ANWENDER-DOKUMENTATION	Anleitung für den Systemverantwortlichen Inbetriebnahme und Wartung	MOS MUTOS 1700
11/87		

Programmtechnische Beschreibung Teil 2

Anleitung für den Systemverantwortlichen

Inbetriebnahme und Wartung

AC A 7100/A 7150

VEB Robotron-Projekt Dresden

C 8013-0001-1 M 3030

Die vorliegende Systemunterlagendokumentation, Programmtechnische Beschreibung Teil 2 Anleitung für den Systemverantwortlichen, Inbetriebnahme und Wartung, entspricht dem Stand von 11/87.

Die Ausarbeitung erfolgte durch ein Kollektiv des VEB Robotron-Elektronik Dresden.

Herausgeber:

VEB Robotron-Projekt Dresden, Leningrader Str. 9, Dresden 8010

Kurzreferat

Die vorliegende Schrift ist eine Anleitung zur Inbetriebnahme, Wartung und Pflege des Betriebssystems MUTOS 1700 (Multi-User-Timesharing-Operating-System) für den Arbeitsplatzcomputer A 7100/A 7150 des VEB Kombinat Robotron.

Inhaltsverzeichnis Seite

1. Inbetriebnahme

1.1. Auslieferungsform des Betriebssystems

MUTOS wird auf mehreren Disketten ausgeliefert, die den gesamten Umfang des Betriebssystems sowie die Hilfsmittel zum Installieren auf der Festplatte enthalten. Eine der Disketten enthält ein MUTOS-Root-Filesystem mit einem File /mutos, das durch das Monitor-Kommando "Boot" (b) gestartet werden kann. Diese Diskette hat folgenden Aufbau:

```
Spur    0         - FM,  16*128 Byte
Spuren  1 ... 159 - MFM, 16*256 Byte
```

Auf der Spur 0 befindet sich ein MUTOS-spezifisches Boot-Programm, ab Spur 1 existiert das Root-Filesystem im Format "dxmf". Alle anderen Disketten des Liefersatzes sind im Format

```
DS, DD, 80*9*512 Byte ("dmf"-Format).
```

Sie enthalten im MUTOS-tar-Format alle restlichen Komponenten des Betriebssystems, die sich nicht auf der Root-Diskette befinden.

Das Root-Filesystem umfaßt einen Basisumfang an Komponenten, der für das Installieren auf der Festplatte benötigt wird.
Es ist aber auch für einfache Arbeiten bei Nur-Disketten-Betrieb benutzbar. Der geringe freie Bereich auf der Root-Diskette begrenzt allerdings die Arbeitsmöglichkeiten. Im Abschnitt 1.5. wird erläutert, wie in diesem Fall gearbeitet werden kann.

1.2. Kopieren des Auslieferungssatzes

Sollen die Disketten vor der aktiven Nutzung gedoppelt werden, geschieht das am besten physisch unter einem anderen Betriebssystem. Als Beispiel wird das Kopieren unter SCP mittels COPYDISK beschrieben: (Eingaben sind mit <CR> abzuschließen)

Ausschrift von COPYDISK	Eingabe beim Kopieren	
	von ROOT1	der restl. Disketten
Source disk?	A	A
Using current drive parameters (N/<CR>)?	N	N
Cyls /Disk :80	<CR>	<CR>
Tracks/Cyl :2	<CR>	<CR>
Sectors/Track :16	<CR>	9
Bytes/Sector :256	<CR>	512
Track 0 handling? Yes	<CR>	N
Destination disk?	B	B
Format Destination Disk? (Y/N)	Y	Y
Interleave 1	4	4
Insert destination disk in drive B Type <CR> to continue	<CR>	<CR>
Insert source disk in drive A Insert destination disk in drive B Type <CR> to continue	<CR>	<CR>

Unter MUTOS wird folgende Herangehensweise empfohlen:

Root-Diskette: Mittels mkbf(8) wird auf einer leeren Diskette das Formatieren und das Schreiben des Boot-Programms auf die Spur 0 vorgenommen.
Das Root-Filesystem wird mit dd(1) physisch übertragen (spurweises Kopieren):

```
dd if=/dev/rdxmf0 of=/dev/rdxmf1 bs=4k
```

tar-Disketten: Formatieren der Disketten

```
format /dev/rdmf?
```

Physisches übertragen (2 Spuren bei jedem übertragungsschritt).

```
dd if=/dev/rdmf0 of=/dev/rdmf1 bs=9k
```

Sollen mehrere Kopien erstellt werden, kann auf einem System mit Festplatte auch folgender Weg beschritten werden:

übertragen des gesamten Disketteninhalts als File auf die Festplatte:

```
dd if=/dev/rdxmf? of=ROOT1 bs=4k (Root-Diskette)
dd if=/dev/rdmf? of=ROOT2 bs=9k (Bsp.für tar-Disketten)
```

Vorbereiten der Zieldisketten über mkbf(8) bzw. format(8) wie oben beschrieben.
übertragen der Files auf die neuen Disketten:

```
dd if=ROOT1 of=/dev/rdxmf? bs=4k (Root-Diskette)
dd if=ROOT2 of=/dev/rdmf? bs=9k (Bsp. für tar-Disketten)
```

Dieser Weg ist zeitlich effektiver, besonders beim Herstellen mehrerer Kopien.

1.3. Start des Betriebssystems

Die mit ROOT1 bezeichnete Diskette wird in ein Laufwerk (günstig: 0) eingelegt und dieses bereit gemacht. Das Monitor-Kommando "b<CR>" veranlaßt das Einlesen und den Start des Betriebssystemkerns (siehe auch boot(8)). Nach dem Austesten der vorgesehenen Anschlußsteuerungen auf ihr Vorhandensein und entsprechenden Ausschriften werden die generierten Werte für RAM-Floppy-Emulation, Root-, Pipe- und Swap-Gerät sowie für Beginn und Größe des Swap-Bereiches angezeigt. Diese Standardwerte können verändert werden. Es ist nicht erforderlich, wenn die Root-Diskette im Laufwerk 0 liegt. Dann kann die Abfrage

```
Do you want changes?
```

verneint werden (<CR>).
Das Betriebssystem meldet sich mit Angabe einer Versionsnummer und dem für Nutzerprozesse zur Verfügung stehenden Hauptspeicherbereich (in kByte). Nach

```
single user login:
```

meldet man sich mit "root<CR>" an. Daraufhin erscheint das Zeichen "#" vom Kommandoprozessor shell(sh(1)) als Kennzeichen der Eingabebereitschaft. Jede eingegebene Zeile ist mit <CR> oder <LF> abzuschließen. Vor Abschluß der Eingabezeile ist eine Zeichenirrung über die Taste DEL, eine Zeilenirrung über CTRL-U erreichbar.
In den nächsten beiden Abschnitten wird die unterschiedliche Herangehensweise beim Installieren des Betriebssystems auf der Festplatte und beim Vorbereiten einer Floppy-orientierten Arbeit erläutert.

1.4. Installieren des Betriebssystems auf der Festplatte

Als Voraussetzung für das Installieren des MUTOS auf der Festplatte muß auf dieser ein Master-Bootblock (Zylinder 0, Spur 0, Sektor 0) vorhanden und darin eine MUTOS-Partition eingerichtet sein. Ist dies nicht schon der Fall, ist die zum Gerät gehörende Diskette dafür zu nutzen.
Die zum Auslieferungsumfang gehörende Root-Diskette enthält Shell-Prozeduren und Programme für folgende Funktionen:

- Formatieren der MUTOS-Partition in geeigneter Form.
- Installieren des MUTOS-Boot-Programms auf der 1.Spur der MUTOS-Partition.
- Einrichten der Filesysteme für Root-, User- und Temp-Bereich als Subpartitions innerhalb der MUTOS-Partition.
- übertragen aller Files von den Disketten auf die Platte.

Nach dem Start des Betriebssystems von der Diskette entsprechend Abschnitt 1.3., wird das Installieren auf der Festplatte über die Shell-Prozedur mkhd(8) eingeleitet:

```
mkhd
```

Innerhalb dieser Prozedur wird zunächst das Programm hdinstall(8) abgearbeitet. Es testet aus, ob auf der Festplatte ein Master-Bootblock vorhanden und eine MUTOS-Partition eingerichtet ist. Die Parameter der Festplatte und die Partitionaufteilung werden angezeigt, bei Vorhandensein auch die Liste defekter Spuren. Danach wird abgefragt, ob ein Formatieren des MUTOS-Bereiches auf der Platte vorgenommen werden soll. Das ist bei Auslieferung des Gerätes für die gesamte Platte bereits erfolgt, und die Abfrage kann meist verneint werden. Sollten im späteren Betrieb zusätzliche Defektspuren auftreten oder sollte aus anderen Gründen eine Neuformatierung erforderlich sein, wird das durch Bejahung der Abfrage möglich.

> hdinstall zeigt während des Formatierungslaufs die aktuellen Zylinder/Kopf-Wertepaare absolut für die Platte, sowie (in Klammern) relativ zum Beginn der MUTOS-Partition an. Außerdem wird sichtbar gemacht, wenn vorgegebene defekte Spuren als solche formatiert werden. Zusätzlich erkannte defekte Spuren werden gemeldet und in die Defektspurliste aufgenommen.

Nach dem (eventuellen) Formatieren werden die Filesysteme für Root, User und Temp eingerichtet. Abfragen ermöglichen es, dies schrittweise zu übergehen. Bei der Erstinstallation müssen diese Schritte alle ausgeführt werden. Beim Einrichten der Filesysteme über mkfs(1) werden Blöcke, die auf defekten Spuren liegen, durch Eintragen in den Inode 1 von der späteren ausgeschlossen. Dieser Vorgang nimmt einige Zeit in Anspruch, da eine Schreib-/Leseprüfung erfolgt.
Als nächster Schritt wird das Root-Filesystem auf der Platte an /mnt angebunden. Dann wird das File mutos von der Diskette auf die Platte übertragen und dort den aktuellen Platten- und Partition-Parametern angepaßt. Zum Abschluß von hdinstall(8) wird noch ein Textfile diskpar in die Root-Directory auf der Platte geschrieben, das die Platten- und Partition-Parameter enthält, wie sie für die Generierung in /usr/sys/cfg/c5170.c benötigt werden. Die Zelle "Genboot" im File mutos wird rückgesetzt und die

Ausgabe des Anfangsdialogs beim Systemstart dadurch unterdrückt. Der weitere Ablauf wird wieder durch mkhd(8) gesteuert. Der Inhalt der Diskette ROOT1 wird auf die Platte übertragen, die Geräteeinträge in dev werden aufgebaut, und einige zusätzliche Schritte werden ausgeführt.
Zum Schluß der Installationsprozedur ist ein arbeitsfähiges MUTOS auf der Platte vorhanden, das unmittelbar gestartet werden kann. Der gesamte Vorgang erfolgt dialoggesteuert bzw. automatisch. Die Größen der Subpartitions werden nach folgenden Gesichtspunkten (automatisch) festgelegt:

Root: 30% der Gesamtpartition, max. jedoch 60 Zylinder
Swap: 10%
Temp: 10%
User: Rest der Gesamtpartition

mkhd(8) zeigt zum Abschluß an, wie ein Systemstart von der Platte vorgenommen werden kann. Danach geht das bis zu diesem Zeitpunkt laufende MUTOS von der Vertriebsdiskette in einen definierten Systemhalt.
Nach Neustart des Systems von der Platte ist die Shell-Prozedur install(8) zu starten, die den Inhalt der restlichen Vertriebsdisketten auf die Platte überträgt. Auch dies geschieht dialoggesteuert

> Sollte eine der Disketten übertragungsfehler anzeigen, ist eine der lesbaren tar-Disketten zweimal einzulegen. Natürlich fehlen dann Komponenten, doch kann so die install-Prozedur fortgesetzt werden. Nach Ersatz der defekten Diskette kann jederzeit nachträglich eine übertragung des Inhalts mit tar(1) erfolgen.

Aus Platzgründen hat das von der Root-Diskette übertragene File /mutos keine Symboltabelle mehr. Falls änderungen im von der Platte zu startenden System (z.B. über adb(1)) erfolgen sollen, kann das File /usr/sys/conf/mutos herangezogen werden, das bis auf die Werte für rootdev, swapdev, pipedev, swplo, nswap und Genboot dem aktuellen System entspricht. Für eine Neugenerierung sind die Werte aus /diskpar nach /usr/sys/cfg/c5170.c zu übernehmen. Zu beachten ist noch, daß nach dem Neustart des Systems von der Festplatte die Filesysteme

/dev/user an /usr

und

/dev/temp an /tmp

zu binden sind (mount(1)).

1.5. Hinweise zur Floppy-orientierten Arbeitsweise

Für diese Arbeitsweise ist es erforderlich, daß der Nutzer sein System so zusammenstellt, wie es seinem Aufgabenspektrum entspricht. Eine allgemeingültige Variante kann es aufgrund der begrenzten Externspeicherkapazität nicht geben.
Für diese Arbeiten sind gute Kenntnisse des MUTOS erforderlich. Sie sollten erfahrenen Systemprogrammierern vorbehalten bleiben. Einfache Nutzungen sind schon mit der Auslieferungsvariante möglich, selbst übersetzungen kleinerer Files.
Folgende Hinweise können gegeben werden:

1. Löschen aller Komponenten von der Root-Diskette, die für die Installation auf der Festplatte erforderlich sind (rm mkhd, install, hdinstall).
2. Aufbau einer getrennten Boot-Diskette, damit das File /mutos nicht auf der Root-Diskette geführt werden muß (s.mkbf(8)). Dabei ist es sinnvoll, das File sys/conf/mutos von der Diskette SYS zu nutzen, damit die Symboltabelle vorhanden ist, um änderungen über adb(1) vornehmen zu können.
3. Das Root-Filesystem ist im Format "dmf" oder "bdmf" aufzubauen, um die Diskette maximal auszunutzen.
4. Im Hauptspeicher wird ein Teil als Swap-Bereich reserviert. Seine Größe sollte etwa 20% des für Nutzerprozesse insgesamt zur Verfügung stehenden Bereiches betragen.
5. Es werden "User"-Disketten aufgebaut, die an /usr angebunden werden können (mount(1)) und aufgabenspezifisch zusätzliche Systemkommandos sowie Nutzerkomponenten enthalten.
   Beim Kopieren der Standardbibliothek /lib/libc.a ist so vorzugehen, daß die Bibliothek auf ein Filesystem mit möglichst viel freiem Platz kopiert und gleich danach ranlib(1) für das neue File gestartet wird.
6. Für bestimmte Zwecke ist es günstig, einen weiteren RAM-Bereich für das Einrichten eines Filesystems zu nutzen. Mit den RAM-Bereichen kann wie mit Externspeichern gearbeitet werden (mkfs(1), mount(1), umount(1)). über diese RAM-Filesysteme können Kopiervorgänge vorgenommen werden, Programme für schnelles Abarbeiten sind dort stationierbar, und als Bereich für temporäre Files erhöht diese Arbeitsweise die Abarbeitungsgeschwindigkeit.
7. Beim Vorhandensein von nur zwei Diskettenlaufwerken ist es nicht möglich, den gesamten Inhalt einer tar-Diskette der Auslieferungsform in einem Arbeitsgang einzulesen. In Teilen kann das Einlesen über einen RAM-Floppy-Bereich oder über die Root-Diskette erfolgen, wenn dort genügend Platz vorhanden ist.
8. Möglichkeiten der änderung einiger Systemparameter ohne Neugenerierung bestehen mittels adb(1) unter Nutzung des Files sys/conf/mutos (Diskette SYS des Auslieferungsumfangs), das die Symboltabelle des Kerns enthält.

Beispiel: (Bedienereingaben sind hervorgehoben)

```
adb -w mutos              Aufruf adb mit Schreiberlaubnis
~~~~~~~~~~~~

swapdev?*d                Abfrage "swapdev" dezimal
~~~~~~~~~~
_swapdev:    44           Ausgabe durch adb


?*w 256                   Major/Minor-Nr. von /dev/ram0
~~~~~~~                   eingeben
_swapdev:    054 = 0400   Ausgabe der änderung durch adb


swplo?*D                  Abfrage "swplo" long-dezimal
~~~~~~~~
_swplo:      1800         Ausgabe durch adb


?*W 0                     Neuer Wert von "swplo"
~~~~~
_swplo:      03410 = 0    Ausgabe der änderung durch adb


nswap?*d                  Abfrage "nswap" dezimal
~~~~~~~~
_nswap:      900          Ausgabe durch adb


?*w 112
~~~~~~~
_nswap:      0


ramcfg,8?*d               Abfrage der Tabelle "ramcfg"
~~~~~~~~~~~
_ramcfg:     0    0    0    0    0    0    0    0
                          Ausgabe von adb

ramcfg,2?*w  400 112      ändern des Wertpaares für /dev/ram0
~~~~~~~~~~~
_ramcfg:     0 = 0620     Ausgabe der änderung durch adb
_ramcfg+#2:  0 = 0160


$q                        Verlassen des adb
~~
```

Beim Boot-Prozeß des so modifizierten Systems wird /dev/ram0 als Swap-Gerät ausgewiesen. Der Swap-Bereich beginnt ab 0 innerhalb des Bereiches von 400 KByte bis 512 KByte und belegt diesen vollständig.
ähnliche änderungen sind für andere Parameter ebenfalls möglich, wenn sie Hauptspeicherplatz belegen. Die Header mit den entsprechenden Definitionen sind in /usr/include/sys enthalten.
Soll der Anfangsdialog unterdrückt werden, ist in analoger Weise zum obigen Beispiel die Zelle "Genboot" auf 0 zu setzen. Das ist in dem Fall günstig, wenn die betreffenden Parameter der Konfiguration angepaßt sind und nicht verändert werden sollen.

Als weiteres Beispiel für eine derartige änderung soll die Einstellung der Schrittrate für die verschiedenen Floppy-Laufwerktypen genannt werden. Dafür ist das File /usr/sys/cfg/c5170.c im Zusammenhang mit dem Header /usr/sys/h/k5170.h als Orientierung zu benutzen (siehe auch kes(4)).

2. Wartung des Betriebssystems MUTOS

2.1. überblick

In diesem Abschnitt werden Besonderheiten des Betriebssystems MUTOS und daraus abzuleitende Konsequenzen für den Normalbetrieb sowie für die Wartung erläutert.
MUTOS ist ein modernes Mehrnutzer-Timesharing-Betriebssystem. Es ist in vollem Umfang nur auf einer mit großem Externspeicher (Festplatte) ausgerüsteten Konfiguration lauffähig. Für eingeschränkte Nutzung ist jedoch auch eine Floppy-orientierte Arbeitsweise möglich (minimal zwei doppelseitige Laufwerke).
MUTOS wird deshalb in einer Form ausgeliefert, die für das Installieren auf einer Festplatte am geeignetsten ist. Soll das System auf einem Rechner ohne Festplatte genutzt werden, sind eine Reihe von Maßnahmen erforderlich bzw. sinnvoll, die in einem speziellen Abschnitt beschrieben werden. Diese eingeschränkte Betriebsweise sollte die Ausnahme sein und auf spezielle Einsätze zugeschnitten werden.
Festplattenorientiert erlaubt MUTOS durch Anschluß weiterer Terminals einen Mehrnutzer-Betrieb. Für seinen effektiven und störungsfreien Betrieb ist eine angepaßte Organisation des Rechenbetriebes notwendig. Die Nutzer des Systems können in sehr unterschiedlichem Grade Kenntnisse des Betriebssystems haben. Vom "Superuser", der gründliches Wissen um die internen Zusammenhänge haben sollte, bis zum Nutzer, der gar nicht wissen muß, mit welchem Betriebssystem er arbeitet und nur seine spezifischen Probleme lösen möchte, reicht das mögliche Spektrum.
Entsprechend müssen die Rechte und Pflichten der Nutzer festgelegt werden.
Eine zentrale Stellung nimmt der Systemverwalter ein. Kein Mehrnutzersystem sollte auf ihn verzichten. Ihm obliegen alle Aufgaben, die mit dem Einrichten des Systems und der laufenden Wartung zusammenhängen. Dafür muß er alle Zugriffsrechte zu zentralen Files und Ausführungsrechte für "kritische" Programme besitzen. Er sollte der einzige sein, der im Status des "Superuser" arbeiten darf. Auf seine Aufgaben wird im Abschnitt 2.3. näher eingegangen.
Für alle anderen Nutzer ist eine Einteilung in Gruppen sinnvoll, deren Basis die gemeinsame Aufgabenstellung sein sollte. Mitgliedern einer Gruppe können zu gemeinsam zu nutzenden Files entsprechende Zugriffsrechte eingeräumt werden.
Alle diese Nutzer sollten in ihren Zugriffsrechten so eingestellt werden, daß sie die Funktionsfähigkeit des Gesamtsystems nicht stören können. Das Abarbeiten von gefährlichen Programmen sowie Schreibzugriffe auf zentrale Files sind ihnen nicht zu gestatten. Dadurch sind Bedien- oder Programmierfehler mit Auswirkungen auf andere Nutzer unterbunden.

2.2. Besonderheiten im Betriebssystem

2.2.1. Puffermechanismus

Eine für das Gesamtsystem wesentliche Arbeitsweise des MUTOS-Kerns besteht in der Pufferung von Blöcken bei der E/A mit blockstrukturierten Geräten. Das heißt, daß E/A-übertragungen von oder zu derartigen Geräten immer über systeminterne Puffer ausgeführt werden. Bei Lesezugriffen wird erst nachgeschaut, ob der gewünschte Block nicht schon in einem Puffer steht. Beim Schreiben wird die physische übertragung verzögert, bis der betreffende Puffer für andere Daten benötigt wird. Für einen hohen Prozentsatz der Fälle kommt das System dadurch ohne wirkliche Datenübertragung mit dem Gerät aus.
Dieser Cache-Mechanismus trägt zur Geschwindigkeitserhöhung des Systems bei, führt bei Fehlbedienungen oder "Systemabstürzen" jedoch zu einem Zustand auf dem Datenträger, der nicht der Wirklichkeit entspricht. Deshalb wird unter bestimmten Umständen eine übereinstimmung des Inhalts der im Hauptspeicher befindlichen Puffer und der zugehörigen Blöcke auf dem Datenträger hergestellt. Diese Wirkung hat der Ruf update(2). Er ist im Kommando sync(1) enthalten, das zu jedem beliebigen Zeitpunkt gegeben werden kann. Im Mehrnutzerbetrieb geschieht das automatisch aller 30 Sekunden. Außerdem erfolgt ein "update" im Zusammenhang mit anderen Funktionen, z.B. bei umount(1). Damit werden größere Datenverluste vermieden. Trotzdem kann es vorkommen, daß defekte File-Systeme entstehen. Ein typisches Beispiel ist das Abschalten eines Laufwerkes ohne vorheriges umount(1). Die Beseitigung derartiger Störungen im File-System wird im Abschnitt 2.4. beschrieben.

2.2.2. Zugriffsrechte

Jedes File im MUTOS-File-System ist mit bestimmten Zugriffsrechten versehen. Sie können getrennt für den Eigentümer, die Mitglieder der gleichen Nutzergruppe und für alle anderen Nutzer vergegeben werden. Zugriffe werden dabei in Lesen (read, r), Schreiben (write, w) und Ausführen (execute, x) unterschieden, wobei unter dem "Ausführen" von Directories das Durchsuchen zu verstehen ist. Da alle Geräte im MUTOS wie Files behandelt werden, können auch die Zugriffsrechte darauf in gleicher Weise festgelegt werden.
Eine besondere Rolle spielt auch hier der "Superuser". Er unterliegt keinen Zugriffsbeschränkungen. Deshalb muß seine Tätigkeit auch besonders verantwortungsvoll ausgeführt werden. Im Gesamtsystem wird durch die Vergabe der Zugriffsrechte darüber entschieden, ob gegenseitige Störungen der Nutzer untereinander möglich sind.
Das ausgelieferte Basissystem kann als Orientierung herangezogen werden. Die Zugriffsrechte sind darin so festgelegt, daß untergeordnete Nutzer Schreibrechte nur von ihrer Home-Directory an abwärts besitzen. Physische Schreibzugriffe auf Platten- und Folienspeicher sind ihnen nicht erlaubt. Die Kommandos mkdir(1), mount(1) und umount(1) können sie nur von ihrer Home-Directory in der Hierarchie abwärts ausführen. Das Gerät /dev/lp ist nur über den Print-Spooler lpr(1) ansprechbar.
Unter diesen Bedingungen sind ein physisches Kopieren von Datenträgern (z.B. mit dd(1)) sowie das Reparieren von gestörten File-Systemen (clri(1), fsck(8)) nur dem "Superuser" erlaubt.

Die Anzeige von Zugriffsrechten und den Eigentümer eines Files erhält man mit ls(1). änderungen können der Eigentümer und der "Superuser" über chmod(1) und chown(1) vornehmen.
Ausführbare Files können einen "Set-User-ID" Modus erhalten. Das bedeutet, daß sie bei der Abarbeitung nicht die Zugriffsrechte des aufrufenden Nutzers erhalten, sondern die des Eigentümers des Files. Diese Arbeitsweise dient dazu, Programme mit Rufen abzuarbeiten, die der Nicht-Superuser nicht geben darf (z.B. mkdir(1)). Es wird dadurch erreicht, daß nur fehlerfreie Programme Eingriffe vornehmen dürfen, die auf die Funktionsfähigkeit des Gesamtsystems Einfluß haben.

2.2.3. Einzel- und Mehrnutzerbetrieb

Der Einzelnutzerbetrieb ist die Ausnahme im MUTOS-System, wenn mit einer Festplatte gearbeitet wird. Bei Floppy-orientierter Arbeitsweise ist er der Normalfall und die folgenden Ausführungen haben nur informativen Charakter. Er dient hauptsächlich dazu, vorbereitende oder korrigierende Maßnahmen für den Mehrnutzerbetrieb vorzunehmen. Das betrifft z.B. änderungen in den Files

```
/etc/passwd
/etc/ttys
/etc/rc,
```

das Prüfen und Reparieren von File-Systemen sowie ähnliche Schritte, bei denen das Gesamtsystem "in Ruhe" sein sollte.
In den Einzelnutzerbetrieb gelangt man, wenn man sich nach dem Systemanlauf und der Aufforderung

```
single user login:
```

anmeldet (im Basissystem mit "root"). Der übergang zum Mehrnutzerbetrieb wird durch Beenden der Einzelnutzerarbeit mit CTRL/Z erreicht. MUTOS eröffnet dann für jedes Terminal, das im File /etc/ttys enthalten und aktiv gesetzt ist, einen Prozeß. Somit ist die Mehrnutzerbetriebsart auch für nur ein einziges Terminal möglich und sinnvoll.
Beim Anlauf des Mehrnutzerbetriebs wird u.a. die Shell-Prozedur /etc/rc abgearbeitet. In der Auslieferungsform beinhaltet sie das Herstellen eines definierten Anfangszustandes des Systems durch Löschen temporärer Files, das Abfragen und Setzen von Datum und Uhrzeit, das Prüfen des Root-File-Systems, das Prüfen des User- und des Temp-File-Systems sowie den Anstoß des Uhr-Dämons. Das Prüfen der File-Systeme geschieht über fsck(8), wobei Standardfehler automatisch korrigiert werden.
Waren derartige Fehlerkorrekturen für das Root-File-System erforderlich, ist ein neuer Systemlauf entsprechend boot(8) notwendig.
In den Mehrnutzerbetrieb wird nur dann übergegangen, wenn das Root-File-System fehlerfrei ist, das User- und das Temp-File-System fehlerfrei oder automatisch korrigierbar sind und das mount(1)-Kommando für User- und Temp-File-System in Ordnung gehen.
Der Inhalt von /etc/rc kann unproblematisch erweitert oder auf andere Standardwerte eingestellt werden, um den konkreten Bedingungen des Einsatzfalles zu genügen. Im Mehrnutzerbetrieb wird aller 30 Sekunden über update(8) ein Aktualisieren aller eingegliederten File-Systeme ausgelöst, was im Einzelnutzerbetrieb nicht der Fall ist. Dort muß eine übereinstimmung von im Hauptspeicher gepufferten Blöcken und Datenträger durch Aufruf von

sync(8) explizit ausgelöst werden.
Eine ordnungsgemäße Beendigung der Arbeit des Betriebssystems sollte folgendermaßen aussehen:
Alle Nutzer werden über das bevorstehende Ende der Arbeit informiert, indem der Systemverwalter über wall(1) eine Mitteilung sendet. Daraufhin beenden die Nutzer mit CTRL/Z ihre Arbeit. Mit dem Kommando

```
/etc/shutdown(8)
```

fährt der Systemverwalter das System definiert bis zum Rechnerhalt ab.
Aus dem Einzelnutzerbetrieb ist das Abfahren des Systems über

```
/etc/haltsys(8)
```

nach vorherigem umount(1) aller File-Systeme vorzunehmen.

2.3. Aufgaben des Systemverwalters

Die Zuständigkeit des Systemverwalters für alle kritischen Aktionen wurde im Abschnitt "überblick" bereits betont. Hier sollen seine Aufgaben im einzelnen umrissen werden.
Als erstes hat er den Auslieferungssatz zu übernehmen. Bei der übergabe des Systems werden ein Root- und ein User-File-System aufgebaut. Ihr Inhalt umfaßt ein Basissystem mit allen benötigten Programmen in abarbeitungsfähiger Form, die Dokumentation und die Quellmodule des Kerns für die Generierung. Dieses System ist für die Anfangszeit i.a. ausreichend.
Der Systemverwalter hat zunächst für alle vorgesehenen Nutzer mit dem Editor Einträge im Password-File /etc/passwd vorzunehmen. Die Auslieferungsform kann als Orientierung dienen. Der Aufbau von /etc/passwd ist aus passwd(5) zu ersehen. Ein Password ist nicht einzutragen. Das kann jeder Nutzer über passwd(1) selbst tun.
Das File /etc/group ist zu aktualisieren (siehe group(5)).
Danach sind die Home-Directories, die im Password-File eingetragen wurden, über mkdir(1) einzurichten. Mit chown(1) ist der Nutzer zum Eigentümer seiner Home-Directory zu machen, mit chgrp(1) ist die Nutzergruppe für die Home-Directory einzustellen. Ein probeweises Anmelden als der angenommene Nutzer muß sofort zu dieser Directory führen.
Eine weitere Aufgabe des Systemverwalters besteht im Führen des Files /etc/ttys. Hier sind alle Terminals des Systems einzutragen und "aktiv" zu setzen, die im Mehrnutzersystem berücksichtigt werden sollen (siehe ttys(5)). Neben dem Console-Terminal können weitere Geräte über /dev/ifss bzw. /dev/v24 angeschlossen werden (z.B. K 8911). Nicht zu bedienende Terminals sind in /etc/ttys passiv zu setzen.
Durch Einfügen weiterer mount(1)-Kommandos kann das Eingliedern nutzerspezifischer File-Systeme angewiesen werden. Für die Reparatur defekter File-Systeme mittels fsck(8) ist auf jedem der File-Systeme in der Root-Directory eine Directory lost+found erforderlich. In dieser Directory müssen leere Eintragungen vorhanden sein. Das Einrichten dieser Directory kann durch Abarbeiten von mklost+found für jedes File-System vorgenommen werden. Dieses Kommandofile wird in der Root-Directory der Root- und der User-Platte ausgeliefert und kann auf jedes neue File-System kopiert und dort abgearbeitet werden.
Die bisher beschriebenen Aufgaben sind meist einmalig, zumindest aber sehr selten zu erledigen. Die Hauptaufgaben des Systemver-

walters liegen in der laufenden Betreuung des Betriebssystems. Dazu gehören die Inbetriebnahme über boot(8) von der Platte, das Arbeiten im Einzelnutzerbetrieb und der übergang zum Mehrnutzerbetrieb.
Eventuelle Tagesmitteilungen an die Nutzer sind im File /etc/motd einzutragen. /etc/ttys ist, falls erforderlich, den tatsächlichen Bedingungen anzupassen. Danach kann mit CTRL/Z in den Mehrnutzerbetrieb umgeschaltet werden.
Treten beim Prüfen der Root- oder der User-Platte Fehler auf, die nicht automatisch korrigiert werden können, muß der Systemverwalter eine interaktive Reparatur mittels fsck(8) vornehmen (siehe Abschnitt 2.4.).
Danach kann erneut der übergang zum Mehrnutzerbetrieb erfolgen.
Hardware-Fehlermeldungen der Geräte-Driver erscheinen auf /dev/console, dem Terminal des Systemverwalters. Dieser hat sie zu registrieren und eventuelle Maßnahmen einzuleiten.
Soll das System seine Arbeit beenden, startet der Systemverwalter die Kommandos

     /etc/haltsys(8)

bei Arbeit im Einzelnutzerbetrieb bzw.

     /etc/shutdown(8)

beim Abfahren aus dem Mehrnutzerbetrieb. Diese Kommandos sorgen für ein definiertes Verlassen des Systems bis zum Rechnerhalt.
Werden auf den Systemplatten Nutzer-Files angelegt und geführt, ist eine Archivierung in bestimmten Zeitabständen sinnvoll. Das Rückspeichern einer Archiv-Version ist meist einfacher und zeitsparender als das Reparieren eines defekten File-Systems. Das Archivieren geänderter Files der beiden Systemplatten ist eine weitere Aufgabe des Systemverwalters. Dazu können dump(1) und restor(1) oder tttaaarrr(1) genutzt werden.
Tritt einmal ein Crash des MUTOS auf (panic-Meldung auf /dev/console), sind die für die Fehleranalyse notwendigen Ausschriften zu notieren.
In größeren Zeitabständen sollte der Systemmanager das File /usr/adm/wtmp kontrollieren. Darin werden alle Nutzeran- und -abmeldungen registriert. Wird das File zu groß, ist es zu löschen und als leeres File (cat </dev/null >/usr/adm/utmp) neu zu erstellen.
Abschließend sei bemerkt, daß ein Systemverwalter nicht unbedingte Voraussetzung für den Betrieb von MUTOS ist. Er kann sich aber auf eine Reihe von Handlungen spezialisieren, die sonst jeder Nutzer des Systems ausführen müßte. Der Systemverwalter muß auch nicht ständig am Rechner anwesend sein. Er sollte jedoch bei Unregelmäßigkeiten derjenige sein, der Maßnahmen einleitet und bei Fehlermeldungen seitens Hardware oder Betriebssystem aussagefähiger Partner ist.

2.4. Reparieren von defekten File-Systemen

2.4.1. Fehlerursachen

2.4.1.1. Nutzerbedingte Fehler

Folgende Fehlbedienungen können zu Störungen im File-System führen:

- unkorrektes Abschalten des Systems
- Eingliedern von fehlerbehafteten File-Systemen ("mount")
- Weiterarbeit trotz kritischer übertragungsfehler

Der erste Anstrich beinhaltet, daß die Arbeit des Betriebssystems mit nicht übereinstimmenden File-Informationen im Hauptspeicher und auf der Platte beendet wird. Typisch dafür sind das unvorschriftsmäßige Abschalten des Rechners oder das Ausschalten von Laufwerken ohne vorheriges uount(1)-Kommando. Bemerkt werden die Auswirkungen dieses Fehlers bei der nächsten Benutzung der betreffenden Platte (Fehlermeldungen bei fsck(8) oder falsche oder fehlende Files).
Die zweite Fehlerursache kann durch einen Test des anzubindenden File-Systems über fsck(8) vor dem mount(1) vermieden werden.
Die letzte Ursache bezieht sich auf Hardware-Lesefehler während der Abarbeitung von fsck(8). Als "kritische Fehler" sind diejenigen Fehlermeldungen zu betrachten, die während aller automatischen Wiederholungen erneut auftraten. Siehe auch Abschnitt 2.4.1.2.

2.4.1.2. Ein- und Ausgabefehler bei übertragungen vom und zum File-System

Derartige Fehler sind durch Ausschriften vom Typ

```
err on dev x/y(n)             x = major number (Gerätetyp)
                              y = minor number (Laufwerksnummer)
bn = ...  cmd = ... sts = ... bn = Blocknummer (dezimal)
                              cmd = E/A-Kommando
                              sts = Fehlerstatus
```

erkennbar, die auf /dev/console erscheinen. Nach erfolgloser Wiederholung wird dem jeweiligen Programm ein Fehlerrückkehrcode übergeben. Die meisten Programme geben daraufhin eine Fehlermeldung auf dem auslösenden Terminal aus. Außerdem ist durch icheck(1) zu ermitteln, welche Files den Block mit der Nr. hinter "bn=" enthalten. Betrifft das nur Nutzerfiles, spielt der Fehler für das System keine Rolle. Der Eigentümer kann entscheiden, was mit dem File geschehen soll (Versuch zu kopieren oder zu löschen). Liegt "bn" jedoch in Systemfiles (z.B. Root-Directory), sollte der Datenträger (wenn möglich) auf ein anderes Laufwerk gelegt werden. Eine Weiterarbeit bringt meist Folgefehler und ist oft auch gar nicht möglich.
Derartige Fehler sind gerätetechnisch bedingt (meist unterschiedliche Justage der Laufwerke). Bringt ein Laufwerkswechsel keine Verbesserung, sollte man eine Weiterarbeit mit diesem Datenträger nicht riskieren, wenn keine Backup-Versionen wichtiger Files existieren. Keinesfalls sind Reparaturmaßnahmen an File-Systemen vorzunehmen, wenn E/A-Fehler kritischer Art auftreten!

2.4.1.3. Crash des Betriebssystems

Kommt das Betriebssystem in eine Situation, in der keine vernünftige Weiterarbeit mehr möglich ist, beendet es seine Arbeit mit einer Ausschrift der Form

panic: ...,

wobei "..." die Ursache bzw. Art des Crash angibt. Eine Liste der Meldungen befindet sich in crash(8).
Einige Meldungen dieser Art beinhalten Zusatzinformationen. In jedem Fall ist es wichtig, derartige Meldungen für spätere Auswertungen zu notieren.
Das File-System wird im Crash-Fall nur beeinflußt, wenn gerade Aktivitäten liefen, die nicht zu Ende gekommen sind (z.B. geöffnete Pipe-Files). Die Crash-Routine versucht, die übereinstimmung zwischen Hauptspeicher- und Platteninformationen noch herzustellen, so daß die wichtigsten Informationen im File-System im allgemeinen den richtigen Stand haben. Auf jeden Fall sind nach einem erneuten Boot(8) des Betriebssystems die schon beschriebenen Prüfungen und evtl. Reparaturen vorzunehmen.

2.4.2. Reparatur eines gestörten File-Systems

2.4.2.1. überblick

Für das Prüfen und Reparieren von File-Systemen steht das Programm fsck(8) zur Verfügung. Es ist leistungsfähiger als die Vorgängerprogramme icheck(1), dcheck(1) und nchhck(1) zusammengenommen. Deshalb wird empfohlen, fsck(8) anstelle der genannten Programme zu verwenden, die jedoch für Einzelfunktionen weiterhin ihre Bedeutung haben.
Bei jeder Inbetriebnahme des MUTOS sollte ein Prüfen der File-Systeme erfolgen. Werden dabei Unstimmigkeiten gemeldet, müssen Korrekturen vorgenommem werden. Fsck(8) hat zwei Arbeitsmodi. Normalerweise wird es automatisch im Systemanlauf abgearbeitet. Dabei werden nur solche änderungen im File-System vorgenommem, die unproblematisch sind. Beim Erkennen anderer Unregelmäßigkeiten beendet fsck seine Arbeit mit einem Ende-Status ungleich Null. Das System läuft dann im Einzelnutzer-Modus. Der Operator startet in diesem Fall fsck interaktiv. In diesem Arbeitsmodus werden jedes ermittelte Problem angezeigt und eine Korrekturmaßnahme vorgeschlagen. Der Operator muß entscheiden, ob die vorgeschlagene Korrektur ausgeführt werden soll. Korrekturen können nur erfolgen, wenn Schreiberlaubnis zum File-System besteht.
Das zu prüfende File-System muß während des Laufs von fsck(8) im Ruhezustand sein. Das ist nach umount(1) gewährleistet. Ist dieser Fall nicht möglich, sollten während des Prüfens keine anderen Aktivitäten mit dem File-System erfolgen.
Das Root-File-System ist immer mit der Angabe des blockorientierten Gerätes zu behandeln (z.B /dev/hda und nicht /dev/rhda).
Erkennt fsck(8) Unregelmäßigkeiten im File-System, zeigt es diese an. Es bietet eine Korrekturmaßnahme an und wartet auf die Reaktion des Bedieners.

Wird das Root-File-System modifiziert, muß anschließend das Betriebssystem neu gestartet werden. Zu diesem Zwecke löst fsck einen definierten Haltzustand aus. Mit Hilfe des Monitor-Boot-Kommandos ist in diesem Falle ein Systemneustart unproblematisch möglich.

Im Normalbetrieb des MUTOS werden einige zentrale Blöcke und Verzeichnisse des File-Systems bei Veränderungen aktualisiert. Das betrifft den Superblock, die Inodes, die Indirekt-Blöcke, die Datenblöcke für Directory-Einträge und die Liste der freien Blöcke. Treten bei übertragungen zu diesen Einheiten Störungen auf, wie sie im Abschnitt 2.4.1. beschrieben wurden, führt das zu Unregelmäßigkeiten im betroffenen File-System.
Die Bedeutung dieser zentralen Verzeichnisse soll zunächst erläutert werden. Genauere Angaben zum Aufbau der Blöcke des File-Systems sind aus dir(5), filesys(5), types(5) zu ersehen.

Superblock

Der Superblock enthält Informationen über die Größe des File-Systems, die Größe der Inode-Liste, einen Teil der Liste freier Blöcke, einen Teil der Liste freier Inodes sowie die Zähler für freie Blöcke und freie Inodes.
Der Superblock eines eingegliederten File-Systems (das Root-File-System ist immer eingegliedert) wird bei jedem umount(1)- und bei jedem sync(8)-Kommando physisch beschrieben.
Das Betriebssystem verwaltet im Superblock außerdem eine Information über das ordnungsgemäße Ausgliedern von vorher eingegliederten File-Systemen.

Inodes

Der Inode ist die zentrale Stelle für jedes File. Er enthält Informationen über den Typ des Inodes (Directory, Daten-File oder Spezial-File), die Anzahl von Directory-Einträgen mit Links zu diesem Inode, die Liste der vom Inode belegten Blöcke und die Größe des Inodes.
Ein Inode wird bei jedem Schließen des Files, das mit dem Inode verknüpft ist, auf den Datenträger geschrieben.

Indirekt-Blöcke

Es gibt drei Arbeiten von Indirekt-Blöcken:

- einfach-indirekte
- doppelt-indirekte und
- dreifach-indirekte Blöcke.

Ein einfach-indirekter Block enthält eine Liste von Blocknummern, die vom Inode belegt sind. Jeder der 128 Einträge in solch einem Block weist auf einen Datenblock.
Ein doppelt-indirekter Block enthält eine Liste von Blocknummern einfach-indirekter Blöcke.
Ein dreifach-indirekter Block enthält eine Liste von Blocknummern doppelt-indirekter Blöcke.
Indirekt-Blöcke werden zur übertragung auf den Datenträger eingereiht, wenn sie vom Betriebssystem modifiziert und freigegeben wurden. Die eigentliche übertragung erfolgt jedoch erst, wenn der belegte Blockpuffer anderweitig benötigt oder ein sync ausgelöst wird.

Daten-Blöcke

Ein Datenblock kann File-Informationen oder Directory-Einträge enthalten.
Das Schreiben auf den Datenträger erfolgt unter den gleichen Bedingungen wie bei Indirekt-Blöcken.

Liste freier Blöcke

Blöcke werden als frei geführt, wenn sie weder dem Superblock, den Inodes, den Indirekt-Blöcken noch den Datenblöcken zugeordnet sind. Der Superblock enthält den ersten Teil der Liste freier Blöcke. In jedem Block, der Teile dieser Liste führt, sind ein Zähler über die Anzahl von Einträgen in diesem Block und ein Zeiger auf den nächsten Block dieser Art enthalten.
Das Schreiben auf den Datenträger erfolgt unter den gleichen Bedingungen wie bei Indirekt-Blöcken.

2.4.2.2. Möglichkeiten zum Erkennen und Beseitigen von Störungen

Superblock

Im Superblock treten am häufigsten Probleme auf, da er bei jeder änderung in den Blöcken des File-Systems oder in Inodes modifiziert wird. Deshalb ist er besonders empfindlich gegen ein Abschalten des Rechners ohne unmittelbar vorher erfolgtes sync.
Der Superblock kann auf Unstimmigkeiten in den Werten für die Größe des File-Systems und der Inode-Liste, in der Liste freier Blöcke und den Zählern für freie Blöcke und freie Inodes geprüft werden.

Größe des File-Systems und der Inode-Liste

Der Wert für die Größe des File-Systems muß größer sein als die Anzahl von Blöcken, die der Superblock und die Inode-Liste belegen. Er muß kleiner als 65535 sein. Diese Größeninformationen sind für fsck kritische Werte. Alle anderen Prüfungen im File-System hängen von der Korrektheit dieser Angaben ab.
Da keine Möglichkeit besteht, die aktuelle Größe des File-Systems und der Inode-Liste zu bestimmen, kann fsck nur auf die Einhaltung dieser Grenzwerte achten.

Liste freier Blöcke

Die Liste freier Blöcke beginnt im Superblock und setzt sich in verketteten Blöcken des File-Systems fort. Jeder Block, der Teile dieser Liste enthält, kann auf die Einhaltung von Grenzwerten bei Zählern und Blocknummern überprüft werden. Außerdem kann festgestellt werden, ob Blöcke schon an anderer Stelle im File-System vergeben wurden. Es wird auch geprüft, daß alle Blöcke des File-Systems erfaßt sind und die Summe der in dieser Liste enthaltenen und von Inodes belegten Blöcke der Anzahl von Blöcken des File-Systems entspricht.
Werden Unstimmigkeiten ermittelt, kann fsck die Liste freier Blöcke neu aufbauen, wobei alle belegten Blöcke ausgeschlossen werden.

Zähler freier Blöcke

Fsck vergleicht diesen Wert im Superblock mit dem aktuell ermittelten. Stimmen beide nicht überein, kann der Zähler im Superblock durch den aktuellen Wert ersetzt werden.

Zähler freier Inodes

Fsck vergleicht diesen Wert im Superblock mit dem aktuell ermittelten. Stimmen beide nicht überein, kann der Zähler im Superblock durch den aktuellen Wert ersetzt werden.

Inodes

Der einzelne Inode ist nicht so sehr gefährdet wie der Superblock, doch führt die i. a. recht große Zahl aktiver Inodes zu ähnlicher Fehlerwahrscheinlichkeit.
Jeder Inode in der Inode-Liste wird auf Unregelmäßigkeiten bezüglich Format, Typ, Link-Zähler, doppelt vergebener Blöcke, Adresse der Blöcke und Größe des Inodes überprüft.

Format und Typ

Das Schreiben falscher Daten in einen Inode (z. B. durch Hardware-Fehler) kann zu unerlaubten Werten in diesen Angaben führen. Fsck kann in diesem Fall nur den Inode löschen, was zu Datenverlust führen kann.

Link-Zähler

Dieser Wert beinhaltet die Gesamtzahl von Directory-Eintragungen, die auf diesen Inode Bezug nehmen.
Fsck überprüft diesen Zähler, indem es die gesamte Directory-Struktur verfolgt und einen aktuellen Link-Zähler für jeden Inode ermittelt. Dabei können folgende Fehlerfälle auftreten.

- alter Link-Zähler ungleich Null, gültiger Link-Zähler gleich Null:
  kein Directory-Eintrag für diesen Inode vorhanden
  File kann nach lost+found gebracht werden

- alter und gültiger Link-Zähler ungleich Null, jedoch nicht übereinstimmend:
  Directory wurde gelöscht bzw. hinzugefügt, ohne daß der Inode aktualisiert wurde
  Ersetzen des alten Wertes durch den gültigen

Mehrfach vorhandene Blöcke

Fsck ermittelt, ob die an einen Inode vergebenen Blöcke schon anderweitig zugewiesen wurden. Tritt solch ein Fall ein, wird der Inode ermittelt, der den doppelt vergebenen Block ebenfalls enthält. Meist ist der Inode mit dem frühesten Modifikationsdatum fehlerhaft. Er sollte gelöscht werden.
Treten zu viele derartige Blöcke auf, kann die Ursache darin liegen, daß das Schreiben eines Indirekt-Blockes nicht erfolgreich war.

Fsck fragt den Operator für beide Inodes ab, ob sie gelöscht werden sollen.

Falsche Blöcke
~~~~~~~~~~~~~~
Fsck überprüft für jeden Block, der an einen Inode vergeben wurde, ob seine Nummer innerhalb bestimmter Grenzen liegt. Blocknummern müssen zwischen der des ersten Datenblocks und der des letzten Blocks im File-System liegen.
Der Operator wird abgefragt, ob der Inode gelöscht werden soll.

File-Größe
~~~~~~~~~~
Jeder Inode enthält einen 32-Bit-Wert für die Größe des Inodes. Diese Größe stellt die Anzahl von Bytes dar, die dem Inode zugeordnet sind. Eine überprüfung der Sinnfälligkeit dieses Wertes ist möglich. Die Größe eines Directory-Inodes muß ein vielfaches von 16 sein. Aus der Anzahl enthaltener Bytes wird die Anzahl benötigter Blöcke ermittelt und mit den wirklich vergebenen verglichen. Gibt es Diskrepanzen, wird eine Warnung ausgegeben. Weitere Korrekturmaßnahmen sind nicht möglich.

Indirekt---Blöcke

Indirekt-Blöcke sind dem Inode zugeordnet. Fehler in ihnen wirken sich direkt auf den Inode aus. überprüfungen erfolgen auf doppelt vergebene Blöcke und Blocknummern außerhalb sinnvoller Grenzen (s. o.).

Daten---Blöcke

Es gibt zwei Arten von Datenblöcken - einfache Datenblöcke und Directory-Datenblöcke. Einfache Datenblöcke enthalten die in einem File gespeicherten Informationen, während Directory-Datenblöcke Directory-Einträge enthalten. Fsck kann die Gültigkeit des Inhalts einfacher Datenblöcke nicht überprüfen. Jeder Directory-Datenblock wird auf Unregelmäßigkeiten folgender Art getestet:

- Inode-Nummern, die auf nicht vergebene Inodes verweisen,
- Inode-Nummern größer als die Zahl der Inodes im File-System,
- fehlerhafte Inode-Nummern für "." und ".." und
- Directories, die vom File-System abgetrennt wurden.

In den ersten beiden Fällen kann fsck den Directory-Eintrag löschen. Die fehlerhaften Inode-Nummern für "." und ".." können durch die richtigen Werte ersetzt werden.
Der letzte Fall bedeutet, daß eine Directory gefunden wird, die keinen Zeiger im File-System hat. Es kann durch Eintrag in die Directory lost+found wieder zugänglich gemacht werden.

Blöcke der Liste freier Blöcke

Diese sind dem Superblock zugeordnet. Unstimmigkeiten in ihnen beeinträchtigen deshalb den Superblock. Erkannt werden Zähler und Blocknummern außerhalb des zulässigen Bereiches und Blöcke, die schon anderweitig vergeben wurden (siehe auch 2.4.2.1. -Liste der freien Blöcke).

2.4.2.3. Vereinbarungen

Fsck arbeitet in mehreren Durchläufen. Jeder Durchlauf aktiviert eine andere Phase des Programms.
Nach dem Anlaufteil arbeitet fsck aufeinanderfolgende Phasen ab, wobei Blöcke und Größenangaben, Pfadnamen, die Verbundenheit (connectivity), Referenz-Zähler und die Liste freier Blöcke überprüft werden.
Wird eine Unregelmäßigkeit erkannt, teilt fsck die Fehlerbedingung mit. Wird eine Antwort vom Operator erwartet, erscheint eine entsprechende Mitteilung, auf die reagiert werden muß.
Der folgende Abschnitt erläutert die Bedeutung jedes Fehlerzustandes, die möglichen Reaktionen und verwandte Fehlerbedingungen.
Die Fehlerzustände sind den Phasen des Programms zugeordnet, in denen sie auftreten können. Zu Beginn werden die Fehler behandelt, die in mehr als einer Phase möglich sind.

2.4.2.4. Fehlerzustände

Die folgenden Fehlerzustände resultieren aus Fehlern bei der Eingabe der Kommandozeile, bei der Bereitstellung von Speicherplatz, beim Eröffnen und Statusabfragen von Files, bei der Prüfung der File-Systemgröße und beim Einrichten des Arbeitsfiles.

C option?

C ist keine legale Option für fsck. Programmabbruch. Siehe auch fsck(8).

Bad -t option

Benutzung der -t Option ohne folgenden File-Namen. Programmabbruch. Siehe auch fsck(8).

Invalid -s argument, defaults asumed

Die -s Option wurde ohne zulässiges Folgeargument benutzt. Siehe auch fsck(8).

Incompatible options: -n and -s

Die Liste freier Blöcke kann nicht ohne Modifikation des File-Systems neu gebildet werden. Programmabbruch. Siehe auch fsck(8).

Can't get memory

Kann normalerweise nicht auftreten. Programmabbruch.

Can't open checklist file: F

Das Standardfile für zu prüfende File-Systeme (/etc/fstab) kann nicht für Lesen geöffnet werden. Programmabbruch. Zugriffsrechte zu F sind zu überprüfen.

Can't stat root

Die Statistikabfrage für die Root-Directory mißlang. Sollte niemals auftreten. Programmabbruch.

Can't stat F

Die Statistikabfrage für das File-System F mißlang. Dieses File-System wird übergangen. Zugriffsrechte zu F sind zu überprüfen.

F is not a block or character device

Es wurde kein File-System, sondern ein normales File angegeben. Fortsetzungbeim nächsten angegebenen File-System. File-Typ von F ist zu überprüfen.

Can't open F

Das File-System F kann nicht für Lesen geöffnet werden. Es wird übergangen. Zugriffsrechte zu F sind zu überprüfen.

Size check::: fsize X isitze Y

Y ist größer als X, oder es gibt mehr als 65535 Inodes im File-System. Das File-System wird übergangen. Es kann mit fsck nicht geprüft werden.

Can't create F

Das Arbeitsfile F konnte nicht eingerichtet werden. Das File-System wird übergangen. Zugriffsrechte zu F sind zu überprüfen.

CAN NOT SEEK: BLK B (CONTINUE)

CAN NOT READ: BLK B (CONTINUE)

CAN NOT WRITE: BLK B (CONTINUE)

Die jeweilige Funktion für den angegebenen Block B mißlang. Dieser Fall sollte nie auftreten.

Mögliche Antworten auf die CONTINUE-Anfrage sind:

YES - Versuch, den Prüflauf fortzusetzen. Ein vollständiges Prüfen des File-Systems ist jedoch nicht möglich. Es sollte ein zweiter fsck-Lauf vorgenommem werden.
Unter bestimmten Bedingungen bricht fsck mit "Fatal I/O error" ab.

NO - Programmabbruch

Jede Phase des fsck-Laufs wird angezeigt. Die während der Phasen möglichen Meldungen werden im Folgenden beschrieben.

*** Phase 1: Check Blocks and Sizes

In dieser Phase wird die Inode-Liste überprüft. Fehlerzustände können den Inode-Typ, die Inode-Größe, das Inode-Format, die zugewiesenen Blocknummern betreffen oder beim Aufbau der Tabelle von Inodes mit einem Link-Zähler von Null auftreten.

UNKNOWN FILE TYPE I=I (CLEAR)

Das Modus-Wort des Inodes hat einen unzulässigen Wert.

Mögliche Antworten auf die CLEAR-Anfrage sind:

YES - Inode I wird gelöscht. Damit tritt immer in Phase 2 der UNALLOCATED-Fehlerzustand für jeden Directory-Eintrag auf, der auf diesen Inode zeigt.

NO - übergehen des Fehlerzustandes.

LINK COUNT TABLE OVERFLOW (CONTINUE)

Eine interne Tabelle der Inodes mit einem Link-Zähler von Null ist ausgelastet. Fsck ist mit einem größeren Wert für MAXLNCNT zu übersetzen.

Mögliche Antworten auf die CONTINUE-Anfrage sind:

YES - Fortsetzung des Programms. Eine vollständige Prüfung des File-Systems ist nicht möglich. Beim Auftreten weiterer Inodes mit einem Link-Zähler von Null wiederholt sich der Fehlerzustand.

NO - Programmabbruch

B BAD I=I

Der Inode I enthält eine Blocknummer B, die außerhalb des zulässigen Bereiches liegt. Treten zu viele derartige Blocknummern im Inode I auf, kann das zur "EXCESSIVE BAD BLKS"-Fehlermeldung führen. Die Phasen 2 und 4 bringen in jedem Fall die "BAD/DUP"-Meldung.

EXCESSIVE BAD BLKS I=I (CONTINUE)

Es treten mehr Blocknummern außerhalb des zulässigen Bereiches im Inode I auf, als verarbeitet werden können (gewöhnlich 10).

Mögliche Antworten auf die CONTINUE-Anfrage sind:

YES - Die verbleibenden Blöcke des Inodes werden übergangen. Eine vollständige Prüfung des File-Systems ist nicht möglich. Ein zweiter Lauf wird empfohlen.

NO - Programmabbruch

B DUP I=I

Der Inode I enthält einen Block B, der schon durch einen anderen Inode belegt wurde. Treten zu viele derartige Blocknummern im Inode I auf, kann das zur "EXCESSIVE DUP BLKS"-Fehlermeldung führen. Die Phasen 2 und 4 bringen in jedem Fall die "BAD/DUP"-Meldung.

EXCESSIVE DUP BLKS I=I (CONTINUE)

Es treten mehr schon von anderen Inodes belegte Blöcke auf, als verarbeitet werden können (gewöhnlich 10).

Mögliche Antworten auf die CONTINUE-Anfrage sind:

YES - Die verbleibenden Blöcke des Inodes werden übergangen. Eine vollständige Prüfung des File-Systems ist nicht möglich. Ein zweiter Lauf wird empfohlen.

NO - Programmabbruch

DUP TABLE OVERFLOW (CONTINUE)

Eine interne Tabelle in fsck für doppelt vergebene Blöcke hat keinen freien Platz mehr. Fsck ist mit einem größeren Wert für DUPTBLSIZE zu übersetzen.

Mögliche Antworten auf die CONTINUE-Anfrage sind:

YES - Fortsetzung des Programmlaufs. Eine vollständige Prüfung des File-Systems ist nicht möglich. Ein zweiter Lauf wird empfohlen. Beim Auftreten weiterer doppelt vergebener Blökke wiederholt sich dieser Fehlerzustand.

NO - Programmabbruch

POSSIBLE FILE SIZE ERROR I=I

Der Wert für die Größe des Inodes I paßt nicht mit der Anzahl belegter Blöcke zusammen. Dies ist nur eine Warnung.

DIRECTORY MISALIGNED I=I

Die Größe eines Directory-Inodes ist kein Vielfaches der Größe eines Directory-Eintrags (gewöhnlich 16). Dies ist nur eine Warnung.

PARTIALLY ALLOCATED INODE I=I (CLEAR)

Der Inode I ist weder belegt noch freigegeben (unnormaler Zustand).

Mögliche Antworten auf die CLEAR-Anfrage sind:

YES - Freigabe des Inodes durch Löschen seines Inhalts.

NO - übergehen der Fehlerbedingung.

** Phase 1B: Rescan For More Dups

Wird ein doppelt vergebener Block im File-System gefunden, wird das File-System erneut durchsucht, um den Inode zu ermitteln, der

den Block schon belegt hatte. Folgende Mitteilung wird ausgegeben, wenn der doppelt vergebene Block gefunden wird:

B DUP I=I

Inode I enthält einen Block B, der schon durch einen anderen Inode belegt wurde. Die Phase 2 bringt daraufhin immer die "BAD/DUP"-Fehlermeldung. Durch Vergleich dieser Meldung mit den Meldungen aus der Phase 1 ist zu ermitteln, welche Inodes den gleichen Block enthalten.

** Phase 2: Check Pathnames

Diese Phase entfernt Directory-Eintragungen, die auf Inodes zeigen, die zu Fehlermeldungen in den Phasen 1 und 1B führten.

In diesem Abschnitt werden Fehlerbedingungen aufgezeigt, die aus dem Modus bzw. Status des Root-Inodes, aus Bereichsüberschreitungen von Directory-Inode-Zeigern und aus Directory-Einträgen, die auf falsche Inodes zeigen, resultieren.

ROOT INODE UNALLOCATED. TERMINATING.

Der Root-Inode (gewöhnlich Inode-Nummer 2) enthält den Modus "nicht belegt". Diese Fehlerbedingung dürfte niemals auftreten. Das Programm wird beendet.

ROOT INODE NOT DIRECTORY (FIX)

Der Root-Inode (gewöhnlich Inode-Nummer 2) ist nicht vom Typ "Directory".

Mögliche Antworten auf die FIX-Anfrage sind:

YES - Der Typ "Directory" wird eingetragen. Sind die zum Root-Inode gehörenden Datenblöcke keine Directory-Blöcke, führt das zu sehr vielen weiteren Fehlerbedingungen!

NO - Programmabbruch

DUPS/BAD IN ROOT INODE (CONTINUE)

In den Phasen 1 oder 1B wurden doppelt vergebene Blöcke oder falsche Blocknummern im Root-Inode gefunden.

Mögliche Antworten auf die CONTINUE-Anfrage sind:

YES - übergehen der Fehlerbedingung und Versuch einer Fortsetzung der Prüfung des File-Systems. Ist der Root-Inode fehlerhaft, kann dies zu einer großen Anzahl anderer Fehlerbedingungen führen.

NO - Programmabbruch

I OUT OF RANGE I=I NAME=F (REMOVE)

Eine Directory-Eintragung F hat eine Inode-Nummer I, die oberhalb des Endes der Inode-Liste liegt.

Mögliche Antworten auf die REMOVE-Anfrage sind:

YES - Die Directory-Eintragung F wird gelöscht.

NO - übergehen der Fehlerbedingung.

UNALLOCATED I=I OWNER=O MODE=M SIZE=S
MTIME=T NAME=F (REMOVE)

Eine Directory-Eintragung F hat einen Inode I, bei dem der Modus "BELEGT" nicht gesetzt ist. Der Eigentümer O, der Modus M, die Größe S, der Modifikationszeitpunkt T und der File-Name F werden ausgegeben.

Mögliche Antworten auf die REMOVE-Anfrage sind:

YES - Die Directory-Eintragung F wird gelöscht.

NO - übergehen der Fehlerbedingung.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T
DIR=F (REMOVE)

In den Phasen 1 oder 1B wurden doppelt vergebene Blöcke oder falsche Blocknummern beim Directory-Eintrag F, Directory-Inode I gefunden. Der Eigentümer O, der Modus M, die Größe S, der Modifikationszeitpunkt T und der Directory-Name F werden ausgegeben.

Mögliche Antworten auf die REMOVE-Anfrage sind:

YES - Die Directory-Eintragung F wird gelöscht.

NO - übergehen der Fehlerbedingung.

DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T
FILE=F (REMOVE)

In den Phasen 1 oder 1B wurden doppelt vergebene Blöcke bzw. falsche Blocknummern beim Directory-Eintrag F, Inode I gefunden. Der Eigentümer O, der Modus M, die Größe S, der Modifikationszeitpunkt T und der File-Name F werden ausgegeben.

Mögliche Antworten auf die REMOVE-Anfrage sind:

YES - Die Directory-Eintragung F wird gelöscht.

NO - übergehen der Fehlerbedingung.

** Phase 3: Check Connectivity

Diese Phase behandelt die Verbundenheit von Directories, die in Phase 2 erkannt wurde. Es werden Fehlerbedingungen angezeigt, die sich aus Directories ohne Bezugnahme sowie fehlendem oder vollständig belegtem lost+found-Directory ergeben.

```
      UNREF DIR I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T (RECONNECT)
```

Der Directory-Inode I wurde ermittelt, der keine Bezugnahme auf eineDirectory-Eintragung besitzt. Der Eigentümer O, der Modus M, die Größe S und der Modifikationszeitpunkt T des Directory-Inodes I werden angezeigt.

Mögliche Antworten auf die RECONNECT-Anfrage sind:

YES - Der Directory-Inode I wird dem Directory für verlorene Files (gewöhnlich lost+found))) zugeordnet. Dies kann zu der lost+found-Fehlerbedingung in Phase 3 führen, falls es Probleme beim Einbinden des Directory-Inodes I in lost+found gibt. Außerdem kann die CONNECTED-Meldung in Phase 3 auftreten, wenn das Einbinden erfolgreich verlief.

NO - übergehen der Fehlerbedingung. Dies führt immer zur UNREF-Fehlerbedingung in Phase 4.

```
      SORRY. NO lost+found DIRECTORY
```

Es existiert in der Root-Directory des File-Systems kein lost+found-Directory. Fsck übergeht das gewünschte Einbinden einer Directory in lost+found. Dies führt zur UNREF-Fehlerbedingung in Phase 4.
Die Zugriffsmodi von lost+found sind zu überprüfen.
Siehe fsck(8) für Einzelheiten.

```
      SORRY. NO SPACE IN lost+found DIRECTORY
```

Es existiert in der lost+found-Directory in der Root-Directory des File-Systems nicht mehr genügend Platz, einen weiteren Eintrag vorzunehmen. Fsck übergeht das gewünschte Einbinden einer Directory in lost+found. Dies führt immer zur UNREF-Fehlerbedingung in Phase 4.
Nicht mehr benötigte Einträge sind aus lost+found zu streichen, oder lost+found ist größer anzulegen (siehe fsck(8)).

```
      DIR I=I1  CONNECTED. PARENT WAS I=I2
```

Dies ist ein Hinweis, daß der Directory-Inode I1 erfolgreich der lost+found-Directory zugeordnet wurde. Der übergeordnete (Parent-)Inode I2 des Directory-Inode I1 wird durch die Inode-Nummer der lost+found-Directory ersetzt.

```
    ** Phase 4: Check Reference Counts
```

Diese Phase behandelt die Link-Zähler, die in den Phasen 2 und 3

ermittelt wurden. Es werden Fehlerbedingungen angezeigt, die sich aus Files oder Directories ohne Bezugnahme, aus fehlender oder vollständig belegter lost+found-Directory, aus falschen Link-Zählern für Files, Directories oder Spezial-Files, aus falschen oder doppelt vergebenen Blöcken in Files und Directories sowie einem fehlerhaften Gesamtzähler freier Inodes ergeben.

```
      UNREF FILE I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T (RECONNECT)
```

Beim Durchsuchen des File-Systems wurde der Inode I ohne Bezugnahme zu einer Directory-Eintragung gefunden. Der Eigentümer O, der Modus M, die Größe S und der Modifikationszeitpunkt T des Inodes I werden angezeigt.

Mögliche Antworten auf die RECONNECT-Anfrage sind:

YES - Einbinden des Inode I in die Directory für verlorene Files (gewöhnlich lost+found). Dies kann zur lost+found-Fehlerbedingung in Phase 4 führen, wenn es dabei Probleme gibt.

NO - übergehen der Fehlerbedingung. Dies führt immer zur CLEAR-Fehlerbedingung in Phase 4.

```
      SORRY... NO       lost+found DIRECTORY
```

Es existiert in der Root-Directory des File-Systems kein lost+found-Directory. Fsck übergeht das gewünschte Einbinden eines Files in lost+found. Dies führt immer zur CLEAR-Fehlerbedingung in Phase 4.
Die Zugriffsrechte zu lost+found sind zu überprüfen.

```
      SORRY... NO       SPACE IN lost+found DIRECTORY
```

Es existiert in der lost+found-Directory in der Root-Directory des File-Systems nicht mehr genügend Platz, einen weiteren Eintrag vorzunehmen. Fsck übergeht das gewünschte Einbinden eines Files in lost+found. Dies führt immer zur CLEAR-Fehlerbedingung in Phase 4.
Größe und Inhalt von lost+found sind zu überprüfen.

```
      (CLEAR)
```

Der in der unmittelbar vorhergehenden Fehlerbedingung genannte Inode kann nicht eingegliedert werden.

Mögliche Antworten auf die CLEAR-Anfrage sind:

YES - Freigeben des Inodes aus der vorhergehenden Fehlermeldung durch Löschen seines Inhalts.

NO - übergehen dieser Fehlerbedingung.

```
      LINK COUNT FILE I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T  COUNT=X  SHOULD BE Y  (ADJUST)

      LINK COUNT DIR I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T  COUNT=X  SHOULD BE Y  (ADJUST)

      LINK COUNT F  I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T  COUNT=X  SHOULD BE Y  (ADJUST)
```

Der Link-Zähler des Inodes I (I kann je nach Ausschrift ein File, eine Directory oder F sein) ist X, sollte aber Y sein. Der Eigentümer O, der Modus M, die Größe S und der Modifikationszeitpunkt T werden angezeigt.

Mögliche Antworten auf die ADJUST-Anfrage sind:

YES - Ersetzen des Link-Zählers des Inode I durch Y.

NO - übergehen der Fehlerbedingung.

```
      UNREF FILE I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T  (CLEAR)

      UNREF DIR I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T  (CLEAR)
```

Der Inode I (I kann je nach Ausschrift ein File oder eine Directory sein) hat keine Verbindung zu einem Directory-Eintrag. Der Eigentümer O, der Modus M, die Größe S und der Modifikationszeitpunkt T des Inode I werden angezeigt.

Mögliche Antworten auf die CLEAR-Anfrage sind:

YES - Freigabe des Inode I durch Löschen seines Inhalts.

NO - übergehen der Fehlerbedingung.

```
      BAD/DUP FILE I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T  (CLEAR)

      BAD/DUP DIR I=I  OWNER=O  MODE=M  SIZE=S
      MTIME=T  (CLEAR)
```

In den Phasen 1 oder 1B wurden doppelt vergebene Blöcke oder unzulässige Blocknummern im Zusammenhang mit dem Inode I ermittelt. I kann je nach Ausschrift ein File oder eine Directory sein. Der Eigentümer O, der Modus M, die Größe S und der Modifikationszeitpunkt T werden angezeigt.

Mögliche Antworten auf die CLEAR-Anfrage sind:

YES - Freigabe des Inode I durch Löschen seines Inhalts.

NO - übergehen der Fehlerbedingung.

```
      FREE INODE COUNT WRONG IN SUPERBLK (FIX)
```

Der Zähler freier Inodes im Superblock stimmt nicht mit dem ermittelten gültigen Wert überein.

Mögliche Antworten auf die FIX-Anfrage sind:

YES - Ersetzen des Zählers im Superblock durch den gültigen Wert.

NO - übergehen der Fehlerbedingung.

** Phase 5: Check Free List

Diese Phase behandelt die Liste freier Blöcke. Es werden Fehlerbedingungen angezeigt, die sich aus unzulässigen Blocknummern oder doppelt vergebenen Blöcken in der Liste freier Blöcke, aus Unstimmigkeiten bei Zählern freier Blöcke und aus Blöcken, die weder im File-System noch in der Liste freier Blöcke erscheinen, ergeben.

EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE)

EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE)

Die Liste freier Blöcke enthält mehr unzulässige Blocknummern (BAD) oder doppelt vergebene Blöcke (DUP), als verarbeitet werden können (gewöhnlich 10).

Mögliche Antworten auf die CONTINUE-Anfrage sind:

YES - Der Rest der Liste freier Blöcke wird übergangen. Diese Fehlerbedingungen führen immer zu den Meldungen BAD BLKS IN FREE LIST bzw. DUP BLKS IN FREE LIST der Phase 5.

NO - Programmabbruch

BAD FREEBLK COUNT

Der Zähler freier Blöcke in einem Block der Liste freier Blöcke ist größer als 50 oder kleiner als 0. Diese Fehlerbedingung führt immer zur Meldung BAD FREE LIST in Phase 5.

X BAD BLKS IN FREE LIST

X DUP BLKS IN FREE LIST

X BLK(S) MISSING

In der Liste freier Blöcke wurden X Blöcke ermittelt, die je nach Fehlerausschrift eine unzulässige Blocknummer haben oder doppelt vergeben wurden oder weder im File-System noch in der Liste freier Blöcke erscheinen. Diese Fehlerbedingungen führen immer zur Meldung BAD FREE LIST in Phase 5.

FREE BLK COUNT WRONG IN SUPERBLOCK (FIX)

Der Zähler freier Blöcke im Superblock stimmt nicht mit dem ermittelten gültigen Wert überein.

Möglche Antworten auf die FIX-Anfrage sind:

YES - Ersetzen des Zählers im Superblock durch den gültigen Wert.

NO - übergehen der Fehlerbedingung.

BAD FREE LIST (SALVAGE)

In der Phase 5 wurden beim überprüfen der Liste freier Blöcke unzulässige Blocknummern, doppelt vergebene Blöcke oder Blöcke ermittelt, die weder im File-System noch in der Liste freier Blöcke erscheinen.

Mögliche Antworten auf die SALVAGE-Anfrage sind:

YES - Esetzen der bisherigen Liste freier Blöcke durch eine neu ermittelte. Diese neue Liste wird dabei so geordnet, daß die Positionierzeiten der Platten reduziert werden.

NO - übergehen der Fehlerbedingung.

** Phase 6: Salvage Free List

Diese Phase rekonstruiert die Liste freier Blöcke. Es werden Fehlerbedingungen angezeigt, die sich aus den Werten für die Blockanzahl pro Zylinder und für die Anzahl zu überspringender Blöcke ("skip") ergeben.

Default free-block list spacing assumed

Dies ist ein Hinweis, daß die Anzahl zu überspringender Blöcke größer als die Blockanzahl pro Zylinder oder kleiner als 1 ist, oder daß die Blockanzahl pro Zylinder größer als 500 oder kleiner als 1 ist. Es werden Standardwerte benutzt (siehe fsck(8)).

CLEANUP

Zum Abschluß der Prüfung eines File-Systems werden Hinweise zum File-System und zu eventuellen Modifikationen ausgegeben.

X files Y blocks Z free

Diese Mitteilung gibt die Anzahl enthaltener Files (X), die Zahl belegter Blöcke (Y) und die verbleibende Anzahl freier Blöcke (Z) im geprüften File-System an.

***** FILE SYSTEM WAS MODIFIED *****

Diese Mitteilung informiert darüber, daß fsck das aktuelle File-

System modifiziert hat. Das erfolgt auch beim "Säubern" eines unvorschriftsmäßig verlassenen File-Systems zur Vorbereitung für das nächste Eingliedern.

```
      ***** ROOT FILESYSTEM WAS MODIFIED *****
      ***** REBOOT MUTOS AFTER AUT0MATIC SYSTEM SHUTDOWN *****
```

Das Root-File-System wurde modifiziert. Fsck löst in diesem Fall einen definierten System-Halt aus. über das Monitor-Boot-Kommando ist das Betriebssystem neu zu starten.

3. Regenerierung des Systems

3.1. überblick

MUTOS wird in Form eines Basissystems ausgeliefert, das nicht unbedingt den konkreten Anforderungen des Anwenders entsprechen wird. Deshalb besteht die Möglichkeit, durch Generierung eines neuen Systemkerns mit veränderten Parametern und Einbeziehung anderer Geräte eine einsatzfallspezifische Anpassung vorzunehmen. In diesem Kapitel wird dafür eine Anleitung gegeben.
Die Generierung geschieht aus Quellmoduln. Alle benötigten Module und Kommandofiles sind im Vertriebssystem in sechs Subdirectories von /usr/sys enthalten:

/usr/sys/h	Header-Files, die über "#include..." in Systemmoduln benutzt werden
/usr/sys/io	Gerätedriver und einige angelehnte Module
/usr/sys/sys	überwiegender Teil des Systems
/usr/sys/mdep	Maschinenabhängige Module
/usr/sys/cfg	Files, die Adressen und Tabellen enthalten
/usr/sys/conf	Allgemeine Files zur Generierung

Diese sechs Subdirectories können auch an beliebiger anderer Stelle im File-System angeordnet werden. Sie bilden einen in sich abgeschlossenen Komplex.
Die Subdirectories io, cfg, mdep und sys enthalten Bibliotheken, in denen die Objektmodule dieser Subdirectories archiviert werden.
Für den Generierungslauf wird das Systemprogramm make(1) genutzt. Das benötigte "makefile" steht in /usr/sys/conf. Nach Beendigung von "make" steht der neue Systemkern als File mit dem Namen "mutos" in /usr/sys/conf.
Wesentliche Eigenschaften des Systems werden in /usr/sys/h/param.h festgelegt. Daran sollte nichts verändert werden. Andere Parameter werden im "makefile" bzw. in c.c in /usr/sys/conf eingestellt. Darauf wird im Punkt 3.2. eingegangen.

3.2. Directory /usr/sys/conf

Diese Directory enthält das File "makefile", das als Eingabefile für make(1) dient.
"make mutos" baut einen Systemkern unter dem Namen "mutos" auf. Bei Veränderungen in Modulen bzw. zugehörigen Header-Files (/usr/sys/h) werden die entsprechenden Module des Kerns neu übersetzt. Dabei sind die Modifikations- bzw. Erstellungsdaten der einzelnen Files ausschlaggebend.
Das "makefile" enthält u.a. Definitionen, die das Einbinden bestimmter Driver veranlassen. In der Auslieferungsform sind alle optionalen Driver eingebunden. Durch Streichen des entsprechenden Ausdruckes in der Zeile "CONFOPT =" kann der zugeordnete Driver für den Generierungslauf ausgeschlossen werden. Weitere Veränderungen sind in c.c möglich.
über Parameter werden die Größen verschiedener systeminterner Tabellen sowie Grenzwerte festgelegt. Die ordnungsgemäße Funktion des Gesamtsystems hängt von einer geeigneten Wahl dieser Parameter ab. Dieser Abschnitt soll dafür Erläuterungen geben. Im allgemeinen ist eine Änderung der folgenden Werte nicht notwendig.

NBUF	Das System MUTOS arbeitet mit einem Cache-Puffer-Mechanismus für Plattenspeicher. Je größer die Anzahl dieser Puffer ist, um so effektiver erfolgt die Plat-

tenarbeit.

NINODE Legt die Größe der Inode-Tabelle fest. Jeweils einen Eintrag in dieser Tabelle benötigt
- jedes geöffnete File
- jede Arbeits-Directory
- "sticky"-Text-Segment
- jedes geöffnete Gerät
- jedes über "mount" eingegliederte Gerät

Pro Inode werden 74 Byte benötigt.

NFILE Legt die maximale Anzahl geöffneter Files fest. Bei jedem open(2) oder creat(2) für ein File wird eine Eintragung vorgenommen.
Jede Eintragung belegt 8 Byte.

NMOUNT Legt die maximale Anzahl eingegliederter File-Systeme fest. Das Root-File-System ist zu berücksichtigen.

MAXUPRC Maximale Anzahl von Prozessen, die ein Nutzer gleichzeitig aktiviert haben darf.
Der Wert sollte so festgelegt werden, daß sinnvoll gearbeitet werden kann, jedoch ein Nutzer nicht alle möglichen Prozesse belegt.

MAPSIZ Bestimmt die Anzahl von Fragmenten, in die der Hauptspeicher bzw. der Auslagerungsbereich unterteilt werden können. Das theoretische Maximum liegt bei der doppelten Anzahl von möglichen Prozessen (2 x NPROC).
Es werden 8 Byte x (MAPSIZ + 1) belegt.

NCALL ist die Anzahl möglicher simultaner Aufrufe der Timeout-Routine (systemintern).
Die Callout-Tabelle belegt 6 Byte/Aufruf.

NPROC Legt die maximale Zahl von Prozessen fest und hängt vom Anforderungsspektrum typischer Nutzer ab. Je Prozeß werden 28 Byte benötigt.

NTEXT bestimmt die maximale Anzahl reiner Textsegmente (Code), d.h. simultan arbeitender reiner Programme.
Pro Textsegment werden 11 Byte belegt.

NCLIST Anzahl von Segmenten der "clist", in der die Daten zeichenweise arbeitender Geräte zwischengespeichert werden. Jedes Segment enthält Speicherplatz für 14 Zeichen. NCLIST sollte so groß gewählt werden, daß pro Terminal (einschließlich Seriendrucker) etwa 80 Zeichen (eine Zeile) zwischengespeichert werden können.

Entsprechend den Angaben zu den einzelnen Parametern können solche Werte gewählt werden, die ein effektives System entstehen lassen. Als Orientierung ist c.c in der Auslieferungsform zu nutzen. Die dort angegebenen Parameter wurden für das ausgelieferte Basissystem eingestellt. Die Größe bestimmter Komponenten (z.B. Tabellen, Felder, Routinen, Driver) ist aus der Symboltabelle des Files /mutos ersichtlich, die man mit nm(1) erhält. Die Werte für die beim Systemstart eingestellten Größen "rootdev", "pipedev", "swapdev", "swplö und "nswap" sind ebenfalls in c.c zu finden und bei Bedarf zu modifizieren. Dafür wird ein Makro "makedev" genutzt, dessen Parameter die Major- und Minor-Nummer entsprechend den Einträgen in /dev sind.

3.3. Directory /usr/sys/cfg

Diese Directory enthält Files für die einzelnen E/A-Schnittstellen zur Festlegung von Adressen und bestimmten Parametern. änderungen entsprechend der gewünschten Konfiguration sind hier möglich. Von Bedeutung sind vor allem die Files c5170.c für den KES K5170 und cram.c für eventuell gewünschte Standardeinstellungen zur Behandlung von Hauptspeicherbereichen in spezieller Form. Für den KES können die konkreten Parameter der Festplatte (Anzahl Zylinder/Spuren/Sektoren, Partitionaufteilung) sowie die Schrittrate bei Floppy-Laufwerken sinnvoll modifiziert werden (siehe auch KES(4)). Prinzipielle änderungen an den Tabellen sind zu vermeiden (z.B. Streichen von Einträgen), da dadurch die Bezüge zu den Minor-Nummern in /dev verändert werden.

3.4. Ergebnis des Generierungslaufs

In der Directory /usr/sys/conf entsteht das File mutos. Es kann für eine Floppy-orientierte Arbeitsweise über strip(1) noch um die Symboltabelle verkürzt werden.
Für die Erprobung des neuen Systems ist es günstig, das bisher benutzte /mutos aufzuheben, z.B. durch Umbenennung:

```
mv /mutos /mutos.old
```

Danach kann das neue System in die Root-Directory kopiert werden:

```
cp mutos /mutos
```

Arbeitet das neue System nach boot(8) ordnungsgemäß, kann /mutos.old gelöscht werden. Im Fehlerfall ist ein Einlesen des alten Systems immer noch möglich.

3.5. Zusätzliche Schritte

Einige Systemprogramme greifen auf Informationen in den Header-Files für den Systemkern zu (z.B. auf param.h). Diese werden in der Directory /usr/include/sys erwartet. Durch Kopieren der Header-Files aus /usr/sys/h in diese Directory sind diesen Programmen die gültigen Header zur Verfügung zu stellen.
Programme, die generierungsabhängig übersetzt werden müssen, stehen in der Directory /usr/src/cmd. Dort steht auch ein "makefile" zum Bilden dieser Programme zur Verfügung.

3.6. Anschluß von Terminals, die nicht zum Systembestand gehören

Sollen Terminals anderen Typs als K8911 angeschlossen und im Mehrnutzerbetrieb bedient werden, ist das durch Ergänzung der Tabelle im Kommando getty(8) möglich. Die Quelle zu diesem Programm befindet sich in /usr/src/cmd. Für den neuen Terminaltyp ist ein weiteres Kennzeichen einzuführen, und die Parameter in der zugehörigen Tabelle sind entsprechend einzustellen (siehe tty(4)). Das eingeführte Kennzeichen ist im File /etc/ttys zu benutzen. Nach Übersetzung von getty.c und kopieren von getty nach /etc wird der neue Terminaltyp im Mehrnutzeranlauf berücksichtigt.
Voraussetzung für einen derartigen Schritt ist natürlich die prinzipiell ähnliche Arbeitsweise des neuen Gerätes wie K8911.
Dazu gehören:

- Asynchronbetrieb
- Datenbreite 7 oder 8 Bit
- Paritätsbehandlung beliebig
- übertragungsgeschwindigkeiten (Baud) 9600, 4800, 2400, 1200
  600, 300, 150, 75, 50
- keine Übertragungsprozedur benutzt