfigurationsabhaengig (z.B. 50).
s_free[0] ist die erste Blockadresse einer Reihe weiterer Bloecke, der Freiliste. Diese sind unter die normalen Datenbloecke gemischt. Das Format dieser Bloecke ist im Include-File (sys/fblk.h) definiert:

```
struct fblk
{
        int     df_nfree;
        daddr_t df_free[NICFREE];
};
```

Die Variablen df_nfree und df_free werden darin genauso behandelt wie s_nfree und s_free im Superblock.

Zuweisen eines Blockes:

Soll ein Block vergeben werden, ist s_nfree zu dekrementieren , die neue Blocknummer ist dann s_free[s_nfree].

Wurde s_nfree = 0, muss ein neuer Block (s_free[0]) in s_nfree und s_free eingelesen werden.

Wird die Blockadresse 0 bereitgestellt, exisitieren keine weiteren verfuegbaren Bloecke mehr, ein Fehler wird angezeigt.

Freiziehen eines Blocks:

Bei der Freigabe eines Blocks ergibt sich s_free[s_nfree] aus der freigewordenen Blockadresse, s_nfree wird inkrementiert.
Falls s_nfree gleich NICFREE ist, wird s_nfree und s_free in den freigewordenen Block geschrieben und s_nfree auf 0 gesetzt. (s_free[0] zeigt dann auf die Freiliste im soeben geschriebenen Block).

S_ninode im Superblock ist die Anzahl der freien i-Nummern in s_inode, dem Feld der freien i-Nummern.

Zuweisen eines i-Nodes:

Soll eine i-Node vergeben werden, ist s_ninode zu dekrementieren. Die i-Nummer ergibt sich dann aus s_inode[s_ninode].
War s_ninode 0, muss die i-Liste gelesen werden, um daraus alle freien i-nodes zu ermitteln, die dann in das Feld s_inode (bis NICINOD) einzutragen sind.

Freiziehen eines i-nodes:

Das freigewordene i-node ist an die Stelle s_inode[s_ninode] zu schreiben und s_ninode zu inkrementieren. Wurde bereits NICINODE erreicht, wird kein Erweitern des Feldes vorgenommen, da es lediglich zum Beschleunigen des Zuweisungsprozesses dient. Ob ein i-node tatsaechlich frei ist, ergibt sich letztlich aus dem i-node selbst.

S_flock und s_ilock im Superblock sind Flags in der Speicherkopie des Filesystems waehrend es in das System eingegliedert (mounted) ist. Ihre Werte auf der Diskette sind uninteressant.
Der Wert von s_fmod wird ebenfalls nur speicherintern gefuehrt. Er zeigt an, dass der Superblock geaendert worden ist. Bei der naechsten periodischen Uebertragung sollten die Filesysteminformation auf Diskette ausgegeben werden.

s_ronly ist Schreibschutz-Indikator; sein "Diskettenwert" ist unwichtig.
s_time ist die Zeit, bei der der Superblock des Filesystems das letzte Mal aktualisiert wurde.
Die Felder s_tfree, s_tinode, s_fname und s_fpack werden nicht laufend aufgebaut.

Die "interleave"-Faktoren s_m und s_n geben den logischen Blockabstand der Freibloecke innerhalb eines bestimmten Bereiches an (Standard 3,500).

i-nodes

Die Speicherung von i-nodes in der i-Liste erfolgt ab Block 2.

Jedes i-node repraesentiert ein File. I-nodes werden mit 1 beginnend fortlaufend numeriert (i-Nummer). Die Anzahl ist abhaengig vom Umfang des Dateisystems. Im i-node 1 koennen Fehlerbloecke abgelegt sein, die bei mkfs(1) festgestellt wurden. I-node 2 wird fuer die Root-Direktory des Filesystems reserviert, andere Festlegungen existieren nicht.

I-nodes sind 64 Bytes lang und haben folgende Struktur (vgl. <sys/ino.h>):

```
/*
 * Inode structure as it appears on
 * a disk block.
 */
struct dinode
{
        unsigned short   di_mode;  /* mode and type of  */
                                   /* file              */
        short  di_nlink;           /* number of links to*/
                                   /* file              */
        short  di_uid;             /* owner's user id   */
        short  di_gid;             /* owner's group id  */
        off_t  di_size;            /* number of bytes in*/
                                   /* file              */
        char   di_addr[40];        /* disk block        */
                                   /* addresses         */
        time_t di_atime;           /* time last accessed*/
        time_t di_mtime;           /* time last modified*/
        time_t di_ctime;           /* time created      */
};
#define INOPB   8                  /* 8 inodes per block*/
/*
 * the 40 address bytes:
 *       39 used; 13 addresses
 *       of 3 bytes each.
 */
```

Di_mode kennzeichnet die Art des Files (Modus), die entsprechend dem st_mode Feld von stat(2) verschluesselt wird.
Di_nlink ist die Anzahl von Direktory-Eintragungen (Links), die zu diesem i-node zugreifen.
Di_uid und di_gid sind die Nutzer- und Gruppen-IDs des Eigentuemers.

Size gibt die Laenge des Files in Bytes an.

Das Feld di_addr enthaelt (fuer regulaere Files und Directories) 10 direkte Blockadressen (je 3 Byte gepackt), die durch das File belegt werden. Die letzten 3 Adressen sind einfach, zweifach und dreifach indirekte Blockadressen, die jeweils auf einen Block von 128 Blockzeigern verweisen.

Die Zeiger in den Indirektbloecken haben den Typ daddr_t (vergleiche types(5)).
Wird von einem File der Block b belegt, muessen nicht unbedingt auch die Bloecke kleiner b benutzt werden.

Tritt die Blocknummer 0 in den i-nodes oder in einem der Indirekbloecke auf, so wurde der korrespondierende Block noch niemals zugewiesen. Solch ein fehlerhafter Block wird so gelesen, als enthielte er nur leere Worte.

Di_atime und di_mtime sind die Zeiten des letzten Zugriffes und der letzten Modifikation des Fileinhalts durch read(2), write(2) oder create(2) (siehe times(2)).
Di_ctime kennzeichnet die Zeit der letzten Modifikation bezogen auf den i-node bzw. das File und sagt aus, ob es zu uebetragen ist.

Special-Files werden im Modus gekennzeichnet. Ein Special-File vom "Block-Type" kann wie ein Filesystem eingegliedert (mounted) werden, ein Special-File vom Character-Type jedoch nicht, obwohl es nicht unbedingt zeichenorientiert sein muss. Bei Special-Files enthaelt das di_addr Feld den Geraetekode (siehe types(5)). Die Geraetekodes der Block- und Character-Special-Files ueberlappen sich.

SIEHE AUCH

mkfs(1), mknod(1), icheck(1), dcheck(1), dir(5), mount(1), stat(2), types(5), inet(7), update(8)

NAME

group - Gruppen-File

BESCHREIBUNG

Group beinhaltet fuer jede Gruppe folgende Informationen:

Gruppen-Name,
verschluesseltes Passwort,
numerische Gruppen-ID,
eine durch Komma getrennte Liste aller in der Gruppe zugelassenen Nutzer.

Das group-File ist ein ISO-File, seine Felder sind durch Doppelpunkte (:) getrennt. Jede Gruppe ist von der naechsten durch ein Newline getrennt. Ist das Passwortfeld Null, wird kein Passwort verlangt.

Dieses File ist im Directory /etc abgelegt. Da die Passworte verschluesselt sind, hat es allgemeine Leseerlaubnis und kann benutzt werden, um z.B. die numerischen Gruppen-IDs den entsprechenden Namen zuzuordnen.

FILES

/etc/group

SIEHE AUCH

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

NAME

mtab - Tabelle der eingegliederten Filesysteme

BESCHREIBUNG

Mtab befindet sich im Directory /etc und beinhaltet eine Tabelle von Geraeten, die durch das mount-Kommando eingegliedert wurden. Diese Tabelle wird nur zur Einsichtnahme durch den Nutzer gefuehrt.
Umount loescht die Eintragungen.

Jede Eintragung ist 64 Bytes lang.
Die ersten 32 Bytes sind der mit Nullen aufgefuellte Name des Directory, in dem das Specialfile eingegliedert (mounted) ist.

Die zweiten 32 Bytes sind der mit Nullen aufgefuellte Name des Specialfiles selbst. Es wird nur der Name des Specialfile ohne den vollstaendigen Pfadnamen gespeichert.
Mtab wird durch das Kommando mount eingerichtet, falls es noch nicht existiert.
Es ist nicht moeglich, ein Filesystem einzugliedern, wenn die entsprechende Eintragung schon vorhanden ist bzw. auszugliedern, wenn der Name nicht gefunden wird.

FILES

/etc/mtab

SIEHE AUCH

mount(1)

PASSWD(5) **PASSWD(5)**

NAME

passwd - Passwort-File

BESCHREIBUNG

Passwd beinhaltet fuer jeden Nutzer folgende Informationen:

Name (Login-Name, enthaelt keinen Grossbuchstaben),
verschluesseltes Passwort,
numerische Nutzer-ID,
numerische Gruppen-ID,
Leeerfeld (systemintern genutzt),
Home-Direktory des Nutzers,
ein Programm, das anstelle von Shell genutzt wird.

Es handelt sich um ein ISO-File. Jedes Feld innerhalb einer Nutzereintragung ist durch einen Doppelpunkt (:) getrennt. Jeder Nutzer ist vom naechsten durch ein Newline getrennt. Ist das Passwortfeld Null, wird kein Passwort verlangt. Ist das Shell-Feld Null, wird Shell selbst benutzt.

Dieses File ist in der Directory /etc abgelegt. Da die Passworte verschluesselt sind, hat es allgemeine Leseerlaubnis und kann z.B. zum Zuordnen der numerischen Nutzer-IDs zu den Namen verwendet werden.

FILES

/etc/passwd

SIEHE AUCH

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

NAME

ttys - Terminal-Initialisierungsdaten

BESCHREIBUNG

Bei jedem Systemanlauf wird durch init(8) ein Prozess fuer jedes im File ttys spezifizierte Terminal eroeffnet. Dies ist die Voraussetzung, dass sich dort ein Nutzer anmelden kann (login(1)). Jede Zeile spezifiziert ein Terminal.

Das 1. Zeichen einer Zeile gibt an, ob die Zeile ausgewertet (1) oder ignoriert (0) werden soll. Das 2. Zeichen wird als Argument fuer getty(8) benutzt. Gewoehnlich ist das 2. Zeichen "0"; andere Zeichen werden bei Terminals benutzt, bei denen die Geschwindigkeitserkennung nicht notwendig ist oder die Specialcharakteristiken haben. Getty muss in einem solchen Fall die entsprechenden Argumente kennen.
Der Rest der Zeile ist der Name des Specialfiles des jeweiligen Terminals im Directory/dev.
Das Single User System "MUTOS 8000" benutzt nur ein Terminal (console).

FILES

/etc/ttys

SIEHE AUCH

init(8), getty(8), login(1)

NAME

types - Einfache Systemdatentypen

UEBERSICHT

#include <sys/types.h>

BESCHREIBUNG

Die Datentypen, wie sie im Include-File definiert sind, werden im MUTOS-System-Kode benutzt. Einige dieser Datentypen sind dem Nutzer zugaenglich:

```
typedef  long            daddr_t;     /* disk address    */
typedef  char *          caddr_t;     /* core address    */
typedef  unsigned int    ino_t;       /* i-node number   */
typedef  long            time_t;      /* a time          */
typedef  int             label_t[9];  /* program status  */
typedef  int             dev_t;       /* device code     */
typedef  long            off_t;       /* offset in file  */
     /* selectors and constructor for device code       */
#define  major(x)        (int)(((unsigned)x)>>8)
#define  minor(x)        (int)(x&0377)
#define  makedev(x,y)    (dev_t)((x)<<8|(y))
```

Die Form daddr_t wird fuer Floppyadressen benutzt, ausser in einem I-Node auf dem Floppy, siehe filsys(5). Zeiten werden in Sekunden entschluesselt, ab 00:00:00 GMT, January 1, 1970. Der Major-Teil des Geraetekodes spezifiziert die Art und der Minor-Teil die Unit-Nummer des Geraetes; beide sind installationsabhaengig. Offsets werden vom Beginn eines Files aus in Bytes gemessen. Die label_t-Variablen werden zum Retten des Prozessor-Status benutzt, waehrend andere Prozesse laufen.

SIEHE AUCH

filsys(5), time(2), lseek(2), adb(1)

UTMP(5) **UTMP(5)**

NAME

utmp, wtmp = Login-Saetze

UEBERSICHT

#include <utmp.h>

BESCHREIBUNG

Das utmp File gibt darueber Auskunft, wer sich augenblicklich im System befindet. Es ist nur sinnvoll bei Multiuserbetrieb. Das File hat folgende (im Include-File definierte) Struktur:

```
struct utmp {
     char ut_line[8];          /* tty name */
     char ut_name[8];          /* user id */
     long ut_time;             /* time on */
};
```

Diese Struktur enthaelt den Namen des Special-Files, bezogen auf das Terminal des Nutzers, den Login-Name des Nutzers und die Zeit des Login in Form von time(2).

Das wtmp File bewahrt alle Logins und Logouts. Sein Format enspricht utmp. Ein geloeschter Nutzername kennzeichnet ein Logout auf dem verbundenen Terminal.
Desweiteren kennzeichnet der Terminalname '.', dass das System zu der angegebenen Zeit "rebootet" wurde. Das angrenzende Paar von Eintragungen mit den Terminalnamen '|' und '}' kennzeichnet die systemgestuetzte Zeit gerade vor und genau nach einem date Kommando, mit dem die Systemausgangszeit geaendert wurde.

Wtmp wird durch login(1) und init(8) aufgebaut, jedoch nicht kreiert. Das Aufzeichnen der Login-Records entfaellt, falls das File nicht existiert. Durch ac(1) wird dieses File ausgewertet.

FILES

/etc/utmp /usr/adm/wtmp

SIEHE AUCH

login(1), getlogin(3), init(8), who(1), ac(1)

HIER(7) HIER(7)

NAME

hier - Hirarchie des Filesystems

BESCHREIBUNG

Die folgende Darstellung gibt eine kurze Uebersicht ueber eine repraesentative Directory-Hirarchie.

	root	Wurzelfilesystem
/dev/	Geraete (4)	
	console	Bedieneinheit des Rechners, tty(4)
	kmem	Hauptspeicher des BS-Kerns, kmem(4)
	mem	Hauptspeicher, mem(4)
	swap	Folienspeicher, auf dem der Swap-Bereich resident ist
	null	Datensenke, null(4)
	rk*	Folienspeicher, rk(4)
	rrk*	Folienspeicher, "raw"-Interface, rk(4)
	lp	Drucker, lp(4)
	lp0	Drucker fuer nroff-Texte, nroff(1)
	rlp	Drucker, "raw"-Interface, lp(4)
	tty*	Terminals, tty(4)
	tty	steuerndes Terminal
/bin/	wichtige residente Dienstprogramme (weitere unter /mnt/bin/, /mnt/etc/ und /usr/bin/)	
	cat	Ausgabe und Verketten von Files, cat(1)
	cp	Kopieren von Files, cp(1)
	date	Setzen Datum, date(1)
	kill	Prozessabbruch, kill(1)
	login	Anmelden des Nutzers, login(1)
	ls	Listen Directory, ls(1)
	rm	Loeschen Files, rm(1)
	sh	Kommandointerpreter, sh(1)
	stty	Setzen Terminal-Optionen, stty(1)
	sync	Aktualisieren Filesystem, sync(1M)

/etc/ Wesentliche Daten und kritische Dienstprogramme

getty	Teil von login(1),getty(8)
group	Gruppenfile, group(5)
init	Vater aller Prozesse, init(8)
motd	Aktuelle Nachrichten
mount	Eingliedern Filesystem, mount(1)
umount	Ausgliedern Filesystem, umount(1)
mtab	Tabelle montierter Filesysteme, mtab(5)
passwd	Passwort-File, passwd(5)
ttys	Terminal-Initialisierung, ttys(5)
utmp	Tabelle der aktiven Nutzer, utmp(5)
update	Periodische Aktualisierung Filesystem, update(8)

/tmp/ Temporaere Files. Maximalen Bereich freihalten, falls erforderlich ein anderes, weniger belegtes Filesystem auf tmp montieren.
Wird genutzt z. B. von

e*	ed(1)
ctm*	cc(1)

/.profile Shell-Kommandofile. Hier sollte ein Suchpfad spezifiziert werden, der alle Directories einschliesst. Die Umgebung von Shell kann gesetzt werden, sh(1), environ(5).

/mnt/ Allgemeines Directory, die 'Utility-Diskette' wird normalerweise darauf montiert.

/mnt/bin/ Weitere Dienstprogramme zu /bin/

adb	Debugger, adb(1)
ar	Archivar, ar(1)
chmod	Aendern Zugriffsrechte, chmod(1)
chown	Aendern Eigentuemer, chown(1)
cmp	Vergleich von Files, cmp(1)
convert	Konvertieren Filesystem von bzw. zum K1600, convert(1)
dd	Konvertieren Files, dd(1)
df	Freier Bereich auf Filesystem, df(1)
echo	Echo der Argumente, echo(1)
ed	Editor, ed(1)
hex	Ausgabe Files hexa, hex(1)
ln	Erzeugen links, ln(1)
mkdir	Erzeugen Directory, mkdir(1)
mv	Umbenennen von Files, mv(1)
ncheck	Ermitteln von Pfadnamen, ncheck(1M)
nm	Ausgabe Symboltabelle, nm(1)
od	Oktalausgabe von Files, od(1)

passwd	Aendern Nutzer-Passwort, passwd(1)
pr	Ausgabe von Files, pr(1)
ps	Prozess-Status, ps(1)
pstat	Ausgabe Systeminformationen (1M)
pwd	Aktuelles Directory, pwd(1)
rmdir	Loeschen Directory, rmdir(1)
size	Groesse eines Objektfiles, size(1)
strip	Entfernen Symbole, strip(1)
wc	Wortzaehler, wc(1)

/mnt/etc/ Kritische Dienstprogramme

clri	Loeschen i-node, clri(1)
dcheck	Pruefen Directories, dcheck(1)
icheck	Pruefen Speicherbelegung Filesystem icheck(1M)
mkfs	Anlegen Filesystem, mkfs(1M)
mknod	Anlegen Special-File, mknod(1M)

/usr/ Allgemeines Directory. Die Compiler-Diskette muss unter diesem Namen montiert werden.

/usr/bin/ Dienstprogramme, die in Zusammenhang mit dem C-Compiler stehen.

as	PLZ/ASM-Assembler, as(1)
cc	C-Compiler-Exekutive, siehe auch unter /usr/lib/ und cc(1)
ed	Editor, ed(1)
ld	Lader, ld(1)

/usr/include/ Auswahl der Standard-Include-Files (2,3,3S)

a.out.h	Objektmodulformat, a.out(5)
ar.h	ar(5)
ctype.h	
exexarpg.h	
math.h	
pwd.h	
sgtty.h	ioctl(2)
signal.h	signal(2)
stdio.h	standard-E/A, stdio(3)
symbol.h	
sys/stat.h	stat(2)
sys/types.h	types(5)
time.h	

/usr/lib/ Objekt-Bibliotheken und anderes

libc.a	Systemrufe, Standard-E/A usw. (2,3,3S)
c[012]	C-Compiler-Paesse, cc(1)
cpp	C-Praeprozessor
crt0.o	Anlauf-Modul fuer C-Programme

NAME

ISO-Tabelle des ISO-Zeichensatzes

BESCHREIBUNG

Nachfolgend wird die Tabelle des benutzten ISO-Zeichensatzes hexadezimal dargestellt.

	0	1	2	3	4	5	6	7
00	nul	del	sp	0	@	P	`	p
01	soh	dc1	!	1	A	Q	a	q
02	stx	dc2	"	2	B	R	b	r
03	etx	dc3	#	3	C	S	c	s
04	eot	dc4	$	4	D	T	d	t
05	enq	nak	%	5	E	U	e	u
06	ack	syn	&	6	F	V	f	v
07	bel	etb	'	7	G	W	g	w
08	bs	can	(	8	H	X	h	x
09	ht	em	)	9	I	Y	i	y
0A	lf	sub	*	:	J	Z	j	z
0B	vt	esc	+	;	K	[	k	{
0C	ff	fs	,	<	L	\	l	|
0D	cr	gs	-	=	M	]	m	}
0E	so	rs	.	>	N	^	n	~
0F	si	us	/	?	O	_	o	del

NAME

ms - Makros zum Textformatieren

UEBERSICHT

nroff -ms [options] file ...

BESCHREIBUNG

Dieses Makro-Paketes unterstuetzt das Erzeugen von Texten fuer verschiedene Zwecke, wie Veroeffentlichungen, Dokumentationen u.s.w.. Durch geringfuegige Veraenderungen in einzelnen Makros kann der Nutzer das auszugebende Format mit seinen speziellen Anforderungen in Uebereinstimmung bringen (z.B. Gestalten des Seitenendes und Seitenkopfes, Laenge und Breite der auszugebenden Seiten, usw.). Dazu sind allerdings einige Kenntnisse ueber nroff(1) notwendig, die bei Benutzung des unveraenderten Makrosatzes nicht erforderlich sind.

Die folgenden nroff-Befehle koennen nach dem ersten .PP weiterhin genutzt werden:

```
.bp   Beginn einer neuen Seite
.br   Ende der aktuellen Ausgabezeile (break)
.sp n Einfuegen von n Leerzeilen
.ls n Zeilenabstand: n=1 einfach, n=2 doppelt
.na   Kein Ausrichten des rechten Randes
```

FILES

/usr/lib/tmac/tmac.s

SIEHE AUCH

nroff(1)

DIE SCHRIFTARTEN

Schriftart	englische Bezeichnung	Ausgabe mit A5120.16 auf Drucker 1152	1157
Normalschrift	Roman	X	X
Breitdruck	Bold	Fettdruck in Normalschrift	X
Schraegdruck	Italic	Normalschrift mit Unterstreichung	Normalschrift mit Unterstreichung

DIE MAKROS

Aufruf	Anfgs.-Wert	Break bei Auslassung	Erklaerung
.1C	ja	ja	Einspaltenformat auf einer neuen Seite
.2C	nein	ja	Beginn Zweispaltenformat
.AB	nein	ja	Anfang des Kurzreferates
.AE	-	ja	Ende des Kurzreferates
.AI	nein	ja	Die Arbeitsstelle des zuvor angegebenen Autors folgt.
.AU	nein	ja	Ein Autorenname folgt.

Aufruf	Anfgs.-Wert	Break bei Auslassung	Erklaerung
.B x	nein	nein	Ausgabe von x in der Schriftart "Breitdruck". Das auszugebende Argument sollte keine Umlaute enthalten. Andernfalls ist der Aufruf .B .SF Argument zu benutzen. Falls beim Aufruf von .B kein Argument angegeben ist, so wird auf die Schriftart "Breitdruck" umgeschaltet, d.h. die Ausgabe des weiteren Textes erfolgt im Breitdruck bis eine neue Schriftart eingestellt wird.
.B1	nein	ja	Beginn des eingerahmten Textes
.B2	nein	ja	Ende des eingerahmten Textes
.BR	-	ja	Anstelle von .br zur Zeilentrennung in .SH- und .NH-Ueberschriften verwenden.
.B'	Datum	nein	Text fuer Seitenende; wird automatisch am Ende einer Seite aufgerufen. Kann * veraendert werden.
.BX x	nein	nein	Eingerahmte Ausgabe von x
.DA x	nroff	nein	'Datum' am Seitenende ist x. Standard ist 'Heute'.
.DE	-	ja	Ende des Display-Textes; schliesst .KE ein.
.DS x	nein	ja	Start von Display-Text. Der folgende Text (bis .DE) wird als untrennbar aufgefasst und auf einer Seite ausgegeben. Die Eingabe-Zeilen werden nicht aufgefuellt. x =I (indented) - fuer eingerueckte Ausgabe des Textes (ist Standard), x = Ausgabe des des Textes am linken Rand der Seite, x =C (centered) - zentrierte Ausgabe des Textes, x =B zum Erzeugen eines links geraden Blockes, der dann zentriert ausgegeben wird; schliesst .KS ein.

Aufruf	Anfgs.-Wert	Break bei Auslassung	Erklaerung
.FE	-	ja	Ende einer Fussnote.
.FS	nein	nein	Start einer Fussnote. Die Fussnote wird am unteren Ende der Seite ausgegeben.
.I x	nein	nein	Analog .B - jedoch wird statt Breitdruck die Schriftart "Schraegdruck" (Kursiv-Schrift) angewaehlt.
.IP x y	nein	ja	Beginn eines eingerueckten Paragraphen mit vorangestelltem x. Die Einruecktiefe betraegt y ens (Standard bei nroff sind 5 Zeichen).
.KE	-	ja	Ende Keep-Modus. Der festgehaltene Text wird auf der folgenden Seite ausgegeben, falls auf der aktuellen Seite nicht genuegend Platz fuer den gesamten Text ist.
.KF	nein	ja	Start Floating-Keep. Falls der von nun an bis .KE festgehaltene Text auf einer neuen Seite ausgegeben werden muss, so wird der darauffolgende Text zum Fuellen der nicht vollstaendig genutzten Seite verwendet.
.KS	nein	ja	Start Keep-Modus. Der folgende Text wird bis .KE festgehalten.
.LP	ja	ja	Beginn eines linksbuendigen Abschnittes.
.ND date	-	nein	Keine Ausgabe des Datums am Seitenende; falls date angegeben ist, so wird das aktuelle Datum (fuer spaetere Benutzung) gleich date gesetzt.
.NH n	-	ja	Wie .SH, aber mit automatischem Numerieren der Ueberschrift. Die Nummern werden gestaffelt angegeben (z.B. 1.2.3.), wobei die Staffeltiefe durch n bezeichnet wird .(Standard ist 1).

Aufruf	Anfgs.-Wert	Break bei Auslassung	Erklaerung
.OC	nein	ja	Ausgabe des Inhaltsverzeichnisses. Auf einer neuen Seite wird ein Verzeichnis aller bis zum Aufruf dieses Makros aufgetretenen Ueberschriften (vom Typ SH und NH) mit Angabe der entspr. Seitennummer ausgegeben. Zweckmaessig ist ein Aufruf von OC am Ende des Textes.
.OM x y z	nein	nein	Fettdruck (Overprint), die drei moeglichen Argumente werden je dreifach ueberdruckt ausgegeben. Nach .OM ist Normalschrift eingestellt.
.OP x y z	nein	nein	Fettdruck (Overprint), Ausgabe von y dreifach ueberdruckt in der aktuell eingestellten Schriftart. x und z werden einfach in der davor gueltigen Schriftart ohne Zwischeraum vor bzw. nach y ausgegeben. Werden nur zwei Argumente angegeben, so wird das erste dreifach ueberdruckt in der aktuellen Schriftart und das zweite ohne Zwischenraum dazu einfach in der davor gueltigen Schriftart ausgegeben. Wird nur ein Argument angegeben, so wird dies in der aktuellen Schriftart dreifach ueberdruckt ausgegeben. Nach .OP ist die Schriftart "Normal" eingestellt.
.PP	nein	ja	Beginn eines Abschnittes; erste Zeile ist eingerueckt.
.PT	S.-Nr.*	-	Seitenueberschrift - automatisch zu Beginn jeder Seite ausgegebene Ueberschrift; kann veraendert werden.
.QE	-	ja	Ende des links und rechts eingerueckten Textes.
.QP	-	ja	Beginn eines einzelnen links und rechts eingerueckten Abschnittes.

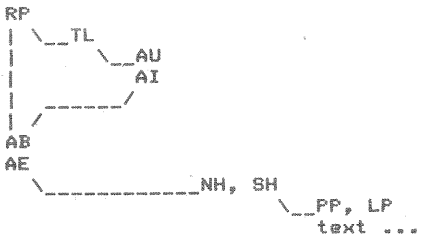
Aufruf	Anfgs.-Wert	Break bei Auslassung	Erklaerung
.QS	-	ja	Beginn von links und rechts eingeruecktem Text.
.R x y z	ja	nein	Analog .B - jedoch wird statt Breitdruck Normalschrift angewaehlt.
.RE	-	ja	Ende einer relativen Einrueckung.
.RP	nein	-	Bewirkt die Ausgabe eines Deckblattes mit Titel, Autorennamen u. - Institution sowie Kurzreferat; muss vor allen anderen Makros aufgerufen werden.
.RS	-	ja	Start einer weiteren Stufe relativer Einrueckung. Folgende .IP's werden von der gueltigen Einrueckung aus gemessen.
.SF x y z	-	nein	Wie .OP - aber alle Argumente werden nur einfach ausgegeben; dient zur Ausgabe einzelner Worte, die Umlaute enthalten (in den Schriftarten Breitdruck und Schraegdruck).
.SG x	nein	ja	Ausgabe je einer Zeile mit dem Namen eines der Autoren. x wird, falls angegeben, anschliessend ausgegeben (z.B. Initialen von Autor u. Schreibkraft).
.SH	-	ja	Der bis zum Beginn des naechsten Abschnittes angegebene Text wird als Ueberschrift in Normalschrift ueberdruckt ausgegeben. Soll eine Zeilentrennung vorgenommen werden, so sind die Zeilen durch .BR zu trennen.
.TA x...	5...	nein	Setzen neuer Tabulatormarken (in ens). Anfangswert fuer nroff sind die Positionen 5 10 15 ...

Aufruf	Anfgs.-Wert	Break bei Auslassung	Erklaerung
.TE	-	ja	Ende einer Tabelle.
.TH	-	ja	Ende der Ueberschrift einer Tabelle.
.TL	nein	ja	Die folgenden Zeilen (bis zu einem Makro) werden als Titel des Textes zentriert in Normalschrift ausgegeben (ggf. Zeilentrennung mittels .br).
.TS x	-	ja	Beginn einer Tabelle; falls x gleich H gesetzt ist, so hat die Tabelle eine sich auf jeder Seite wiederholende Ueberschrift. Diese besteht aus dem folgenden, bis .TH reichenden Text.

* S.-Nr. = Seitennummer

REIHENFOLGE

Werden bestimmte Makros benutzt, so ist folgende Reihenfolge ihres Aufrufes einzuhalten.



Makros, die in der Hirarchie hoeher liegen, sollten, falls sie benutzt werden, vor den weiter unten liegenden Makros aufgerufen werden. Makros, die senkrecht ueber einem anderen Makro liegen, muessen zur Einhaltung der vorgesehenen Form vor diesem Makro aufgerufen werden.

BOOT(8) BOOT(8)

NAME

boot - Start-Prozeduren

BESCHREIBUNG

Das MUTOS-8000-System wird in mehreren Phasen gestartet.

- Allgemeiner Start des BC A5120.16
  Hierbei wird die BOOT-Diskette in ein FD-Laufwerk eingelegt und durch Einschalten des BC A5120.16 (bzw. Restart) ein minimales BOOT-System geladen.
  Im Anschluss werden dem Nutzer dieses Komponenten zur Auswahl angeboten:

FORM.B	- Formatieren von BOOT-Disketten
COPY.B	- Kopieren von BOOT-Disketten
FORM.M	- Formatieren von MUTOS-Disketten
COPY.M	- Kopieren von MUTOS-Disketten
BOOT MUTOS	- BOOT-Prozeduren (Standard: SIO-Technik)
BOOT MUTOS.PIO	- BOOT-Prozeduren (PIO-Technik)

- BOOT-Prozeduren
  Durch Eingabe von BOOT MUTOS (bzw. BOOT MUTOS.PIO) werden die entsprechenden Programm-Teile des E/A-Sub-Systems geladen und aktiviert.
  Anschliessend erfolgt das Laden des MUTOS8000-Kerns auf den RAM (256 K). Fuer weitere Nutzer-Handlungen erscheint eine Mitteilung auf TTY.

  * BOOT-Diskette aus FD-Laufwerk entfernen
  * ROOT-Diskette in entsprechendes FD-Laufwerk einlegen.

  Ein Zuweisen der rkx (0...3) kann durch Eingeben neuer rk-Nummern (Minor) in der Reihenfolge der FD-Laufwerke (0...3) erfolgen. Danach wird durch Eingabe eines (r) (ready) der MUTOS8000-Kern gestartet (reset).
  Den Abschluss des Systemanlaufes bildet der Aufruf von /etc/init.

SIEHE AUCH

init(8)

NAME

COPY.B - Kopieren Boot-Diskette
COPY.M - Kopieren MUTOS-Diskette

UEBERSICHT

COPY.B [S TO D][v]
COPY.M [S TO D][v]

BESCHREIBUNG

Im Boot-System koennen Disketten physisch kopiert werden.

Ohne Optionen wird standardmaessig vom Laufwerk 0 nach Laufwerk 1 kopiert.
Mit der Option S TO D koennen andere Quell (S) und Ziellaufwerke (D) gewaehlt werden (0-3).

Die Option V (Verification) bewirkt gleichzeitig ein Kontrollesen der Duplikatdiskette.

Fuer Boot-Disketten ist COPY.B die einzige Vervielfaeltigungsmoeglichkeit.

Neben dem physischen Kopieren von MUTOS-Disketten mit dem Programm COPY.M koennen nach dem Systemanlauf von MUTOS8000 z.B. mit den Kommandos cp81), cat(1) logische Teile kopiert werden. Ein Filesystem auf der Zieldiskette ist dazu Voraussetzung (Kommando mkfs(1)).

SIEHE AUCH

cp(1),cat(1),mkfs(1),FORM(8),vk(4),boot(8)

NAME

crash - Massnahmen bei 'Systemabstuerzen'

BESCHREIBUNG

In diesem Abschnitt werden einige Massnahmen beschrieben, die nach einem 'Systemabsturz' eingeleitet und ausgefuehrt werden koennen. Es ist in jedem Fall davon auszugehen, dass kein ordnungsgemaesser Systemabschluss erfolgte.

Beim Restart nach einem Systemabsturz ist immer im Einzelnutzermodus zu arbeiten (siehe init (8)). Als erstes sollten die Kommandos dcheck und icheck(1) fuer alle zum Zeitpunkt des Absturzes montierten Filesysteme (einschliesslich Wurzel-Filesystem) abgearbeitet werden.

Werden dabei irgendwelche Unstimmigkeiten gefunden, sollten sie sofort beseitigt werden. Sind die Filesysteme in Ordnung, sollte das Datum ueberprueft, gegebenenfalls modifiziert und durch Eingabe von CTRL/Z in den Mehrnutzerbetrieb uebergegangen werden.

Um MUTOS ueberhaupt noch starten (boot) zu koennen, muessen drei Files (und die in der Hierarchie darueberliegenden Directories) in Ordnung sein. Zunaechst muss das Initialisierungsprogramm /etc/init vorhanden und ausfuehrbar sein.
Ist das nicht der Fall, steht die CPU im Nutzer-Modus auf Adresse 0x1000A im dynamischen Halt.Damit init richtig abgearbeitet werden kann, muessen die Files /dev/console und /bin/sh verfuegbar sein.
Kann das System nicht gestartet (boot) werden, muss ein anderes lauffaehiges System auf einem Reserve-Medium (Backup-Medium) vorhanden sein, mit dem dann gearbeitet werden kann. Das Root-Filesystem wird in diesem Fall wie ein montiertes Filesystem behandelt.

Reparatur einer Diskette. Die wichtigste Regel besagt, dass zerstoerte Disketten nicht montiert werden duerfen, ausser, wenn es unbedingt notwendig ist. Weiterhin wird empfohlen, eine Kopie herzustellen, ehe mit der Diskettenreparatur begonnen wird. Gibt es beim Herstellen einer Kopie Schwierigkeiten, muss evtl. die defekte Diskette im "raw"-Modus blockweise gelesen und auf die Sicherungsdiskette ausgegeben werden. Nicht lesbare Bloecke (512 Byte) sind dabei als Bloecke mit binaeren Nullen zu kopieren. Sollen die Inhalte einzelner Sektoren nicht lesbarer Bloecke gerettet werden, empfiehlt sich eine Behandlung unter einem dafuer geeigneten Betriebssystem (siehe rk(4)).
Die beim Abarbeiten des Kommandos icheck (1) ausgegebenen Mitteilungen werden in zwei Arten eingeteilt. Es koennen Unstimmigkeiten in der Liste der freien Speicherbereiche (free list) auftreten, wie z.B. doppelte Eintragungen in dieser Liste oder freie Speicherbloecke in Files. Diese Fehler werden mit dem Kommando icheck -s beseitigt. Ist derselbe Block in mehreren Files enthalten oder enthaelt ein File fehlerhafte (bad) Bloecke, dann sollten die dazugehoerigen Files mit clri(1) geloescht und die Liste der freien Speicherbereiche neu erzeugt werden. Ist irgendeines dieser Files jedoch sehr wichtig, kann auch versucht werden, es zuerst auf ein anderes Geraet zu kopieren.

Files, die mehr Directory-Eintragungen als "links" haben, werden mit dem Kommando dcheck gefunden, da mit dem Kommando clri nur ein spezieller Fall dieses Problems untersucht wird. Solche Situationen sind sehr gefaehrlich, daher sind alle Directory-Eintragungen des entsprechenden Files zu loeschen. Gibt es andererseits mehr "links" als Directory-Eintragungen, so besteht keine Gefahr der Ausbreitung des Fehlerzustandes. Es geht aber Platz auf der Diskette fuer die allgemeine Verwendung verloren. Es empfiehlt sich, zunaechst dieses File zu kopieren, wenn es eine Eintragung hat und noch gebraucht wird. Danach sind mit clri seine i-Node und mit rm alle existierenden Directory-Eintragungen zu loeschen.

Mit dem Kommando dcheck werden auch Files gefunden, die weder "links" noch Directory-Eintragungen haben. Sie erscheinen auf dem Root-Geraet, wenn das System angehalten wird, obwohl es noch offene Pipes gab, und in anderen Filesystemen, wenn das System angehalten wurde und noch Files enthaelt, die im geoeffneten Zustand geloescht wurden. Mit dem Kommando clri wird der i-Node geloescht und mit icheck -s jeder fehlerhafte Block (missing block) repariert.

Ursachen des Systemabsturzes. Bei den meisten Systemabstuerzen gibt MUTOS eine Meldung auf das Steuerterminal aus. In der folgenden Liste sind genug Informationen enthalten, um daraus Wege zur Fehlerbeseitigung ableiten zu koennen. Prinzipiell sind Hardware- oder Softwarefehler die Ursache. Die Meldungen haben die Form 'panic: ...', die je nach Moeglichkeit mit anderen Informationen ergaenzt werden.

blkdev
: Die Routine getblk wurde mit einem nicht existierenden Geraetetyp (major number) als Argument aufgerufen. Hardware- oder Softwarefehler.

devtab
: Als Geraetetyp (major number) fuer getblk wurde ein Nullgeraet angegeben. Hardware- oder Softwarefehler.

iinit
: Waehrend des Initialisierens trat beim Lesen des Superblockes fuer das Root-Filesystem ein E/A-Fehler auf.

out of inodes
: Ein montiertes Filesystem hat beim Erzeugen neuer Files keine i-Nodes mehr zur Verfuegung. Mit icheck ist zu pruefen, ob das Geraet noch verfuegbar ist.

no fs
In der Tabelle, die die montierten Geraete enthaelt, fehlt ein Geraet. Hardware- oder Softwarefehler.

no imt
Wie 'no fs', aber der Fehler wird an anderen Stellen ausgeloest.

no inodes
Die im Speicher enthaltene i-Node-Tabelle ist gefuellt. Es ist zu versuchen, im File param.h den Parameter NINODE zu erhoehen (Nutzerfehler).

swap error
Beim Auslagern entstand ein nicht korrigierbarer E/A-Fehler.

unlink - iget
Das Directory, das das zu loeschende File enthalten soll, wird nicht gefunden. Hardware- oder Softwarefehler.

out of swap space
Ein Programm muss ausgelagert werden, aber der Swap-Bereich ist gefuellt. Es ist zu versuchen, ihn zu vergroessern.

trap
Das System hat bei einer Unterbrechung einen Fehler erkannt. Er wird von folgender Angabe begleitet: Adresse ("seg" und "aps") der Speicherzelle, in die der Prozessorstatus zum Unterbrechungszeitpunkt gerettet wurde, Befehlszaehler (Segmentnummer und Verschiebung:"pc") und Prozessorstatus zum Unterbrechungszeitpunkt ("ps"), die vom Prozessor gelieferte Unterbrechungsidentifikation ("ident") und eine der folgenden "Trap"-Typen:

0 nicht autorisierte E/A-Unterbrechung
1 Befehl aus dem erweiterten Befehlssatz
3 Single Step
6 rekursiver Systemruf

parity
Waehrend der Arbeit des U8001-Prozessors ist ein Paritaetsfehler aufgetreten. Die Adresse des unterbrochenen Befehls wird ausgegeben.

SIEHE AUCH

clri(1), dcheck(1), icheck(1), init(8)

NAME

FORM.B - Formatieren Boot-Diskette
FORM.M - Formatieren MUTOS-Diskette

BESCHREIBUNG

Beide Programme koennen innerhalb des Boot-Systems benutzt werden. Sie dienen dem Formatieren von Disketten.

Mit dem Programm FORM.B koennen Disketten fuer das Boot-Systems formatiert werden.
Die Programmausschriften erfolgen in Grossbuchstaben. Das Diskettenlaufwerk (Ø...3) kann gewaehlt werden (Vorzugsweise LW 1). COPY.(8) dient nachfolgend zum Kopieren der Boot-Diskette.

Durch FORM.M wird eine Diskette fuer MUTOS8000 formatiert (die Ausschriften erfolgen in Kleinbuchstaben). Es koennen

- das Laufwerk (0...3, vorzugsweise 1) und
- die Sektorlaenge
  128 - 0
  512 - 1

gewaehlt werden.
Das Standardformat fuer MUTOS8000 ist 512 Byte/Sektor. Die Sektorlaenge von 128 Byte kann fuer den Informationsaustausch (Kompatibilitaet) zu anderen Anlagen genutzt werden (z.B. K1630,CM4,...).
Bei der Sektorlaenge von 512 Byte kann zusaetzlich die Sektorsequenz gewaehlt werden (1...4).

Nachfolgend koennen MUTOS-Disketten im Boot-System physisch kopiert werden (COPY.M(8)).
Zum allgemeinen Weiterverarbeiten in MUTOS 8000 muss mit dem Kommando mkfs(1) ein Filesystem erstellt werden.

SIEHE AUCH

COPY(8),mkfs(1),cp(1),rk(4),boot(8)

NAME

getty - Setzen des Terminal-Modus

UEBERSICHT

/etc/getty [char]

BESCHREIBUNG

Getty wird von init(8) unmittelbar aufgerufen. Es liest den 'login'-Namen des Nutzers und aktiviert login(1) mit dem Namen als Argument.

init(8) ruft getty mit einem einzelnen Zeichen als Argument auf, das aus der Eintragung fuer dieses Terminal in ttys(5) entnommen wird. Dieses Argument fuehrt zu einer Tabelle terminalspezifischer Parameter in getty. Sie enthaelt u.a. den Text der 'login:'-Meldung, die auch Zeichenfolgen enthalten kann, die das Terminal in einen bestimmten Zustand setzen.

Der sich anmeldende Nutzer beendet die Eingabe seines Namens mit CR, LF bzw. ENTER. Im ersten Fall wird fuer das Terminal CRMOD eingestellt (siehe ioctl(2)).
Der eingegebene Name wird durchsucht, ob er Kleinbuchstaben enthaelt. Ist das nicht der Fall, wird angenommen, dass das Terminal nur Grossbuchstaben kennt. Daraufhin wird LCASE eingestellt.

Abschliessend wird 'login' mit dem Namen des Nutzers als Argument aufgerufen.

Die folgenden Argumente aus dem ttys-File werden verstanden:

w Tastatur und BAB 2 des BC 5120.16
2 Fuer Terminals mit 9600 Baud Uebertragungsrate

SIEHE AUCH

init(8), login(1), stty(1), ioctl(2), ttys(5)

NAME

init, rc - Initialisieren der Prozesssteuerung

UEBERSICHT

/etc/init
/etc/rc

BESCHREIBUNG

Init wird in der letzten Stufe der Boot-Prozedur aufgerufen (siehe boot(8)). Im allgemeinen besteht seine Aufgabe darin, einen Prozess fuer jedes Terminal zu eroeffnen, von dem sich ein Nutzer anmelden (login) kann.

Wird init das erste Mal ausgefuehrt, wird das Steuerterminal /dev/console fuer Lesen und Schreiben geoeffnet und unmittelbar danach die Shell aufgerufen. Mit diesem Herangehen wird ein Einzelnutzersystem aufgebaut. Ist das Shell-Programm beendet, initialisiert init das Mehrnutzersystem und startet den folgend beschriebenen Prozess.

Hat init das Mehrnutzersystem initialisiert, wird die Shell aufgerufen und das Kommandofile /etc/rc abgearbeitet. Es fuehrt solche systeminternen Routinen wie Loeschen (removing) temporaerer Files, Montieren des Filesystems und Starten interner Filesystem-Synchronisation aus.
Danach liest init das File /etc/ttys und erzeugt (fork) einen Prozess fuer jedes im File spezifierte Terminal.
Jeder dieser Prozesse oeffnet sein Terminal fuer Lesen und Schreiben. Die Kanaele werden den Filedeskriptoren 0, 1 und 2 zugeordnet und bilden in dieser Reihenfolge die Standard-Eingabe, die Standard-Ausgabe und die Fehlerfiles. Dann wird /etc/getty mit dem durch das letzte Zeichen der entsprechenden Zeile im File ttys spezifizierten Argument aufgerufen. Getty liest den Nutzernamen und ruft login(1) auf, um den Nutzer anzumelden (login) und das Shell-Programm auszufuehren.
Zum Schluss beendet die Shell infolge eines EOF, das entweder explizit eingegeben oder im Resultat eines Hangup- Signals erzeugt wurde, seine Arbeit. Der Teil von init, der auf ein solches Ereignis wartet, wird gestartet. Er loescht (remove) die entsprechende Eintragung im File utmp (die aktuellen Nutzer) und erzeugt eine Eintragung im File /usr/adm/wtmp, indem alle An- und Abmeldungen (login) registriert werden. Dann wird das entsprechende Terminal erneut geoeffnet und getty wieder aufgerufen.

Init faengt das Hangup-Signal SIGHUP auf und interpretiert es als Aufforderung, das System aus dem Mehrnutzer- in den Einzelnutzerzustand zu ueberfuehren. Das Hangup-Signal wird mit "kill -1 1" gesendet.

FILES

/dev/tty?, /etc/utmp, /usr/adm/wtmp, /etc/ttys, /etc/rc

SIEHE AUCH

login(1), kill(1), sh(1), ttys(5), getty(8)

UPDATE(8) **UPDATE(8)**

NAME

update - Periodisches Aktualisieren des Filesystems

UEBERSICHT

/etc/update [on/off]

BESCHREIBUNG

Mit update kann bewirkt werden, dass der MUTOS-Kern intern alle 30 Sekunden ein sync (siehe sync (2)) ausfuehrt. Das sichert, dass das Filesystem bei einem Systemabsturz mit grosser Wahrscheinlichkeit aktualisiert ist.

Mit der on-Option wird die interne periodische Synchronisation ein-, mit der off-Option ausgeschaltet. Update, ohne Argument aufgerufen, zeigt an, ob die interne Synchronisation ein- oder ausgeschaltet ist.

FILES

/dev/kmem

SIEHE AUCH

sync(2), sync(1), init(8)