SOFTWARE

WEGA

DIENSTPROGRAMME
BAND C

EAW *electronic*

P8000

Version 1.1 (2008-03-30)

EAW-electronic P8000

W E G A - S o f t w a r e

Dienstprogramme

(Band-C)

Diese Dokumentation wurde von einem Kollektiv des Kombinates

VEB ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW "FRIEDRICH EBERT"

erarbeitet.

Herausgeber:

Kombinat
VEB ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW "FRIEDRICH EBERT"
Hoffmannstrasse 15-26
BERLIN
1193

WAE/03-0204-02

Ausgabe: 12/88

Die vorliegende Dokumentation unterliegt nicht dem Aenderungsdienst.

Spezielle Hinweise zum aktuellen Stand der Softwarepakete befinden sich in README-Dateien auf den entsprechenden Vertriebsdisketten.

Dieser Band-C enthaelt folgende Unterlagen:

Teil 1: VI
Beschreibung

Teil 2: MM
Makropaket

Teil 3: ADB
Debugger

Teil 4: FSCK
Dateisystemueberpruefung

Gesamtinhaltsverzeichnis Seite

Teil 1: VI-Beschreibung

Teil 2: MM-Makropaket

Hinweise des Lesers zu diesem Dokumentationsband

Wir sind staendig bemueht, unsere Unterlagen auf einem qualitativ hochwertigen Stand zu halten. Sollten Sie deshalb Hinweise zur Verbesserung dieses Dokumentationsbandes bzw. zur Beseitigung von Fehlern haben, so bitten wir Sie, diesen Fragebogen auszufuellen und an uns zurueckzusenden.

Titel des Dokumentationsbandes: WEGA-Dienstprogramme (Band-C)

Ihr Name / Tel.-Nr.:

Name und Anschrift des Betriebes:

Genuegt diese Dokumentation Ihren Anspruechen? ja / nein
Falls nein, warum nicht?

Was wuerde diese Dokumentation verbessern?

Sonstige Hinweise:

Fehler innerhalb dieser Dokumentation:

Unsere Anschrift: Kombinat VEB ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW "FRIEDRICH EBERT"
Abteilung Basissoftware
Hoffmannstrasse 15-26
BERLIN
1193

Notizen:

Beschreibung des bildschirmorientierten Editors vi

Vorwort

Vi (visuell) ist ein bildschirmorientiertes interaktives Bildschirmaufbereitungsprogramm, in dem der Bildschirm wie ein Fenster in der aufzubereitenden Datei wirkt. Veraenderungen werden auf dem Bildschirm dargestellt, wodurch Modifizierungen erleichtert werden. Die Regelmaessigkeit und mnemonische Zuordnung der Befehle erleichtern den Gebrauch des Befehlssatzes und unterstuetzen die Merkfaehigkeit. Der volle Befehlssatz des traditionellen, zeilenorientierten Editors ex steht mit vi zur Verfuegung und eine Umschaltung zwischen beiden ist leicht moeglich.

Inhaltsverzeichnis Seite

1. Einleitung

1.1. Allgemeines

Vi kann an vielen Bildschirmterminals eingesetzt werden. Neue Terminals werden nach Aufbereitung einer das Terminal beschreibenden Datei (/etc/termcap) betrieben. Obwohl der Einsatz eines intelligenten Terminals, mit dem man von einem lokalen Bildschirmgeraet aus Zeilen und Zeichen einfuegen und loeschen kann, vorteilhaft ist, wird auch eine einwandfreie Funktion des Editors an nichtintelligenten Terminals gewaehrleistet. Der Editor optimiert die Reaktionszeit, indem ein kleineres Fenster und ein differnzierter Algorithmus fuer die Aktualisierung des Bildschirms angewandt wird.

1.2. Befehlsdarstellung

Fuer die Beschreibung von Befehlen werden folgende Darstellungen benutzt:

< > Die einzugebenden beschreibenden Namen der Daten oder Datenelemente werden in spitze Klammern eingeschlossen. Z.B. <dateiname>.

[ ] Optionelle Daten werden in eckige Klammern eingeschlossen.

| Senkrechter Strich bezeichnet eine Oder-Funktion.

ESC Bezeichnung fuer die Escape-Taste (die auf einigen Tastaturen als ALT bezeichnet wird).

RUB Bezeichnet die Delete-Taste (die auf einigen Tastaturen als DEL bezeichnet wird).

CTRL Bedeutet CONTROL-Taste. An manchen Terminals wird CTRL mit dem Zeichen (^) dargestellt. Dieses Zeichen ist nicht mit dem nach oben gerichteten Pfeil zu verwechseln

^ Bedeutet Pfeil nach oben.

1.3. Spezial-Zeichen

ESC-Taste: Diese Taste loescht teilweise eingegebene Kommandos. Sie beendet ebenfalls Operationen im Textmodus. Befindet sich das Terminal bereits im Ruhezustand, kann diese Taste auch ein Tonsignal ausloesen.

RETURN-Taste: Diese Taste loest das Ausfuehren der meisten Kommandos aus. Mit ihr werden ebenfalls csh-Kommandos (C-Shell) gestartet.

RUB-Taste: Diese Taste unterbricht und stoppt den Editor.

Ein Unterbrechen des Editors waehrend des Umwandelns oder Aktualisierens groesserer Teile des Bildschirmes, kann eine durcheinander geratene Darstellung verursachen. Tritt dies auf, ist es immer noch moeglich, die Aufbereitung fortzusetzen, indem:

a) das Kommando CTRL-z eingegeben wird, wobei die Bildschirmdarstellung wiederhergestellt wird.

b) die Darstellung auf dem Bildschirm ignoriert wird und das Verschieben oder Suchen wiederholt wird.

In dieser Unterlage ist der Gebrauch von RUB einem Interrupt aequivalent.

Schraegstrich (/): Dieses Symbol bezeichnet einen Such-String. Bei Betaetigen dieser Taste bewegt sich der Cursor auf die letzte Zeile des Bildschirms, wo er als ein Prompt wirkt. Um den Cursor an die aktuelle Position zurueckzubringen, wird RUB betaetigt. Die Eingabe des Schraegstriches mit nachfolgender Betaetigung der Backspace-Taste beendet ebenfalls die Suche.

"Zeile loeschen" und "Zeichen loeschen" sind nutzerprogrammierbar. Sie koennen mit dem Programm stty (siehe STTY(1) im WEGA-Programmierhandbuch) veraendert werden.

Zeile loeschen: Das Zeichen hierfuer ist normalerweise CTRL-x.

Zeichen loeschen: Das Loesch-Zeichen ist normalerweise CTRL-h.

1.4. Aufrufen von vi

Wenn das System eingeschaltet und angelaufen ist, wird der Terminaltyp wie folgt gesetzt:

```
%setenv TERM <code> (RETURN)
```

dabei bedeuten:

% ist der Systemprompt

setenv Kommando fuer das Setzen der Umgebung

TERM ein erforderliches Schluesselwort

<code> der einzugebende Code fuer den Terminaltyp

Fuer das P8000-Terminal ist <code> P8 (fuer die ADM31 kompatible Variante des Terminals oder PV (fuer die VT100 kompatible Variante) und (RETURN) ist die RETURN-Taste

Das Kommando-Format lautet:

% vi [-t <tag>] [-r[<filename>]] [+[<command>]] [+<n>] [+/<string>] [-1] [<filename>] (RETURN)

dabei bedeuten:

% Systemprompt.

vi Kommando zum Aufrufen des visuellen Bildschirmeditors.

-t <tag> Option zum Aufbereiten der Datei mit dem Identifizierungskennzeichen <tag> bei <tag>

-r [<filename>] Option fuer die Rueckgewinnung einer Datei nach einem Editor- oder Systemzusammenbruch (siehe Abschn. 7.2.).

+ [<command>] fuehrt das ex-Kommando <command> vor dem Eintritt in den visuellen Modus aus. Ohne <command> beginnt der visuelle Editor am Ende der Datei (siehe auch Abschn. 5.2.).

+<n> startet den visuellen Editor bei Zeile Nummer <n>.

+/<string> veranlasst vi nach einem String <string> zu suchen und dort zu beginnen.

-l Option fuer das Setzen der Aufbereitungsoptionen fuer LISP (siehe Abschn. 6.).

<filename>Name der aufzubereitenden Datei.

Achtung

Eckige Klammern ([]) und spitze Klammern (<>) duerfen in das Kommando nicht einbezogen werden.

Beispiele:

1. Das einfachste vi-Kommando besteht im Aufrufen von vi fuer die Aufbereitung einer einfachen Datei:

   %vi <filename< <RETURN>

   dabei ist <filename> der Name der aufzubereitenden Datei.

   Achtung

   Alle Eingaben muessen mit einem RETURN abgeschlossen werden. Im weiteren wird weder RETURN noch System-Prompt in den Systemkommandos aufgefuehrt, es wird jedoch davon ausgegangen, dass jedes Kommando mit RETURN abgeschlossen wird.

2. Fuer den Start von vi bei Zeile Nummer n benutzt man die Form

%vi [+<n>] <filename>

3. Fuer den Start von vi bei einem String <string> benutzt man die Form

%vi [+/<string>] <filename>

vi sucht nach <string> und beginnt, wenn es dieses gefunden hat, bei <string>. Fuer zusaetzliche Details siehe Abschn. 5.2. Nachdem das Kommando eingegeben ist, erscheint der Dateiname auf dem Bildschirm. Der Editor modifiziert nicht direkt die Datei, die aufbereitet wird. Er kopiert die Datei in einen Zwischenspeicher und merkt sich den Dateinamen. Der Inhalt der Datei wird nicht beruehrt, bis die Veraenderungen in die Originaldatei zurueckgeschrieben werden. Nachdem die Datei kopiert worden ist, bereitet vi diese Datei auf. Der Bildschirm wird geloescht und der Text der Datei erscheint. Wenn dies nicht geschieht, ist folgendes zu unternehmen:

1. Es ist zu ueberpruefen, ob der Code des Terminaltyps korrekt ist. Die Eingabe eines falschen Code erzeugt eine unkorrekte Bildschirmausgabe. Fuer die Kontrolle wird vi verlassen und

:q

eingegeben. Nachdem RETURN wird die Kontrolle an die Shell (Kommando-Interpreter) zurueckgegeben. Um den Terminalcode zu ueberpruefen gibt man

printenv TERM

ein und wiederholt die Eingabe, wie oben beschrieben.

2. Kontrolle der Richtigkeit des Dateinamens. Ein falscher Dateiname kann auf dem Bildschirm zu einer Fehleranzeige fuehren. Ist dies der Fall, kehrt man zur Shell zurueck (wie in 1. gezeigt) und beginnt von neuem.

3. Reagiert der Editor nicht, so unterbricht man ihn mit der Loeschtaste DEL (oder RUB). Danach kehrt man zur Shell (wie in 1. dargestellt) zurueck.

1.5. Betriebsarten

Vi verfuegt ueber 4 Betriebsarten:

1. Kommandobetriebsart. Dies ist der Ausgangszustand. Die anderen Betriebsarten kehren zu dieser Art zurueck. Die Taste ESC loescht alle teilweise eingegebenen Kommandos.

2. Textbetriebsart. In diese Betriebsart gelangt man ueber die folgenden Operatoren:

   a A i I o O c C s S R

   In dieser Art kann jeder beliebige Text eingegeben werden. Die Texteingabe wird normalerweise mit ESC abgeschlossen. Auch die Eingabe von Text kann mit RUB (aus besonderen Gruenden) beendet werden.

3. Last-line-Betriebsart. Diese Art wird durch die folgenden Operatoren ausgeloest:

   : / ? !

   Nach Eingabe von RETURN oder ESCAPE laufen Kommandos oder Stringsuchen ab. Kommandos werden mit DEL (oder RUB) geloescht. In dieser Betriebsart des Editors werden die Kommandos auf der letzten Zeile ausgegeben. Befindet sich der Cursor in der ersten Position der letzten Zeile, so fuehrt der Editor eine Berechnung aus, wie z.B. die Berechnung einer neuen Position in der Datei nach einer Suche, oder er laesst ein Kommando ablaufen, um einen Teil des Zwischenspeichers umzuformatieren. Waehrend dieses geschieht, ist es moeglich, den Editor mit RUB zu unterbrechen. In manchen Systemen ist es nicht moeglich weiterzuschreiben, wenn sich der Cursor auf der letzten Zeile befindet und der Editor unterbrochen wurde.

4. Offene Betriebsart (open mode). Diese Betriebsart wird in Abschnitt 1.9. beschrieben.

1.6. Umschalten auf die Shell

Um in vi ein Shell-Kommando auszufuehren, nutzt man ein Kommando der Form

:!<command>

wobei <command> das Shell-Kommando ist. Das System laesst das <command> ablaufen und kehrt zu vi zurueck, wenn das Kommando abgearbeitet ist. Dem Bediener wird gemeldet:

Hit RETURN to continue (zur Fortsetzung RETURN-Taste druecken)

Nachdem RETURN eingegeben wurde, loescht der Editor den Bildschirm und wiederholt das vorhergehende Bild. Vi uebernimmt wieder die Kontrolle und die Aufbereitung kann

fortgesetzt werden. Wenn jedoch ein weiteres (:)-Kommando vor dem RETURN eingegeben wurde, wird das vorhergehende Bild nicht wiederholt. Fuer die Ausfuehrung mehrerer Kommandos im Shell wird das Kommando

:sh

eingegeben. Sind alle notwendigen Shell-Kommandos ausgefuehrt, kehrt man mit der Eingabe

CTRL-d

zu vi zurueck. Vi loescht den Bildschirm und die Bearbeitung kann fortgesetzt werden. Um mehr als ein Kommando in der C-Shell auszufuehren, gibt man das Kommando

:csh

ein.

1.7. Verlassen von vi

Um vi zu verlassen und in die Shell zurueckzukehren, wird das Kommando

ZZ

benutzt. Wenn Veraenderungen am Text vorgenommen wurden, wird der Inhalt des vi-Zwischenspeichers in die Originaldatei zurueckgeschrieben und der Editor wird verlassen. Sind keine Veraenderungen erfolgt, wird der Editor verlassen.

Es ist auch moeglich, Veraenderungen in die Datei zu schreiben, ohne vi zu verlassen, indem das Kommando

:w

benutzt wird.

Um vi zu verlassen (quit), ohne die Veraenderungen zu schreiben, wird das Kommando

:q!

benutzt. Damit werden alle Textveraenderungen aufgegeben. Das Kommando ist dafuer geeignet, um Veraenderungen am Inhalt des Zwischenspeichers vorzunehmen und die Originaldatei unveraendert zu lassen. Muessen die Veraenderungen erhalten bleiben, darf dieses Kommando nicht benutzt werden.

1.8. vi und ex

Vi ist eine Editierart im zeilenorientierten Editor ex. Einige Operationen sind in ex leichter auszufuehren als in vi, z.B. systematische Veraenderungen im zeilenorientierten Material. Erfahrene Anwender vermischen vi- und ex- Kommandos, um ihre Arbeit zu erleichtern.

Wenn vi gerade laeuft, ist es moeglich, mit dem Kommando

Q

auf ex umzuschalten. Es meldet sich mit einem Doppelpunkt (:). Die in dieser Unterlage beschriebenen und mit einem vorangestellten Doppelpunkt versehenen vi-Kommandos, sind fuer ex verfuegbar. Aehnliches trifft fuer die meisten ex-Kommandos zu, die fuer vi verfuegbar sind, wenn sie mit einem vorangestellten Doppelpunkt versehen sind.

In seltenen Faellen kann ein interner Fehler in vi auftreten. In einem solchen Fall wird auf dem Bildschirm eine Fehlerdiagnose ausgegeben, vi verlassen und die Kontrolle an die Kommandobetriebsart von ex zurueckgegeben. Es ist dann entweder moeglich:

1. die begonnene Arbeit abzuspeichern und mit der Eingabe des Kommandos

   x

   die Arbeit abzubrechen oder,

2. vi mit dem Kommando

   vi

   wieder aufzurufen.

1.9. Anwendung von vi auf Hardcopyterminals

Es ist moeglich, vi auf einem Hardcopy-Terminal oder einem Terminal zu benutzen, dessen Cursor nicht von der Grundlinie weg bewegt werden kann. Auf diesen Terminals laeuft vi in der "offenen" Betriebsart (open mode). Der Name stammt vom open-Kommando in ex ab, das den open mode aufruft. Auf einem nichtintelligenten Terminal tritt vi automatisch in die offene Betriebsart (open mode) ein.

Um open mode manuell aufzurufen, wird ex und danach, von ex aus, das Kommando

open

eingegeben. Fuer die Rueckkehr aus open mode in ex gibt man das Kommando

Q

ein. Fuer die Rueckkehr aus ex nach vi gibt man

vi

ein. Die Unterschiede zwischen der visuellen und der offenen Betriebsart sind:

1. Die Art und Weise der Textdarstellung. Im open mode benutzt der Editor ein einzeiliges Fenster in der Datei. Eine Vorwaerts- und Rueckwaertsbewegung in der Datei ergibt neue Zeilen auf dem Bildschirm, die immer unterhalb der vorhergehenden Zeile erscheinen.

2. Das Kommando

   z

   nimmt keine Parameter an, sondern gibt ein "zusammenhaengendes Fenster", bestehend aus Zeilen, die der aktuellen Zeile vorausgehen oder nachfolgen, aus und kehrt zur aktuellen Zeile zurueck.

3. Auf einem Hardcopy-Terminal wird mit dem Kommando

   CTRL-r

   die aktuelle Zeile wiederholt. Auf diesen Terminals benutzt der Editor gewoehnlich zwei Zeilen, um die aktuelle Zeile darzustellen. Dabei ist die erste Zeile eine Kopie der Originalzeile und die zweite Zeile, die Arbeitszeile, d.h. auf ihr werden alle Veraenderungen dargestellt. Wenn Zeichen geloescht werden, gibt der Editor eine Reihe von (\), um zu zeigen, welche Zeichen geloescht wurden. Der Editor gibt auch die aktuelle Zeile aus, nachdem solche Veraenderungen vorgenommen wurden, so dass diese sichtbar werden.

1.10. Terminals mit Grossbuchstaben

Alle Zeichen werden in Kleinbuchstaben umgewandelt. Jedem Grossbuchstaben muss jedoch ein nach links geneigter Schraegstrich vorangestellt sein. Die Kombination "\-Zeichen" wird solange nicht angezeigt, bis der Schraegstrich von einem zweiten Zeichen gefolgt wird.

Die folgenden Zeichen sind auf Terminals mit Grossbuchstaben nicht verfuegbar:

{ } | ~ `

Anstelle dieser Zeichen kann folgendes eingegeben werden:

{ wird ersetzt durch \(
} wird ersetzt durch \)
~ wird ersetzt durch \^
| wird ersetzt durch \!
` wird ersetzt durch \'

1.11. Langsam arbeitende Terminals

Der vi-Editor minimiert die Verzoegerungszeit, die fuer die Aktualisierung des Bildschirmes erforderlich ist, indem die Ausgabe an den Bildschirm begrenzt wird. Fuer langsame und nichtintelligente Terminals opitimiert vi die Aktualisierung der Anzeige waehrend des Textmode und ersetzt geloeschte Zeichen mit dem Symbol "@". Auf langsamen Terminals, die vi in vollem Bildschirmmode unterstuetzen, ist es sinnvoll, den "open" Mode zu benutzen.

Vi verfuegt ueber eine Betriebsoption (slowopen), die fuer ein langsames Terminal geeignet ist. Zusaetzliche Informationen siehe Abschn. 6.2.

1.12. Abkuerzungen

Vi verfuegt ueber eine Reihe von Kurzkommandos, die die oben vorgestellten laengeren Kommandos abkuerzen. Diese Kommandos sind in der entsprechenden Liste aufgefuehrt.

1.13. Zeilennummern

Der vi-Editor kann, wenn es gewuenscht wird, jede Zeile nummerieren. Dafuer wird die Editor-Option "number" (Option fuer Zeilennummerierung), die im Abschn. 6.8. beschrieben wird, benutzt.

1.14. Zeilendarstellung auf dem Bildschirm

Der vi-Editor formt lange logische Zeilen in kurze physische Bildschirmzeilen um. Kommandos, die eine Zeile bewegen, bewegen auch eine logische Zeile. Folglich werden alle Segmente einer Zeile in einer Bewegung uebersprungen. Das Kommando

|

bewegt den Cursor auf eine angegebene Spalte. Das kann nuetzlich sein, um ungefaehr in die Mitte einer langen Zeile zu gehen und diese zu teilen. (Das Kommando ist ein senkrechter Strich, keine 1 oder l.) Zum Beispiel setzt das Kommando

80|

den Cursor auf die 80. Spalte eines langen Satzes.

Bei einem nichtintelligentem Terminal bringt der Editor nur volle Zeilen auf den Bildschirm. Wenn auf dem Bildschirm nicht genug Platz ist, um eine logische Zeile unterzubringen, laesst der Editor die physische Zeile leer und setzt ein "@" als Platzmarkierung auf die Zeile. Wenn Zeilen geloescht werden, wird oft nicht der gesamte Bildschirm umgeschrieben, sondern der Editor loescht nur jede Textzeile und setzt ein "@", um Zeit zu sparen. Um ein Maximum an Informationen auf dem Bildschirm darzustellen, gibt man

CTRL-r

ein.

1.15. Hinweiszeichen fuer Dateiende

Wenn das Ende der Datei erreicht ist und die letzte Zeile nicht die letzte Zeile des Bildschirmes ist, setzt der vi-Editor die Tilde (~) an die linke Seite jeder verbleibenden Zeile. Damit wird angezeigt, dass die letzte Zeile der Datei auf dem Bildschirm dargestellt wird, und dass jene Zeilen mit der Tilde ausserhalb der Datei liegen.

1.16. Wiederholungen (counts)

Ein count ist ein Argument, das die Anzahl der Ausfuehrungen eines Kommandos oder die Anzahl der Zeilen beruehrt. Verschiedene vi-Kommandos koennen mit einem vorangestellten count versehen werden, das die Ausfuehrung des Kommandos beeinflusst. Am gebraeuchlichsten sind folgende:

1. Fuer die folgenden Kommandos beeinflusst ein vorangestellter count die Anzahl der Verschiebungen (beim Rollen):

   CTRL-d CTRL-u

2. Fuer die folgenden Kommandos betrifft count die Zeilen- oder Spaltennummer:

   z G | (senkrechter Strich)

3. Fuer die meisten vi-Kommandos bedeutet ein vorangestelltes count die Anzahl der Kommandowiederholungen. Z.B. bewegt das Kommando

```
                5w
```

den Cursor 5 Worte weiter. Das Kommando

```
                5dw
```

loescht 5 Worte und

```
                3.
```

loescht 3 weitere Worte.

2. Steuerung der Bildschirmdarstellung

2.1. Zeilenweises Rollen

Fuer ein zeilenweises Rollen des Bildschirmes werden folgende Kommandos benutzt:

```
[<n>]CTRL-u   n Zeilen nach oben rollen
[<n>]CTRL-d   n Zeilen nach unten rollen
```

Wenn n weggelassen wird, so betraegt der Standardwert die Haelfte der Fenstergroesse.

Achtung

Einige nichtintelligente Terminals koennen nicht nach oben rollen. In solchen Faellen loescht CTRL-u die Anzeige und frischt sie mit einer Zeile auf, die in der Datei weiter zurueckliegt (in Richtung des Anfangs).

2.2. Seitenweises Rollen

Die Funktionen CTRL-f und CTRL-b bewegen das Sichtfenster jeweils eine Seite vor- bzw. rueckwaerts. Beide Kommandos bewirken, dass einige Textzeilen der vorhergehenden Seite erhalten bleiben, um den Zusammenhang zu wahren. Es ist moeglich, eine Datei durchzulesen, wobei anstelle der Zeilenkommandos die Seitenkommandos benutzt werden. Der grundlegende Unterschied besteht darin, dass die Zeilenkommandos den Text langsam weiterruecken und mehr vom vorhergehenden Text erhalten bleibt, waehrend die Seitenkommandos eine Seite auf einmal veraendern, wobei nur einige Textzeilen fuer die Fortsetzung erhalten bleiben.

2.3. Suche nach Zeichenketten (string)

Die Suchfunktion positioniert den Bildschirm ebenfalls innerhalb einer Datei. Diese Funktion durchsucht die Textdatei nach einer bestimmten Zeichenkette (string) und positioniert den Cursor an die Stelle, wo die angegebene Zeichenkette zunaechst auftritt. Das Suchkommando lautet:

```
/<string>
```

Um von der Cursorposition ab rueckwaerts zu suchen, benutzt man das Kommando:

```
?<string>
```

Um die Vorwaerts- oder Rueckwaertssuche nach der Zeichenkette bis zum Auftreten des naechsten <string> zu wiederholen, wird das Kommando

n

benutzt. Um die Zeichenkettensuche in der umgekehrten Richtung zu wiederholen, gibt man

N

ein. Wenn <string> in der Textdatei nicht vorhanden ist, gibt vi die Nachricht "pattern not found" auf der letzten Zeile des Bildschirmes aus und bewegt den Cursor auf seine Ursprungsposition zurueck. Die string-Suche wird normalerweise so ausgefuehrt, dass sie nach Erreichen des Dateiendes am Anfang der Datei wieder fortgesetzt wird. So kann die Zeichenkette gefunden werden, auch wenn sie nicht in der urspruenglich im Kommando angegebenen Richtung liegt (vorausgesetzt die Zeichenkette befindet sich tatsaechlich in der Datei). Die Umlaufsuchfunktion (wraparound) kann ueber die Editor-Option "nowrapscan" (oder "nowrs") ausgeschaltet werden. Diese Option ist eine der Optionen, die in Abschnitt 6 kurz beschrieben werden.

Wenn die gesuchte Zeichenkette am Anfang einer Zeile stehen soll, dann ist dem Such-String ein nach oben gerichteter Pfeil voranzustellen. Soll die Zeichenkette nur mit dem Ende einer Zeile uebereinstimmen, dann ist das Such-String mit einem $-Zeichen zu beenden.

Beispiele:

/^search

sucht nach dem Wort "search" am Beginn einer Zeile, und

/last$

sucht nach dem Wort "last" am Ende einer Zeile.

Wenn die Suchzeichenkette einen Schraegstrich (/) enthaelt, muss ihm ein Backslash (\) vorangestellt werden. Das trifft ebenfalls zu, wenn die Editor-Option "magic" gesetzt ist (siehe Abschnitt 6).

Am Ende der Zeichenkettensuche plaziert vi den Cursor an die Stelle des naechsten oder des vorhergehenden Auftretens der Zeichenkette.

Es koennen ganze Textzeilen bis zu der Zeile, in der sich die Zeichenkette befindet, betroffen sein. Dafuer wird das Suchkommando mit der Form

/<string>/-<n>

benutzt.

Dabei sind:

<string> Teil des Suchkommandos und
<n> die Anzahl der Zeilen, die sich vor der Zeile befinden, die die Zeichenkette enthaelt.

Das "-" kann durch ein "+" ersetzt werden. Dabei werden <n> Zeilen nach der Zeile, die das <string> enthaelt, angezeigt. Wenn kein Zeilen-Offset einbezogen ist, behandelt der Editor anstelle von ganzen Zeilen nur Zeichen bis zum Punkt der Uebereinstimmung der Zeichenkette. Somit ist "+0" zu benutzen, um die Zeile zu treffen, die uebereinstimmt. Der Editor ignoriert die Gross- oder Kleinschreibung von Woertern in der Zeichenkettensuche, wenn das im Kommando beruecksichtigt wird. Eine kurze Beschreibung wird dafuer in Abschnitt 6 unter "Weglassen der Unterscheidung von Gross- und Kleinschreibweise" gegeben.

Die Suche nach Zeichenketten kann ebenso in Verbindung mit den Operatoren "d" und "c" (siehe Abschnitt 3.4) und "y" (siehe Abschnitt 4.3) benutzt werden.

2.4. Steuerung der Cursorposition

Um den Cursor auf eine bestimmte Zeile zu positionieren, wobei die Zeilen durch eine Nummer identifiziert werden, wird das Kommando

[<n>]G

benutzt, wobei n eine Zeilennummer ist. Somit bewegt 1G den Cursor auf die erste Zeile in der Datei. Wenn <n> weggelassen wird, bedeutet der Standardwert die letzte Zeile der Datei.

Der Cursor kann aufwaerts, abwaerts, vorwaerts und rueckwaerts mittels folgender Tasten bewegt werden.

aufwaerts: k, CTRL-p oder CTRL-k
abwaerts: j, CTRL-n oder CTRL-j

rueckwaerts: h, CTRL-h oder Ruecktaste
vorwaerts: Leertaste oder kleines l

Einige Terminals haben Tasten mit Pfeilen (4 oder 5 Tasten mit Pfeilen in verschiedene Richtungen), die die gleichen Funktionen besitzen (an manchen Terminals muessen die Funktionstasten ueber die Umschalttaste bedient werden).

Um den Cursor an die erste, mit einem Zeichen belegte Position der naechsten Zeile in der Datei zu bringen, wird die RETURN oder "+"-Taste betaetigt. Wird "-" betaetigt, so wird der Cursor auf die erste, mit einem Zeichen belegte Position auf der vorhergehenden Zeile zurueckgefuehrt. Diese Tasten koennen ebenso fuer das Rollen benutzt werden, wenn sich der Cursor auf der oberen bzw. der unteren Zeile

des Bildschirms befindet.

Vi verfuegt auch ueber Kommandos, um den Cursor an den Anfang, die Mitte oder die letzte Zeile des Bildschirms zu positionieren. Um auf die obere Zeile zu gehen, betaetigt man die H-Taste. Wird

<n>H

betaetigt, wird der Cursor n-Zeilen von der oberen Zeile des Bildschirms herabbewegt. <n> ist eine Option. Der Anfang des Bildschirmes ist die Standardposition. Das Kommando "M" positioniert den Cursor auf die Mitte des Bildschirmes. Das Kommando

<n>L

bringt den Cursor entweder auf die letzte Zeile des Bildschirms oder auf die Zeile, die n Zeilen von der untersten entfernt ist. Wenn <n> weggelassen wird, so wird der Standardwert genommen, der die Positionierung des Cursors auf die letzte Zeile des Bildschirms bedeutet.

Der Cursor kann innerhalb einer Zeile mit den folgenden Kommandos verschoben werden. Um den Cursor auf ein anderes als das erste Wort zu setzen, wird das Kommando

[<n>]w

benutzt, dass den Cursor nach rechts an den Anfang des n-ten Wortes auf der Zeile bringt. Der Standardwert bezieht sich auf ein Wort. Das Kommando

[<n>]b

bewegt den Cursor n Worte zurueck. Der Standardwert ist ein Wort. Das Kommando

[<n>]e

bewegt den Cursor nach rechts an das Ende des n-ten Wortes und nicht an den Anfang dieses Wortes. Der Standardwert ist ein Wort.

Die Kommandos "b", "w" und "e" halten an Satzzeichen an. Um den Cursor vor- oder rueckwaerts ohne Halt am Satzzeichen zu bewegen, werden die Zeichen "W", "B" oder "E" entsprechend benutzt. Die Worttasten schlagen das Ende der Zeile um, d.h. ihre Wirkung wird auf der naechsten Zeile fortgesetzt.

Nachdem der Cursor aus irgendeinem Grund bewegt wurde, kann er an seine vorhergehende Position mit dem Kommando `` (zwei umgekehrte einfache Anfuehrungszeichen) zurueckgesetzt werden. Das Kommando '' (zwei einzelne Anfuehrungszeichen) bewegt den Cursor an das erste

Druckzeichen der Zeile, die die vorhergehende Positionsmarke (die zwei einzelnen Anfuehrungszeichen) enthaelt.

Das ist oft guenstiger, als das Kommando "G", da es keiner Zeilenzaehlung oder anderer Vorbereitung bedarf.

Um den Cursor auf die erste mit einem Druckzeichen versehene Position auf der aktuellen Textzeile zu bewegen, benutzt man entweder "0" oder den nach oben gerichteten Pfeil. Um den Cursor an das Ende der aktuellen Zeile zu fuehren, wird "$" benutzt.

Das Kommando

[<n>]f<c>

bewegt den Cursor an die Stelle, an der das Zeichen <c> zum n-ten Male auftritt. Der Standardwert ist die Position des naechsten Auftretens des Zeichens <c>. Fuer Wiederholungen wird das (;) benutzt. Das Kommando fuer die umgekehrte Richtung lautet

[<n>]F<c>

Damit wird die gleiche Funktion ausgefuehrt, ausser dass der Cursor rueckwaerts in den vorhergehenden Text gefuehrt wird. Wiederholungen erfolgen mit einem Semikolon. Um den Cursor auf das Zeichen zu setzen, dass dem n-ten Auftreten des Zeichen <c> vorangeht, wird

[<n>]t<c>

eingegeben. Um den Cursor rueckwaerts auf das Zeichen, dass dem n-ten Auftreten des Zeichens <c> folgt, wird

[<n>]T<c>

eingegeben. Die Kommandos (f,F,t,T) koennen mit dem Semikolon wiederholt werden oder die Richtung kann mit dem Komma umgekehrt werden.

Um den Cursor auf uebereinstimmende Klammern zu bringen, wird der Cursor entweder auf Eroeffnung oder den Abschluss der Klammer gebracht und die %-Taste betaetigt. Diese Funktion wirkt sowohl fuer geschweifte Klammern ({}) und eckige Klammern ([]).

Um den Cursor auf den Anfang des n-ten folgenden Satzes zu bringen, benutzt man das Kommando:

[<n>])

wobei der Standardwert fuer n 1 ist. Um den Cursor rueckwaerts an den Anfang eines Satzes zu bewegen, wird das Kommando

[<n>](

eingegeben, wobei der Standardwert fuer n 1 ist. Ein Satz wird mit einem Punkt, einem Frage- oder Ausrufezeichen gefolgt entweder von zwei Leerzeichen oder einem Zeilenende, als beendet definiert. Saetze beginnen ebenfalls an Absaetzen oder Kapitelgrenzen. Z.B. bewegt das Kommando

2)

den Cursor einen Satz hinter das Ende des aktuellen Satzes.

Um den Cursor vorwaerts an den Anfang des naechsten Paragraphen zu bringen, benutzt man (}), um den Cursor an den Anfang des vorhergehenden Paragraphen zu bringen, wird ({) benutzt. Um den Cursor ueber mehrere Abschnitte zu bewegen, wird vor die geschweifte Klammer die Anzahl n gesetzt. Z.B. rueckt das Kommando

3}

den Cursor 3 Abschnitte weiter. Ein Abschnitt beginnt nach einer leeren Zeile oder an der Kapitelgrenze.

Mit Hilfe von

]]

wird der Cursor an den Anfang des naechsten Kapitels gebracht. Mit

[[

wird der Cursor zurueck an die Grenze des vorhergehenden Kapitels bewegt.

2.5. Markierungen (tags)

Es ist moeglich, eine Position in der Editordatei mit einem tag, bestehend aus einem Buchstaben zu markieren und dann an jedes spezielle tag zurueckzukehren. Um eine Position im Text mit einem tag zu versehen, wird das Kommando

m<tag>

benutzt, wobei das tag irgendein Buchstabe aus dem Alphabet ist. Um an das tag zurueckzukehren, wird das Kommando

`<tag>

eingegeben.

Wenn Operatoren (wie z.B. der Loesch-Operator) in einer Zeile benutzt werden, die mit tags versehen sind, so kann

es guenstig sein, mit der ganzen Zeile Operationen durchzufuehren (z.B. die gesamte Zeile zu loeschen), anstelle die Operationen bis zur exakten Position des tag durchzufuehren. In diesem Falle wird die Form

anstelle der Form

```
`<tag>
```

benutzt. Z.B. loescht das Kommando

```
d`<tag>
```

ganze Zeilen, beginnend von der Position des Cursors bis zu der Zeile mit dem tag.

2.6. Status der Datei

Mit dem Kommando

```
CTRL-g
```

wird der Dateistatus festgestellt. Der Editor veranlasst die Anzeige von: Name der Datei, die editiert wird, Nummer der aktuellen Zeile, Anzahl der Zeilen im Zwischenspeicher, relative Position im Zwischenspeicher in Prozent.

2.7. Loeschen des Bildschirmes

Wenn aus irgendeinem Grunde die Bildschirmdarstellung auf dem Terminal durcheinander geraten ist, so ist es oft moeglich, ein korrektes Bild wieder herzustellen, indem das Kommando:

```
CTRL-l
```

oder

```
CTRL-z
```

in Abhaengigkeit vom Terminal eingegeben wird. Auf einem nichtintelligentem Terminal ist es moeglich, wenn eine oder mehrere Zeilen geloescht wurden, die Symbole "@" mit dem Kommando

```
CTRL-r
```

oder

```
CTRL-R
```

zu eleminieren. Dadurch wird die Bildschirmanzeige neu erstellt und die geloeschte(n) Zeile(n) geschlossen.

2.8. Groesse des Fensters

Als Fenstergroesse wird die Anzahl der Zeilen bezeichnet, die auf dem Bildschirm abgebildet werden. Vi unterstuetzt die aktuelle oder Standard- Fenstergroesse. Auf Terminals, die mit einer Geschwindigkeit groesser als 1200 Baud arbeiten, benutzt der Editor den vollen Terminal-Bildschirm. Auf langsameren Terminals nutzt der Editor 8 Zeilen als Standard-Fenstergroesse. Auf Terminals, die mit 1200 Baud arbeiten, betraegt die Standard-Fenstergroesse 18 Zeilen.

Die geeignete Fenstergroesse wird benutzt, wenn der Editor den Bildschirm loescht und ihn wieder auffuellt, nachdem eine Suche oder eine andere Bewegung, die ueber die Grenzen des aktuellen Fensters hinausfuehrt, durchgefuehrt wurde. Kommandos, die eine neue Fenstergroesse benoetigen, wie z.B. count (siehe Abschnitt 1.16), bewirken oft, dass das Bild neu dargestellt wird. Im Zusammenhang mit diesen Kommandos kann eine kleinere Fenstergroesse genauso brauchbar sein und es ist moeglicherweise zweckmaessig, eine kleinere Fenstergroesse mit dem entsprechenden Kommando anzugeben. In jedem Falle wird die Anzahl der dargestellen Zeilen erhoeht, wenn:

1. Kommandos wie "-" benutzt werden; diese vergroessern das Fenster nach oben.

2. Kommandos wie "+", RETURN oder CTRL-d benutzt werden; diese bewegen das Fenster nach unten.

Die Roll-Kommandos CTRL-d und CTRL-u haben die zuletzt spezifizierte Groesse fuer das Rollen gespeichert. Der Standardwert ist die halbe Fenstergroesse.

Der Editor vereinfacht das Editieren bei geringen Geschwindigkeiten, indem man mit einem kleinen Fenster beginnt und es in dem Masse erweitert, wie das Editieren voranschreitet. Der Editor ist leicht in der Lage, das Fenster zu erweitern, wenn auf intelligenten Terminals Einfuegungen in die Mitte des Bildschirmes plaziert werden.

Mit dem Kommando

<m>z<n><suffix>

kann das Fenster vergroessert oder verkleinert werden und die aktuelle Zeile oder jede gewuenschte Zeile koennen an jede Stelle des Fensters plaziert werden.

Dabei bedeuten:

<m> die Zeilennummer. Der Standardwert ist die aktuelle Zeile.

z der Kommando-Operator.

<n> Anzahl der Zeilen im Fenster.

<suffix> steuert die Position der gewuenschten Zeile innerhalb des Fensters und kann folgendes sein:

<RETURN> plaziert die Zeile an den Anfang des Bildschirmes

. plaziert die Zeile in die Mitte

- plaziert die Zeile auf die letzte Zeile des Bildschirmes

Z.B. bewirkt das Kommando

z5.

eine Neuauflage der Bildschirmanzeige mit der aktuellen Zeile in der Mitte eines 5zeiligen Fensters, wobei das Kommando

5z5.

die Zeile Nr. 5 in die Mitte eines 5zeiligen Fensters plaziert.

3. Editierkommandos

3.1. Allgemeines

Im allgemeinen benutzen die Editierkommandos den Textmode. Der Textmode wird durch die Eingabe eines der verschiedenen Einfuegekommandos eroeffnet. Nach der Eingabe des Einfuegekommandos werden alle nachfolgenden Tasteneingaben zu Texteinfuegungen. Der Texteinfuegemode wird immer mit der Betaetigung der (ESC)-Taste beendet.

Viele verwandten Editor-Kommandos werden durch die gleiche Buchstabentaste aufgerufen und unterscheiden sich nur darin, dass das eine durch einen Kleinbuchstaben und das andere durch einen Grossbuchstaben angegeben wird. Der Grossbuchstabe unterscheidet sich vom Kleinbuchstaben lediglich im unterschiedlichen Richtungssinn: Der Grossbuchstabe bewirkt eine Operation rueckwaerts und/oder aufwaerts und der Kleinbuchstabe wirkt vorwaerts und/oder abwaerts. Mit allen Kommandos des Textmodes ist es moeglich, einen Buchstaben oder auch mehrere Textzeilen einzufuegen. Um mehr als eine Textzeile einzufuegen, wird die RETURN-Taste waehrend der Eingabe betaetigt. Eine neue Zeile wird damit fuer den Text erstellt und die Einfuegung kann fortgesetzt werden. Bei langsamen oder nichtintelligenten Terminals, kann es vorkommen, dass der Editor wartet, bis der Rest des Bildschirmes neu aufgezeichnet wird. In diesem Falle ueberschreibt der neue Text die existierenden Zeilen auf dem Bildschirm. Dadurch werden Verzoegerungen verhindert, die auftreten, wenn der Editor versucht, den Rest des Bildschirmes zu aktualisieren. Die Bildschirmanzeige wird korrekt aktualisiert, wenn der Textmode beendet wird.

Die Zeichen, die normalerweise auf Systemkommandoebene zum Loeschen von Zeichen oder Zeilen benutzt werden, koennen ebenfalls im Textmode benutzt werden (z.B. CTRL-h oder # und @, CTRL-x oder CTRL-u). Unabhaengig vom Loesch-Zeichen loescht CTRL-h immer das letzte eingegebene Zeichen.

Der Cursor bewegt sich rueckwaerts, aber die Zeichen bleiben auf dem Bildschirm erhalten. Das erweist sich fuer die Eingabe eines aehnlichen Textes als guenstig. Der Bildschirm wird nach einem escape aktualisiert. Um den Bildschirm sofort zu korrigieren, wird die ESC-Taste betaetigt und der Textmode wieder begonnen.

Es ist nicht moeglich im Eingabemodus von einer Zeile auf die vorhergehende Zeile zurueckzugehen. Um eine Korrektur in einer vorhergehenden Zeile vorzunehmen, wird ESC betaetigt und dann der Cursor zurueck in die vorhergehende Zeile bewegt. Die Korrektur wird ausgefuehrt. Es wird zurueckgegangen und dann wird das entsprechende Textkommando erneut eingegeben.

Achtung

Das Zeichen CTRL-w loescht ein ganzes Wort und hinterlaesst ein Leerzeichen nach dem vorangegangenen Wort. Das ist fuer ein schnelles Zurueckkehren und Einfuegen guenstig.

Es ist nicht moeglich, Zeichen mit CTRL-w zu loeschen, wenn sie nicht im Textmode eingegeben wurden.

3.2. Text einfuegen

Die allgemeine Form des Kommandos im Textmode lautet

<n><command><string>ESC

dabei sind:

<n> eine vorangestellte Anzahl, der Standardwert ist 1.

<command> eines der unten aufgefuehrten Kommandos des Einfuegemode.

<string> die eingefuegte Zeichenkette, bestehend aus dem Text.

ESC die Umschalttaste.

Die Wirkung des vorangestellten count <n> besteht darin, die eingefuegte Zeichenkette n mal zu wiederholen. Jeder der folgenden Kommando-Operatoren kann benutzt werden, um den Einfuegemodus zu eroeffnen:

a A i I o O c C s S R

Diese Kommandos und ihre Variationen werden weiter unten beschrieben.

Um einen Text in eine Datei einzufuegen, benutzt man eines der Kommandos aus dem Einfuegemode. Z.B.

i

Alle dem "i" (oder anderen Operatoren des Eingabemodus) folgenden Zeichenketten bestehend aus Zeichen oder Text, die auf dem Terminal eingegeben werden, werden in die Datei eingefuegt, bis der Eingabemode beendet wird. Um den Eingabemode zu beenden, wird ESC (escape) betaetigt. Auf manchen nichtintelligenten Terminals scheint der Bildschirm den Originaltext zu ueberschreiben, wenn ein Text eingefuegt wird. Sobald der Einfuegemode beendet ist, wird der eingefuegte und vorhergehende Text richtig dargestellt.

Eine Variante des "i"-Kommandos ist

^i

die den Text an den Anfang einer Zeile einfuegt. Das Kommando

I

ist ein Aequivalent dazu.

Im allgemeinen kann den meisten Eingabekommandos ein count (Anzahl) vorangestellt werden. Z.B. wiederholt das Kommando

5iApfel

das Wort "Apfel" fuenf mal:

ApfelApfelApfelApfelApfel

In der folgenden Beschreibung wird die vorangestellte Anzahl nicht immer gezeigt.

Das Kommando

a

dient ebenfalls zum Eingang in den vi-Textmode. Der Unterschied zwischen beiden Kommandos besteht darin, dass mit dem Kommando "i" der Text vor dem Cursor (links davon) eingefuegt wird, wobei mit "a" der Text nach dem Cursor (rechts davon) eingefuegt wird. Das Kommando "a" ist guenstig fuer das Anhaengen eines oder mehrerer Buchstaben an ein Wort. Die Anfuegeoperation wird ebenfalls mit der ESC-Taste beendet.

Eine Variante des Kommandos "a" benutzt das $-Zeichen

$a

um den Cursor an das Ende einer aktuellen Zeile zu bringen und einen Text anzufuegen. Ein aequivalentes Kommando ist

A

Ein weiterer Weg, eine oder mehrere Textzeilen an die Datei anzufuegen besteht darin, das Kommando

o

zu benutzen. Dadurch wird der existierende Text geoeffnet und ein neuer Text unter der aktuellen Zeile angefuegt. Das Kommando

O

oeffnet und fuegt neuen Text oberhalb der aktuellen Zeile an. Beide Kommandos werden mit ESC beendet. Ein

vorangestelltes count oeffnet n Zeilen.

Es ist ebenfalls moeglich, nicht druckbare Zeichen in den Text einzufuegen. Dazu ist im Eingabemode CTRL-V und anschliessend das Zeichen einzugeben.

3.3. Loeschen und Einfuegen von Zeichen

Um ein oder mehrere Zeichen zu loeschen, wird der Cursor auf die Position des zu loeschenden Zeichens gebracht. Es wird das Kommando

[<n>]x

benutzt, dabei sind:

<n> die Anzahl der Zeichen oder Leerstellen, die geloescht werden sollen, der Standardwert ist 1.

x das Zeichen fuer das Loeschkommando.

Um ein oder mehrere Zeichen zu loeschen, die vor dem Cursor stehen, wird das Kommando

[<n>]X

benutzt, dabei sind:

<n> die Anzahl der Zeichen und Leerstellen, die geloescht werden sollen, der Standardwert ist 1.

X das Zeichen fuer das Loeschkommando.

Um ein oder mehrere Zeichen zu ersetzen (zu veraendern), benutzt man das Kommando

[<n>]r<c>

dabei sind:

<n> die Anzahl der zu veraendernden Zeichen.

r das Kommando fuer das Ersetzen.

<c> ist auf ein Zeichen begrenzt, das n-mal anstelle der n-mal geloeschten Zeichen wiederholt werden soll.

Um ein oder mehrere Zeichen mit einer Zeichenkette zu ersetzen (zu veraendern), benutzt man das Kommando

[<n>]R<string>

dabei sind:

<n> die Anzahl, wie oft das Ersetzen erfolgen soll.

R das Kommando fuer das Ersetzen.

<string> ist die Zeichenkette, die als Ersatz genommen werden soll. Die Zeichenkette kann beliebig lang sein.

Um eine Anzahl von Zeichen durch mehr als ein Zeichen zu ersetzen:

```
[<n>]s<string>
```

dabei sind:

<n> die Anzahl der zu ersetzenden Zeichen.

s das Ersetzungskommando.

<string> die Zeichenkette, die die geloeschten Zeichen ersetzt. Die Zeichenkette kann beliebig lang sein.

Um die Eingabe der Zeichenkette zu beenden, wird ESC betaetigt.

3.4. Loesch-Operator

Das Kommando

```
d
```

fungiert als Loesch-Operator.

Zum Loeschen von n Worten wird der Cursor positioniert und dann

```
[<n>]dw
```

oder

```
d[<n>]w
```

eingegeben. Der Standardwert ist ein Wort. Um ein Wort rueckwaerts zu loeschen (links vom Cursor), gibt man

```
[<n>]db
```

oder

```
d[<m>]b
```

ein. Der Standardwert ist ein Wort.

Fuer das Loeschen von n einzelnen Zeichen, wird der Cursor auf das entsprechende Anfangszeichen positioniert und das Kommando

[<n>]d<space>

eingegeben. Dieses ist aequivalent zum Kommando x. Der Standardwert ist eine Leerstelle.

Eine Variante des "d"-Kommandos ist

d$

womit der restliche Text auf der aktuellen Zeile geloescht wird. Ein aequivalentes Kommando ist

D

Der Operator "c" veraendert ganze Woerter. Um n Woerter zu aendern, gibt man das Kommando

[<n>]cw

ein. Wenn das Kommando eingegeben ist, wird das Ende des zu veraendernden Textes mit dem Symbol "$" markiert. Der Ersatztext wird eingegeben und die Texteingabe mit ESC abgeschlossen. Der Standardwert ist ein Wort.

Eine Variante des "c"-Kommandos ist

c$

das den Rest des Textes auf der aktuellen Zeile veraendert. Ein aequivalentes Kommando ist

C

Wenn Operationen in einer Textzeile durchgefuehrt werden, wird es oft gewuenscht, dass die Zeichen bis zum erstmaligen Auftreten eines Zeichens geloescht werden. Dafuer wird das Kommando

[<n>]df<x>

benutzt, wobei f<x> das n-malige Auftreten des Zeichens <x>, das nach dem Cursor folgt, lokalisiert. Der Standardwert ist das erstmalige Auftreten von <x>. Dieses Kommando loescht bis zum - und einschliesslich des - Zeichen <x>. Eine Variante des Kommandos lautet

[<n>]dt<x>

dabei wird der Operator f durch t ersetzt. In diesem Falle wird der Text bis zum - aber nicht einschliesslich des - Zeichen (s) <x> geloescht. Das Kommando

T

bewirkt aehnliches, aber arbeitet in umgekehrter Richtung zum t-Operator - d.h. es arbeitet in den vorhergehenden

Text hinein.

Um n ganze Zeilen zu loeschen, wird der Loesch-Operator zweimal gegeben:

[<n>]dd

Der Standardwert ist eine Zeile.

Auf einem nichtintelligenten Terminal kann der Editor manchmal die gesamte Zeile auf dem Bildschirm loeschen und sie durch das Symbol "@" auf der linken Seite ersetzen. Das korrespondiert nicht mit irgendeiner Zeile in der Datei, sondern ist ein Platzindikator. Es hilft ein laenger dauerndes Umzeichnen des Bildschirmes zu umgehen, dass erforderlich waere, um die geloeschten Zeilen aufzufuellen.

Der Operator

[<n>]cc

ist dem Kommando "dd" aehnlich, aber es belaesst vi im Textmode, waehrend das bei "dd" nicht erfolgt. Das Kommando "cc" ist fuer das Aendern einer ganzen Zeile guenstig. Der Cursor ist entsprechend zu positionieren, dann gibt man das Kommando und danach den Ersatztext ein. Die Operation im Textmode wird mit ESC abgeschlossen. Das Kommando

[<n>]S

stellt ein Synonym fuer das Kommando "cc" dar und verhaelt sich analog zum Kommando "s". Dabei ist "s" als eine Zeichensubstitution und "S" als eine Zeilensubstitution zu betrachten. Es gibt weitere Varianten fuer das Zeilen-Loeschkommando. Das Kommando

d<n>L

loescht alle Zeilen, die sich unterhalb des Cursors befinden, bis zur n-ten Zeile von der Grundzeile des Bildschirmes aus gerechnet. Der Standardwert bezieht sich auf alle Zeilen, bis zur letzten Zeile des Bildschirmes.

Es ist weiterhin moeglich, eine Suche nach einer Zeichenkette mit dem Loesch-Operator zu benutzen:

d/<string>

Dieses Kommando loescht Zeichen von der Cursorposition an bis zum Punkt des Uebereinstimmens der Zeichenkette. Das Kommando

d/<string>/-n

loescht Zeichen von der Cursorposition an bis zur n-ten Zeile, die der ensprechenden Zeichenkette vorangeht. Das

Kommando

d/<string>/+n

loescht Zeichen vom Cursor bis zur Zeile n, die der entsprechenden Zeichenkette folgt. Aehnliche Kommandos koennen dafuer genutzt werden, um ganze Zeilen, die in Beziehung mit einer Zeichenkette stehen, zu veraendern:

c/<string>/-n

und

c/<string>/+n

Beim Editieren eines Textes ist es gewoehnlich am einfachsten, satzweise, abschnittsweise und kapitelweise zu editieren. Der Operator "(" und ")" kann mit dem Loesch-Operator benutzt werden. Z.B. loescht das Kommando

[<n>]d)

die restlichen n Saetze. Der Standardwert betrifft den Bereich ab Cursorposition bis zum Ende des aktuellen Satzes. Auf aehnliche Art wird mit

[<n>]d

eine von zwei Loescharten ausgefuehrt:

Befindet sich der Cursor am Anfang eines Satzes, so loescht das Kommando die vorhergehenden n Saetze. Befindet sich der Cursor nicht am Anfang eines Satzes, so loescht das Kommando den Text vom Cursor rueckwaerts bis zum Anfang von n Saetzen. Der Standardwert ist der Anfang des aktuellen Satzes. Der Editor zeigt die Veraenderungen an. Er zeigt ebenfalls an, wenn eine Veraenderung den Text beruehrt, der nicht auf dem Bildschirm dargestellt ist.

Um ein Kommando mehr als einmal zu wiederholen, wird die Punkt (.)-Taste betaetigt.

3.5. Der Aufhebungs-Operator (undo)

Vi verfuegt ueber einen Aufhebungsoperator

u

der die letzte Veraenderung rueckgaengig macht. Das Aufhebungskommando kann das vorhergehende Aufhebungskommando aufheben - d.h. das erste Aufhebungskommando kann den Text in seinen Originalzustand zurueckfuehren und das zweite Kommando kann die Veraenderung wieder einfuegen. Dabei koennen mehrere Zeilen einbezogen sein. Das Aufhebungskommando macht nur eine Veraenderung rueckgaengig. Man kann

jedoch, nachdem man mehrere Veraenderungen vorgenommen hat, die Zeile auf ihren Originalzustand mit dem Kommando

U

zurueckfuehren.

3.6. Editieren von Programmen

Der Editor verfuegt ueber eine Reihe von Kommandos zum Editieren von Programmen. Eines der praktischsten ist die Option zum automatischen Einruecken, wodurch korrekt eingerueckte Programme erzeugt werden koennen. Ein weiteres betrifft die Option zum Verschieben der Spalten, das fuer das Neusetzen des Wertes des Ruecksetztabulators angewandt wird. Beide Kommandos werden in Abschnitt 6.5. beschrieben.

Die Operatoren "<" und ">" werden benutzt, um einzelne Zeilen entweder nach links oder nach rechts um eine Verschiebebreite zu verschieben. Um eine Zeile zu verschieben, werden die Doppel-Operatoren wie folgt benutzt: [<n>]<< verschiebt die Zeile um eine Verschiebebreite nach links und [<n>]>> verschiebt die Zeile um eine Verschiebebreite nach rechts.

Dabei wird mit n die Anzahl der Zeilen festgelegt; der Standardwert ist eine Zeile.

Es ist ebenfalls moeglich, alle Zeilen vom Cursor an bis zur letzten Zeile auf dem Bildschirm entweder nach links oder nach rechts zu verschieben. Dafuer wird das Kommando

<L

oder

>L

benutzt. Mit einem weiteren Zusatzkommando kann man die Uebereinstimmung des Oeffnens und Schliessens von Klammern fuer komplizierte Ausdruecke herbeifuehren. Um die passende Klammer zu finden, wird der Cursor entweder an eine sich oeffnende oder eine sich schliessende Klammer plaziert und die Prozenttaste (%) betaetigt. Das ist sowohl fuer geschweifte ({}) und eckige Klammern ([]) anwendbar.

Fuer das Editieren von C-Programmen bewirken doppelte eckige Klammern ([[]]) ein Vor- oder Zurueckspringen auf eine Zeile, die mit einer geschweiften Klammer ({) beginnt. Werden sich schliessende doppelte eckige Klammern (]]) mit einem Operator benutzt, dann wird nach einer Zeile angehalten, die mit einer geschweiften Klammer ({) beginnt. Das ist manchmal in Verbindung mit dem Kommando "y" wie folgt gebraeuchlich:

```
y]]
```

wobei der y-Operator eine Zeile herauszieht und diese in einem Zwischenspeicher ablegt.

3.7. Loeschen von Zeichen und Zeilen

Der gebraeuchlichste Weg, den eingegebenen Text zu korrigieren, besteht darin, CTRL-h zu betaetigen, um ein falsches Zeichen zu loeschen oder CTRL-w, um ein falsches Wort zu loeschen. Wenn das System als Loeschzeichen das Doppelkreuz (#) benutzt, so wirkt es wie CTRL-h in vi.

Das Zeichen zum Zeilenloeschen ist normalerweise eines der folgenden:

```
`
CTRL-x
CTRL-u
```

Damit wird die gesamte Eingabe auf der aktuellen Zeile geloescht. Im allgemeinen wird durch das Loeschzeichen nicht mehr als die zurueckliegende Zeile geloescht, und es werden auch keine Zeichen geloescht, die nicht mit dem aktuellen Kommando des Textmode eingefuegt worden waren. Um Korrekturen auf der vorangegangenen Zeile vorzunehmen - nachdem eine neue Zeile begonnen wurde - wird folgendermassen vorgegangen:

1. ESC betaetigen, um den Eingabemode zu beenden.

2. Den Cursor in die entsprechende Position bringen, um die Korrektur vorzunehmen.

3. Zurueckkehren und den Eingabemode fortsetzen. Fuer die Fortsetzung erweist sich oft der Operator "A" zum Anfuegen an die aktuelle Zeile als guenstig.

4. Neuanordnen und duplizieren des Textes

4.1. Allgemeines

Nach Definition wird ein Satz mit einem Punkt (.), einem Ausrufezeichen (!) oder einem Fragezeichen (?) beendet und wird entweder von einem "end of a line" (Zeilenende) oder zwei Leerzeichen gefolgt. Nach den abschliessenden Satzzeichen, aber vor den Leerraeumen oder new line kann jede Anzahl von sich schliessenden runden oder eckigen Klammern oder Anfuehrungszeichen erscheinen.

Die Operatoren "(" und ")" bringen den Cursor an den Anfang bzw. das Ende des vorhergehenden bzw. naechsten Satzes. Aehnlich wirken die Operatoren { und } und [[und]], die eine Bewegung ueber Absaetze bzw. Kapitel ermoeglichen. Die Operatoren, die aus der eckigen Klammer bestehen, erfordern eine Doppeleingabe, da sie den Cursor ueber eine beachtliche Distanz bewegen koennen. Obwohl es einfach ist mit einfachen Anfuehrungsstrichen '' zurueckzukehren, koennen diese Kommandos zu Verwirrungen fuehren, wenn sie zufaellig ausgefuehrt werden.

Nach Definition beginnt ein Absatz nach jeder leeren Zeile und ebenfalls bei jedem Satz von Paragraphen-Macros. (Siehe NROFF und TROFF-Beschreibung) Die Paragraphen-Macros koennen veraendert oder erweitert werden, indem eine andere Zeichenkette der Paragraphen-Option in EXINIT zugewiesen wird. Die Satz- und Paragraphen-Kommandos koennen mit Zahlen (counts) versehen werden, um ueber Satz- und Absatz-Gruppen hinweg zu wirken. Im Editor beginnen Kapitel (sections) nach jedem macro in der sections-Option. Kapitelgrenzen sind immer Zeilen- und Absatz-Grenzen.

Es ist moeglich, einen laengeren Text durchzusehen, indem Kapitel (section) Kommandos benutzt werden. Es ist ebenfalls moeglich, mit jedem dieser Section- und Paragraphen-Kommandos ein vorangestelltes count zu benutzen. Die Section-Kommandos interpretieren ein vorangestellten count als eine andere Fenstergroesse, in der die Bildschirmdarstellung auf dem neuen Platz wiederzugeben ist. Diese Fenstergroesse ist die Basisgroesse fuer neu zu zeichnende Fenster, bis eine andere Groesse angegeben wird. Das erweist sich als guenstig, wenn man auf einem langsamen Terminal nach einem bestimmten Kapitel sucht. Es ist moeglich, dem ersten Section-Kommando einen kleinen count zu geben und dann jede nachfolgende Kapiteluebeschrift in einem kleinen Fenster zu betrachten.

4.2. Pufferspeicher

Vi verfuegt ueber folgende Pufferspeicher:

1. Einen einzelnen, unbenannten Puffer, indem der zuletzt geloeschte oder veraenderte Text gespeichert ist

2. Einen Satz von benannten Puffern - a bis z - der dafuer benutzt wird, Text innerhalb einer Datei zu speichern oder zu verschieben oder zwischen den Dateien auszutauschen.

Die Puffer werden von den Operatoren "yank" und "put" benutzt, die in Abschnitt 4.3. beschrieben sind.

4.3. Textmanipulationen

Der Operator "yank" wird benutzt, um Text in den nicht bezeichneten Pufferspeicher oder in irgendeinen bezeichneten Puffer abzulegen. Die Kommandoschreibweise lautet

"[<buffer>][<n>]yw

dabei sind:

" bedeuten das das folgende Zeichen ein Puffer und kein Kommando ist.

<buffer> einen Puffernamen a-z; der Standardwert ist der unbenannte Puffer.

<n> die Anzahl der herauszuziehenden Woerter; Standardwert ist ein Wort.

y der yank-Operator.

w der word-Operator.

Dieses Kommando loescht den herausgezogenen Text nicht. Satzzeichen werden als Woerter gezaehlt. Um ein vollstaendiges Wort herauszuziehen, muss der Cursor unter die ersten Buchstaben des Wortes gebracht werden. Befindet sich der Cursor nicht am Wortanfang, dann werden alle Zeichen von der Cursorposition an bis zum naechsten Leerzeichen (am Ende des Wortes) herausgezogen.

Der Operator "yy" ist aequivalent "Y"; das Kommando

"[<buffer>][<n>]Y

zieht eine gesamte Zeile, auf der sich der Cursor befindet heraus und plaziert sie in einen Puffer, wie oben beschrieben. Das count <n>, das dem Y-Operator vorangestellt ist, zieht n-Textzeilen heraus. Der Standardwert ist eine Zeile.

Beispiele:

Das Kommando

```
yw
```

zieht das Wort, bei dem sich der Cursor befindet, heraus. Das Kommando

```
4yw
```

zieht das Wort, bei dem der Cursor steht und die folgenden 3 Woerter heraus, und gibt sie in den nicht bezeichneten Puffer. Das Kommando

```
"a12yw
```

laedt 12 Woerter in den Zwischenspeicher a.

Ein gewoehnliches Loeschkommando speichert den Text im unbenannten Puffer, so dass ein gewoehnliches put-Kommando (p oder P, siehe weiter unten) es woanders hin schicken kann. Der Inhalt des nicht bezeichneten Zwischenspeichers geht jedoch verloren, wenn Dateien veraendert werden; um Text von einer Datei in eine andere zu geben, ist es deshalb wichtig, einen bezeichneten Zwischenspeicher zu benutzen.

Text, der herausgenommen wurde und zwischengespeichert wird, kann mit den Operatoren p oder P in den Text wieder eingefuegt (put) werden, wobei die Kommandoschreibweise lautet:

```
"[<buffer>]p
```

dabei bezeichnen die "<buffer> den Namen des Zwischenspeichers, indem der herausgezogene Text gespeichert war. Der Operator "p" fuegt den herausgezogenen Text nach oder unter dem Cursor wieder ein und der Operator "P" fuegt den Text vor oder ueber dem Cursor ein. Die Kommandobeschreibung ist fuer beide P- und p-Operatoren identisch. Wenn kein Puffer angegeben wird, ist der Standardwert der unbezeichnete Puffer.

Der Text, der herausgezogen und zwischengespeichert wird, kann Teil einer Zeile sein oder ein Satz, der mehr als eine Zeile umfasst. Wenn der Text wieder eingesetzt wird, wird er in Abhaengigkeit vom Kommando entweder vor oder hinter den Cursor gesetzt. Besteht der Text aus vollstaendigen Zeilen, dann wird er in vollstaendigen Zeilen ohne Veraenderung der aktuellen Zeile wieder eingesetzt.

Das Kommando

```
[<n>]YP
```

zieht eine Kopie von n Zeilen heraus und fuegt dann sofort den gleichen Text vor der aktuellen Zeile ein. Dadurch

erhaelt man zwei identische Textzeilen und der Cursor begibt sich in die obere Zeile. Aehnlich verhaelt sich das Kommando

[<n>]Yp

hierbei werden aber n Zeilen kopiert und diese nach (unter) die aktuelle Zeile plaziert, so dass zwei identische Zeilen entstehen. Z.B. wiederholt das Kommando 3YP die Textzeile dreimal. Der Standardwert ist eine Textzeile.

Das yank-Kommando wie auch die Kommandos zum Loeschen und Veraendern, koennen im Zusammenhang mit einer Zeichenkettensuche gebraucht werden. Das Kommando

y/<string>/-<n>

zieht die Zeichen ab Cursorposition bis zur n-ten Zeile, die der Zeichenkettenuebereinstimmung vorausgeht, heraus. Aehnlich wirkt das Kommando

y/<string>/+<n>

Es zieht alle Zeichen heraus und speichert diese zwischen, beginnend von der Cursorposition bis zur n-ten Zeile, die dem der Zeichenkette folgt.

Die gleichen Zwischenspeicher koennen mit den Operatoren zum Loeschen benutzt werden, um Textbloecke innerhalb der Datei oder zu einer anderen Datei zu verschieben. Das Verschieben eines Textblockes erfordert drei Operationen:

1. Loeschen und speichern von n Zeilen.

2. Cursor an die neue Position bringen.

3. Den Text anbringen ("put").

Beispiel:

Es werden 5 Textzeilen geloescht und zeitweilig in Zwischenspeicher a gespeichert:

"a5dd

Die Anfuehrungszeichen bezeichnen einen Puffernamen und nicht das Kommando "a". Als naechstes wird der Cursor an die neue Textstelle gebracht und das Kommando

"ap

oder das Kommando

"aP

eingegeben, um den Text an der neuen Stelle einzufuegen.

Um zu einer anderen Datei zum Zwecke des Editierens umzuschalten, bevor der zwischengespeicherte Text eingefuegt wird, ist ein Kommando der Form

:e <filename>

zu benutzen, wobei <filename> die andere zu editierende Datei darstellt. (Diese Kommandos werden in einem spaeteren Abschnitt beschrieben.)

MERKE

Wenn der Inhalt des aktuellen Editor-Zwischenspeichers veraendert worden ist, so muss er entweder zurueckgeschrieben oder verworfen werden, bevor zur anderen Datei umgeschaltet wird.

5. Dateimanipulationen

5.1. Schreiben, Abbrechen und Editieren von neuen Dateien

Die grundlegenden Schreibe- und Abbruch-Kommandos sind in Abschnitt 1.7. beschrieben.

Wenn der Text veraendert worden ist, aber die Veraenderungen nicht in die Datei geschrieben werden sollen, so verwirft das Abbruch-(quit)Kommando (:q!) die Veraenderungen. Um die gleiche Datei von neuem zu editieren (noch einmal zu beginnen), wird das Kommando:

```
:e!
```

eingegeben. Dieses Kommando wird selten benutzt, da die Veraenderungen nicht durchgefuehrt werden koennen, nachdem sie verworfen wurden.

Um eine andere Datei zu editieren, ohne das der vi-Editor verlassen wird, wird

```
:e <filename>
```

eingegeben. Wenn die Veraenderungen nicht in die Datei geschrieben worden sind (vor diesem Kommando), gibt vi die Nachricht

```
No write since last change (:edit! overrides)
```

aus und verzoegert das Editieren der anderen Datei. Die Antwort darauf wird mit dem Kommando

```
:w
```

eingegeben, um die Veraenderungen in der ersten Datei zu speichern. Nachdem die Veraenderungen geschrieben sind, wird das Kommando ":e <filename>" wiederholt oder das Kommando

```
:e!
```

benutzt, um die Veraenderungen in der ersten Datei zu verwerfen und die zweite Datei aufzurufen. Um die Veraenderungen automatisch abzuspeichern, ist die autowrite-Option (automatisches Schreiben) zu setzen. Wenn autowrite gesetzt ist, wird das Kommando

```
:n
```

anstelle von

```
:e
```

benutzt.

5.2. Kommandos zur Dateimanipulation

Die Tabelle 5-1 enthaelt die vi-Kommandos zur Dateimanipulation. Diesen Kommandos folgt ein Wagenruecklauf (RETURN) oder eine Umschaltung (ESC). Die meisten Kommandos bedueerfen keiner Erklaerung. Im folgenden wird beschrieben, wie diese Kommandos benutzt werden.

Tabelle 5-1 Kommandos zur Dateimanipulation

Kommando	Funktion
:w	Veraenderungen in die Datei zurueckschreiben
:wq	Veraenderungen zurueckschreiben und Arbeit abbrechen
:x	wenn notwendig, schreiben und abbrechen
:e<name>	Editieren der Datei <name>
:e!	Veraenderungen verwerfen und neu editieren
:e+<name>	Datei <name> editieren, am Ende beginnen
:e+<n><name>	Datei <name> beginnend mit Zeile n oder mit dem Kommando n editieren
:e#	Editieren einer alternativen Datei, die durch den zuletzt eingegebenen Dateinamen bestimmt wird, bevor der aktuelle Dateiname eingegeben wurde
:e%	aktuelle Datei editieren
:w <name>	Datei <name> schreiben
:w! <name>	Datei <name> ueberschreiben
:<x>,<y>w <name>	Zeilen <x> bis <y> nach <name> schreiben
:r <name>	Datei <name> in den Zwischenspeicher einlesen
:r!,cmd>	Ausgabe von <cmd> in den Zwischenspeicher lesen
:n	naechste Datei in der Argumentenliste editieren
:n!	Veraenderungen zur aktuellen Datei verwerfen und die naechste Datei editieren
:n <arglist>	neue Argumentenliste <arglist> spezifizieren
:ta <tag>	die Datei, die das Identifizierungskennzeichen <tag> enthaelt, bei <tag> editieren

Das grundsaetzliche Schreibkommando lautet

:w

das Veraenderungen in die Datei schreibt. Wenn das Editieren einer einzelnen Datei beendet ist, sind die Veraenderungen zurueckzuschreiben und vi ist mit dem Kommando

ZZ

zu verlassen.

Fuer das Editieren langer Texte ist es guenstig, die Veraenderungen desoefteren mit dem Kommando ":w" zurueckzuschreiben und mit dem Kommando "ZZ" zu beenden.

Wenn mehr als eine Datei editiert wird, sind die Veraenderungen mit dem Kommando ":w" zurueckzuschreiben und das Editieren einer neuen Datei mit dem Kommando ":e" zu beginnen. Eine weitere Moeglichkeit besteht darin, die autowrite-Option zu setzen (siehe Abschnitt 6) und das Kommando

:n <file>

zu benutzen, um die naechste Datei zum Editieren aufzurufen.

Wenn die Veraenderungen zur aktuellen Datei nicht zurueckgeschrieben wurden, ist dieses Kommando funktionsunfaehig.

Wenn Veraenderungen an der Editorkopie einer Datei vorgenommen wurden, diese aber nicht zurueckgeschrieben werden sollen, dann wird das Ausrufezeichen (!) dem Kommando hinzugefuegt. Dadurch verwirft der Editor alle vorgenommenen Veraenderungen. Mit diesem Kommando sollte sehr aufmerksam umgegangen werden.

Den verschiedenen ":e"-Kommandos koennen Argumente beigegeben werden. Mit dem Argument "+" wird das Editieren am Ende der Datei begonnen und mit

+<n>

beginnt der Editor auf Zeile n. Darueberhinaus kann n auch jedes Editor-Kommando ohne Leerzeichen sein, wie z.B. ein Durchsuchen

+/<string>

oder

+?<string>

wobei der Editor nach <string> sucht.

Andere Argumente fuer ":e" beinhalten das Zeichen "%", das, wenn es in einem Kommando benutzt wird, als der aktuelle

Dateiname interpretiert wird. Ein weiteres Argument ist "#", das als alternativer Dateiname angesehen wird, wobei der alternative Dateiname der zuletzt eingegebene, anders als der aktuell lautende, ist. Z.B. angenommen es wurde das Kommando

:e

eingegeben und es erscheint eine Meldung, dass die Datei nicht geschrieben wurde. Dann besteht eine Moeglichkeit darin, das Kommando

:w

einzugeben, das die Datei schreibt und danach das Kommando

:e#

folgen zu lassen, um das vorherige ":e" zu wiederholen. Das Kommando

CTRL-^

fuehrt die gleiche Funktion aus.

Um einen Teil eines Zwischenspeichers in eine Datei zu schreiben, sind zuerst die Zeilennummern zu bestimmen, die den zu schreibenden Abschnitt umschliessen. Fuer die Anzeige der Zeilennummer, auf der der Cursor steht, wird das Kommando

CTRL-g

genommen oder die Option number gesetzt. Danach wird das Kommando

: <x>,<y>w <name>

eingegeben, wobei:

<x>,<y>die obere und untere Zeilennummer angeben

<name> der Dateiname der Zieldatei

Wenn die Zieldatei nicht existiert, wird sie erstellt. Andernfalls gibt vi die Diagnosemeldung

"<name>" File exists - use "w! <name>" to overwrite command

Man kann anstelle der Zeilennummer die Adressenmarkierungen im Kommando benutzen. Wenn z.B. das Kommando

ma

die erste Zeile in Register a und

```
mb
```

die letzte Zeile in Register b markiert schreibt das Kommando

```
:'a,'bw! <name>
```

diese Zeilen in die Datei <name>.

Es ist moeglich, eine weitere Datei nach der aktuellen Zeile in den Pufferspeicher einzulesen. Dafuer nimmt man das Kommando

```
:r <name>
```

Um einen Satz von aufeinanderfolgenden Dateien zu editieren, sind zuerst alle Dateinamen als Argumente im Kommando

```
:n <name1> <name2> .... <namex>
```

einzugeben und dann jede Datei der Reihe nach unter Benutzung des Kommandos

```
:n
```

zu editieren.

Das Kommando

```
:ta
```

ist fuer das Editieren von grossen Programmen guenstig. Es nutzt eine Datenbank aus Funktionsnamen und deren Adressen (die durch das Programm ctags(1) erstellt werden kann). Bevor ein ":ta"-Kommando aufgerufen wird,um von einer Datei auf die andere zu wechseln, muss die gesamte aktuelle Arbeit in eine Datei geschrieben oder sie wird verworfen.

Um eine Shell-Kommando auszufuehren, wird ein Ausrufezeichen in Zusammenahng mit einem Shell-Kommando <cmd> benutzt:

```
:! <cmd>
```

6. Optionen

6.1. Allgemeines

Wie oben bereits bemerkt, sind die Optionen aus dem Editor ex ebenfalls fuer vi verfuegbar und leicht anzuwenden. Die gebraeuchlichsten sind in Tabelle 6-1 aufgelistet.

Tabelle 6-1 Haeufig gebrauchte Optionen

Option	Standardwert	Funktion
autoindent	noai	automatisches Einruecken
autowrite	noaw	automatisch Schreiben vor :n, :ta,CTRL-^ und !
ignorecase	noic	ignorieren der Gross-/Klein-schreibweise beim Suchen
lisp	nolisp	Kommandos behandeln S-Aus-druecke
list	nolist	Tabs werden mit CTRL-i ausge-geben; Zeilenenden werden mit $ markiert
magic	magic	die Zeichen . [und * sind spezielle Zeichen fuer Durch-suchen (scans)
number	nonu	Zeilen werden mit vorangestell-ten Zeilennummer abgebildet
paragraphs	para= IPLPPPQPP LIbp	Macro-Namen, die einen Para-graphen (Abschnitt) beginnen
redraw	nore	ein "smart"-Terminal auf einem nichtintelligentem Terminal simulieren
scroll	1/2	Anzahl der gerollten Zeilen
sections	sect=NHSHH HU	Macro-Namen, die einen neuen Abschnitt (Section) beginnen
shiftwidth	sw=8	um eine bestimmte Distanz verschieben fuer <, >, Ein-gabe von CTRL-d und CTRL-t
shwomatch	nosm	die passende Klammer (oder { zeigen, wenn) oder } einge-geben werden
slowopen	noslow	die Bildschirmaktualisierung wird waehrend der Einfuegungen zeitlich verschoben
term	vtz	benutzter Terminaltyp
terse	noterse	kuerzere Fehlerdiagnose
window	speed dependent	Anzahl der Zeilen im Bild-schirmfenster
wrapmargin	wm=0	rechten Rand von rechts her-einbringen
wrapscan	ws	bei end of line von vorn be-ginnen

Im allgemeinen gibt es drei Arten von Optionen: numerische Optionen, String-Optionen und Toggle-Optionen. Numerische und String-Optionen werden durch Kommandos folgender Form

gesetzt:

set <opname>=<val>

dabei sind:

<opname> der Name der Option

<val> der entsprechende String- oder numerische Wert fuer die Option

Toggle-Optionen koennen mit folgenden Kommandos gesetzt bzw. rueckgesetzt werden:

set <opname>
set <noopname>

Diese Optionen koennen waehrend der Arbeit in vi eingegeben werden, indem dem Setz-Kommando ein Doppelpunkt vorangestellt wird. Das Kommando lautet dann wie folgt:

:se <opname>=<value>

oder

:se <opname>

Um auf dem Bildschirm eine Liste jener Optionen auszugeben, die gesetzt worden sind, ist das Setz-Kommando ohne Optionsnamen wie folgt einzugeben:

:set

Um den Wert einer einzelnen Option anzuzeigen, wird folgendes Kommando eingegeben:

:set <opname>?

Um eine Liste aller moeglichen Optionen und ihren aktuellen Werten zu erhalten, gibt man das Kommando ein:

:set all

Es ist zu beachten, dass o.g. Kommandos auch abgekuerzt werden koennen und das mehrfache Optionen nur mit einem Optionskommando gesetzt werden koennen:

:se ai as nu

Die Optionen, die waehrend einer Editier-Sitzung gesetzt wurden, sind nur solange wirksam, bis der Editor verlassen wird. Es kann jedoch guenstig sein, eine Liste von Optionen zu haben, die immer, wenn der Editor benutzt wird, gesetzt werden. Das kann erreicht werden, indem eine Liste von ex-Kommandos geschaffen wird - d.h. Kommandos, die vom Texteditor ex genutzt werden - die immer dann genommen werden

sollen, wenn die Programme ex, edit oder vi aufgerufen werden. (Es ist zu beachten, dass nur Kommandos, die mit einem Doppelpunkt beginnen ex-Kommandos sind.) Es erweist sich als guenstig, wenn man diese Kommandos auf einer einzelnen Zeile auflistet.

Es ist moeglich, jede Anzahl von Optionskommandos in die Umgebungsvariable EXINIT einzubauen, wenn Optionen in der Umgebung gesetzt werden, dann werden sie automatisch bei jedem Aufruf von vi gesetzt. Z.B., um autoindent, autowrite und terse zu setzen, wuerde das Kommando lauten (unter Benutzng von csh):

```
setenv EXINIT 'set ai aw terse'
```

6.2. Das Editieren an langsamen Terminals

Der Textmode eines langsamen Terminals wird durch die slowopen-Option gesteuert. Diese Option wird durch das Kommando

```
:se slow
```

gesetzt. In langsamen Systemen begrenzt diese Option die Ausgabe an das Terminal. Es ist ebenfalls moeglich, den Editor zu veranlassen, diese Option auch auf schnelleren Terminals zu benutzen, indem man diese Option eingibt. Um die slowopen-Option auszuschalten, wird das Kommando

```
:se noslow
```

benutzt. Weiterhin ist es moeglich, ein intelligentes Terminal mit der redraw-Option zu simulieren. Diese Simulation erzeugt eine grosse Menge an Ausgabedaten und ist im allgemeinen nur auf leicht belasteten Systemen und schnellen Terminals zu tolerieren. Diese Option wird mit dem Kommando

```
:se redraw
```

gesetzt und mit dem Kommando

```
:se noredraw
```

wieder geloescht.

6.3. Unterscheidung von Gross-/Kleinbuchstaben

Der Editor ist in der Lage, wenn es ihm befohlen wird, die Gross- bzw. Kleinschreibweise von Woertern im Such-String zu ignorieren. Das entsprechende Kommando lautet:

```
:se ic
```

Um die ignore case option auszuschalten, gibt es das Kommando

```
:se noic
```

6.4. Sonderzeichen (magic)

Zeichenketten, die in einer Zeichenkettensuche vorkommen, koennen Zeichen enthalten, die "magic"-Bedeutungen fuer vi haben, enthalten. Wenn diese Funktion nicht gewuenscht wird, dann wird die magic-Option mit dem Kommando

```
:se nomagic
```

zurueckgesetzt. Bei nomagic sind nur die Zeichen "^" und "$" spezielle Zeichen in Strukturen. Das Zeichen "\" stellt ebenfalls ein Spezialzeichen dar (wie fast ueberall im System) und kann fuer die Funktion der erweiterten Strukturuebereinstimmung genutzt werden.

Sowohl fuer magic als auch fuer nomagic ist es notwendig, einen "\" (Backslash) vor einem "/" in einer vorwaerts gerichteten Zeichenkettensuche oder einem "?" in einer rueckwaerts gerichteten Zeichenkettensuche zu benutzen. Das heisst, wenn die Zeichenkettensuche entweder fuer ein "/" (vorwaerts) oder eine "?" (rueckwaerts) gilt, dann muss dem Zeichen ein Backslash vorangestellt werden. In Tabelle 6-2 sind die erweiterten Formen aufgelistet, die benutzt werden, wenn die magic-Option gesetzt wird.

Tabelle 6-2 Erweiterte Operatoren fuer die magic-Option

Operator	Funktion
^	Am Anfang einer Struktur (pattern), bringt den Anfang einer Zeile in Uebereinstimmung
$	Am Ende einer Struktur, bringt das Ende einer Zeile in Uebereinstimmung
.	Bringt jedes Zeichen in Uebereinstimmung
<	Bringt den Anfang eines Wortes in Uebereinstimmung
>	Bringt das Ende eines Wortes in Uebereinstimmung
[str]	Bringt jedes einzelne Zeichen in einem String str in Uebereinstimmung
[^str]	Bringt jedes einzelne Zeichen, das nicht im String str ist, in Uebereinstimmung
[x-y]	Bringt jedes Zeichen zwischen x und y in Uebereinstimmung, wobei x und y alphanumerische Zeichen sind
*	Bringt jegliche Anzahl von vorhergehenden Strukturen in Uebereinstimmung

Es ist zu beachten, dass im nomagic-Mode die Zeichen

@ [and *

mit einem vorangestellten "\" benutzt werden.

6.5. Autoindent und Shiftwidth (Automatiaches Einruecken und Verschiebebreite)

Die Option zum automatischen Einruecken (autoindent option) erweist sich fuer die Erstellung von korrekt eingerueckten Programmen als guenstig. Um die autoindent-Option zu setzen, benutzt man das Kommando

:se ai

Zur Demonstration der Wirkungsweise dieser Option, ist eine neue Zeile mit dem Buchstaben "o" zu eroeffnen, einige Tabs und einige Zeichen einzugeben und dann eine weitere Zeile zu beginnen. Der Editor liefert ein Leerzeichen am Beginn der neuen Zeile, so dass sie mit der vorhergehenden Textzeile in Uebereinstimmung gebracht wird. Es ist zu beachten, dass es nicht moeglich ist, ueber die automatisch eingerueckte Stelle zurueckzuschreiten.

Bei der Anwendung der autoindent-Option, ist es teilweise angebracht, an den Rand zurueckzukehren - z.B., um eine Marke an den Rand anzubringen. Um die autoindent-Option in diesem Falle zu uebergehen wird das Kommando

CTRL-d

genommen, wodurch ueber die automatische Einrueckung zurueckgesprungen werden kann. Jedesmal, wenn dieses Kommando eingegeben wird, springt der Cursor eine Verschiebebreite zurueck. Wenn die Verschiebebreite auf 8 gesetzt ist, springt der Cursor 8 Stellen zurueck. Dabei ist zu beachten, dass dies nur sofort nach der automatischen Einrueckung wirkt.

Um alle Einrueckungen einschliesslich der naechsten Zeile zu stoppen, gibt man ein:

0CTRL-d

Eine einfache Art und Weise, am linken Rand eine Markierung zu setzen, besteht darin, (^) und dann CTRL-d einzugeben. Der Editor bringt den Cursor an den linken Rand der Zeile und erzeugt auf der naechsten Zeile ein neues Einruecken.

Die linke Begrenzung besteht normalerweise aus 8 Stellen. Um diese Begrenzung zurueckzusetzen, ist die shiftwidth-Option zu benutzen, die durch das Kommando

:se sw=<n>

eingegeben wird, wobei <n> die Anzahl der Stellen ist, die

die Breite der Begrenzung setzt.

6.6. Kontinuierliche Texteingabe

Werden lange Texte eingegeben, ist es oftmals guenstig, dass die Zeilen in der Naehe des rechten Randes automatisch umgebrochen werden. Um den Text n Stellen vom rechten Rand umbrechen zu lassen, benutzt man das Kommando

```
:se wm=<n>
```

Wenn der Editor eine eingegebene Zeile umbricht, kann sie wieder mit dem Kommando

```
[<n>]J
```

zusammengefuegt werden, wobei n die Anzahl der Zeilen ist, die wieder zusammengefuegt werden sollen. Standardmaessig wird die folgende Zeile an das Ende der aktuellen Zeile angefuegt. Der Editor liefert eine Leerstelle an der Verbindungsstelle der zusammengefuegten Zeilen und plaziert den Cursor auf diese Leerstelle. Um diese Leerstelle zu loeschen, gibt man das Kommando "x" ein.

6.7. Editieroptionen und -kommandos fuer LISP

Der vi-Editor verfuegt ueber einige Optionen zum Editieren von LISP-Programmen. Das Setzen der Lisp-Option erfolgt mit dem Kommando

```
:se lisp
```

Diese Option veraendert die Klammer-Kommandos "(" und ")", so dass sie rueckwaerts und vorwaerts ueber s-Ausdruecke springen koennen. Die Klammern "{" und "}" verhalten sich aehnlich wie die "(" und ")"-Kommandos, aber stoppen nicht an Teilen. Diese Kommandos koennen dafuer benutzt werden, um schnell durch einen Kommentar oder bis zur naechsten Liste durchzugehen.

Die autoindent Option unterscheidet sich fuer LISP von der normalen Funktion. Sie liefert eine Einrueckung, um die ersten Argumente der letzten offenen Liste in Uebereinstimmung zu bringen. Wenn es ein solches Argument nicht gibt, dann ist die Einrueckung zwei Stellen groesser als die letzte Ebene.

Die showmatch Option ist fuer das Schreiben in LISP geeignet. Vorausgesetzt, dass der Klammeranfang "(" auf dem Bildschirm erscheint, wenn das Klammerende ")" eingegeben wird, bewegt sich der Cursor kurzzeitig an die Position des Klammeranfangs. Um diese Option zu setzen, gibt man das Kommando

```
:se sm
```

ein. Der vi Editor nutzt auch den Operator

```
=
```

der existierende Zeilen so in Uebereinstimmung bringt, als ob sie mit den gesetzten lisp und autoindent Optionen eingegeben worden waeren. Z.B. bringt das Kommando

```
=%
```

am Anfang einer Funktion alle Zeilen der Funktionsvereinbarung in Uebereinstimmung.

Beim Editieren von LISP bewirken die doppelten Klammern "[[" und "]]", dass der Cursor vorwaerts oder rueckwaerts auf Zeilen bewegt wird, die mit einem Klammeranfang beginnen. Das ist fuer die Handhabung von Funktionsdefinitionen guenstig.

6.8. Zeilennummern

Falls gewuenscht, kann der Editor Zeilennummern vor jede Textzeile auf dem Bildschirm setzen. Dafuer lautet das Kommando

```
:se nu
```

Um die Option der Zeilennummerierung auszuschalten, gibt es das Kommando

```
:se nonu
```

6.9. Tabs und End of Line Indikatoren

Es ist moeglich, dass der Bildschirm Tabs als CTRL-i und Zeilenenden durch das Symbol "$" darstellt, wenn die list Option benutzt wird. Dafuer wird das Kommando

```
:se list
```

eingegeben. Diese Option kann mit dem Kommando

```
:se nolist
```

abgeschaltet werden.

6.10. Automatisches Schreiben von Dateien

Wenn eine Datei vor dem Wechsel zu einer neuen Datei nicht ausgeschrieben wurde, gibt vi die Meldung

"No write since last change (edit! overrides)"

aus. Damit der Editor die Veraenderungen automatisch speichert, ist die "autowrite" Option zu setzen

```
:se aw
```

Um Dateien zu aendern, nimmt man das Kommando

```
:n
```

anstelle von

```
:e
```

Um diese Optionen auszuschalten, wird das Kommando

```
:se noaw
```

eingegeben.

6.11. Definieren von Abschnitten (Paragraphen) und Kapiteln (Sections)

Es sind Editoroptionen verfuegbar, um einen Abschnitt oder/und ein Kapitel fuer NROFF Macros zu definieren (siehe Abschnitt 7 des P8000-Programmierhandbuchs). Ein Abschnitt beginnt normalerweise nach jeder Leerzeile. Diese Abschnittsgrenzen werden von den Operatoren "{" und "}" benutzt (siehe Abschnitt 2.4). Die "Paragraph" Option wird durch

```
set para=<macro name>
```

gesetzt, wobei <macro name> ein nroff macro(s) ist/sind, das/die den Anfang eines Abschnittes definiert/en. Kapitel koennen durch

```
set sections=<macro name>
```

neu definiert werden. Nach Definition beginnt ein Kapitel nach jeder Zeile mit einem Seitenvorschub CTRL-l in der ersten Spalte. Kapiteltgrenzen sind auch Zeilen- und Abschnittsgrenzen. Diese Begrenzungen werden von den Operatoren "[[" und "]]" benutzt (siehe Abschnitt 2.4.).

6.12. Terminaltyp

Der Terminaltyp wird von der Umgebung bestimmt, wenn

```
% setenv TERM <type>
```

ausgefuehrt wurde (siehe Abschnitt 1.4.). Diese Option

```
:se term
```

gibt den Terminaltyp aus.

6.13. Rollen

Die Laenge des Rollens bei Benutzung von CTRL-d, CTRL-u und "z" Kommandos kann durch

```
:se scroll=<val>
```

veraendert werden, wobei <val> die Rollgroesse (Anzahl der Zeilen) ist.

6.14. Terse

Die Fehlermeldungen koennen mit dem Kommando

```
:se terse
```

abgekuerzt und mit

```
:se noterse
```

wieder verlaengert werden. Das ist fuer den erfahrenen Nutzer guenstig.

6.15. Fenster

Die Anzahl der Zeilen in einem Textfenster kann mit dem Kommando

```
:se window=<val>
```

veraendert werden. Fuer langsame Terminals (600 baud oder weniger) betraegt die Fenstergroesse 8, fuer Terminals mit mittlerer Geschwindigkeit (1200 baud) ist die Groesse 16 und fuer Hochgeschwindigkeitsterminals - die gesamte Bildschirmgroesse minus 1.

6.16. Umschlagen um das Ende von Dateien

Die Suche nach Zeichenketten erfolgt normalerweise durch eine Datei hindurch und wird am Anfang wieder fortgesetzt. Diese Eigenschaft kann mit

```
:se nows
```

ausgeschaltet werden.

Anlage 1

Korrekturzeichen waehrend der Eingabe

Operator	Funktion
CTRL-H	Loeschen letztes Zeichen
CTRL-W	Loeschen letztes Wort
erase	Erase-Funktion, wie CTRL-H
kill	Kill-Funktion, Loeschen der Eingabezeile
\	Hebt folgende CTRL-H, erase bzw. kill-Funktion auf
ESC	Ende der Eingabe, Rueckkehr zum Kommandomodus
DEL	Interrupt, beendet Eingabe
CR	Carriage return, startet neue Eingabezeile
CTRL-D	Backtab ueber autoindent (automatisches-Einruecken)
^CTRL-D	loescht autoindent nur fuer eine Zeile
0CTRL-D	loescht alle autindent
CTRL-V	nichtdruckbare Zeichen koennen editiert werden, wenn CTRL-V vorangestellt wird

Anlage 2

Vi Symboluebersicht

Diese Anlage gibt einen zusammenfassenden Ueberblick ueber die Bedeutung aller Editorzeichen. Die Zeichen sind in der Reihenfolge der ASCII-Zeichen aufgefuehrt: als erstes CRTL-Zeichen, dann Sonderzeichen, dann Zahlen, dann Gross- und Kleinbuchstaben.

Die Beschreibung jedes Zeichens beinhaltet die Bedeutung des Zeichens als Kommando und die Bedeutung waehrend des Eingabemodus. Viele Kommandos koennen vorangestellte Zahlen haben, die die Anzahl der Wiederholungen beschreibt. Im folgendem ist diese Moeglichkeit nicht in jedem Fall extra angefuehrt.

CTRL-B Kommandomode: Fenster rueckwaerts. Eine Zahl gibt die Anzahl der Wiederholungen an.

CTRL-D Kommandomode: Rollt der Text ein halbes Fenster nach unten. Eine vorangestellte Zahl rollt den Text um n Zeilen nach unten. Die Zahl wird sich fuer folgende CTRL-D und CTRL-U Kommandos gemerkt.

Eingabemodus: Tabulatoren, die mit autoindent erzeugt wurden, werden geloescht.

CTRL-F Kommandomodus: Fenster vorwaerts. Eine Zahl gibt die Anzahl der Wiederholungen an.

CTRL-G Kommandomodus: Aequivalent zu :f ,es wird der Name der aktuellen Datei ausgegeben, ob diese veraendert wurde, die aktuelle Zeilennummer, die Zeilennummer der Datei und der aktuelle Standpunkt in der Datei in Prozent.

CTRL-H (BS) Kommandoebene: Cursur nach links.(siehe h)

Eingabemodus: Waehrend des Eingabemodus wird das letzte Zeichen geloescht. Der Cursor geht zurueck, das Zeichen bleibt aber auf dem Bildschirm sichtbar, bis es ueberschrieben wird bzw. der Eingabemodus verlassen wird.

CTRL-I (TAB) Eingabemodus: Tabulator

CTRL-J (LF) Kommandomodus: Cursor nach unten (siehe k).

CTRL-L Kommandoebene: Cursor nach rechts

CTRL-M Kommandoebene: Carriage return. Setzt Cursor auf erstes vom Leerzeichen verschiedenen Zeichen der naechsten Zeile. Eine vorangestellte Zahl setzt

den Cursor auf den Anfang der n-ten Zeile nach der aktuellen.

Eingabemodus: Beginn einer neuen Eingabezeile.

CTRL-N Kommandomodus: Cursor nach unten (siehe j)

CTRL-P Kommandomodus: Cursor nach oben (siehe k).

CTRL-Q Eingabemodus: Hebt die Bedeutung des folgenden Zeichens auf (fuer die Eingabe von ESC, CTRL-H usw.; siehe auch CTRL-V). Bei den meistem Terminals wird dieses Zeichen vom Terminal abgefangen, so dass es nicht moeglich ist dieses Zeichen einzugeben.

CTRL-R Kommandomode: Die gleiche Bedeutung wie r. Im open mode bei Hardcopy-Terminals erzeugt dieses Zeichen eine erneute Darstellung der aktuellen Zeile.

CTRL-S ungenutzt. Bei den meisten Terminals wird dieses Zeichen vom Terminal abgefangen, so dass es nicht moeglich ist, dieses Zeichen im Editor einzugeben.

CTRL-T Eingabemodus: Wenn autoindent gesetzt ist werden am Anfang der Zeile Leerzeichen (Anzahl wird mit shiftwidth eingestellt) eingefuegt

CTRL-U Kommandomodus: Bildschirm um ein halbes Fenster nach oben rollen. Eine vorangestellte Zahl rollt den Bildschirm um n Zeilen nach oben. Diese Zahl wird fuer die naechten CTRL-U und CTRL-D Kommandos gemerkt.

CTRL-V Eingabemodus: Hebt die Bedeutung des folgenden Zeichens auf. Dadurch ist es m|glich nichtdruckbare und andere Sonderzeichen einzugeben.

CTRL-W Eingabemodus: Loescht ein Wort. Das geloeschte Wort bleibt auf dem Bildschirm sichtbar und kann ueberschrieben werden.

CTRL-Z Kommandomodus: Erneuerung der Bildschirmanzeige

CTRL-[(ESC) Kommandomode: Abbruch des laufenden Kommandos

Eingabemode: Beenden der Eingabe. Rueckkehr in den Kommandomode.

CTRL-\ Kommandomode: Uebergang zu ex.

Eingabemode: Beenden der Eingabe. Uebergang zu ex.

CTRL-] Kommandomode: Aequivalent zu :ta

CTRL-^ Kommandomode: Aequivalent zu :e #

Space Kommandomode: Cursor nach rechts (siehe l)

! Kommandoebene: Zwei "!", gefolgt von einem Shellkommando, fuegen die Ausgabe hinter die aktuelle Zeile an. Wenn anstelle des ersten "!" ein Operator, der eine Anzahl von Zeilen spezifiziert, angegeben wird, so werden diese Zeilen als Standardeingabe fuer das folgende Shellkommando benutzt, und die Ausgabe wird an die Stelle der alten Zeilen gesetzt. Z.B. 2!}sort sortiert die naechsten beiden Paragraphen mit dem Programm sort.

$ Kommandomode: Cursor wird an das Ende der Zeile gesetzt. Wenn eine Zahl vorangestellt ist, wird der Cursor auf das Ende der folgenden n-ten Zeile gesetzt. Wenn die "list"-Option gesetzt ist, so zeigt $ das Ende einer Zeile an.

% Kommandoebene: Wenn der Cursur auf einer runden oder geschweiften Klammer steht, wird er auf die dazugehoerige Klammer gesetzt.

" Kommandoebene: Gefolgt von einem Buchstaben wird ein Puffer gekennzeichnet.

' Kommandoebene: Gefolgt von einem Buchstaben, wird der Cursor auf die mit dem Buchstaben markierte Stelle (siehe m) gesetzt.

(Kommandoebene: Setzt Cursor auf den Anfang des naechsten Satzes.

+ Kommandoebene: Setzt Cursor auf die naechste Zeile.

, Kommandoebene: Kehrt die letzten f F t oder T Kommandos um und sucht in umgekehrter Richtung.

- Kommandoebene: Setzt Cursor auf den Anfang der vorangegangenden Zeile.

. Kommandoebene: Wiederholt das letze Kommando, das Veraenderungen vorgenommen hat.

/ Kommandoebene: Zeichenkettensuche

0 Kommandoebene: Setzt Cursor an den Anfang der Zeile.

: Kommandoebene: Vorangestelltes Zeichen bei der Dateimanipulation, beim Setzen und Anzeigen von Optionen und zur Kommandoausfuehrung auf Systemebene.

< Kommandoebene: << verschiebt die Zeile um die Verschiebebreite (shiftwidth). Eine vorangestellte Zahl verschiebt die naechsten n-Zeilen.

> siehe <

? Kommandoebene: Zeichenkettensuche rueckwaerts

A Kommandoebene: Anfuegen am Ende der Zeile (aequivalent $a)

B Kommanoebene: Cursor um ein Wort zurueck

C Kommandoebene: Ersetzen des Restes der Zeile (aequivalent $c)

D Kommandoebene: Loeschen des Restes der Zeile (aequivalent $d)

E Kommandoebene: Cursor auf das Ende des Wortes

F Kommandoebene: Gefolgt von einem Zeichen, wird dieses Zeichen rueckwaerts in der aktuellen Zeile gesucht.

G Kommandoebene: Setzt Cursor auf n-te Zeile. Ist keine Nummer angegeben, so wird der Cursor auf die letzte Zeile gesetzt.

H Kommandomode: Setzt den Cursor an die erste Stelle des Bildschimes. Ist eine Zahl vorangestellt, so wird der Kursor auf die n-te Zeile des Bildschirmes gesetzt.

I Kommandomode: Eingabe am Anfang der Zeile. (aequivalent ^i)

J Kommandoebebe: Verbindet die aktuelle Zeile mit der folgenden.

L Kommandoebene: Setzt den Cursor an den Anfang der letzten Zeile des Bildschirms. Eine vorangestellte Zahl bedeutet die n-te Zeile ueber der Letzten Bildschirmzeile.

M Kommandomode: Setzt den Cursor auf die mittelste Zeile des Bildschirms.

N Kommandomode: Sucht die naechste Zeichenkette (vorher mit ? oder /) in umgekehrter Richtung.

O Kommandomode: Eroeffnet eine neue Zeile vor der aktuellen Zeile.

P Kommandomode: Einfuegen des letzten geloeschten Textes. Der geloeschte Text befindet sich in den Puffern 1-9.mit "x kann einer dieser Puffer ausgewaehlt werden und dann mit P eingefuegt werden. Die Puffer a-z sind fuer allgemeine Anwendungen verfuegbar.

Q Kommandoebene: Uebergang von vi zu ex

R Kommandoebene: Ersetzen von Zeichen.

S Kommandoebene: Austausch ganzer Zeilen (aequivalent cc)

T Kommandoebene: Gefolgt von einem Zeichen, wird dieses Zeichen in der aktulellen Zeile rueckwaerts gesucht. Der Cursor wird auf die Stelle vor dem gesuchten Zeichen gesetzt.

U Kommandoebene: Stellt den alten Zustand der aktuellen Zeile wieder her.

W Kommandomode: Setzt den Cursor auf den Anfang des naechsten Wortes.

X Kommandoebene: Loescht das Zeichen vorm Cursor.

Y Kommandoebene: Speichert eine Kopie der aktuelle Zeile im unbenannten Puffer ab. Von dort kann sie mit p bzw. P an anderer Stelle eingefuegt werden.

ZZ Kommandoebene: Verlassen des Editors. Wenn Veraenderungen vorgenommen wurden, wird der Puffer zurueckgeschrieben.

[[Kommandomode: Setzt den Cursor auf den Anfang des vorangegangenden Kapitels, der normalerweise mit ".NH" oder ".SH" beginnt. Aber auch an "{" am Anfang einer Zeile wird angehalten, um den Anfang einer Funktion in C-Programmen zu finden.

]] Kommandomode: Setzt den Cursor auf den Anfang des naechsten Kapitels.

^ Kommandomode: Cursor auf den Anfang der Zeile

a Kommandomode: Fuegt den folgenden Text an die aktuelle Position an. Mit ESC kann die Eingbe abgeschlossen werden. Eine vorangestellte Zahl

fuegt den folgenden Text n-mal an.

b Kommandomode: Setzt den Cursor rueckwaerts an den Beginn eines Wortes.

c Kommandomode: Ersetzt das folgende Objekt (z.B. w fuer Wort) gegen den nachfolgenden Eingabetext,der durch ESC beendet wird. Wenn mehr als eine Zeile Text ersetzt wird, dann wird der alte Text in einen der nummerierten Puffer aufbewahrt. Wenn nur auf der aktuellen Zeile Text ersetzt werden soll, dann wird das Ende des zu ersetzenden Textes durch ein $ gekennzeichnet. Eine vorangestellte Zahl bedeutet die Anzahl der Objekte (z.B. 3c) und c3) bedeuten die naechsten 3 Saetze)

d Kommandomode: Loescht das folgende Objekt (z.B. w fuer Wort). Wenn mehr als eine Zeile geloescht wird, wird der Text in einem nummerierten Puffer gespeichert.

e Kommandoebene: Setzt den Cursor an das Ende des naechsten Wortes.

f Kommndoebene: Gefolgt von einem Zeichen, wird der Cursor auf das naechste Auftreten dieses Zeichens gesetzt.

h Kommandoebene: Cursor nach links

i Kommandoebene: Texteingabe vor der aktuellen Cursorposition.

j Kommandoebene: Cursor nach unten

k Kommandoebene: Cursor nach oben

l Kommandoebene: Cursor nach rechts

m Kommandoebene: Markieren der aktuellen Cursorposition mit Angabe eines der Register a-z. Zur markierten Position kann man mit den Operatoren ' oder ` zurueckkehren.

n Kommandoebene: Wiederholung der letzten Zeichenketten-Such Operation.

o Kommandoebene: Eroeffnen einer neuen Eingabezeile nach der aktuellen Zeile.

r Kommandoebene: Ersetzen eines Zeichens

s Kommandoebene: Ersetzt ein einzelnes Zeichen

u Kommandoebene: Macht die letzte Aenderung rueckgaengig.

w Kommandoebene: Setzt den Cursur auf den Anfang des naechsten Wortes.

x Kommandoebene: Loescht ein Zeichen.

y Kommandoebene: Speichert das folgende Objekt in den unbenannten Puffer. Es ist auch moeglich einen Puffer (a-z) anzugeben (z.B. "xy). Der abgespeicherte Text kann mit p oder P an anderer Stelle eingefuegt werden.

z Kommandoebene: Das folgende Zeichen gibt an, an welcher Stelle auf dem Bildschirm die aktuelle Zeile stehen soll."-" am Ende; "." in der Mitte; RETURN am Anfang. Eine vorangestellte Zahl nimmt anstelle der aktuellen Zeile die n-te Zeile.

{ Kommandomode: Setzt den Cursor auf den Anfang des letzten Abschnitts.

| Kommandoebene: Setzt den Cursor auf das n-te Zeichen der aktuellen Zeile.

} Kommandoebene: Setzt den Cursor auf den Anfang des naechsten Abschnitts.

CTRL-? (DEL) Unterbricht den Editor und setzt in der Kommandoebene fort.

Anlage 3

Kommandozusammenfassung

- Dateimanipulation:

:w	Zurueckschreiben der Aenderungen
:wq	Zurueckschreiben und Verlassen des Editors
:x	Zurueckschreiben und Verlassen des Editors
:q	Verlassen des Editors
:q!	Verlassen, ohne die vorgenommenen Aenderungen abzuspeichern
:e Datei	Editieren der angegebenen Datei
:e!	Reeditieren; Datei wird in den Zustand versetzt, den sie nach dem letzten Schreiben hatte
:e#	Editieren der Datei, die beim Aufruf vor der aktuellen Datei angegeben wurde (auch CTRL-^)
:w Datei	Zurueckschreiben in angegebene Datei
:w! Datei	Zurueckschreiben in angegebene Datei, wenn diese bereits existiert (Ueberschreiben)
:!Kommando	Abarbeiten des angegebenen Kommandos
:n	Editieren der naechsten in der Kommandoliste angegebenen Datei
:f	Anzeige der aktuellen Datei und Zeilennummer (auch CTRL-g)
:sh	Aufruf von Shell (Eingabe von CTRL-d fuer Rueckkehr)
!!Kommando	Abarbeiten des Kommandos und Anhaengen der Ausgabe an die aktuelle Zeile

- Kursorpositionierung innerhalb der Datei:

CTRL-f	Fenster vorwaerts
CTRL-b	Fenster rueckwaerts
CRTL-d	halbe Fenster runter
CRTL-u	halbe Fenster hoch
[n]G	Kursor auf Zeile n (impl. letzte Zeile)
/string	naechste Zeile, die Zeichenkette enthaelt
?string	analog /string aber rueckwaerts im Text
n	Wiederholen des letzten /- oder ?-Kommandos
N	analog n, aber in entgegengesetzter Richtung
/string/+n	n-te Zeile nach Zeichenkette
?string?-n	n-te Zeile vor Zeichenkette
]]	naechste Sektion/Funktion (Zeile, die mit { beginnt)
[[	auf vorherige Sektionsgrenze
%	finde Klammer

- Markieren und Rueckkehr:

``	Rueckkehr zur vorherigen Position im Text
'	der vorherigen Position
mx	Markieren der Position mit Buchstaben x

```
          (beliebiger Buchstabe des Alphabets)
`x        Kursor auf mit x markierte Position
          Markierung enthaelt
```

- Zeilenpositionierung:

```
H         oberste Bildschirmzeile
M         auf Mitte des Bildschirms
L         letzte Bildschirmzeile
+         erste "non-white-Position" der naechsten Zeile
-         erste "non-white-Position" der vorherigen Zeile
RETURN    wie CR; Kursor auf Beginn naechste Zeile
j         naechste Zeile, gleiche Spalte
k         vorherige Zeile, gleiche Spalte
```

- Kursorpositionierung innerhalb einer Zeile:

```
^         erste "non-white-Position"
0         Zeilenbeginn
$         Zeilenende
l oder -> Kursor ein Zeichen nach rechts
h oder <- Kursor ein Zeichen nach links
CTRL-h    wie <-
space     wie ->
fx        Finde x vorwaerts
Fx        Finde x rueckwaerts
tx        auf Zeichen vor x (vorwaerts)
Tx        auf Zeichen nach x (rueckwaerts)
;         Wiederhole letzte f-,F-,t- oder T-Kommando
,         Invers von ;
[n]|      auf angegebene Spalte
```

- Woerter, Folgen, Paragraphen:

```
w         Wort vorwaerts   )
b         Wort rueckwaerts )  stoppen bei Punktuation
e         ans Wortende     )
)         an den Beginn der naechsten Folge
}         an den Beginn des naechsten Paragraphen
(         an den Beginn der vorherigen Folge
{         an den Beginn des vorherigen Paragraphen
W  )
B  )      analog w-, b-, e-Kommando ohne Stop bei
E  )      Punktuation
```

- Korrektur waehrend der Eingabe:

```
CRTl-H    Loeschen letzte Zeichen
CRTL-W    Loeschen letzte Wort
erase     Erase-Funktion, wie CRTL-H
kill      Kill-Funktion, Loeschen Eingabezeile
         vor CRTL-H verhindert es Loeschfunktion
```

ESC	Ende der Eingabe, Rueckkehr zum Kommandomodus
CRTL-?	Interrupt, beendet Eingabe
CRTL-D	Backtab, ueber autoindent
^CRTL-D	Loescht autoindent nur fuer eine Zeile
0CRTL-D	Loescht alle autoindent (autom. Einrueckung)
CRTL-V	nichtdruckbare Zeichen koennen editiert werden, wenn CRTL-V vorangestellt wird

- Eingabe und Ersetzen:

a	Anhaengen nach Kursor
i	Einfuegen vor Kursor
A	Anhaengen am Ende der Zeile
I	Einfuegen vor dem ersten "non-blank-Zeichen"
o	Eroeffnen Zeile (Eingabe von Textzeilen nach der aktuellen Zeile)
O	Eingabe von Textzeilen vor der aktuellen Zeile
rx	Ersetzen eines Zeichens durch das nach r angegebene Zeichen
Rstring	Ersetzen durch angegebene Zeichenkette

- Operatoren:

d	Streichen
c	Aendern von Woertern
<	Linksverschiebung einer Zeile um 1 Zeichen
>	Rechtsverschiebung einer Zeile um 1 Zeichen
!	Ausfuehren eines Kommandos vom Editor aus
=	Einruecken fuer LISP
y	Herausloesen einer Zeile und Speichern in einem Puffer

- Sonstige Operatoren:

C	Aendern des Restes der Zeile
D	Streichen bis Zeilenende
s	Ersetzen von Zeichen
S	Ersetzen von Zeilen
J	Verbinden von Zeilen
x	Streichen des Zeichens auf Kursorposition
X	Streichen des Zeichens vor Kursorposition
Y	Herausloesen von Zeilen

Herausloesen und Wiedereinbinden:

p	Zurueckspeichern von Zeilen nach der aktuellen Zeile (die durch y, Y herausgeloest wurden)
P	analog p, aber vor der aktuellen Zeile
xp	Zurueckspeichern aus dem durch x gekennzeichneten Puffer
"xd	Streichen in dem durch x gekennzeichneten Puffer
"xy	Herausloesen aus dem durch x gekennzeichneten Puffer

- Rueckgaengigmachen, Wiederholen, Wiederbekommen:

u	letzte Aenderung rueckgaengig machen
U	alle in der Zeile getaetigten Aenderungen rueckgaengig machen
"dp	Wiederholen der letzten Streichungen

- Aufruf und Verlassen von vi:

%vi Datei	Editiere die angegebene Datei, Kursor steht am Anfang der Datei
ZZ	Austritt aus vi, retten der durchgefuehrten Aenderungen

- Das Display:

letzte Zeile	enthaelt Fehlernachrichten, Eingaben zu den Operatoren :, /, ? und ! werden dort angezeigt, ebenso wie Meldungen ueber Ein-/Ausgaben und groessere Aenderungen
@Zeilen	nur auf dem Bildschirm, nicht in der Datei (bei einfachen Terminals)
~Zeilen	Zeilen ueber das Dateiende hinaus
CRTL-x	Control-Zeichen, CRTL-? entspricht delete
tabs	werden als Leerzeichen expandiert, der Kursor steht auf dem letzten Zeichen

- Beispiele fuer einfache Kommandos:

dw	Streichen eines Wortes
de	Streichen eines Wortes, Punkt. bleibt erhalten
dd	Streichen einer Zeile
3dd	Streichen von drei Zeilen
iTextESC	Einfuegen von Text
cwNeuESC	Aendern eines Wortes in angegebenes neues Wort

Das Makropaket

MM

Inhaltsverzeichnis Seite

1. Einfuehrung

1.1. Beschreibung

Die Memorandum-Makros (MM) sind ein Paket von textformatierenden Makros fuer die Textformatierungsprogramme 'nroff' und 'troff'. Mit dem MM-Paket wird dem Anwender ein einheitlicher flexibler Werkzeugsatz fuer die Erstellung von Dokumenten zur Verfuegung gestellt. Obwohl das MM-Paket dem Nutzer auch fuer spezielle Anwendungsfaelle Makros anbietet, bleibt das MM-Paket ein allgemein nutzbares Makropaket passend fuer jede Art von Textformatierungsaufgaben. Die Anwendung des MM-Paketes reicht dabei von einem einseitigen Brief bis zu Dokumenten von mehreren hundert Seiten Umfang. Das MM-Paket kann angewendet werden bei der Erstellung von:

- Briefen
- Berichten
- technischen Beschreibungen
- Handbuechern
- Buechern

1.2. Vereinbarungen

Jede der folgenden Sektionen beschreibt eine bestimmte Funktion des MM-Paketes. Bei der Reihenfolge wurde ihre Allgemeingueltigkeit beruecksichtigt. Weiter vorn stehende Sektionen werden haeufiger benutzt werden, als weiter hinten stehende. Einige der letzteren Sektion koennen voellig uebergangen werden, wenn die standardmaessig verwendeten MM-Makros ausreichend sind. Innerhalb einer Sektion erfolgt die Beschreibung vom Normalfall zum Spezialfall. Man sollte das Lesen einer Sektion nur soweit betreiben, bis man einen hinreichenden Ueberblick erhalten hat. Den Rest sollte man nur kurz "ueberfliegen", da er sicher nur im speziellen Anwendungsfall interessant ist.
Zahlen in geschweiften Klammern { } beziehen sich auf eine Sektionsnummer innerhalb dieser Beschreibung (z.B. {1.2}).
Sektionen oder Abschnitte, die die Kenntnis der in {1.4} erwaehnten Formatierungsprogramme voraussetzen, sind am Ende der Ueberschrift der entsprechenden Sektion mit einem !*! gekennzeichnet. In der Syntax der Makroaufrufe zeigen eckige Klammern [ ] ein optionales Argument an. Argumente, die mehr als einmal auftreten koennen, werden durch drei aufeinanderfolgende Punkte (...) angezeigt. Ein Verweis der Form name (N) zeigt auf das Kommando name in der Sektion N des WEGA-Referenz-Handbuches. Erzeugen die beiden Formatierungsprogramme eine unterschiedliche Ausgabe, wird zuerst die mit nroff erzeugte angegeben, die mit troff anstelle folgt dann in runden Klammern.
Beispiel: Der "Titel" ist "unterstrichen (kursiv gedruckt)" bedeutet, dass der Titel bei der Erstellung mittels nroff unterstrichen und bei Erstellung mit troff kursiv gedruckt wird.

1.3. Aufbau eines Dokumentes

Fuer das Formatieren eines Dokumentes kann die Eingabe vier Hauptsegmente enthalten, von denen jedes ggf. auch ausgelassen werden kann. Bei Angabe der Segmente muss jedoch unbedingt folgende Reihenfolge eingehalten werden:

- Setzen von Parametern - Durch das Setzen von Parametern wird das aeussere Bild eines Dokumentes festgelegt. Der Nutzer kann Seitengroesse, Randeinstellung, Form der Numerierung von Ueberschriften oder Listen, Seitenueberschriften, Fussnoten und viele andere Elemente des Dokumentes gestalten. Ausserdem kann der Nutzer durch das Setzen von entsprechenden Parametern Makros zum MM-Paket hinzufuegen oder bestehende Makros neu definieren. Dieses Segment kann weggelassen werden, wenn die Standardeinstellung fuer den Anwender ausreichend ist. Durch das Setzen von Parametern wird keine Ausgabe erzeugt, sondern der Aufbau des Dokumentes festgelegt.
- Beginn - Das Beginnsegment definiert die Angaben eines Dokumentes, die nur einmal zu Beginn erscheinen (z.B. Titel, Name des Autors, Datum).
- Hauptteil - Der Hauptteil eines Dokumentes kann einen einzelnen Abschnitt oder aber auch hunderte von Seiten umfassen. Er kann eine Hierarchie von bis zu sieben Ueberschriften beinhalten. Die Ueberschriften werden auf Wunsch automatisch numeriert. Sie koennen ausserdem (ebenfalls auf Wunsch) fuer eine spaetere Verwendung, z.B. zur Erstellung eines Inhaltsverzeichnisses, abgespeichert werden. Fuenf zusaetzliche Ebenen von Unterordnungen werden durch einen Satz von Makros fuer die automatische Numerierung, fuer die alphabetische Anordnung bzw. fuer das Kennzeichnen von Listenangaben zur Verfuegung gestellt. Der Hauptteil eines Dokumentes kann ausserdem verschiedenste Arten von Tabellen, Abbildungen, Referenzen und Fussnoten beinhalten.
- Ende - Dieser Teil des Dokumentes enthaelt nur Angaben, die am Ende eines Dokumentes erscheinen sollen, notwendigenfalls einschliesslich verschiedener Unterschriftsleisten oder Listenbezeichnungen. Hier koennen ausserdem Makros aufgerufen werden, um Angaben komplett oder teilweise aus dem Inhaltsverzeichnis oder dem Deckblatt zu uebernehmen.

Ob die vier beschriebenen Segmente eines Dokumentes auftreten und in welchem Umfang, haengt von der Art des zu erstellenden Dokumentes ab.

1.4. Definitionen

Der im weiteren Text benutzte Begriff "Formatierungsprogramm" bezieht sich auf die Textformatierungsprogramme nroff und troff. Aufrufe (Anforderungen) sind Kommandos, die durch die Formatierungsprogramme erkannt werden. Obwohl durch die Verwendung des MM-Paketes diese Aufrufe selten

direkt benoetigt werden {3.10}, wird sich im vorliegenden Dokument auf einige Formatierkommandos bezogen.
Beispiel: .sp fuegt eine Leerzeile in die Ausgabe ein.

M a k r o s bestehen aus einer Gruppe von Formatierungsaufrufen, denen ein Makroname (Kommandoname) zugeordnet wird. Daraus folgend ist jeder Makroname eine Abkuerzung fuer eine Liste von Aufrufen. Andernfalls wuerde eine sich staendig wiederholende Liste von Anforderungsaufrufen benoetigt. Durch das MM-Paket wird dem Nutzer ein Satz von Standard-Makros bereitgestellt. Zusaetzlich kann er entsprechend seiner Anforderungen spezielle Makros definieren. Makros und Aufrufe teilen sich den gleichen Satz von Namen und werden in der gleichen Art benutzt.

Z e i c h e n k e t t e n werden oft in Kopfteilen{9}, Fussnoten{8} oder Listen{5} benutzt. Auch hier erfolgt der Aufruf durch Angabe ihres Namens. Sie teilen sich ebenfalls den Satz von Namen, der schon von Aufrufen und Makros benutzt wird. Mit dem Aufruf .ds (definiere string) kann einer Zeichenkette ein Wert zugewiesen werden. Den Wert der Zeichenkette enthaelt man durch Angabe ihres Namens. Besteht der Name aus einem Zeichen, muss ein "*" und besteht er aus zwei Zeichen, muss ein "*(" vorangestellt werden.
Beispiel: Die Zeichenkette 'DT' enthaelt normalerweise das aktuelle Datum. Durch Eingabe von: Heute ist der *(DT, kann folgende Ausgabe erzeugt werden: Heute ist der 17. November 1986.
Zur Veraenderung des Datums gibt es zwei Moeglichkeiten: Erstens durch Aufrufen von: .ds DT 01/02/87. Zweitens durch Aufruf des Makros .ND {6.7.1}.

N u m b e r r e g i s t e r dienen zur Speicherung von Zahlen und werden fuer Flags, arithmetische Operationen und zur automatischen Numerierung verwendet. Durch den Aufruf von .nr bzw. durch den Namen des Numberregisters kann diesem ein Wert zugewiesen werden. Besteht der Name aus einem Zeichen, muss ein "\n" und besteht er aus zwei Zeichen, muss ein "\n(" vorangestellt werden.
Beispiel: Der Wert des Registers d soll um 1 groesser sein als der Wert des Registers dd.
.nr d 1 + \n(dd

Anmerkung:
Bei der Festlegung von Namen fuer Aufrufe, Makros, Zeichenketten und Numberregister sind die Vereinbarungen von ((14.1)) zu beachten.

2. Das Aufrufen von Makros

In dieser Sektion werden der Aufbau einiger Kommandozeilen und die Kommandoflags beschrieben.

2.1. Das mm Kommando

Das mm (1) Kommando wird zum Ausdrucken von Dokumenten benutzt, die durch nroff oder durch das MM-Paket erstellt wurden. Der Aufruf von nroff erfolgt mit dem -cm flag {2.2}. Das mm Kommando besitzt Optionen, um die Vorbehandlung mit Hilfe der Praeprozessoren tbl (1) und/oder negn (1) durchzufuehren und eine Verarbeitung durch verschiedene Ausgabehilfen festzulegen. Argumente oder Flags, die nicht durch mm (1) erkannt werden (z.B. -rc3), werden an nroff oder das MM-Paket weitergeleitet. Die Optionen koennen in beliebiger Reihenfolge auftreten, muessen jedoch immer vor den Dateinamen erscheinen (siehe MM(1) im WEGA Programmierhandbuch).

2.2. Das -cm oder -mm Flag

Eine weitere Moeglichkeit des Aufrufes des MM-Paketes besteht in der Einbeziehung des -cm oder -mm Flags als ein Argument der Formatierungsprogramme nroff und troff. Das -cm Flag laedt die vorverdichtete Version der Makros. Durch das -mm Flag wird die Datei /usr/lib/tmac/tmac.c vor allen anderen Dateien gelesen und verarbeitet. Diese Aktion definiert die MM-Makros, setzt die Standardwerte fuer die verschiedenen Parameter und initialisiert das Formatierungsprogramm fuer die Verarbeitung der eingegebenen Textdatei.

2.3. Typische Kommandozeilen

Die Prototyp-Kommandozeilen haben folgenden Aufbau (die verschiedenen Optionen sind in {2.4} erklaert).

- Text ohne Tabellen oder Gleichungen:

```
        mm [options] filename ...
 oder   nroff [options] -mm filename ...
        mmt [options] filename ...
 oder   troff -mm [options]
```

- Text mit Tabellen:

```
        mm -t [options] filename ...
 oder   tbl filename ... | nroff [options] -cm
        mmt-t [options] filename ...
 oder   tbl filename ... | troff [options] -cm
```

- Text mit Gleichungen:

```
        mm -e [options] filename ...
 oder   negn filename ... | nroff [options] -cm
        mmt -e [options] filename ...
```

```
   oder   egn filename ... | troff [options] -cm
```

- Text mit Tabellen und Gleichungen:

```
          mm -t -e [options] filename ...
   oder   tbl filename ...| negn | nroff [optitons] -cm
          mmt [options] -t -e filename ...
   oder   tbl filename ... | egn | troff [options] -cm
```

Erfolgt ein Formatieren mit nroff, sollten bei der Erstellung der Ausgabedatei die Besonderheiten des Ausgabegeraetes (z.B. Rueckwaertstransport des Papiers oder richtungsabhaengige Halbzeilenschritte) beruecksichigt werden. Einige Ausgabegeraete und Kommandozeilen, die dafuer geeignet sind, werden unten angegeben. Siehe auch {2.4} sowie 300(1), 450(1), col(1) und terminals(7) fuer detaillierte Information.

- DAS I 450 im '10-Zeichen per Inch '- und '6 Zeilen per Inch'-Mode mit 0,75 Inch Offset und einer Zeilenlaenge von 6 Inches (60 Zeichen). Ist dies die Standardausgabe, braucht die -T Option nicht benutzt werden (es sei denn, die Shell-Variante $ TERM ist auf einen anderen Wert gesetzt):

```
          mm Dateiname ...   oder
          nroff -T450 -n -cm Dateiname ...
```

- Das I 450 im '12-Zeichen per Inch' - und 'Zeichen per Inch'-Mode mit 0,75 Inch Offset und einer Zeilenlaenge von 6 Inches (72 Zeichen):

```
          mm-12 Dateiname ...   oder
          nroff -T450 - 12 -h -cm Dateiname ...
```

oder um die Zeilenlaennge auf 80-Zeichen zu erhoehen und den Offset auf 3 Zeichen zu verringern:

```
          mm - 12 - rW80 - ro3 Dateiname ...
```

Wenn notwendig muessen, wie schon zu Beginn dieser Sektion in den Prototyp-Kommandozeilen gezeigt wurde, die Praeprozessoren tbl(1) und eqn(1) (fuer troff) oder negn(1) (fuer nroff) aufgerufen werden.

Enthalten durch nroff erzeugte Dokumente zwei oder mehrere Spalten (Kolumnen) {12.4}, muss entweder mm(1) mit der -c Option aufgerufen werden oder die Ausgabe muss mittels col(1) nachbehandelt werden. Im ersten Fall ist zu beachten, dass mm(1) fuer viele Terminaltypen {2.1} automatisch col(1) benutzt. Im zweiten Fall muss der -T37 Terminaltyp fuer nroff angegeben werden; die Angabe der -n Option kann dabei entfallen. Das Ausgabeprodukt von col(1) muss durch einen entsprechenden Terminalfilter verarbeitet werden. (z.B. 450(1)).

2.4. Parameter der Kommandozeile

Numberregister beinhalten Parameter, die die verschiedenen Ausgabearten kontrollieren. Viele von ihnen koennen innerhalb von Textdateien durch Aufruf von .nr geaendert werden. Zusaetzlich dazu koennen einige dieser Register in der Kommandozeile selbst gesetzt werden. Das ist vor allem fuer jene Parameter sinnvoll, die nicht staendig in einem Text vorhanden sind. Werden diese Register benutzt (mit Ausnahme des Registers P, siehe unten), muessen sie in der Kommandozeile oder vor der Abarbeitung der definierten MM-Makros gesetzt sein. Bedeutung der Numberregister:

-rA1n n=1 hat die Wirkung eines argumentlosen .AF Makroaufrufes {6.8} n=2 gestattet die Anwendung eines akustischen Signals (falls vorhanden).

-rBn definiert die Makros fuer die Erstellung des Inhaltsverzeichnisses und fuer das Deckblatt. n=1 gestattet das Anlegen des Inhaltsverzeichnisses. n=2 gestattet die Erstellung eines Deckblattes. n=3 erzeugt beides, d.h., hat B einen Wert groesser 0, werden die .TC {10.1} und/oder .CS {10.2} Makros definiert. Um eine Wirkung zu erzielen, muessen diese Makros dann auch aufgerufen werden.

-rCn n gibt die Art der Kopie an (z.B. DRAFT), die unten auf jede Seite gedruckt wird {9.5}.

n = 1 fuer OFFICIAL FILE COPY
N = 2 fuer DATE FILE COPY
n = 3 fuer DRAFT mit einfacher Sperrung und Standardabsatz
n = 4 fuer DRAFT mit doppelter Sperrung und der Absatz ist 10 Zeichen eingerueckt.

-rD1 stellt den Debug-Mode ein. Dieses Flag zwingt das Formatierungsprogramm, die Abarbeitung auch bei Fehlern, die durch das MM-Paket entdeckt wurden, fortzusetzen. Andernfalls wuerden diese Fehler zum Abbruch fuehren. Es umfasst auch einige Debuggerinformationen im standardmaessigen Kopfteil der Seiten {9.2, 12.3}.

-rEn gibt den Schriftsatz der Felder Thema/Datum/Autor an. Bei n=0 erscheinen diese Felder im Fettdruck (Standard fuer troff) und bei n=-1 sind diese Felder regulaerer Text (Standard fuer nroff).

-rLk k gibt die Anzahl der Zeilen an, die pro Seite gedruckt werden (fuer nroff repraesentiert k die Zeilen oder Zeichenanzahl; fuer troff repraesentiert k eine entsprechende Bezugsgroesse). Als Standardwert werden 66 Zeilen pro Seite eingestellt.

-rNn gibt die Art der Seitennumerierung an. Bei n=0 (Standardwert) wird der Seitenkopf auf allen Seiten gedruckt {9.2}. Bei n=1 wird der Fussteil (aber nur auf der 1. Seite) durch den Kopfteil ersetzt. Bei n=2 wird der auf der

ersten Seite erscheinende Kopfteil weggelassen. Bei n=3 erscheint eine Seitennumerierung fuer jede Sektion {4.5}; siehe auch .FD {8.3} und .RP {11.4}. Bei n=4 wird der standardmaessig erscheinende Seitenkopf unterdrueckt, ohne dass der vom Nutzer definierte Kopfteil beeinflusst wird. Bei n=5 werden die Seiten und Abbildungen innerhalb einer Sektion numeriert.

n	Seite 1	Seite 2
0	Kopfteil	Kopfteil
1	Kopfteil ersetzt Fussnote	Kopfteil
2	kein Kopfteil	Kopfteil
3	Sektionsseitennummer als Fussteil	
4	kein Kopfteil	kein Kopfteil, wenn nicht PH angegeben wurde

Der Text des Kopf- und Fussteils haengt nicht vom Wert des Numberregister ab. n gibt nur an, ob und wo der Kopfteil (und fuer n = 3 oder 5 auch der Fussteil) gedruckt wird, sowie die Art der Seitennumerierung. Besonders dann, wenn Kopf- und Fussteil nicht vorhanden sind {9.2, 9.5} ist der Wert von n irrelevant.

- rOk offset der Ausgabe um k Plaetze nach rechts (fuer nroff repraesentiert k die Zeilen- oder Zeichenpositionen; fuer troff repraesentiert k eine entsprechende Bezugsgroesse). Die Angabe eines Offset ist sinnvoll fuer die Positionierung der Ausgabe auf manchen Terminals. Wurde das Register nicht in der Kommandozeile gesetzt, ist das Standardoffset .75 Inches.
Beachte: Der Name des Registers ist der Grossbuchstabe O, nicht die Zahl Null (0)!

- rPn gibt an, dass die zu numerierenden Seiten des Dokumentes mit n beginnen. Dieses Register kann auch im Eingabetext mit der .nr-Anforderung gesetzt werden.

- rSn stellt die Punktgroesse und den Vertikalabstand fuer das Dokument. Der Standardwert fuer n ist 10 fuer die Punktgroesse und 12 Punkte fuer den vertikalen Abstand (6 Zeilen pro Inch). Dieser Parameter wird nur fuer troff angewendet.

- rTn stellt das Register auf Werte entsprechend der verwendeten Geraete. Bei n = 1 werden die Zeilenlaenge auf 80 und der Seiten-Offset auf 3 gesetzt. Bei n = 2 wird die Zeilenanzahl auf 84 Zeilen pro Seite geaendert und ein Unterstreichen wird verboten.
Der Standardwert fuer n ist 0. Dieser Parameter wird nur fuer nroff angewendet.

- rU1 es erfolgt ein Unterstreichen der Sektionsueberschriften. Es werden jedoch nur Buchstaben und Zahlen unterstrichen. Soll die gesamte Ueberschrift (einschliess-

lich Leerzeichen) unterstrichen werden, siehe {4.2.2.4}. Dieser Parameter wird nur fuer nroff angewendet.

- rWk mit diesem Parameter wird die Groesse einer Seite angegeben (z.B. Zeilenlaenge und Titellaenge) (fuer nroff repraesentiert k die Zeilen- oder Zeichenpositionen; fuer troff repraesentiert k eine entsprechende Bezugsgroesse). Damit kann die Standardseitengroesse von 6 Inches (60 Zeichen im '10-Zeichen per Inch '-Mode oder 72 Zeichen im '12-Zeichen per Inch '-Mode) veraendert werden.

2.5. Das Weglassen des -cm oder -mm Flags

Ist eine grosse Anzahl von Argumenten in der Kommandozeile gefordert, ist es zweckmaessig, die erste (oder einzigste) Eingabedatei eines Dokumentes wie folgt anzugeben:
Null oder mehrere Initialisierungswerte der Register sind in {2.4} angegeben.

```
.so/usr/lib/tmoc/tmac.m.
Rest des Textes.
```

In diesem Fall muessen weder das -cm oder -mm Flag, noch das mm(1) oder mmt(1) Kommando angegeben werden. Die gleiche Wirkung wird erreicht durch den .so Aufruf, jedoch muessen zuvor die Register von {2.4} geladen werden. Ihre Werte sind nur von Bedeutung, wenn sie gesetzt sind, bevor die Makros abgearbeitet werden. Wird diese Methode angewendet, ist es am besten nur jene Parameter in die Eingabedatei einzubeziehen, die selten geaendert werden.
Beispiel:

```
. nr  W 80
. nr  O 10
. nr  N  3
. so /usr/lib/tmoc/tmac.m
. H 1 "EINFUEHRUNG"
  .
  .
```

gibt die Zeichen fuer nroff an, eine Zeile von 80 Zeichen Laenge; eine Seitenoffset von 10; 'Sektion-Seite'-Numerierung und Verarbeitung des Inhaltsverzeichnisses.

3. Formatierungskonzept

3.1. Grundbegriffe

Die Aufgabe eines Formatierungsprogrammes besteht darin, eine eingegebene Datei so aufzubereiten, dass daraus eine druckbare Datei mit einem sauberen und uebersichtlichen Druckbild entsteht. Der optische Eindruck eines Druckbildes wird bestimmt durch

- eine sinnvolle Einteilung in Absaetze
- einen geraden rechten Rand
- silbenrichtige Worttrennung am Zeilenende

Es ist auch moeglich, diese Modes ein- und auszuschalten (siehe .P {4.13}, .SA {12.2}, .Hy {3.4}, .nf- und fi. Aufrufe). Dabei ist allerdings zu beachten, dass ein Abschalten des Fill-Modes durch den .nf-Aufruf auch ein Abschalten der Justierung und Silbentrennung zur Folge hat.

Bestimmte Formatierungskommandos (Anforderungen und Makros) fuehren zum Umbruch in der Zeile, d.h., dass der nachfolgende Text auf einer neuen Zeile beginnt. Diese Art des Zeilenendes wird als Break bezeichnet. Die Formatierungsaufrufe und MM-Makros erzeugen solche Breaks.

Die Erstellung von Dokumenten ist auf zwei Arten moeglich. Erstens durch die Benutzung der Formatierungsaufrufe und zweitens durch Verwendung des MM-Paketes. Bei der gleichzeitigen Benutzung von Formatierungsaufrufen aus dem MM-Paket heraus, muss sich der Anwender eventueller Nebeneffekte bewusst sein. Die Erstellung mit Hilfe des MM-Paketes ist vorteilhafter, da

- es viel einfacher ist, die Ausfuehrungsart eines Dokumentes zu kontrollieren und zu einem spaeteren Zeitpunkt zu aendern;

- komplizierte Funktionen (z.B. die Erstellung von Fussnoten oder Inhaltsverzeichnisse) bequem erreicht werden koennen;

- der Nutzer sich nicht um die Besonderheiten der Formatierungssprache zu kuemmern braucht.

Formatierungsanforderungen sollten nur verwendet werden, wenn unbedingt notwendig {3.10}. Um das Ueberarbeiten eines eingegebenen Textes zu einem spaeteren Zeitpunkt zu erleichtern, sollten die eingegebenen Zeilen kurz sein und immer mit einem Newline beendet werden.

3.2. Argumente, Anfuehrungszeichen (") und Softspaces

Fuer jeden Makroaufruf ist ein Null-Argument ein Argument der Weite Null. Solch ein Argument hat oft eine spezielle

Bedeutung. Die bevorzugte Form dafuer ist " ". Man sollte beachten, dass das Weglassen eines Argumentes nicht einem Null-Argument gleichzusetzen ist (siehe .MT-Makro in {6.6}). Weiterhin koennen auszulassende Argumente nur am Ende einer Argumentenliste auftreten. Im Gegensatz dazu koennen Null-Argumente ueberall erscheinen. Ein Argument, das gewoehnliche Leerzeichen (Softspaces) enthaelt, muss in " " (Anfuehrungszeichen) eingeschlossen sein. Andernfalls wird es wie mehrere einzelne Argumente behandelt.

Anmerkung:
Das "-Zeichen (Anfuehrungszeichen) ist ein Zeichen, das durch eine einzige Tasteneingabe erzeugt wird. Es sollte nicht mit den zwei Apostrophen ' ' oder den zwei Akzentzeichen ` ` verwechselt werden, fuer die zwei Tasteneingaben erforderlich sind.

Die Benutzung des "-Zeichens ist weder als Teil eines Wertes fuer ein Makroargument, noch als Zeichenkette gestattet, wenn diese Zeichenkette als Makroargument verwendet wird. Moechte man dieses Zeichen doch verwenden, sollte man auf die Apostrophs oder-/und Akzentzeichen ausweichen. Diese Einschraenkung ist notwendig, da viele Makroargumente beliebig oft verarbeitet werden, z.B. koennen Textueberschriften auch im Inhaltsverzeichnis erscheinen.

3.3. Undehnbare Leerzeichen (Hardspaces)

Soll eine Ausgabe so erstellt werden, dass sie einen sauberen geraden rechten Rand ergibt, kann es sein, dass zu den schon vorhandenen Leerzeichen noch weitere hinzugefuegt werden muessen. Das kann jedoch die erwuenschte Anordnung des Textes nachteilig beeinflussen. Um dieses Problem zu umgehen, ist es notwendig, ein Leerzeichen anzugeben, das waehrend der Druckaufbereitung nicht in seiner Groesse veraendert werden kann. Dies kann durch sogenannte undehnbare Leerzeichen (Hardspaces) geschehen. Fuer ihre Erzeugung gibt es mehrere Wege:

1. Durch Eingabe eines Backslash-Zeichens mit anschliessendem Leerzeichen ("\ "). Dieses Paar erzeugt unmittelbar ein undehnbares Handspace-Leerzeichen.

2. Nicht - oder selten benutzte Sonderzeichen koennen als Hardspaces fuer die Ausgabe verwendet werden. Das dafuer am meisten verwendete Sonderzeichen ist die Tilde (~). In diesem Fall muss dann aber am Anfang der Aufruf

```
.tr~
```

erfolgen.

Sollte die Tilde im Ausgabetext selbst verwendet werden, kann die durch Aufruf von

```
.tr~~
```

zurueckerhalten werden.

Die vorangegangene Bedeutung der Tilde kann durch ein Wie-

derholen des .tr~-Aufrufes erlangt werden. Dieser Aufruf wirkt aber nur nach einem Break oder nach der Zeile, die die Tilde erzwungen hat.

Anmerkung:
Diese Art der Erzeugung von Hardspaces ist nicht empfehlenswert, wenn ein Dokument viele Gleichungen besitzt, die die Tilde selbst enthalten.

3.4. Silbentrennung

Durch die Formatierungsprogramme wird keine silbenmaessige Worttrennung durchgefuehrt wenn der Nutzer es nicht ausdruecklich fordert. Die Silbentrennung kann durch den Aufruf

```
.nr Hy 1
```

zu Beginn des Textes eingeschaltet werden. (Silbentrennung innerhalb der Fussnoten siehe {8.3}).
Ist die Silbentrennung gestattet, wird sie durch die Formatierungsprogramme automatisch an den entsprechenden Wortstellen durchgefuehrt. Der Nutzer kann jedoch, falls eine spezielle Darstellungsart fuer ein Wort erwuenscht wird, Indikatoren fuer die Silbentrennung angeben. Dies geschieht entweder durch Benutzung spezieller Zeichen oder durch das Erstellen einer kleinen Liste von Worten mit fest vorgegebenen Punkten fuer die Trennungsstricheinfuegung (ungefaehr 128 Zeichen).
Durch die Benutzung des .hw-Aufrufes kann der Nutzer eine kleine Liste von Worten mit zuvor fuer die Silbentrennung markierten Stellen zusammenstellen.
Beispiel: Um die entsprechende Stelle im Wort 'printout' anzuzeigen, ist folgende .hw-Anforderung erforderlich.

```
.hw print-out
```

Standardmaessig wird die Zeichenfolge "\%" als Indikator fuer die Silbentrennung verwendet. Erscheinen diese Zeichen zu Beginn eines Wortes, wird es nicht in die Silbentrennung einbezogen. Diese Zeichen werden alternativ verwendet, d.h. sie werden auch zur Markierung der Silbe(n) innerhalb eines Wortes benutzt. In beiden Faellen erscheinen sie nicht in der Ausgabe.
Dem Nutzer steht es aber auch frei, ein anderes Zeichen als Silbentrennungssindikator zu definieren. Dies geschieht durch:

```
.HC [ Trennungssindikator ]
```

Oft wird das "^"-Zeichen fuer diesen Zweck benutzt. In diesem Fall ist zu Beginn des Dokumentes

```
.HC ^
```

anzugeben.

Anmerkung:
Es ist zu beachten, dass Worte, die Binde- oder Gedankenstriche (nroff em dashes) enthalten, auch dann nach diesen Zeichen getrennt werden, wenn die Silbentrennung abgeschaltet ist. Die Silbentrennung erfolgt auch dann nur wenn notwendig.

3.5. Tabs

Die Makros .MT {6.6}, .TC {10.1} und .CS {10.2} benutzen fuer das Setzen und Rueckspeichern der Standardwerte der Tabstops (alle 8 Zeichen fuer nroff; alle 1/2 inch fuer troff) die .ta-Anforderung. Werden durch den Nutzer die Tabs auf andere Werte als die standardmaessig verwendeten gesetzt, geschieht das auf eigene Verantwortung.

Es ist zu beachten, dass ein Tabzeichen immer von seiner Stellung in der Eingabezeile her betrachtet wird und nicht von seiner Position in der Ausgabezeile. Generell sollten Tabs nur in Zeilen auftreten, die im No-fill-Mode {3.1} verarbeitet wurden.
Weiterhin ist zu beachten, dass tbl (1) {7.3} zwar die Tabstops aendert, jedoch die Standardwerte fuer die Lage der Tabs nicht nachlaedt.

3.6. Die spezielle Benutzung des BEL-Zeichens

Das nichtdruckbare BEL-Zeichen wird in vielen Makros, in denen es noetig ist die Argumentgroesse zu berechnen bzw. einen willkuerlichen Text zu begrenzen (wie in Kopfteilen und Fussnoten {9}, in Ueberschriften {4} und Markenlisten {5}), als Begrenzer benutzt. Nutzer, die dass BEL-Zeichen in ihren Eingabetext mit einbeziehen (speziell in den Makroargumenten), erhalten eine fehlerhafte Ausgabe.

3.7. Das "+"Zeichen (Bulletin-Zeichen)

Das "+"Zeichen wird auf einem Typenraddrucker so erzeugt, indem ein o durch ein + ueberdruckt wird. Fuer die Kompatibilitaet mit troff wird dieses Zeichen durch MM erstellt. Dies geschieht durch Angabe von

```
\* ( BU
```

wo immer solch ein Zeichen erwuenscht ist.
Es ist zu beachten, dass die Bulletin-List Makros (= BL) {5.3.3.2} diese Zeichenkette (string) fuer die automatische Erzeugung der Bulletins in der Postenliste benutzen.

3.8. Gedankenstriche, Minuszeichen und Bindestriche

Im Gegensatz zu nroff hat troff feste Graphikcodes fuer Gedankenstriche, Minuszeichen und Bindestriche. Wird nur nroff benutzt, kann das Minuszeichen ("-") fuer alle drei Zeichen benutzt werden.

Nutzer, die ueberwiegend troff benutzen, sollten die Vereinbarungen des NROFF User Manuals beachten.

Aber jene Anwender, die beide Formatierungsprogramme anwenden, muessen bei der Texterstellung beachten, dass diese Zeichen nicht so dargestellt werden, dass sie fuer beide Formatierungsprogramme geeignet sind. Deshalb werden folgende Vereinbarungen getroffen:

Gedankenstrich — Eingabe von "*(EM" fuer beide Formatierungsprogramme. Diese Zeichenkette erzeugt ein em dash (-)in troff und ein (--) in nroff. Dabei ist zu beachten, dass die Dash-List Makros (.DL) {5.3.3.3} automatisch die em dashes (Gedankenstriche) fuer die Postenliste erzeugen.

Bindestrich — Eingabe von "-" fuer beide Formatierungsprogramme. Bei Benutzung von nroff erfolgt die Ausgabe als "-" und bei troff als Bindestrich.

Minus — Eingabe von "\-" fuer ein wirkliches Minuszeichen ungeachtet des Formatierungsprogrammes. nroff ignoriert das \, waehrend troff ein wahres Minuszeichen ausgibt.

3.9. Das TM-Zeichen (Trademark)

Die Zeichenkette "*(TM" fuer Trademark ist ebenfalls durch MM verfuegbar. Sie wird eine halbe Zeile ueber den Text geschrieben, dem sie folgt.
Beispiel: P8000 *(TM Handbuch plaziert das TM-Zeichen nach P8000.

3.10. Benutzung der Formatierungsaufrufe

Die meisten Formatierungsaufrufe sollten nicht mit MM benutzt werden. MM realisiert die entsprechenden Formatierungsfunktionen mehr nutzerorientiert als die Grundformatierungsanforderungen {3.1}. Einige Formatierungsanforderungen sind jedoch auch mit MM sinnvoll.

```
.af  .br  .ce  .de  .ds
.fi  .h w .ls  .nf  .nr
```

```
.nx  .rm  . rr .rs  .so
.sp  .ta  .ti  .tl  .tr
```

Fuer troff sind manchmal auch die .fp, .lg und .ss Anforderungen sinnvoll. Die Benutzung anderer Anforderungen, ohne ein Verstehen ihrer vollen Bedeutung, kann zu Fehlern fuehren.

4. Absaetze, Abschnitte und Ueberschriften

Um ein Dokument uebersichtlich zu gestalten, ist es sinnvoll, den Text in Abschnitte und Absaetze zu gliedern und durch verschiedene (evtl. numerierte) Ueberschriften hervorzuheben. Diese Sektion beschreibt das Erstellen von Absaetzen bzw. Abschnitten und Ueberschriften. Auf das Anlegen von zusaetzlichen Abschnitten und Listen wird in {5} naeher eingegangen.

4.1. Absaetze

```
.P [Typ]
ein oder mehrere Textzeilen
```

Dieses Makro wird benutzt, um Absaetze nach Wunsch einzuruecken oder zu numerieren. Im allgemeinen werden alle Absaetze linksbuendig angeordnet, wobei die erste Zeile auf Wunsch auch eingerueckt sein kann. Diese Anordnung wird erreicht durch Aufruf des .P-Makros und Register Pt vor Absaetzen, denen keine Ueberschrift vorangestellt ist {4.2}. Der Initialwert des Registers Pt ist 0, was bedeutet, dass nach einem Aufruf von .P 0 der folgende Text auf einer neuen Zeile beginnt (und somit einen neuen Abschnitt kennzeichnet), aber die erste Zeile nicht eingerueckt ist.

```
.P 0
```

Soll die erste Zeile eines neuen Absatzes eingerueckt werden, muss ein Aufruf von .P 1 erfolgen.
Durch Setzen des Pt-Registers auf 1 werden alle Absaetze, einschliesslich des Typs 0, standardmaessig eingerueckt.
Sollen alle Absaetze (ausser nach Ueberschriften, Listen und Hervorhebungen) eingerueckt werden, muss zu Beginn des Dokumentes das Pt-Register auf 2 gestellt werden.

```
.nr Pt 2
.P 1
```

In den beiden zuletzt genannten Faellen erfolgt ein Einruecken der ersten Zeile jeweils um 5 Stellen (Standardwert). Sollte der Nutzer mit diesem Betrag nicht zufrieden sein, kann er durch Laden des Pi-Registers eine andere Einrueckgroesse erzwingen. Um einen Absatz um 10 Stellen einzuruecken, muss

```
.nr Pi 10
.P 1
```

zu Beginn des Dokumentes angegeben sein.
Das Numberregister Ps gibt den Zwischenraum zwischen den Absaetzen an. Im Standardfall hat Ps den Wert 1, was einen Abstand von einer Leerzeile (1/2 vertikaler Abstand) bedeutet.

Achtung:
Die Einrueckwerte muessen direkt angegeben sein und werden wie Zeichenpositionen behandelt, z.B. als eine Zahl von 'ens'. Im troff repraesentiert ein 'en' die Zahl der Punkte (1 Punkt = 1/72 Inch) gleich der halben aktuellen Punktgroesse. Im nroff ist ein 'en' gleich der Weite eines Zeichens.

Ohne Ruecksicht auf den Wert von Pt kann bei einzelnen Absaetzen eine Ausrichtung auf den linken Rand oder ein Einruecken erzwungen werden. .P 0 erzwingt immer ein Ausrichten nach dem linken Rand, waehrend .P 1 ein Einruecken entsprechend des Wertes des Pi-Registers zur Folge hat. Erscheint P1 innerhalb einer Liste, wird der Wert von Pi zu den angegeben Leerzeichen der aktuellen Liste hinzuaddiert {5}. Ein Numieren der Absaetze wird erreicht durch das Setzen des Np-Registers auf 1. Die dabei erzeugten Absaetze werden innerhalb der ersten Ueberschriftsebene numeriert, z.B. 1.01, 1.02, 1.03, 2.01 usw.

Eine andere Art der Numerierung erhaelt man durch die Benutzung des .nP-Makros anstelle des .P-Makros. In diesem Fall werden die Absaetze innerhalb der zweiten Ueberschriftsebene numeriert und die ersten beiden Zeichen werden eingerueckt. Die dadurch entstehende Numerierung hebt sich deutlicher ab.

```
.H 1 "ERSTE UEBERSCHRIFT"
.H 2 "Zweite Ueberschrift"
.nP
eine oder mehrere Textzeilen
```

4.2. Numerierte Ueberschriften

```
.H Ebene [ Ueberschrift ] [Suffix]
keine oder mehrere Textzeilen
```

Das .H Makro liefert bis zu sieben Ebenen von numerierten Ueberschriften. Die Zahl 1 verkoerpert die hoechste, die Zahl 7 die niedrigste Ebene. Der Suffix wird an die Ueberschrift angehaengt und kann fuer Fussnoten verwendet werden, die nicht mit der Ueberschrift zusammen im Inhaltsverzeichnis erscheinen sollen.

Hinweis:
Da das .H Makro (oder .HU {4.3}) auch die Funktion des .P Makro durchfuehrt, ist eine Benutzung des .P Makro nach Aufruf des .H Makro nicht sinnvoll und wird ignoriert {4.2.2.2.}.

4.2.1. Standardmaessige Ueberschriften

Die Darstellungsart der Ueberschrift ist abhaengig von ihrer Ebene. Einer Ueberschrift der ersten Ebene gehen zwei

Leerzeilen (ein vertikaler Abstand) voraus. Allen anderen Ueberschriften geht eine Leerzeile (1/2 vertikaler Abstand) voraus.

.H1 UEBERSCHRIFT	erzeugt eine Ueberschrift im Fettdruck, gefolgt von einer Leerzeile (1/2 vertikaler Abstand). Der folgende Text beginnt auf einer neuen Zeile und ist entsprechend des aktuellen Absatztypes eingerueckt. Damit sich diese Ueberschrift besser abhebt, sollte sie komplett in Grossbuchstaben erscheinen.
.H2 Ueberschrift	erzeugt ebenfalls eine Ueberschrift im Fettdruck, gefolgt von einer Leerzeile (1/2 vertikalem Abstand). Der folgende Text beginnt auf einer neuen Zeile und ist entsprechend des eingestellten Absatztyps eingerueckt. Normalerweise werden nur die Anfangsbuchstaben grossgeschrieben.
.Hn Ueberschrift	fuer 3 < n < 7 wird eine unterstrichene (kursivgedruckte) Ueberschrift erzeugt, der zwei Leerzeichen folgen. Der folgende Text erscheint auf der gleichen Zeile (eingefuegte Ueberschriften).

Die entsprechende Numerierung und Platzfreilassung (horizontal und vertikal) erfolgt selbst dann, wenn bei einem Aufruf des .H Makros kein Text fuer die Ueberschrift angegeben wurde.

4.2.2. Ueberschriften mit unterschiedlichem Aussehen

Nutzer, die mit der Standarderstellung von Ueberschriften zufrieden sind, koennen diesen Abschnitt ueberspringen und bei {4.3} weiterlesen. Durch das Setzen von bestimmten Registern und Zeichenketten (strings) ist es moeglich, den optischen Eindruck von Ueberschriften zu veraendern.

4.2.2.1. Seitenvorschub und Zwischenraum

Einer Ueberschrift der ersten Ebene gehen normalerweise, wie schon erwaehnt, zwei Leerzeilen (ein vertikaler Abstand) voraus. Alle anderen Ueberschriften werden durch eine Leerzeile (1/2 vertikalen Abstand) eingeleitet. Passt eine aus mehreren Zeilen bestehende Ueberschrift nicht mehr komplett auf eine Seite, erfolgt automatisch ein Seitenvorschub. Jede Ueberschrift der ersten Ebene kann durch Angabe von

```
. nr Ej 1
```

an den Anfang einer neuen Seite gerueckt werden. Auch diese

Angabe muss am Beginn des Dokumentes stehen. Der hinter Ej angegebene Wert entspricht der Ebene (ausgehend von 1), bis zu der der Seitenvorschub fuer Ueberschriften wirkt. Es tritt also immer ein Seitenvorschub auf, wenn die Ebene der Ueberschrift kleiner oder gleich Ej ist.
Beispiel: Durch die Angabe von .nr Ej 3 erzwingen alle Ueberschriften der 1. bis 3. Ebene einen Seitenvorschub.

4.2.2.2. Abstand nach Ueberschriften

Das Druckbild eines Textes, das einem .H-Aufruf folgt, wird durch drei Register bestimmt. Das sind das Hb-Register (Heading break), das Hs-Register (Heading space) und das Hi-Register (Post-Heading indent).

Ist das Niveau der Ueberschrift kleiner gleich Hb, erscheint nach der Ueberschrift ein Break {3.1}. Ist das Niveau kleiner gleich Hs wird nach der Ueberschrift eine Leerzeilel (1/2 vertikaler Abstand) eingefuegt. Die Standardwerte fuer Hb und Hs sind 2. Ist das Niveau der Ueberschrift groesser als Hb und auch groesser als Hs, dann reicht die Uberschrift (falls vorhanden) in den Text hinein. Diese Register gestatten es, Ueberschriften innerhalb eines Dokumentes in einer einheitlichen Art vom Text zu trennen.

Fuer alleinstehende Ueberschriften, das sind Ueberschriften, die nicht in den folgenden Text uebergehen, wird die Anordnung der naechsten Ausgabezeile durch das Register Hi bestimmt. Ist Hi = 0, wird der Text linksbuendig ausgerichtet. Ist Hi = 1 (Standardfall), wird der Text entsprechend des durch Register Pt {4.1} eingestellten Abschnittypes eingerichtet. Ist Hi = 2, wird der Text so eingerueckt, dass er unter dem ersten Wort der Ueberschrift steht und sich somit die Numerierung noch staerker abhebt.
Beispiel: Um folgende Anordnung zu erreichen:

- eine Leerzeile (1/2 vertikaler Abstand) nach den ersten drei Ueberschriftsebenen

- keine Ueberschriften, die in den Text uebergehen

- die Texte, die den Ueberschriften folgen, sollen (ohne Ruecksicht auf den Pt-Wert) linksbuendig ausgerichtet sein,

muessen am Anfang des Dokumentes folgende Anweisungen stehen:

```
. nr Hs 3
. nr Hb 7
. nr Hi 0
```

4.2.2.3. Zentrierte Ueberschriften

Um zentrierte Ueberschriften zu erhalten, kann das Hc-Register benutzt werden. Eine Ueberschrift wird zentriert, wenn ihr Niveau kleiner oder gleich Hc ist und wenn sie allein steht {4.2.2.2}. Der standardmaessige Ladewert fuer Hc ist 0 (nicht zentrierte Ueberschriften), soll er so veraendert werden, dass auch Ueberschriften der 2. Ebene zentriert werden, muss . nr Hc 2 angegeben werden.

4.2.2.4. Ueberschriften im Fett- oder Kursivdruck und mit Unterstreichungen

a) Die Einstellung erfolgt durch einen Schluessel.
Jede Ueberschrift, die durch nroff unterstrichen wird, erscheint bei troff in Kursivschrift. Die Zeichenkette HF (Heading Font) enthaelt sieben Codes (1-7), die den Schriftsatz fuer die Ueberschriftsebenen 1 - 7 angeben. Die gueltigen Codes , ihre Interpretationen und die Standardeinstellungen fuer HF sind der nachfolgenden Tabelle zu entnehmen:

Format.-progr.	HF Code 1	2	3	HF Standard-einstellg.
(Ebene				1 2 3 4 5 6 7)
nroff	keine U	U	U	3 3 2 2 2 2 2
troff	Antiqua	kursiv	fett	3 3 2 2 2 2 2

U = Unterstreichung

So erzeugt die Standardeinstelung fuer die Ebenen 1 und 2 eine Unterstreichung der Ueberschrift (Fettdruck) und fuer die Ebenen 3 bis 7 eine Unterstreichung (Kursivdruck). Der in der HF-Zeichenkette am weitesten links stehende Wert gibt den Schriftsatz fuer die Ueberschriftsebene 1 an. Ausgelassene Werte werden als 1 gedeutet.
Im folgenden Beispiel ist der Schriftsatz fuer die Ebenen 1 bis 5 angegeben und bedeutet Fettdruck und zwei Nichtunterstreichungen bzw. Antiqua (Ebene 6 und 7).

```
.ds HF 3 3 3 3 3
```

b) Unterstreichungen bei nroff
Unterstreichungen durch nroff koennen auf zwei Arten erfolgen.In der ersten Art (.ul Anforderung) werden nur Buchstaben und Zahlen unterstrichen. In der zweiten Art werden alle Zeichen einschliesslich Leerzeichen unterstrichen. Im Standardfall werden durch das MM-Paket die Ueberschriften entsprechend der zweiten Art unterstrichen. Dies geschieht jedoch nur dann, wenn die Ueberschriften so kurz sind, dass sie in eine Zeile passen. Ist eine zu unterstreichende Ueberschrift zu lang, erfolgt die Unterstreichung nach der ersten Art. Ein Un-

terstreichen von Ueberschriften durch die erste Art kann durch Benutzung des -rU1 Flags beim Aufruf von nroff {2.4} erzwungen werden.

c) Punktgroesse der Ueberschrift bei troff
Der Nutzer kann auch mittels der Hp-Zeichenkette die gewuenschte Punktgroesse fuer jede Ueberschriftsebene angeben

.ds HP [ps1] [ps2] [ps3] [ps4] [ps5] [ps6] [ps7]

Im Standardfall wird die Ueberschrift (.H und .HU) in der gleichen Punktgroesse wie der Rest des Textes gedruckt. Die Ausnahme bilden alleinstehende Ueberschriften im Fettdruck, sie werden einen Punkt kleiner als der Rest des Textes gedruckt. Die Zeichenkette HP kann, aehnlich der HF-Zeichenkette, bis zu sieben Werte enthalten. Diese Werte entsprechen den sieben Ebenen der Ueberschriften.
Beispiel: .ds HP 12 12 10 10 10 10 10

gibt an, dass die ersten beiden Niveauebenen der Ueberschriften in der Punktgroesse 12 und der Rest in der Punktgroesse 10 zu drucken sind.
Es ist auch moeglich, die Punktgroesse als relative Werte (als Punktgroessenaenderungen) anzugeben.
Beispiel: .ds HP +2 +2 -1 -1

Werden absolute Punktgroessen verwendet, geschieht das ohne Beruecksichtigung der Punktgroesse des Textteils. Werden relative Punktgroessen verwendet, werden diese relativ zum verwendeten Text angegeben, selbst dann, wenn der Text geaendert wird. Ausgelassene Werte oder Null bewirken, dass die Standardpunktgroesse fuer die entsprechende Ueberschriftsebene genommen wird.

Achtung:
Es wird nur die Punktgroesse der Ueberschriften beeinflusst. Die Angabe einer grossen Punktgroesse ohne Vergroesserung des vertikalen Abstandes (mit .HX und/oder .HZ) kann ein "Ueberdrucken" zur Folge haben.

4.2.2.5. Moeglichkeiten zur Numerierung und Kennzeichnung von Ueberschriften

.HM [arg 1] ... [arg 7]

Die Register H1 bis H7 werden als Zaehler fuer die sieben Ebenen der Ueberschrift benutzt. Ihre Werte werden normalerweise in arabischen Ziffern gedruckt. Durch Anwendung des .HM Makros stehen dem Nutzer auch andere Moeglichkeiten der "Numerierung" zur Verfuegung. Das Makro kann bis zu sieben Argumente haben. Jedes Argument ist eine Zeichenkette, die die Art der benutzten Kennzeichnung anzeigt. Die

gueltigen Werte und ihre Bedeutung sind der unten angegebenen Tabelle zu entnehmen. Ausgelassene Werte werden als 1 gewertet, waehrend ungueltige Werte keine Wirkung haben.

Wert	Bedeutung
1	arabische Ziffern (Standard fuer alle Niveaus)
0001	arabische Ziffern mit fuehrenden Nullen, um eine entsprechende Zahl zu erhalten
A	Grossbuchstaben
a	Kleinbuchstaben
I	roemische Zahlen mit Grossbuchstaben (I, V, X)
i	roemische Zahlen mit Kleinbuchstaben (i, v, x)

Im Standardfall erfolgt die vollstaendige Ueberschriftsgestaltung fuer eine gegebene Ebene in der Weise, dass die Kopfzeile mit ihrer der Ebene entsprechenden Kennzeichnung rechts an die Kennzeichnungen aller hoeheren Ebenen angehaengt wird. Um dieses Aneinanderreihen von Kennzeichnungen zu verbieten, d.h. um lediglich die Kopfzeile der angegebenen Ebene, die mit einem Punkt abgeschlossen wird, zu erhalten, ist das Ht-Register auf 1 zu setzen.
Beispiel: .HM I A 1 a i
.nr Ht 1

erzeugt die allgemein benutzte Darstellungsart.

4.3. Unnumerierte Ueberschriften

.HU Ueberschrift

Das .HU-Makro ist ein Spezialfall des .H-Makro und wird in der gleichen Art und Weise behandelt, mit der Ausnahme, dass keine Ueberschriftskennzeichnung gedruckt wird. Um die hierarchische Struktur von Ueberschriften zu bewahren, wenn H und .HU-Makros abwechselnd auftreten, wird jede .HU-Ueberschrift niveaumaessig so behandelt, wie das Niveau im Hu-Register angegeben wurde. Der Initialwert des Hu-Registers ist 2. Der Unterschied zwischen

.HU Ueberschrift
und
.H2 Ueberschrift

besteht nur darin, dass im letzteren Fall die Kennzeichnung mit gedruckt wird. In beiden Faellen wird der Zaehler fuer das Niveau 2 inkrementiert und fuer die Niveaus 3 bis 7 auf 0 zurueckgesetzt. Der Wert von Hu sollte so gesetzt sein, dass die Ueberschriften in der untersten (niedrigsten) Ebene unnumeriert erscheinen.

Die Verwendung von .HU ist besonders bei der Erstellung von Appendices und anderen Sektionen sinnvoll, die nicht gut in das Numerierungsschema des Gesamtdokumentes {14.2} einzuordnen sind.

4.4. Ueberschriften und Inhaltsverzeichnis

Der Text der Ueberschriften und ihre entsprechenden Seitennummern koennen auch automatisch in das Inhaltsverzeichnis uebernommen werden. Dafuer muessen folgende drei Aktionen durchgefuehrt werden:

- Festlegung im Register Cl, welche Niveauebene der Ueberschriften fuer das Inhaltsverzeichnis verwendet werden.

- Aufruf des .TC-Makros {10.1} am Ende des Dokumentes

- Angabe von -rBn {2.4} in der Kommandozeile

Jede Ueberschrift, deren Niveau kleiner oder gleich dem Wert des Cl-Registers (Contents Level) ist, wird aufgehoben und erscheint spaeter im Inhaltsverzeichnis. Der Standardwert fuer Cl ist 2, d.h. die ersten beiden Ueberschriftsebenen erscheinen automatisch im Inhaltsverzeichnis.

Durch das Sichern von Ueberschriften kann es zur Speicherueberschreitung kommen. Besonders dann, wenn viele Ebenen von Ueberschriften waehrend der Abarbeitung von Displays {7} und Fussnoten {8} gespeichert werden. In solchen Faellen erscheint dann die Fehlerausschrift 'Out of temp file space" {Anhang C}. Zur Vermeidung solcher Faelle gibt es zwei Massnahmen:

- kuerzere Ueberschriften
- Streichen von Ueberschriften der unteren Ebenen aus dem Inhaltsverzeichnis

4.5. Ueberschriften der ersten Niveauebene und die Seitennumerierung in der Art 'Sektion-Seite'

Standardmaessig werden die Seiten am oberen Blattrand sequentiell durchnumeriert. Fuer Dokumente groesseren Umfanges ist es oft empfehlenswert, die Sektionen einzeln in der Art 'Sektion - Seite' zu numerieren. Als Sektionsnummer wird dabei die gegenwaertig erste Niveauebene der Ueberschrift genommen. Diese Art der Seitennumerierung wird durch die Angabe des -rN3 oder -rN5 Flag in der Kommandozeile erreicht {9.9}. Dabei ist zu beachten, dass als Nebeneffekt das Setzen von Ej auf 1 auftritt, d.h., jede Sektion beginnt auf einer neuen Seite. Bei dieser Numerierungsart erscheint die Seitennummer am unteren Blattrand.

4.6. Durch den Nutzer erstellte Makros

Dieser Abschnitt betrifft nur die Nutzer, die ihre Formatierungsmacros selbst definieren moechten. Es gibt folgende Makros:

```
.HX dlevel rlevel Ueberschrift
.HY dlevel rlevel Ueberschrift
.HZ dlevel rlevel Ueberschrift
```

Die Makros .HX, .HY und .HZ sind ein Mittel, durch die der Nutzer die Moeglichkeit erhaelt, den vorangegangenen Mechanismus der Ueberschriftenerstellung zu beeinflussen. Durch die Selbsterstellung dieser Makros kann er sowohl Struktur als auch Behandlungsweise von Ueberschriften veraendern. Der Aufruf dieser Makros erfolgt automatisch durch das Aufrufen des .H-Makros. Wobei .HX und .HY vor dem Erstellen der Ueberschrift aufgerufen werden und .HZ danach. Nach dem Aufruf von .HX wird der benoetigte Raum fuer die Ueberschrift berechnet, wobei die in .HX enthaltenen Informationen (wie z.B. eine .ti-Anforderung) mit beruecksichtigt werden. Anschliessend erfolgt der Aufruf des .HY-Makros. Hier kann der Nutzer bestimmte, fuer die Ueberschriftserstellung benoetigte Angaben, neu festlegen. Alle diese Standardaktionen werden nur durchgefuehrt, wenn diese Makros nicht definiert sind.

Wurden die Makros .HX, .HY oder .HZ durch den Nutzer definiert, werden die darin enthaltenen Angaben zum entsprechenden Zeitpunkt benutzt. Hat der Nutzer das .H-Makro aufgerufen, entsprechen die abgeleiteten Ebenen (dlevel) und die wirkliche Ebene (rlevel) gleich der in dem .H-Aufruf angegebenen Ebene. Ruft der Nutzer das .HU Makro {4.3} auf, ist der dlevel gleich dem Inhalt des HU-Registers (standardmaessig 2) und der rlevel ist 0. In beiden Faellen besteht der Ueberschriftstext aus dem Text des Originalaufrufes.

Bis zu dem Zeitpunkt, wo .HX durch .H aufgerufen wird, hat das H-Makro den Zaehler fuer die gegebene Ueberschriftsebene inkrementiert {4.2.2.5}, eine oder mehrere Leerzeilen (vertikaler Abstand) erzeugt, die der Ueberschrift vorausgehen {4.2.2.1} und das Ueberschriftskennzeichen gespeichert (z.B. eine Zeichenkette bestehend aus Zahlen, Buchstaben und Punkten, die fuer die Kennzeichnung der Ueberschrift benoetigt werden). Wird .HX aufgerufen, koennen ausser den vom Nutzer zugaenglichen Registern und Zeichenketten noch folgende benutzt werden:

string }0 Ist rlevel ungleich 0, enthaelt diese Zeichenkette die Ueberschriftenkennzeichnung. Es werden zwei Leerzeichen (zur Trennung der Kennzeichnung von der Ueberschrift) an die Zeichenkette angehaengt. Ist rlevel gleich 0, ist die Zeichenkette leer.

register ;0 Dieses Register gibt die Art des Abstandes an, der der Ueberschrift folgt {4.2.2.2}. Eine 0 bedeutet, dass die Ueberschrift in den Text uebergeht. Eine 1 bedeutet, dass der Ueberschrift ein Break (jedoch keine

Leerzeile) folgt. Eine 2 bedeutet, dass der Ueberschrift eine Leerzeile (1/2 vertikaler Abstand) folgt.

string } 2 — Ist das Register ;0 auf 0 gesetzt, enthaelt diese Zeichenkette zwei Leerzeichen zur Trennunng einer (in den Text uebergehenden) Ueberschrift vom nachfolgenden Text. Ist das Register ;0 auf ungleich 0 gesetzt, ist die Zeichenkette leer.

register ; 3 — Dieses Register enthaelt einen Justierungsfaktor fuer eine .ne Anforderung, die vor dem Ausdruck der Ueberschrift erscheinen muss. Zu Beginn von .HX hat das Register den Wert 3, wenn dlevel gleich 1 ist und in allen anderen Faellen den Wert 1. Die .ne Anforderung gibt die Anzahl der nachfolgenden Zeilen an: der Inhalt des Registers ; 0 wird fuer die Leerzeilen benutzt (die Haelfte des Wertes fuer den vertikalen Abstand) plus Inhalt des Registers ; 3 als Leerzeilen (die Haelfte als vertikaler Abstand) plus Zeilenanzahl der Ueberschrift.

Der Nutzer kann die Werte der Zeichenkette }0, }2 und das Register;3 innerhalb von .HX aendern.
Im folgenden werden einige Beispiele aufgefuehrt:

a) Aendern des Formats fuer die Kennzeichnung von Ueberschriften der ersten Ebene von der Form n. in die Form n.0

```
.if \\$1=1 .ds}0 \\n (H1.0\(\[
                              [ steht fuer Leerzeichen
..
```

b) Trennen einer in den Text uebergehenden Ueberschrift vom nachfolgenden Text durch einen Punkt und zwei Leerzeichen

```
.if \\nl;0=0 .ds }2 .\(\(
..
```

c) Ausgabe von mindestens 15 Leerzeilen vor Ausdruck einer Ueberschrift der ersten Ebene

```
.if\\$l=1 .nr ;3 15-\\n(;0
..
```

d) Hinzufuegen von 3 zusaetzlichen Leerzeilen vor jeder Ueberschrift der ersten Ebene

```
.if \\ $l = 1 .sp 3
..
```

e) Unterschiedliches Einruecken

```
.in \\ $l *2 -2
..
```

f) Einruecken einer in den Text uebergehenden Ueberschrift der dritten Ebene um 5 Plaetze

```
.if \\ $l = 3 .ti 5n
..
```

Bei der Vergabe der Namen fuer nur zeitweilig benutzte Zeichenketten und Makros innerhalb von .HX sind die unter {14.1} getroffenen Vereinbarungen zu beachten. .HY wird aufgerufen, nachdem eine .ne-Anforderung ausgegeben wurde. Bestimmte Dinge, die in .HX gefordert sind, muessen wiederholt werden.
Beispiel:

```
.de HY
.if \\ $l = 3 .Zi 5n
..
```

Das .HZ-Makro wird am Einde eines .H-Makros aufgerufen, um dem Nutzer ausser einer Veraenderung der Ueberschrift noch weitere Veraenderungen, abweichend vom Standard, zu gestatten. So koennen z.B. in einem umfangreichen Dokument die Sektionen mit den Kapiteln eines Buches uebereinstimmen und der Nutzer moechte den Seitenkopf oder die Fussnote aendern.
Beispiel:

```
.de HZ
.if $L=1 .PF Sektion S3
..
```

4.7. Hinweise fuer grosse Dokumente

Grosse Dokumente sind oft aus Bequemlichkeit so angelegt, dass jede Sektion aus einer Datei besteht. Fuer die Numerierung solcher Dateien ist es ratsam, viele Ziffern zu verwenden, z.B. ist es besser, die Zahlen von 01 bis 02 zu verwenden, als von 1 bis 9 und 10 bis 20.

Oft moechten Nutzer individuelle Sektionen grosser Dokumente formatieren. Um dies mit den richtigen Sektionsnummern durchzufuehren, ist es notwendig, das Register H1 um 1 kleiner anzugeben, als die Nummer der Sektion vor dem entssprechenden . H1-Aufruf. Zum Beispiel ist am Anfang der Sektion 5

```
.nr H1 4
```

einzufuegen.

Achtung:
Da diese Praxis die automatische Numerierung bei Streichungen oder Hinzufuegungen von Sektionen stoert, sind solche Zeilen so bald als moeglich zu

streichen.

4.8. Komplexbeispiel

Eingabestrom

```
 .fi
 .ds HF 3 3 3 2
 .nr Hc 1
 .nr Hb 3
 .nr Hi 0
 .nr Pi 6
 .nr Ps 0
 .nr Hy 1
 .nr Hu 1
 .HM 1 1 a I
 .SA 1
 .H 1 "HEAD 1"
Die Ueberschrift der ersten Ebene erscheint zentriert.
 .H 2 "Head 2"
Hier steht der Text der zweiten Ebene.
 .nr Hi 2
 .H 3 "head 3"
Das Setzen des Numberregister Hb auf 3 bewirkte, dass
diese Ueberschrift der dritten Ebene (im Gegensatz zur
Standardeinstellung) separat ausgegeben wird. Durch das
Numberregister Hi=2 wird die erste Zeile soweit einge-
rueckt, dass sie mit der Ueberschrift uebereinstimmt. Die
Kennzeichnung der dritten Ebene geschieht mittels Klein-
buchstaben - fortlaufend. In Abaenderung zur Standardein-
stellung erfolgt auch die Ausgabe von Ueberschriften der
dritten Ebene im Fettdruck. Fuer diesen Zweck wurde die HF-
Zeichenkette geaendert in:
HF 3 3 3 2
 .nr Hi 0
 .P 1
Das Einruecken entsteht durch Aufruf des Makros .P 1
nachdem zuvor das Numberregister Ps auf 0 eingestellt wur-
de.
Die Standardeinstellung des Hs-Register (Hs 2) erzwingt,
dass der Text ohne Leerzeile an die Ueberschrift ange-
schlossen wurde.
 .HU Achtung
 .DS 1 1 8
Das Wort "Achtung" ist eine unnumerierte Ueberschrift und
wurde aufgrund des Numberregister Hu=1 wie eine Titelzeile
der hoechsten Ebene behandelt, d.h. (in diesem Fall) er-
folgt die Ausgabe zentriert und in jedem Fall ein Weiter-
zaehlen der ersten Ebene. Die Erstellung dieses Abschnittes
wurde mit dem .DS-Makro (.DS 1 1 8) vorgenommen {7.1}.
 .DE
 .H 1 UEBERSCHRIFT
Das ist dritte Ueberschrift auf der ersten Ebene.
 .H 2 Ueberschrift
Text der zweiten Ebene.
```

```
 .H 2 Kopfzeile
Textkoerper einer anderen zweiten Ebene.
 .H 3 Titelzeile
Textteil dieses Abschnittes. Kennzeichnung dieser Titelzeile "a".
 .H 3 Titel
Text eines anderen Abschnittes. Kennzeichnung des Titels "b".
 .H 4 "Ebene 4"
Die Ueberschrift "Ebene 4" geht in den Text ueber und ist nur durch zwei Leerzeichen vom nachfolgenden Text getrennt. Die Kennzeichnung der vierten Ebene erfolgt mit roemischen Zahlen (grossbuchstabig).
 .H 4 "Ebene 4.1"
nachfolgender Text
 .H 4 "Ebene 4.2"
 .H 4 "Ebene 4.3"
```

Ausgabe

1. HEAD 1

Die Ueberschrift der ersten Ebene erscheint zentriert.

1.1 HEAD 2

Hier steht der Text der zweiten Ebene.

1.1.a head 3

Das Setzen des Numberregister Hb auf 3 bewirkte, dass diese Ueberschrift der dritten Ebene (im Gegensatz zur Standardeinstellung) separat ausgegeben wird. Durch das Numberregister Hi=2 wird die erste Zeile soweit eingerueckt, dass sie mit der Ueberschrift uebereinstimmt. Die Kennzeichnung der dritten Ebene geschieht mittels Kleinbuchstaben - fortlaufend. In Abaenderung zur Standardeinstellung erfolgt auch die Ausgabe von Ueberschriften der dritten Ebene im Fettdruck. Fuer diesen Zweck wurde die HF-Zeichenkette geaendert in: HF 3 3 3 2.

Das Einruecken entsteht durch Aufruf des Makros .P 1, nachdem zuvor das Numberregister Ps auf 0 eingestellt wurde. Die Standardeinstellung des Hs-Register (Hs 2) erzwingt, dass der Text ohne Leerzeile an die Ueberschrift angeschlossen wurde.

Achtung

Das Wort "Achtung" ist eine unnumerierte Ueberschrift und wird aufgrund des Numberregister Hu=1 wie eine Titelzeile der hoechsten Ebene behandelt, d.h. (in diesem Fall) erfolgt die Ausgabe zentriert und in jedem Fall ein Weiterzaehlen der ersten Ebene. Die Erstellung dieses Abschnittes wurde mit dem .DS-Makro (.DS 1 1 8) vorgenommen.

3. UEBERSCHRIFT

Das ist dritte Ueberschrift auf der ersten Ebene.

3.1 Ueberschrift

Text der zweiten Ebene.

3.2 Kopfzeile

Textkoerper einer anderen zweiten Ebene.

3.2.a Titelzeile
Textteil dieses Abschnittes. Kennzeichnung dieser Titelzeile "a".

3.2.b Titel
Text eines anderen Abschnittes. Kennzeichnung des Titels "b".

3.2.b.I Ebene 4 Die Ueberschrift "Ebene 4" geht in den Text ueber und ist nur durch zwei Leerzeichen vom nachfolgenden Text getrennt. Die Kennzeichnung der vierten Ebene erfolgt mit roemischen Zahlen (grossbuchstabig).

3.2.b.II Ebene 4.1 nachfolgender Text

3.2.b.III Ebene 4.2

3.2.b.IV Ebene 4.3

5. Listen

Diese Sektion beschreibt die Erstellung von verschiedenen Listenarten, wie z.B.

- Listen mit aus Zahlen gebildetem Kennzeichen
- Listen mit aus Buchstaben gebildetem Kennzeichen
- Bulletin-Listen
- Dash-Listen
- Listen mit selbstgewaehltem Kennzeichen
- Listen, die mit einer Zeichenkette starten, die zuvor zu definieren ist.

5.1. Grundbetrachtung

Im weiteren Text steht der Begriff 'Item' fuer kleine Absaetze oder Unterpunkte, die oft der Uebersicht wegen mit Stabstrichen (-Dash), Plus (+Bullet), Buchstaben, Zahlen oder anderen Zeichen versehen werden und somit eine weitere Unterteilung der schon in {4} beschriebenen Absaetze bzw. Abschnitte vornehmen.

Um Items in einer Liste zu erstellen, ist oft eine sich staendig wiederholende Eingabe von Argumenten notwendig. Mit diesen Argumenten wird der Aufbau dieser Items naeher beschrieben. Durch das MM-Makro-Paket wird dem Nutzer das Erstellen von Listen erleichtert. Alle Listen bestehen grundsaetzlich aus folgenden Teilen:

- Einem Listen-Initialisierungs-Makro, das das aeussere Bild einer Liste angibt, wie z.B. Art der Kennzeichnung (Zahlen, Buchstaben), Zeilenabstand und Einrueckwerte.

- Einem oder mehreren Listen-Item-Makros (.LI). Jedem .LI ist der eigentliche Text nachgestellt..

- Dem Listen-End-Makro (.LE). Dieses beendet die Liste und laedt die zuvor eingestellten Werte zurueck.

Die Listen koennen bis zu fuenf Ebenen tief verschachtelt sein. Das Listen-Initialisierungsmakro rettet den urspruenglichen Status einer Liste (wie z.B. Art der Kennzeichnung, Einrueckwerte usw.). Das .LE-Makro laedt ihn wieder zurueck.
Auf diese Art braucht das Format einer Liste nur einmal zu Beginn definiert zu werden. Es ist jedoch auch moeglich, aufbauend auf diese Struktur, durch den Nutzer eigene Makros muehelos zu erstellen {5.4, Anhang A}.

5.2. Ein Beispiel verschachtelter Listen

Im folgenden Abschnitt wird die Eingabe mehrerer Listen und die daraus resultierende Ausgabe angefuehrt. Die darin enthaltenen .AL und .DL Makro-Aufrufe sind Beispiele von Listen-Initialisierungs-Makros {5..3.3}.

```
Eingabe:
 .AL A
 .LI
Dies ist ein Item mit einem aus Buchstaben gebildeten
Kennzeichen.
Der Text zeigt die Anordnung der zweiten Zeile.
 .AL
 .LI
Dies ist ein Item mit einem aus Zahlen gebildeten Kennzei-
chen.
Der Text zeigt die Anordnung der zweiten Zeile.
 .DL
 .LI
Dies ist ein Dash-Item.
Der Text zeigt die Anordnung der zweiten Zeile.
 .LI + 1
Dies ist ein Dash-Item mit einem "Plus" als Praefix.
Der Text zeigt die Anordnung der zweiten Zeile.
 .LE
 .LI
Dies ist der 2. Item mit einem aus Zahlen gebildeten Kenn-
zeichen.
 .LE
 .LI
Dies ist der 2. Item mit einem aus Buchstaben gebildeten
Kennzeichen.
Der Text zeigt die Anordnung der zweiten Zeile.
 .LE
 .P
Diese Zeile wird linksbuendig ausgegeben.

Ausgabe:
A. Dies ist ein Item mit einem aus Buchstaben gebildeten
   Kennzeichen. Der Text zeigt die Anordnung der zweiten
   Zeile.

   1. Dies ist ein Item mit einem aus Zahlen gebildeten
      Kennzeichen. Der Text zeigt die Anordnung der zwei-
      ten Zeile.

      - Dies ist ein Dash-Item. Der Text zeigt die Anord-
        nung der zweiten Zeile.

     +- Dies ist ein Dash-Item mit einem "Plus" als Prae-
        fix. Der Text zeigt die Anordnung der zweiten
        Zeile.

   2. Dies ist der 2. Item mit einem aus Zahlen gebildeten
      Kennzeichen. Der Text zeigt die Anordnung der zwei-
      ten Zeile.

B. Dies ist der 2. Item mit einem aus Buchstaben gebildeten
   Kennzeichen. Der Text zeigt die Anordnung der zweiten
   Zeile.
Diese Zeile wird linksbuendig ausgegeben.
```

5.3. Allgemeine Listen Makros

Zuerst werden die Listen-Makros betrachtet, die allen Listen gemeinsam sind. Die einzelnen Listen-Initialisierungs-Macros werden dann in {5.3.3} naeher beschrieben.

5.3.1. Listen Item

.LI [Kennzeichnung] [1]
eine oder mehrere Zeilen, die die Liste bilden.

Das .LI Makro wird fuer alle Listen benutzt. Der Aufruf dieses Makros bewirkt die Ausgabe einer Leerzeile (1/2 vertikaler Abstand) zur Trennung seines Textes vom vorangegangenen Text. Durch Setzen des Ls-Registers kann dies aber durch den Nutzer verhindert werden {5..3.3.1}. Wurden beim .LI-Aufruf keine Argumente angegeben, erfolgt die Kennzeichnung des Items entsprechend des zuletzt aufgerufenen Listen-Initialisierungs-Makro. Bei Verwendung nur eines Argumentes wird dieses zur Kennzeichnung benutzt. Wurden beide Argumente beim Aufruf angegeben, erscheint das erste als Praefix vor der aktuellen Kennzeichnung. (Das zweite Argument ist immer 1).Auf diese Weise kann der Nutzer bestimmte Items in einer Liste hervorheben. Zwischen dem Praefix und der Kennzeichnung erscheint zur raeumlichen Trennung immer ein hartes Leerzeichen.
Beispiel:

Eingabe:

```
.BL 6
.LI
Dies ist ein Bulletin-Item
.LI:
Das Bulletin-Zeichen wird durch den Doppelpunkt ersetzt.
.LI + 1
Jetzt erscheint der Doppelpunkt als Praefix zum Bulletin-Zeichen.
.LE
```

Ausgabe:

```
+ Dies ist ein Bulletin-Item.
: Das Bulletin-Zeichen wird durch den Doppelpunkt ersetzt.
:+ Jetzt erscheint der Doppelpunkt als Praefix zum Bulletin-Zeichen.
```

Hinweis:
Das Kennzeichen darf keine gewoehnlichen (Soft)-Spaces enthalten, da die Anordnung des Items bei eingeschalteter rechter Randeinstellung verloren

geht {3.3}.

Ist die Zeichenkette des aktuellen Kennzeichens leer und das erste Argument des LI-Aufrufes ist Null bzw. wurde nicht angegeben, wird die erste Zeile des folgenden Textes ausgerueckt und beginnt auf der Hoehe des (nicht vorhandenen) Kennzeichens {5.3.3.6}

5.3.2. Listen Ende

.LE [1]

Durch Aufruf des .LE Makro erfolgen zwei Aktionen:
- Beenden des zuvor erfolgten Listen-Initialisierungs-Makro-Aufrufes
- Zurueckkehren auf die naechst hoehere Ebene (wenn vorhanden) durch Rueckspeichern der durch den Listen-Initialisierungs-Makro geretteten Werte.

Ist das optionale Argument angegeben, erfolgt durch Aufruf des .LE-Makro die Ausgabe einer leerzeile (1/2 vertikaler Abstand). Auf die Angabe des Argumentes kann aber verzichtet werden, wenn anschliessend ein Makro aufgerufen wird, das seinerseits eine oder mehrere Leerzeilen erzeugt (wie z.B. .P, .H oder .LI).

Da durch anschliessende .H und .HU-Aufrufe saemtliche Kennzeichnungsinformationen geloescht werden, waere es denkbar, abschliessende .LE-Aufrufe wegzulassen. Diese "Einsparung" ist nicht empfehlenswert, da bei eineem spaeteren Einfuegen eines neuen Textes (vor der Ueberschrift des .H-Aufrufes) oftmals uebersehen wird, dass dieser Text dem letzten .LI-Aufruf zugeordnet und entsprechend behandelt wird.

5.3.3. Listen-Initialisierungs-Makro

Im folgenden Abschnitt werden die verschiedenen Initialisierungs-Makros aufgefuehrt.

5.3.3.1. Listen mit einem aus Zahlen oder Buchstaben gebildeten Kennzeichen

.AL [Typ] [Einrueckwert] [1]

Der .AL [Typ] wird zur Erzeugung von Listen verwendet, die entweder mit einem aus Zahlen oder Buchstaben gebildeten Kennzeichen versehen werden sollen. Bei Auslassung aller Argumente erfolgt automatisch eine Kennzeichnung durch Zahlen und der folgende Text wird entsprechend des Li-Registers und natuerlich in Abhaengigkeit seiner Ebene eingerueckt {5.2}. Standardmaessig erfolgt ein Einruecken um 6 Stellen (5 in troff). Der so entstandene Raum wird genutzt fuer ein Space, zwei Zahlen, einen Punkt und zwei

weitere Leerzeichen, die das Kennzeichen vom Textkoerper trennen.

Durch Laden des Numberregister Ls kann die automatische Ausgabe einer Leerzeile, die jedem .AL-Aufruf vorausgeht, ebenenabhaengig unterdrueckt werden. Standardmaessig wird Ls mit dem Wert 6 geladen, was bedeutet, dass die niedrigste Ebene nicht durch eine Leerzeile vom vorangegangenen Text getrennt wird. Das Setzen von

```
.nr Ls 0
```

bewirkt, dass alle Items einer Liste "lueckenlos" nacheinander ausgegeben werden.
Das erste Argument des .AL-Makros gibt die Art der Kennzeichnung an. Folgende Zeichen sind fuer das Argument zulaessig: 1, A, a, I und i. Die daraus resultierende Kennzeichnung ist {4.2.2.5} zu entnehmen.

Achtung

Das dort angefuehrte 0001 Format ist hier nicht gestattet

Bei Nichtangabe (oder bei 0) wird standardmaessig eine 1 als Argument eingesetzt. Das zweite Argument gibt die Raumgroesse der Einrueckung an. Ist das Argument ungleich Null, wird es anstelle des Li-Wertes verwendet und ebenenabhaengig eingerueckt. Ebenenabhaengig heisst, dass der Argumentwert zum aktuellen Einrueckwert, der durch eventuell hoehere Ebenen schon vorgegeben ist, addiert wird. Bei Argumentauslassung erfolgt standardmaessig ein Einruecken um den im Li-Register angegebenen Wert.
Durch Angabe des dritten Argumentes werden die Items nicht durch eine Leerzeile (1/2 vertikaler Abstand) voneinander innerhalb der Liste getrennt. Eine Ausnahme bildet der erste Item, ihm wird in jedem Fall eine Leerzeile vorangestellt. Ein automatisches "Weiterzaehlen" der Kennzeichnung erfolgt, wenn in einer Ebene das gleiche .AL-Makro noch einmal angerufen wird {5.2}.

5.3.3.2. Bulletin-Liste

```
.BL [Einrueckwert] [1]
```

Durch Aufruf des .BL-Makro wird eine Bulletin-Liste erstellt, in der jedem Item als Kennzeichen das Bulletin-Zeichen "+" vorangestellt ist. Das Bulletin-Zeichen ist vom Text durch ein Leerzeichen getrennt. Wurde beim Aufruf kein Einrueckwert angegeben, wird der gesamte Item entsprechend des Pi-Registers eingerueckt (standardmaessig um 5 Plaetze). Bei Angabe des Einrueckwertes ersetzt dieser den Pi-Wert. Bei Aktivierung des .P-Makros aus dem .BL-Makro heraus, erfolgt ein Einruecken der ersten Zeile des erstellten Absatzes in der Weise, dass der Wert des Pi-Registers zum gegenwaertigen Einrueckwert addiert wird.

5.3.3.3. Dash-Liste

.DL [Einrueckwert] [1]

Die Dash-Liste entspricht in allen Punkten der Bulletin-Liste, nur, dass anstelle des Bulletin-Zeichens dem Textkoerper ein Stabsstrich '-' vorausgeht.

5.3.3.4. Liste mit selbstgewaehltem Kennzeichen

.ML Kennzeichen [Einrueckwert] [1]

Das .ML-Makro entspricht den .BL- und .DL-Makros aber mit dem Unterschied, dass der Nutzer eine Kennzeichnung selbst definieren kann. Diese Kennzeichnung kann auch aus mehreren Zeichen bestehen. Der Text wird dem angegebenen Argument entsprechend eingerueckt. Andernfalls erfolgt ein Einruecken um die um 1 erhoehte Stellenzahl des selbstgewaehlten Kennzeichens. Durch Angabe des dritten Argumentes werden die Items nicht durch Leerzeilen voneinander innerhalb einer Liste getrennt. Durch die zuletzt beschriebenen 3 Makros (.BL, .DL, .ML) erfolgt keine automatische Kennzeichenweiterzaehlung. Bei Nichtangabe des Einrueckwertes erfolgt ein Einruecken entsprechend des Pi-Registers (standardmaessig um 5 Stellen).

5.3.3.6. Variable Item-Liste

.VL Text-Einrueckwert [Kennzeichen-Einrueckwert] [1]

Das .VL-Makro liefert selbst keine Kennzeichnung, sondern verwenden die durch das .LI-Makro bereitgestellten. Ihre Benutzung geschieht oft zur Darstellung von Definitionen. Der Kennzeichen-Einrueckwert ist ein Ausdruck fuer den Abstand des Kennzeichens zum gerade aktuellen Einrueckwert. Erfolgt keine Angabe dieses Argumentes wird standardmaessig 0 vorausgesetzt. Der Text-Einrueckwert muss immer angegeben werden und drueckt den Abstand des Textes zum aktuellen Einrueckwert aus. Durch ein drittes Argument wird festgelegt, ob die Items einer Liste durch Leerzeilen voneinander getrennt sind. Im nun folgenden Beispiel wird die Benutzung des .VL-Makro gezeigt.

Eingabe:

```
      .tr~
      .VL 20 2
      .LI Kennz ~1
Es folgt die Beschreibung von Kennz 1;
Kennz 1 in der .LI-Zeile enthaelt eine Tilde (~), die in
ein hartes Leerzeichen umgewandelt wird, um zusaetzliche
Leerzeichen zwischen Kennz und 1 zu vermeiden {3.3}.
      .LI zweites ~Kennz
Dies ist das zweite Kennz. Auch hier wird die Tilde (~) in
```

ein Hardspace zur raeumlichen Trennung umgewandelt.

```
      .LI drittes ~Kennz~laenger~als~Einrueckwert:
```

Dieser Item veranschaulicht die Wirkung einer Kennzeichnung, die laenger als der Text-Einrueckwert ist. Ein Leerzeichen trennt das Kennzeichen vom nachfolgenden Text.

```
      .LI~
```

Dieser Item hat keine Kennzeichnung, da die dem .LI-Makroaufruf folgende Tilde in ein Leerzeichen umgewandelt wird.

```
      .LE
```

Ausgabe:

Kennz 1 Es folgt die Beschreibung von Kennz 1; Kennz 1 in der .LI-Zeile enthaelt eine Tilde (~), die in ein hartes Leerzeichen umgewandelt wird, um zusaetzliche Leerzeichen zwischen Kennz und 1 zu vermeiden {3.3}.

zweites Kennz Dies ist das zweite Kennz. Auch hier wird die Tilde (~) in ein Hardspace zur raeumlichen Trennung umgewandelt.

drittes Kennz laenger als Einrueckwert: Dieser Item veranschaulicht die Wirkung einer Kennzeichnung, die laenger als der Text-Einrueckwert ist. Ein Leerzeichen trennt das Kennzeichen vom nachfolgenden Text.

Dieser Item hat keine Kennzeichnung, da die dem .LI-Makroaufruf folgende Tilde in ein Leerzeichen umgewandelt wird.

Die Tilde im zuletzt angegebenen .LI-Makroaufruf ist unbedingt erforderlich, da andernfalls die erste Textzeile nicht eingerueckt wird. Diese Art von Fehler entsteht immer, wenn einem .VL-Makro ein argumentloses .LI-Makroaufruf folgt.

Eingabe:

```
      .VL 10
      .LI
```

Diese Textzeile wird nicht eingerueckt und beginnt somit am linken Rand. Ab der zweiten Zeile wird der Text um 10 Plaetze eingerueckt.

```
      .LE
```

Ausgabe:

Diese Textzeile wird nicht eingerueckt und beginnt somit am linken Rand. Ab der zweiten Zeile wird der Text um 10 Plaetze eingerueckt.

Hinweis:

Das Kennzeichen darf keine gewoehnlichen (Soft)-Spaces enthalten, da die Anordnung der Items bei

eingeschalteter rechter Randeinstellung verloren geht {3.3}.

5.4. Listen-Beginn-Makro !*!

```
.LB Text-Einrueckwert Kennzeichen-Einrueckwert
 P Typ [Kennzeichen] [LI-Sp] [LB-Sp]
```

In den meisten Faellen werden die eingangs beschriebenen Listen-Initialisierungs-Makros den Anforderungen genuegen. Doch fuer ganz besondere Anwendungsfaelle stellt das MM-Paket dem Nutzer ein Makro zur Verfuegung, durch das er noch mehr Einfluss auf das Anlegen von Listen hat. Dieses spezielle Makro, das .LB-Makro, wird auch von allen Listen-Initialisierungs-Makros benutzt. Das .LB-Makro hat folgende Argumente:
Der "Text-Einrueckwert" spezifiziert den Platz an dem der Text, vom aktuellen Einrueckwert ausgehend, beginnt. Fuer "automatische" Listen wird dieser Wert durch das Li-Register festgelegt. Bei Bulletin- und Dash-Listen erfolgt die Ausgabe des Wertes durch das Pi-Register.

Die Argumente "Kennzeichen-Einrueckwert" und "P" geben an, wo das Kennzeichen vor dem Text erscheint. Das Kennzeichen wird innerhalb eines Feldes angelegt, das durch den "Kennzeichen-Einrueckwert" und den Beginn des eigentlichen Textes begrenzt wird. Im allgemeinen wird das Argument "Kennzeichen-Einrueckwert" mit 0 angegeben.
Ist P=0 wird das Kennzeichen linksbuendig in diesem Feld plaziert. Bei einem Wert fuer P groesser als 0, werden P mal Leerzeichen an das Kennzeichen angehaengt. Die so entstandende Zeichenkette wird unmittelbar vor dem Text angeordnet, wobei der "Kennzeichen-Einrueckwert" ignoriert wird. Das gewuenschte Kennzeichen wird durch die Werte der Argumente "Typ" und "Kennzeichen" bestimmt. Ist "Typ" = 0, wird das Argument "Kennzeichen" direkt als Zeichenkette uebernommen. Ist "Typ" > 0, wird in Abhaengigkeit von "Kennzeichen" ein aus Zahlen oder aus Buchstaben bestehendes Kennzeichen gebildet. Die fuer das Argument "Kennzeichen" zu verwendenden Werte sind 1, 1, a, I und i {5.3.3.1}.

Typ	Kennzeichen	Ergebnis
0	nicht angegeben	erste Zeile wird nicht eingerueckt
0	Zeichenkette	Zeichenkette
>0	nicht angegeben	numerisches Kennzeichen
>0	einer der folg. Werte (1,A,a,I,i)	Zahlen oder Buchstaben-Kennzeichnung mit automatischer Erhoehung

Die unter "Typ" angegebenen sechs moeglichen Werte erzeugen eine unterschiedliche Darstellung des Kennzeichens x.

Typ	Darstellung des Kennzeichens
1	x.
2	x)
3	(x)
4	[x]
5	<x>
6	{x}

Hinweis:
Das Kennzeichen darf keine gewoehnlichen (Soft)-Spaces enthalten, da die Anordnung des Items bei eingeschalteter rechter Randeinstellung verlorengeht {3.3}.

Das Argument LI-Sp steht fuer die Anzahl von Leerzeilen (1/2 vertikaler Abstand) die durch jeden in der Liste vorhandenen .LI-Makro-Aufruf ausgegeben werden. Wurde das Argument nicht angegeben, erfolgt standardmaessig die Ausgabe von einer Leerzeile. Durch Angabe von 0 ist das Erstellen einer Kompaktliste moeglich. Ist das Argument groesser als 0, erfolgt vor der Ausgabe des Kennzeichens durch das .LI-Makro eine .ne-Anforderung fuer die Ausgabe von zwei Leerzeilen.

Das Argument LB-Sp gibt die Anzahl der Leerzeilen (1/2 vertikaler Abstand) an , die durch das .LB-Makro selbst erfolgen. Bei Nichtangabe erfolgt keine Ausgabe von Leerzeilen (Standardeinstellung).

Fuer die Benutzung von LI-Sp und LB-Sp gibt es drei sinnvolle Kombinationen. Im Normalfall sollte LI-Sp auf 1 und LB-Sp auf 0 gesetzt sein. Als Ergebnis erhaelt man vor jedem Item in einer Liste eine Leerzeile. Durch die Beendigung der Liste mit einem .LE 1 Makroaufruf erfolgt am Listenende die Ausgabe einer weiteren Leerzeile.
Im zweiten Fall wird durch das Setzen von LI-Sp auf 0 und LB-Sp auf 1 und durch das Beenden der Liste mit .LE 1 vor dem LB-Makroaufruf und am Ende der Liste je eine Leerzeile ausgegeben.
Sind LI-Sp und LB-Sp auf 0 gesetzt und wird die Liste nur mit .LE abgeschlossen, erhaelt man als Ergebnis eine absolut kompakte Liste ohne jegliche Leerzeilen.

Anmerkung:
Der Anhang A enthaelt die Definitionen des sechs unter 5.3.3 aufgefuehrten Listen-Initialisierungs-Makros. Durch Aenderung dieser Definitionen erhaelt der Nutzer die Moeglichkeit, die Darstellung der Listen zu beeinflussen.

5.5. Komplexbeispiel

Eingabestrom

```
 .nr Hy 1
 .nr Ps 0
 .nr L 72
 .fi
 .SA 1
 .AL A
 .LI
Das  ist  ein .AL-Makro mit  einem  Buchstaben-Kennzeichen.
Dies ist die zweite Zeile des Items.
 .AL
 .LI
Das ist ein .AL-Makro mit einem Zahlen-Kennzeichen. Das ist
die zweite Zeile des Items.
 .DL
 .LI
Das ist ein .DL-Makro. xxxxxxxxx xxxx xxxxx xxxxx xxxxxx
Das war die zweite Zeile
 .BL
 .LI
Das  ist ein .BL-Makro.  Das .BL-Makro ist durch ein +  ge-
kennzeichnet. Es kann aus mehreren Zeilen bestehen.
 .LI * 1
Das ist das zweite .BL-Makro. Der Stern ist Praefix.
 .LE
 .ML?
 .LI
Das ist ein .ML-Makro.  Das selbstgewaehlte Kennzeichen ist
ein  Fragezeichen.  Dieses Makro erscheint auf der gleichen
Ebene wie das .DL-Makro.
 .P 1
Aus dem .ML-Makro heraus erfolgt der Aufruf des  .P-Makros,
wobei  die erste Zeile standardmaessig um 5 Stellen  einge-
rueckt ist.
 .LE
 .LE
 .LE
 .LI
Das  ist das zweite .AL-Makro mit einem Buchstaben-Kennzei-
chen.  Sein Kennzeichen ist der Grossbuchstabe 'B'.
 .BL 10
 .LI
Dies ist ein um 10 Stellen eingeruecktes .BL-Makro.
 .nr Pi 8
 .P 1
Aus  dem .BL-Makro heraus erfolgt der Aufruf des  .P-Makro,
wobei  die erste Zeile in diesem Fall um 8  Stellen  einge-
rueckt ist.
 .LE
 .RL 14 1
 .LI
Das  ist ein .RL-Makro,  das um 14 Stellen eingerueckt  ist
und als erster Item mit einer Leerzeile angeschlossen wird.
```

```
 .LI
Das ist das zweite .RL-Makro, das um 14 Stellen eingerueckt
ist und ohne Leerzeile angeschlossen wird.
 .LE
 .VL 8 2
 .LI #
Das ist ein .VL-Makro,  dessen Text-Einrueckwert 8 und des-
sen Kennzeichen-Einrueckwert 2 ist.
 .LE
 .LE
 .LB 10 2 3 5 a
 .LI
Das ist ein .LB-Makro, dessen Text um 10 Stellen vom linken
Rand eingerueckt ist. Das Kennzeichenfeld beginnt 2 Stellen
vom linken Rand und ist somit 8 Stellen breit. Das Kennzei-
chen "a" erscheint in <> an der fuenften Stelle im Kennzei-
chenfeld.
 .TE
```

Ausgabe

A. Das ist ein .AL-Makro mit einem Buchstaben-Kennzeichen. Dies ist die zweite Zeile des Items.

   1. Das ist ein .AL-Makro mit einem Zahlen-Kennzeichen. Das ist die zweite Zeile des Items.

      - Das ist ein .DL-Makro. xxxxxxxxx xxxx xxxxx xxxxx xxxxxx Das war die zweite Zeile.

        + Das ist ein .BL-Makro. Das .BL-Makro ist durch ein + gekennzeichnet. Es kann aus mehreren Zeilen bestehen.

      * + Das ist das zweite .BL-Makro. Der Stern ist der Praefix.

   ? Das ist ein .ML-Makro. Das selbstgewaehlte Kennzeichen ist ein Fragezeichen. Dieses Makro erscheint auf der gleichen Ebene wie das .DL-Makro.
     Aus dem .ML-Makro heraus erfolgt der Aufruf des .P-Makros, wobei die erste Zeile standardmaessig um 5 Stellen eingerueckt ist.

B. Das ist das zweite .AL-Makro mit einem Buchstaben-Kennzeichen. Sein Kennzeichen ist der Grossbuchstabe 'B'.

     + Dies ist ein um 10 Stellen eingeruecktes .BL-Makro.
       Aus dem .BL-Makro heraus erfolgt der Aufruf des .P-Makros, wobei die erste Zeile in diesem Fall um 8 Stellen eingerueckt ist.

     [1] Das ist ein .RL-Makro, das um 14 Stellen

eingerueckt ist und als erster Item mit einer Leerzeile angeschlossen wird.

[2] Das ist der zweite .RL-Makro, der um 14 Stellen eingerueckt ist und ohne Leerzeile angeschlossen wird.

Das ist ein .VL-Makro, dessen Text-Einrueckwert 8 und dessen Kennzeichen-Einrueckwert 2 ist.

<a> Das ist ein .LB-Makro, dessen Text um 10 Stellen vom linken Rand eingerueckt ist. Das Kennzeichenfeld beginnt 2 Stellen vom linken Rand und ist somit 8 Stellen breit. Das Kennzeichen a erscheint in < > an der fuenften Stelle im Kennzeichenfeld.

6. Makros zur Erstellung spezieller Dokumente

Das MM-Paket kann auch zur Erstellung von Schriften benutzt werden, deren Einband und erste Seite ein von der Norm abweichendes Format aufweisen. Solche speziellen Dokumente sind z.B. Veroeffentlichungen, Mitteilungen, Protokolle, Urkunden, Vereinbarungen und Versionsdokumente. Die hierfuer benoetigten Angaben, wie z.B. Titel, Autor, Datum usw., werden in der gleichen Art und Weise angegeben und nur durch ein Argument unterschieden. Mit den in dieser Sektion beschriebenen Makros ist das Erstellen solcher Dokumente moeglich. Die dabei unbedingt einzuhaltende Aufrufreihenfolge ist in {6.9} angegeben. Werden Einband und eine spezielle erste Seite nicht benoetigt, bzw. sollen den gleichen Aufbau haben, wie der Rest des Dokumentes, kann diese Sektion komplett "ueberblaettert" werden.

6.1. Die Titelzeile

```
.TL [arg 1] [arg 2]
Eine oder mehrere Titelzeilen
```

Die der Titelzeile folgenden Argumente koennen inhaltlich durch den Nutzer selbst bestimmt werden. So waeren z.B. bei einer Patentschrift Registratur-Nummer und Patent-Nummer denkbar. Beide Argumente koennen jeweils aus mehreren Zahlenkolonnen bestehen. Beinhaltet ein Argument Leerzeichen, muss das gesamte Argument in Anfuehrungsstriche eingeschlossen werden. Die nachfolgenden Titelzeilen werden im Fill-Mode {3.1} verarbeitet.
Beispiel:

```
.TL "123, 456" 7812
Pruefanleitung fuer Beisteller
```

Eine .br-Anforderung kann eine Titelzeile in mehrere Zeilen aufspalten. Wenn nur ein Argument angegeben worden ist, erscheint es in der gleichen Zeile und wird nur durch einen Bindestrich vom Titeltext getrennt. Haben beide Argumente den gleichen Wert, werden sie auf der der Titelzeile folgenden Zeile ausgegeben. Ihnen voran steht der Text "Charging and Filing Case". Bei unterschiedlichen Argumenten erfolgt die Ausgabe getrennt auf zwei Zeilen, eingeleitet durch die Texte "Charging Case" und "Filing Case".

6.2. Angabe der Autoren

```
.AU name [Initialien] [Ort] [Abt.] [ext] [Raum]
         [arg 1] [arg 2] [arg 3]
.AT
```

Die Argumente dieses Makros liefern Informationen, die den Autor des Dokumentes betreffen. Bei Angabe der ersten sechs Argumente ist ihre Reihenfolge einzuhalten. Sollte die Aufzaehlung mehrerer Autoren notwendig sein, hat fuer jeden

Autor ein gesonderter .AU-Makro-Aufruf zu erfolgen. Argumente, die Leerzeichen beinhalten, sind in Anfuehrungsstriche zu setzen. Durch das .AT-Makro koennen Titel, Dienststellung oder Funktionen eines Autors angegeben werden. Der Aufruf des .AT-Makros hat unmittelbar hinter dem betreffenden .AU-Makro zu erfolgen. Die durch das .AT-Makro gelieferten Angaben werden durch das .SG-Makro {6.11.1} ausgewertet.

Beispiel:
```
      .AU "Klaus H. Schmidt" KS ZFT WAE H30
      .AT Themenleiter "Winchester Controller"
```
Bei der Ausgabe wird dem Argument "ext" automatisch ein "x" vorangestellt. Die Ausgabe der Argumente [Ort], [Abteilung] [extern] und [Raum] koennen durch Setzen des Registers Au auf 0 verhindert werden (Standardwert 1). Die letzten drei Argumente [arg 1] ... [arg 3] stehen zur freien Verfuegung und erscheinen auf separaten Zeilen.

6.3. Die "Technical Memorandum"-Nummer

Dokumenten vom Typ "Technical Memorandum" kann mit Hilfe des Makros

```
      .TM [Nummer] ...
```

eine Zahlenkennung zugewiesen werden, die maximal aus neun Zahlenkolonnen bestehen darf.

Beispiel:
```
      .TM 7654321   77777777
```

Dieses Makro wird ignoriert bei der Erstellung vom Dokument im "Release-Paper Style" und im "External-Letter-Style".

6.4. Das Abstrakt-Makro

Das Abstrakt-Makro begrenzt einen Text, der fuer die Gestaltung des Deckblatttes angelegt wird.

```
      .AS [arg] [Einrueckwert]
      Text
      .AE
```

Das MM-Paket bietet die Moeglichkeit der Erstellung von Deckblaettern, passend zu den Dokumenten:
- "Technical Memorandum"
- "Memorandum For File"
- "Release-Paper"

Mit Ausnahme fuer "Memorandum For File" ist die Benutzung des Abstrakt-Makros optional.
Auf dem Deckblatt steht der durch .AS/.AE begrenzte Text nach den Angaben ueber den Autor und nach dem zentriert

ausgegebenen Wort "ABSTRACT". Die Art des erzeugten Deckblattes wird durch das erste Argument [arg] und durch das .CS-Makro {10.2} bestimmt. Ist arg = 2 wird automatisch (ohne Aufruf des .CS-Makros) ein Deckblatt der Art "Memorandum For File" erzeugt. Ist arg + 2, wird in Abhaengigkeit des .MT-Types {6.6} und des .CS-Makros das entsprechend andere Deckblatt angelegt. Die Ausgabe von zusaetzlichen Bezeichnungen auf dem "Memorandum For File"-Deckblatt durch die .NS/.NE-Makros {6.11.2} ist gestattet.
Das zweite Argument [Einrueckung] gestattet, den durch das .AS/.AE-Makro begrenzten Text mit einer vom eigentlichen Text abweichenden Randbreite darzustellen. Der Wert ist ein Direktwert (ein Wert von 'ens') und bezieht sich auf beide Raender.
Die Angabe von Ueberschriften und Displays innerhalb der Benutzung des .AS/AE-Makros ist nicht gestattet.

6.5. Andere Schluesselworte

```
.OK [Keyw] ...
```

Durch Verwendung des "Keyword"-Makros ist es moeglich, fuer ein "Technical Memorandum"-Deckblatt bis zu neun Schluesselwoerter oder Wortfolgen anzugeben. Enthalten die Schluesselwoerter Leerzeichen, hat die Angabe in " " zu erfolgen.

6.6. Die unterschiedlichen Dokumenttypen

```
.MT [Typ] [Empfaenger]
```

Durch das MT-Makro wird der obere Teil der ersten Seite eines "Memorandum"- oder "Release-Paper"-Types sowie die Art des Deckblattes beschrieben. Die folgende Tabelle zeigt die moeglichen Werte fuer [Typ] und ihre Bedeutung:

Typ	Bedeutung	
.MT""	-	
.MT 0	-	
.MT	MEMORANDUM_FOR_FILE	x
.MT 1	MEMORANDUM_FOR_FILE	x
.MT 2	PROGRAMMER's_NOTES	x
.MT 3	ENGINEER's_NOTES	x
.MT 4	fuer Versionsdokumente	
.MT 5	fuer spezielles Format	
.MT "Zeichenkette"	Zeichenkette	x

Bei den mit "x" gekennzeichneten Zeilen in der Tabelle erfolgt die Ausgabe des unter "Bedeutung" angegebenen Textes nach dem letzten Argument des .AU-Makros. Das durch den Typ 5 erhaltene spezielle Format zeichnet sich dadurch aus, dass der dem .TL-Makro folgende Titeltext nicht "subject:" vorangestellt ist und dass die Angabe des Titels und die

eventuell vorhandenen .TL-Argumente nicht zentriert erfolgen.
Eine Angabe des zweiten Arguments des .MT-Makros hat zur Folge, dass ab der zweiten Seite und auf allen folgenden Seiten der normale Seitenkopf durch den Namen des Empfaengers ersetzt wird.
Beispiel:

```
.MT 1 "Peter Maier"
```

erzeugt auf der 5. Seite einen Kopfdruck:

```
Peter Maier - 5
```

Zur Bedeutung des zweiten Argumentes beim Typ 4 siehe {6.8}.

Hinweis:
Der beim .MT-Makro-Aufruf angegebene Typ beeinflusst die Ausfuehrung der .TL-, .AU-, .AT- und .AF-Makros.

6.7. Aenderung des Datums

```
.ND Datum
```

Durch Aktivierung des ND-Makros wird der Inhalt der Zeichenkette DT, die normalerweise das aktuelle Tagesdatum enthaelt, auf den Wert des angegebenen Arguments gestellt.

6.8. Moeglichkeiten zur Gestaltung der ersten Seite

```
.AF [Firmenname]
```

Durch Variation des Argumentes des .AF-Makros ist es moeglich, bestimmte Angaben auf der ersten Seite zu erhalten oder zu unterdruecken.
Bei Angabe eines Argumentes erscheint dieses als Name des Betriebes am oberen rechten Blattrand (in Abhaengigkeit vom .MT-Typ!) der ersten Seite. Die Verwendung eines Null-Argumentes " ", bewirkt den Vorschub um mehrere Zeilen fuer ein nachtraegliches Einsetzen der Firmenbezeichnung (z.B. durch Verwendung eines Firmenstempels).
Der Aufruf des .AF-Makros ohne Argumente gestattet den Einsatz von vorgedrucktem Papier. In diesem Fall werden weder Firmenname, noch "subject:", "date" oder "from" ausgegeben.
Die Verwendung des argumentlosen .AF-Makros ist identisch dem Aufruf des -rA1 -Numberregister {2.4}, mit dem Unterschied, dass letzterer zur Aenderung der Zeilenlaenge und/oder des Seitenoffsets benutzt werden muss. Bei der Verwendung von industriell vorgedruckten Formularen ist die Standardeinstellung 5.8 i und 1 i.

6.9. Das Versionsdokument

Eine andere Art der Gestaltung der ersten Seite erhaelt man durch Aufruf des .MT-Makros mit dem Argumentenwert 4.

```
.MT 4 [1]
```

Ein Versionsdokument ist dadurch gekennzeichnet, dass die Titelzeile (angegeben durch das .TL-Makro), der Name des Autors (angegeben im .AU-Makro) und der Firmenname (erhalten durch .AF-Makro) zentriert untereinander erscheinen. Die Verwendung des optionalen zweiten Argumentes ist nur sinnvoll, wenn ein Dokument von mehreren Autoren erarbeitet wurde. In diesem Fall erscheint dann fuer jeden Autor der [Firmenname] bzw. der [Ort].

Weitere Angaben, die z.B. im .AU-Makro-Aufruf enthalten sind, werden bei der Erstellung eines Versionsdokumentes nicht beruecksichtigt.

6.10. Aufrufreihenfolge der "Beginn"-Makros

Werden die unter {6.1 - 6.7} beschriebenen Makros benutzt, muss folgende Aufrufreihenfolge eingehalten werden:

```
(.ND...)
 .TL...
eine oder mehrere Titelzeilen
(.AF...)
 .AU...
(.AT...)
(.TM...)
(.AS...)
eine oder mehrere Textzeilen
(.AE   )
(.NS...)
eine oder mehrere Textzeilen
(.NE   )
(.OK...)
 .MT...
```

Auf die in Klammern ( ) stehenden Makros kann verzichtet werden, wenn die durch sie gelieferten Angaben nicht benoetigt werden. Wurde einmal das .MT-Makro aufgerufen, ist ein wiederholtes Aktivieren der Makros .TL und .AU nicht moeglich, da sie aus Platzgruenden aus der Makrodefinitionstabelle gestrichen wurden.

6.11. Endegestaltung

Fuer den Abschluss spezieller Dokumente stehen mehrere Makros zur Verfuegung, die eine differenzierte Endgestaltung gestatten. Ein Aufrufen dieser Makros wird bei .MT 4 ignoriert.

6.11.1. Die Unterschriftsleiste

```
.FC [Floskel]
```

Standardmaessig wird durch das .FC-Makro "Yours very truly" ausgegeben. Es ist aber auch moeglich, einen anderen Text auszugeben:

```
.FC [Hochachtungsvoll]
```

Bei Benutzung des FC-Makros hat dies vor Aufruf des .SG-Makros zu erfolgen.

```
.SG [arg] [1]
```

Durch ein argumentloses .SG-Makro erscheint der Name des Autors (oder die Namen der Autoren) und die Angabe des .AT-Makros entweder nach dem eigentlichen Text oder bei Verwendung des .FC-Makros nach der 'Floskel'. Die Ausgabe erfolgt zentriert und wird mit 3 Leerzeilen eingeleitet, wodurch Raum fuer eine spaetere Unterschrift gegeben ist. Bei Angabe eines Null-Arguments " " erfolgt anschliessend eine linksbuendige Ausgabe der durch Bindestriche verbundenen .AU-Makro-Argumente [Ort]-[Abt.]-[Initialien]. Sollen die Initialien der schreibenden Sekretaerin ebenfalls erscheinen, sind sie als erstes Argument anzugeben und werden mit Bindestrich an die Initialien des Autors automatisch angehaengt.

Wurde ein Dokument von mehreren Autoren erarbeitet, erfolgt durch Verwendung des zweiten Arguments die Ausgabe aller Autoren in der gleichen Weise wie beim argumentlosen .SG-Makro. Die durch das .AU-Makro gelieferten Angaben werden jedoch nur vom ersten Autor ausgegeben. Wird diese Zeile nicht erwuenscht, sind die entsprechenden Argumente in den zugehoerigen .AU-Makros auszulassen. Sollen im Gegensatz dazu alle Autoren "genau beschrieben" werden, hat die Angabe der Daten manuell nach dem argumentlosen -SG-Aufruf zu erfolgen.

```
.SG
.rs
.Sp - 1v
Textzeilen
```

6.11.2. "Copy to" und andere Bezeichnungen

```
.NS [arg]
eventuell Textzeilen von Bezeichnungen
.NE
```

Bei Benutzung des .NS-Makros hat dies nach Aufruf des .SG-Makros zu erfolgen. Durch Angabe verschiedener Argumente

erfolgt die Bereitstellung von vordefinierten Texten (siehe Tabelle).
Das .NE-Makro beendet alle .NS-Makros.

NS [arg]	Text
.NS" "	Copy to
.NS 0	Copy to
.NS	Copy to
.NS 1	Copy (with att.) to
.NS 2	Copy (without att.) to
.NS 3	Att.
.NS 4	Atts.
.NS 5	Enc.
.NS 6	Encs.
.NS 7	U. S. C.
.NS 8	Letter to
.NS 9	Memorandum to
.NS "Zeichen"	Copy (Zeichen) to

Die .NS und .NE-Makros koennen auch zu Beginn eines Dokumentes benutzt werden, um die Liste der Bezeichnungen auf dem Deckblatt des "Memorandum for File" auszugeben. Dem vorangestellt werden die Aufrufe der Makros .AS 2 und .AE. Fuer Bezeichnungen, die zu Beginn ohne .AS 2 angegeben wurden, erfolgt die Ausgabe am Dokumentende.

6.12. Genehmigungsleiste

```
.AV "K. Ott - Revisor"
```

Der Aufruf dieses Makros erzeugt die Ausgabe von "APPROVED:" (genehmigt, bewilligt, befuerwortet), drei Leerzeilen, eine Linienzeile, den in .AV angegebenen Namen und Datum.

```
     APPROVED:

----------------------------      -----------------
     K. Ott - Revisor             Date
```

6.13. Erzwingen eines Ein-Seiten-Briefes

Ist ein vorangegangener Text (Brief) so lang, dass z.B. die Unterschriftsleiste nicht mehr auf die gleiche Seite passt, ist es zweckmaessig, die Seite durch Benutzung der -rln-Option zu verlaengern. Durch Angabe von -rL90 wird eine 90-zeilige Seite erzwungen.

6.14. Komplexbeispiel

Eingabestrom

```
 .TL 13.03.87 15.07.80
 Memorandum Makro-Paket
 .AF "Kombinat-EAW"
 .AU "PETER MEIER" PM Berlin WAE Tel. 531 R1.6
 .AT "Mitarbeiter F/E"
 .AU "WOLFGANG SCHULZ" WS Leipzig LMAA Tel. 211 R14
 .AT "wiss. Mitarbeiter"
 .MT "Bericht zur LFM 87"
 .FC "Mit freundlichen Gruessen"
 .SG HM 1
 .AV "P. Scharf Abt.Ltr."
 .NS "LIDO & Co"
 .NE
```

Ausgabe (.MT 0)

Kombinat-EAW

subject: Memorandum Makro-Paket
Charge Case 12.03.87
File Case 15.07.80

date: April 4. 1987

from: PETER MEIER
Berlin WAE
R1.6 xTel.531

WOLFGANG SCHULZ
Leipzig LMAA
R14 xTel.211

Mit freundlichen Gruessen

PETER MEIER
Mitarbeiter F/E

Berlin-WAE-PM/WS-

WOLFGANG SCHULZ
wiss. Mitarbeiter

APPROVED:

P. Scharf Abt.Ltr.

Date

Copy (LIDO & Co) to

Ausgabe (MT 4)

Memorandum Makro-Paket

Kombinat-EAW

Mit freundlichen Gruessen

APPROVED:

P. Scharf Abt.-Ltr.

Date

Ausgabe (.MT 5)

Memorandum Makro-Paket
Charge Case 12.03.87
File Case 15.07.80

April 4, 1987

Mit freundlichen Gruessen

PETER MEIER
Mitarbeiter F/E

Berlin-WAE-PM/WS-

WOLFGANG SCHULZ
wiss. Mitarbeiter

APPROVED:

P. Scharf Abt. Ltr.

Date

Copy (LIDO & Co) to

Ausgabe (.MT Bericht zur LFM 1987)

Kombinat -EAW

subject: Memorandum Makro-Paket
Charge Case 12.03.87
File Case 15.07.80

date: April 4, 1987

from: PETER MEIER
Berlin WAE
R1.6 xTel.531

WOLFGANG SCHULZ
Leipzig LMAA
R14 xTel.211

Bericht zur LFM 1987

Mit freundlichen Gruessen

PETER MEIER
Mitarbeiter F/E

Berlin-WAE-PM/WS-

WOLFGANG SCHULZ
wiss. Mitarbeiter

APPROVED:

P. Scharf Abt.Ltr.

Date

Copy (LIDO & co) to

7. Displays

Displays sind Textbloecke, die inhaltlich zusammengehoeren und nicht ueber verschiedene Seiten verteilt werden sollen. Das MM-Makro-Paket stellt zwei Arten von Displays bereit:

- feste Displays (.DS)
- flexible Displays (.DF)

Displays fester Darstellung werden in der Reihenfolge ihrer Eingabe ausgegeben. Daraus folgt, dass die Ausgabe eines Displays auf der gegenwaertigen Druckseite nur erfolgt, wenn der "Restraum" genuegend Platz bietet, um das Display komplett aufzunehmen. Ist dies nicht der Fall, erfolgt ein Seitenvorschub.
Displays flexibler Darstellung sind dadurch gekennzeichnet, dass auch sie auf einer neuen Seite beginnen, wenn auf der vorangegangenen nicht mehr genuegend Platz ist; jedoch mit dem Unterschied, dass der danach eingegebene Text in der Ausgabe dann davor erscheint. Mehrere mit .DF erstellte Texte werden in einer Warteschlange angeordnet, wobei ihre Eingabreihenfolge erhalten bleibt. Im Standardfall werden Displays im "no-fill mode", mit einfachem Abstand und uneingerueckt erstellt. Dies kann aber durch Angabe der Argumente beim Aufruf der Display-Makros durch den Nutzer veraendert werden.
Displays koennen nicht verschachtelt werden und die Benutzung von Ueberschriften (.H oder .HU) {4.2, 4.3} ist nicht gestattet. Ein Aufruf des .P-Makros {4.1} aus dem Display heraus ist erlaubt.
Displays werden in einer vom Textkoerper unterschiedlichen Umgebung verarbeitet (siehe .ev-Anforderung im NROFF-Handbuch).

7.1. Feste Displays

```
.DS [Format] [Fill] [Einrueckwert-R]
eine oder mehrere Textzeilen
.DE
```

Feste Displays beginnen mit dem .DS-Makro und werden durch das .DE-Makro beendet. Bei einem argumentlosen Aufruf erfolgt die Verarbeitung im "no-fill mode", d.h. die Zeilenstruktur der Eingabe bleibt erhalten, ein Einruecken unterbleibt.
Bei Angabe eines Einrueckwertes -R, erfolgt ein Einruecken der Zeilen vom rechten Rand um den angegebenen Wert.
(bei NROFF: Angabe als Direktwert oder behandelt als 'ens')
(bei TROFF: Angabe als Messwert oder im Standardfall wie

Durch das Format-Argument ist es moeglich, den nachfolgenden Text zeilen- oder blockweise zu zentrieren und vom linken Rand her einzuruecken. Die Bedeutung des Format-Argumentes ist nachfolgender Tabelle zu entnehmen.

Format-Argument	Bedeutung

"	kein Einruecken
0 oder L	kein Einruecken
1 oder I	Einruecken
2 oder C	zeilenweises Zentrieren
3 oder CB	blockweises Zentrieren

Die Verwendung des Fill-Argumentes ist nachfolgender Tabelle zu entnehmen:

Fill-Argument	Bedeutung
"	no-fill mode
0 oder N	no-fill mode
1 oder F	fill mode

Nicht angegebene Argumente werden standardmaessig auf '0' gesetzt!
Der durch das Format-Argument (1 oder I) benutzte Einrueckwert (vom linken Rand!) wird durch das Si-Numberregister angegeben (standardmaessig 5).

Obgleich im Normalfall ein linksseitig eingeruecktes Display mit einem standardmaessig eingerueckten Absatz uebereinstimmt {4.1}, werden die Einrueckwerte verschiedenen Registern (Si und Pi) entnommen.

Das Format-Argument 3 (CB) sollte benutzt werden, um mit zentrierten Gleichungen fuer "mark" und "lineup" in den Praeprozessoren eqn (1)/neqn (1) zu arbeiten {7.4}. Die Erstellung des Displays erfolgt blockweise auf Basis der laengsten Zeile, zentriert und linksbuendig.
Im Gegensatz dazu fuehrt die Angabe von 2(C) zu einem Zentrieren jeder einzelnen Zeile. Beim Aufruf und Abschluss des Makros erfolgt standardmaessig jeweils die Ausgabe einer Leerzeile (1/2 vertikaler Abstand). Dies kann durch Setzen des Numberregisters Ds auf 0 unterdrueckt werden.

7.2. Flexible Displays

```
.DF [Format] [Fill] [Einrueckwert R]
eine oder mehrere Textzeilen
.DE
```

Flexible Displays beginnen mit dem .DF-Makro und werden durch das .DE-Makro beendet. Die Bedeutung der Argumente ist analog dem .DS-Makro und ist {7.1} zu entnehmen. Eine gewisse Ausnahme bilden das Einruecken und Zentrieren. Ihre Position wird vom jeweils angegebenen linken Rand neu berechnet, da zwischen Eingabe und Ausgabe des flexiblen Displays eine Aenderung der Randeinstellung stattfinden kann.
Bei Aktivierung und Abschluss des .DF-Makros erfolgt analog zum .DS-Makro die Ausgabe einer Leerzeile (1/2 vertikaler Abstand).

Im Gegensatz zum .DS-Makro werden .DF-Makros nach ihrer Ausfuehrung in eine Warteschlange (queue) eingeordnet und warten dort auf ihre Ausgabe. Nach der Ausgabe der dort in der Reihenfolge ihrer Eingabe angeordneten Displays werden die Displays aus der Warteschlange gestrichen. Bei leerer Warteschlange erfolgt eine sofortige Ausgabe in Abhaengigkeit von der Displaygroesse und der Werte der Numberregister De und Df (siehe nachfolgende Tabellen.)
Befindet sich in der Warteschlange schon mindestens ein flexibles Display, ist eine sofortige Ausgabe nicht moeglich und das neue Display wird in der Warteschlange abgelegt.

Nach einem Seitenvorschub (oder nach Beginn der zweiten Spalte im Zwei-Spalten-Mode) erfolgt die Ausgabe des naechsten Displays, wenn das Df-Register entsprechend geladen wurde und das ausgegebene Display wird aus der Warteschlange gestrichen. Weiterhin erfolgt ein Streichen und Ausgeben aller Displays aus der Warteschlange beim Erreichen von Sektionsende (bei der Seitennumerierung innerhalb der Sektion {9.9}) oder bei Dokumentende. Dies geschieht in jedem Fall, bevor folgende Makros ausgefuehrt werden: .SG-Makro, .CS-Makro oder .TC-Makro.

Die Ausgabe eines Displays auf der gerade "aktuellen" Seite erfolgt nur, wenn:
- der noch freie Platz ausreicht um das Display komplett aufzunehmen;
- bei Displays groesseren Umfanges (d.h. groesser als eine Seite) und wenn noch die Haelfte frei ist.

Werte fuer das De-Register

Wert	Bedeutung
0	Standardwert: Es werden keine besonderen Massnahmen durchgefuehrt
1	Nach Ausgabe jedes flexiblen Displays erfolgt ein Seitenvorschub, so dass auf einer Seite immer nur ein flexibles Display erscheint und ihm kein weiterer Text folgt.

Beachte: Andere Werte fuer De werden wie 1 behandelt.

Werte fuer das Df-Register

Wert	Bedeutung
0	Die Ausgabe von flexiblen Displays erfolgt nicht vor Sektionsende (bei eingeschalteter Sektionsseitennumerierung) oder vor Dokumentende.
1	Ausgabe des flexiblen Displays auf die "aktuelle" Seite, wenn der Platz ausreichend gross ist, sonst erst bei Sektions- oder Dokumentende.

2 Ausgabe des naechsten flexiblen Displays aus der Warteschlange unmittelbar zu Beginn einer neuen Seite oder Spalte (im Zwei-Spalten-Mode).
3 Ausgabe eines flexiblen Displays auf die "aktuelle" Seite, wenn der Platz ausreichend gross ist. Ausgabe eines flexiblen Displays zu Beginn einer neuen Seite oder Spalte.
4 Ausgabe soviel flexibler Displays (mindestens jedoch eines) wie moeglich, die nach einem Seitenwechsel auf eine neue Seite "passen". Dabei ist zu beachten, dass bei gesetztem De 1 jedem Display ein Seitenvorschub folgt.
5 Standardwert: Ausgabe eines neuen flexiblen Displays auf die "aktuelle" Seite, wenn der Platz ausreichend gross ist. Ausgabe mindestens eines (oder mehrerer) Displays zu Beginn einer neuen Seite oder Spalte.

Beachte: Werte groesser 5 werden behandelt wie 5.

Fuer die Erstellung von Displays im Zwei-Spalten Mode ist auch die Benutzung des .WC-Makros {12.4} moeglich.

7.3. Tabellen

```
.TS [H]
globale Optionen
Spalten-Descriptoren
Tabellenname
[.TH [N]]
Tabellendaten
.TE
```

Das .TS-Makro (Tabellen-Start) und das .TE-Makro (Tabellen-Ende) werden vom Praeprozessor tbl (1) benutzt. Sie begrenzen den durch tbl (1) untersuchten Text und definieren den Bereich um die Tabelle herum. Um .TS - .TE - Bloekke, die Tabellen, Gleichungen, Texte (im fill- und no-fill-mode) und Bildunterschriften enthalten, zusammenzuhalten, sollten diese Bloecke innerhalb von Displays (.DS - .DE) eingeschlossen werden. Analog dazu sollten flexible Tabellen innerhalb von flexiblen Displays (.DF -.DE) angelegt werden. Die Displayfunktion und die Begrenzerfunktion sind unabhaengig voneinander.

Das .TS-Makro gestattet auch die Verarbeitung von Tabellen, die sich ueber mehrere Seiten erstrecken. Soll dabei jede Seite mit der Tabellenueberschrift versehen werden, geschieht das durch Angabe des "H"-Argumentes. Es folgen die Optionen, Formatinformationen und der Tabellentitel. Anschliessend muss unbedingt das .TH-Makro aufgerufen werden. Es ist zu beachten, dass das kein Kennzeichen von tbl (1), sondern des durch das MM-Paket definierten Makros ist.

Der Grossbuchstabe 'N' als Argument des .TH-Makros gibt an, dass der Tabellenkopf nur einmal ueber eine Tabelle ge-

druckt wird. Das wird vor allem dann angewendet, wenn eine lange Tabelle aus mehreren kleinen .TS H - .TE Segmenten erstellt wird.
Beispiel:

```
.TS H
globale Optionen
Spalten-Descriptoren
Tabellenname
.TH
Tabellendaten
.TE
.TS H
globale Optionen
Spalten-Descriptoren
Tabellenname
.TH N
Tabellendaten
.TE
```

In dem angefuehrten Beispiel wuerde eine Tabellenueberschrift nur ueber dem ersten Tabellensegment erscheinen, wenn beide Segmente auf der gleichen Seite ausgegeben werden. Auf allen folgenden Seiten, auf denen die Tabellenausgabe fortgesetzt wird, erfolgt eine weitere Ausgabe der Tabellenueberschrift. Sinnvoll ist diese Art der Darstellung auch bei Tabellen, die aufgrund zu vieler Textbloecke zu gross sind. Auch in diesem Fall wuerde jedes .TS H - .TH Segment seine eigene Ueberschrift haben. Wird ab dem zweiten Segment dann .TS H - .TH N benutzt, erscheint eine Tabellenueberschrift nur zu Beginn der Tabelle und dann im Kopfteil jeder neuen Seite (oder Spalte) auf der die Tabelle fortgesetzt wird.

Fuer nroff kann die -e Option (-E fuer mm (1)) {2.1} benutzt werden.

7.4. Gleichungen

```
.DS
.EQ [Name]
Gleichung(en)
.EN
.DE
```

Erfolgt das Erstellen von Gleichungen mit den Praeprozessoren eqn(1) und neqn(1) [6.7] koennen die Makros .EQ (Gleichung-Start) und .EN (Gleichung-Ende) in gleicher Weise wie die .TS- und .TE-Makros fuer tbl (1) als Begrenzer benutzt werden. Im Unterschied zu .TS und .TE muessen .EQ und .EN immer innerhalb eines .DS - .DE Paares erscheinen.

Hinweis:
Werden jedoch .EQ und .EN nur als Begrenzer von linearen Gleichungen verwendet oder um eqn/neqn

Definitionen anzugeben, duerfen .DS - .DE nicht benutzt werden. Andernfalls erscheinen in der Ausgabe ungewollte Leerzeilen.

Das im .EQ-Makro angegebene Argument "Name" kann zur Benennung einer Gleichung benutzt werden. Im Standardfall erscheint der Gleichungsname am rechten Rand, vertikal zentriert bei mehrzeiligen Gleichungen. Durch Setzen des Numberregisters Eq auf 1 erfolgt die Darstellung des Namens links neben der Gleichung. Die Ausgabe der Gleichung erfolgt zentriert, wenn die zentrierte Darstellung (.DS 2 oder .DS 3) benutzt wird. Andernfalls steht die Gleichung in Abhaengigkeit von Eq rechts oder links von ihrem Namen.

7.5. Beschriftung von Bildern, Tabellen, Gleichungen und Ausstellungsstuecken

```
.FG [Titel] [Darst] [Flag]
.TB [Titel] [Darst] [Flag]
.EC [Titel] [Darst] [Flag]
.EX [Titel] [Darst] [Flag]
```

Die angegeben Makros werden innerhalb von .DS - .DE Paaren benutzt, um Bilder oder Zeichnungen (.FG), Tabellen (.TB) und Gleichungen (.EC) zu numerieren (mit automatischer Erhoehung) und mit einem Titel (darueber oder darunter) zu versehen. Die Makros benutzen die Register Fg, Tb, Ec und Ex (siehe {2.4} -rN5 um die Zaehler innerhalb einer Sektion zurueckzusetzen).

Beispiel:
Der Aufruf des .FG-Makros

```
.FG "Blockschaltbild Winchester-Beisteller"
```

erzeugt die Ausgabe:

Figure 1. Blockschaltbild Winchester-Beisteller

Die durch die Makros ausgegebenen Texte sind:

```
.FG Figure ...
.TB TABLE ...
.EC Equation ...
.EX Exhibit ...
```

Die Ausgabe eines einzeiligen Textes erfolgt immer zentriert; andernfalls werden alle Zeilen ausser der ersten soweit eingerueckt, dass sie mit dem ersten Zeichen des Titels uebereinstimmen.
Die Art der Numerierung kann durch die .af-Anforderung geaendert werden. Durch Setzen des Numberregisters Of auf 1, kaann z.B. die Gesamtdarstellung von "Figure 1 Titel" zu "Figure 1 - Titel" geaendert werden.

Das Argument "Darst" wird zur Abaenderung der normalen

Numerierung benutzt. Ist das Flag-Argument ausgelassen oder 0, wird "Darst" als Praefix vor die Nummer gestellt. Ist das Flag =1, erscheint "Darst" als Suffix und ist das Flag =2, wird "Darst" anstatt der Nummer ausgegeben.

Ist das Numberregister N auf 5 gesetzt {2.4}, erfolgt innerhalb der Sektionen eine automatische "Bild"-Numerierung und das Argument "Darst" wird ignoriert.

Hinweis:
Im Gegensatz zu den Tabellenbeschriftungen, die ueber einer Tabelle erscheinen, erfolgt das Beschriften von Bildern, Gleichungen und Ausstellungsstuecken darunter.

7.6. Auslisten von Bildern, Tabellen, Gleichungen und Ausstellungsstuecken

Oft wird vom Anwender gewuenscht, einen Gesamtueberblick ueber saemtliche im Dokument vorhandenen Darstellungen, Tabellen, Gleichungen usw. zu erhalten. Solch eine Auflistung erhaelt man durch Setzen der Numberregister Lf, Lt, Le und Lx auf 1. Die eigentliche Ausgabe erfolgt dann unmittelbar nach Ausgabe des Inhaltsverzeichnisses.

Anmerkung:
Bis auf Le (0) sind standardmaessig alle anderen Numberregister schon 1.

Die Titel dieser Listen sind in den nachfolgenden Zeichenketten enthalten, die hier mit ihren Standardnamen angegeben sind und koennen leicht veraendert werden.

```
.ds Lf LIST OF FIGURES
.ds Lt LIST OF TABLES
.ds Le LIST OF EQUATIONS
.ds Lx LIST OF EXHIBITS
```

8. Fussnoten

Zur Erstellung von Fussnoten stehen dem Nutzer zwei Makros und eine Zeichenkette zur Verfuegung. Die Makros dienen zur Textbegrenzung und zur Gestaltung der aeusseren Form der Fussnoten. Die Zeichenkette dient zur automatischen Numerierung.

8.1. Fussnoten mit automatischer Numerierung

Diese Art der Fussnotenerzeugung geschieht durch das lueckenlose Anhaengen der drei Zeichen "" an das Wort, auf das sich die Fussnote bezieht. Die Kennzeichnung der Fussnote erscheint eine halbe Zeile ueber dem zugehoerigen Text (bei troff in einer kleineren Punktgroesse). Fussnoten mit automatischer Numerierung koennen nicht auf Deckblaettern benutzt werden.

8.2. Erstellung von Fussnoten

```
.FS [Name]
eine oder mehrere Zeilen des Fussnotentextes
.FE
```

Der Beginn einer Fussnote wird durch das Makro .FS (Fussnoten Start) und das Ende durch das Makros .FE (Fussnoten Ende) angezeigt. Ein argumentloses .FS-Makro bezieht sich immer auf ein mit "" gekennzeichnetes Wort im Text. Wird das Argument [Name] angegeben, muss im vorangegangenen Text auch ein Wort mit [Name] gekennzeichnet worden sein. Eine gemischte Verwendung dieser beiden Fussnotenarten ist moeglich. Der zwischen .FS und .FE liegende Text wird immer im Makroaufrufe beinhalten. Diese Art der Fussnoten koennen auch fuer Deckblaetter und Tabellen {7.3} benutzt werden. Der eigentliche Text der Fussnote (in .FS - .FE eingeschlossen) sollte unmittelbar dem Wort folgen, auf das er sicht bezieht (gekennzeichnet durch "" oder [Name] als letztes Zeichen dieser Eingabezeile). In der Praxis wird beiden Kennzeichnungsarten ein undehnbares Leerzeichen " nachgestellt, wenn sich die Fussnote auf das letzte Wort eines Satzes bezieht.

8.3. Aeussere Form der Fussnoten !*!

```
.FD [JW] [1]
```

Durch Variation der .FD-Argumente kann der Nutzer angeben, ob innerhalb der Fussnoten die Silbentrennung eingeschaltet (.hy) oder ausgeschaltet (.nh) ist, ob der Text eingerueckt wird oder nicht und ob die Ausrichtung nach dem rechten Rand eingeschaltet (.ad) oder ausgeschaltet (.na) ist. Weiterhin ist es moeglich, [Name] links- oder rechtsbuendig auszugeben (siehe nachfolgende Tabelle):

JW	Bedeutung
0	.nh .ad Text eingerueckt Name linksbuendig
1	.hy .ad " " " " "
2	.nh .na " " " " "
3	.hy .na " " " " "
4	.nh .na nicht eingerueckt" "
5	.hy .ad " " " "
6	.nh .na " " " "
7	.hy .na " " " "
8	.nh .ad Text eingerueckt Name rechtsbuendig
9	.hy .ad " " " "
10	.nh .na " " " "
11	.hy .na " " " "

Wird fuer [JW] ein Wert > 11 angegeben, wird automatisch 0 eingesetzt. Wurde das erste Argument ausgelassen oder als Nullargument " " angegeben, wird bei nroff automatisch .FD 10 und bei troff .FD angenommen. Dies sind auch die standardmaessigen Werte.

Das zweite Argument zeigt an, ob nach Auftreten einer Ueberschrift der ersten Ebene, die automatische Numerierung der Fussnoten wieder von vorn (mit 1) beginnt. Die Benutzung dieses Argumentes ist vor allem in der Sektionsseitennumerierung sinnvoll.

Fussnoten sind vom Text durch einen halblangen Strich getrennt. Umfangreichere Fussnoten, die auf der folgenden Seite weitergefuehrt werden, sind vom Text durch einen langen Strich getrennt. (In troff erfolgt die Darstellung der Fussnote zwei Punktgroessen kleiner als der uebliche Text).

8.4. Abstand zwischen mehreren Fussnoten

Standardmaessig werden beim Auftreten mehrerer Fussnoten auf einer Seite diese durch eine Leerzeile (drei Punkte vertikal) voneinander getrennt. Der Nutzer kann durch Setzen des Numberregisters Fs einen anderen Abstand definieren.

.nr Fs 2 bewirkt die Ausgabe von zwei Leerzeilen (sechs Punkte ver- tikal) zur Separierung von Fussnoten.

8.5. Komplexbeispiel

Eingabestrom

```
 .fi
 .nr Hy 1
 .nr Pi 6
 .nr Hc 1
 .nr Pt 1
 .HM 1
 .SA 1
 .H 1 "A BICYCLE"*
 .FS  *   Fussnote     mit    selbstgewaehlter     Kennung:
Example   for "English-freaks"
 .FE
 .P 1 They had given their son a bicycle
 .FD 1
 .FS Erste Fussnote mit automatischer  Kennung:  device  to
move on roads
 .FE and   were  watching proudly as he   rode  around  and
around  the  block.   On the  first round he shouted:"Look,
Mom,  no hands." The second  time  around:"Look,   Mom,  no
feet."At the third time: "Look, Mom, no teeth!"
 .FD 5
 .FS Zweite Fussnote mit  automatischer  Kennung:  teeth  =
Zaehne
 .FE
```

Ausgabe

1. A BICYCLE*

They had given their son a bicycle1 and were watching proudly as he rode around and around the block. On the first round he shouted: "Look, Mom, no hands." The second time around: "Look, Mom, no feet." At the third time: "Look, Mom, no teeth!2

---

* Fussnote mit selbstgewaehlter Kennung: Example for "English-freaks"

1. Erste Fussnote mit automatischer Kennung: device to move on roads

2. Zweite Fussnote mit automatischer Kennung: teeth = Zaehne

9. Gestaltung des oberen und unteren Blattrandes

Durch das MM-Paket wird dem Nutzer die Moeglichkeit geboten, ueber bzw. unter jede Seite einen einheitlichen Text zu definieren. Dieser Text wird im Weiteren als "Kopftext" bzw. "Fusstext" bezeichnet. Zur Festlegung eines jeden dieser einzeiligen Texte stehen drei Makros zur Verfuegung:

- fuer einen allgemeinen Text
- fuer einen Text fuer alle geradzahligen Seiten
- fuer einen Text fuer alle ungeradzahligen Seiten

Folglich kann ein "Kopf-" bzw. "Fusstext" aus maximal zwei Zeilen bestehen.

9.1. Standardmaessige Gestaltung

Standardmaessig erscheint nur ueber jeder Seite die zentrierte Seitennumerierung ohne jeglichen Text. Die Seitennumerierung kann durchgaengig sein, oder aber auch mit jeder Sektion neu beginnen {9.9}. Wird das .MT-Makro {6.2} benutzt, erfolgt automatisch ein Unterdruecken der Ausgabe des Textes und der Blattnummer auf der ersten Seite (vorausgesetzt vor Aufruf des .MT-Makros erscheint kein "Break"). Da die unter {6.9} und {9} angegebenen Makros ebenso wie die .nr- und .ds-Anforderungen keinen Break verursachen, ist ihre Benutzung vor dem .MT-Makroaufruf gestattet.

9.2. Gestaltung des oberen Blattrandes

.PH [arg]

Da die Bedeutung des Argumentes auch fuer die nachfolgenden Makros (.EH, .OH, .PF, .EF, OF) identisch ist, erfolgt die Erklaerung nur hier. Das Argument hat die Form:
"'linksbuendig' 'zentriert' 'rechtsbuendig'". Die Angabe von Zeichenketten und Register innerhalb eines Argumentes ist auch gestattet {9.10}. Die Verwendung des Apostrophs (') als Begrenzer ist allerdings nicht empfehlenswert, da es auch innerhalb des Argumentes selbst benutzt werden kann. Es sollte dann einheitlich ein anderes Zeichen als Begrenzer definiert werden. Die Ausgabe erscheint in Abhaengigkeit des Argumentes linksbuendig, zentriert oder rechtsbuendig. Durch Aufruf des .PH-Makros erscheint ueber jeder Seite die Seitennummer (entnommen aus dem P-Register) in arabischen Ziffern. Voran- und nachgestellt wird ein Bindestrich. Das Format der Seitennummer kann durch die .af-Anforderung geaendert werden. Beim Arbeiten im Debug-Mode (-rD1 flag ist in der Kommandozeile gesetzt {2.4}) erscheinen die Debuggerausschriften auf jeder Seite links oben als Bestandteil des standardmaessigen Seitenkopfes. Die Debuggerinformation besteht aus der Versionsnummer des Source Code Control

Systems (SCCS) und der aktuellen Versionsnummer des MM-Paketes {12.3}, gefolgt von der Zeilennummer innerhalb der augenblicklichen Eingabedatei.

9.3. "Kopftext" auf geradzahligen Seitennummern

.EH [arg]

Durch Aufruf des .EH-Makros erfolgt die Ausgabe einer Textzeile am oberen Blattrand aller geradzahligen Seiten. Die Zeile steht unmittelbar hinter der durch das .PH-Makro gelieferten. Standardmaessig ist diese Zeile leer. Bedeutung von [arg] siehe {9.2}.

9.4. "Kopftext" auf ungeradzahligen Seitennummern

.OH [arg]

Die Bedeutung ist analog dem .EH-Makro {9.3} nur mit dem Unterschied, dass die Ausgabe nur auf allen ungeradzahligen Seiten erfolgt.

9.5. Gestaltung des unteren Blattrandes

.PF [arg]

Das .PF-Makro liefert standardmaessig eine Leerzeile, die unter dem Text auf dem unteren Blattrand erscheint. Die Benutzung des -rCn-Flags in der Kommandozeile {2.4} bewirkt die Ausgabe der Art der Kopie auf der dem "Fusstext" folgenden Zeile. Wurde -rC3 oder -rC4 (DRAFT) angegeben, erscheint im "Fusstext" das Datum {6.7}. Die Bedeutung von [arg] ist {9.2} zu entnehmen.

9.6. "Fusstext" auf geradzahligen Seitennummern

.EF [arg]

Durch Aufruf des .EF-Makros erfolgt die Ausgabe einer Textzeile am unteren Blattrand aller geradzahligen Seiten. Die Zeile steht unmittelbar ueber der durch das .PF-Makro gelieferten Zeile. Standardmaessig ist diese Zeile leer. Die Bedeutung von [arg] ist analog {9.2}.

9.7. "Fusstext" auf ungeradzahligen Seitennummern

.OF [arg]

Die Bedeutung ist analog dem .EF-Makro {9.6}, nur mit dem Unterschied, dass die Ausgabe auf allen ungeradzahligen Seiten erfolgt.

9.8. "Fusstext" auf der ersten Seite

Im Standardfall erscheint kein "Fusstext" auf der ersten Seite. Wird vom Anwender jedoch auch auf dieser Seite ein "Fusstext" gewuenscht, muss vor Seitenende im Eingabetext der Aufruf von .PF und/oder .OF erfolgen. Soll auf der ersten Seite anstatt eines "Fusstextes" ein "Kopftext" erscheinen, muss in der Kommandozeile das -rNn-Flag auf 1 gesetzt sein {2.4}.

9.9. "Kopf- und Fusstext" bei der Numerierungsart 'Sektion-Seite'

Wie schon in {4.5} erwaehnt, ist es auch moeglich, Dokumente so durchzunumerieren, dass mit jedem Sektionswechsel die Seitennumerierung von vorn beginnt. Um dies zu erreichen, muss in der Kommandozeile das -rNn-Flag auf 3 oder 5 gesetzt sein. In diesem Fall erscheint kein "Kopftext". Als "Fusstext" wird zentriert die Seitennummer in der Art

9.10. Die Verwendung von Zeichenketten und Registern !*!

Wie schon in {9.2} angegeben, ist es auch moeglich, dass innerhalb eines Argumentes der Name einer Zeichenkette oder eines Registers erscheint. Die zu diesem Namen gehoerige Zeichenkette oder der entsprechende Registerwert werden mit der Ausgabe des entsprechenden "Kopf-" bzw. "Fusstextes" verknuepft. Dem Zeichenketten- oder Registernamen muessen vier Backslashes vorangestellt werden, da der Aufruf dieser Namen dreimal durchgefuehrt wird:

- als Argument eines "Kopf"- oder "Fusstext-Makros
- in einer Formatierungsanforderung innerhalb dieser Makros
- in einem .tl-Aufruf waehrend der Erstellung eines "Kopf"- oder "Fusstextes". Das vierte Backslash dient zur Kennzeichnung von Zeichen- ketten und Registern {1.4}.

Beispiele:

```
1) .PH "' ' ' Seite \\\\nP'"
```

erzeugt einen rechtsbuendigen "Kopftext" mit dem Wort entnommen wird.

```
2) .PF "''-\\\\n(H1-\\\\nP-''"
```

erzeugt einen zentrierten "Fusstext" mit der Numerierungsart 'Sektion-Seite'. Die Sektionsnummer entspricht dem H1-Register, das die Kennzeichnung der hoechsten Ueberschriftsebene enthaelt {4.2.2.5}. Die Seitennummer wird durch das P-Register bestimmt.

3) .PF "'*(a]'''"

erzeugt einen linksbuendigen "Fusstext", mit dem Inhalt der zuvor definierten Zeichenkette 'a]'.

4) .PH "'' Memorandum Makros ''"
   .PF " "
   .EF "'\\\\nP'' Version 3'"
   .OF "'' LIDO & Co''\\\\nP'"

erzeugt den zentrierten Kopftext "Memorandum-Makro". Anstelle eines "Fusstextes" wird eine Leerzeile ausgegeben. Die Ausgabe der aktuellen Seitennummer erfolgt auf geraden Seiten in der linken unteren Ecke, auf ungeraden Seiten in der rechten unteren Ecke. Im angegebenen Beispiel steht auf allen geradzahligen Seiten rechts unten "Version 3" und auf allen ungeradzahligen links unten "LIDO & Co".

9.11. Die "Kopftext" Verarbeitung !*!

Hinweis:
Dieser Abschnitt betrifft nur Nutzer, die ihre Formatierungsmakros selbst definieren moechten.

Fuer die "Kopftext"-Aufbereittung werden vom MM-Paket zwei auch dem Nutzer zugaengliche Makros benutzt. Es handelt sich dabei um das .TP-Makro, das in der "Umgebung" des "Kopftextes" aufgerufen wird (siehe .er-Anforderung) und um das .PX-Makro. Der Aufruf des argumentlosen .PX-Makros erfolgt nach dem Rueckspeichern der "Umgebung" und bei eingeschaltetem "no-space-mode". Standardmaessig geschieht die Einstellung des .TP-Makro (nach der ersten Seite eines Dokumentes) wie folgt :

```
.de TP
.sp 3
.tl \\\*(}t
.if e 'tl \*(}e
.if o 'tl \*(}o
.sp
..
```

Die Zeichenkette }t enthaelt den "Kopftext", die Zeichenkette }e den Text fuer geradzahlige Seiten und die Zeichenkette }o den Text fuer ungeradzahlige Seiten. Die Angabe dieser Texte erfolgt in den Aufrufen der Makros .PH, .EH und .OH. Durch einen "Eingriff" in das .TP-Makro hat der Nutzer die Moeglichkeit, sich mehrere "Kopftext"-Zeilen zu definieren {12.5}. Dabei ist zu beachten, dass das Formatieren innerhalb des .TP-Makros in einer vom Text unterschiedlichen "Umgebung" durchgefuehrt wird.

Beispiel: Um einen "Kopftext" zu erhalten, der aus drei zentrierten Zeilen besteht, die den Namen des Dokumentes, das Erschei- nungsdatum und die Versionsnummer angeben,

kann .TP wie folgt definiert werden:

```
.de TP
.sp
.ce 3
Memorandum Makro
Maerz 1987
Version 3
.sp
..
```

Das .PX-Makro wird fuer die Bereitstellung eines Textes benutzt, der unter dem normalen "Kopftext" einer jeden Seite erscheinen soll. Weiterhin werden durch ihn die Tabstops angegeben, die diesen Text nach den im Textkoerper vorhandenen Spalten ausrichten.

9.12. Die "Fusstext" Verarbeitung

```
.BS
keine oder mehrere Textzeilen
.BE
```

Die zwischen dem .BS- und .BE-Makro angeordneten Textzeilen erscheinen immer linksbuendig als Text am unteren Blattrand nach den (eventuell vorhandenen) Fussnoten aber vor dem eigentlichen "Fusstext".

Hinweis:
Dieser Text erscheint auch auf den Seiten des Inhaltsverzeichnisses und auf dem Deckblatt des "Memorandum for File".

Ein zwischen .BS und .BE definierter Text kann durch die Angabe eines Leerblockes geloescht werden:

```
.BS
.BE
```

9.13. Einstellung des oberen und unteren Blattrandes

```
.VM [Orand] [Urand]
```

Das .VM-Makro gestattet dem Anwender, zusaetzlich ueber dem "Koptext" bzw. unter dem "Fusstext", einen zusaetzlichen Abstand (Leerraum) anzugeben. Die beiden Argumentwerte werden zum schon standardmaessig eingestellten Rand (drei Leerzeilen oben, eine Leerzeile unten {9.11}) hinzuaddiert. Beide Werte muessen immer positive Zahlen sein. Wuenscht der Nutzer die standardmaessig vorgegebene obere bzw. untere Randeinstellung zu verringern, geschieht dies durch Umdefinierung des .TP-Makro {9.11}.

9.14. Spezielle Verwendung des unteren Blattrandes

.PM [Code]

Durch Aufruf des .PM-Makros ist es moeglich, nach dem "Fusstext" einen vorgegebenen Text (siehe Tabelle) noch zusaetzlich auszudrucken.

Code	Bedeutung
.PM	-
.PM N	NOTICE Not for use or disclosure except under written agreement
.PM P	PRIVATE This information should not be disclosed to unauthorized persons. It is meant solely for use by authorized employees.
.PM BP	PROPRIETARY Not for use or disclosure outside except by written approval of the director of the distributing organization
.PM BR	RESTRICTED The information herin is meant solely for use by authorized employees and is not to be disclosed to others.

9.15. Private Dokumente

.nr Pv Wert

Es ist auch moeglich, dass Wort "PRIVATE" zentriert und unterstrichen auf der zweiten Zeile (noch vor dem "Kopftext") in einem Dokument anzugeben (siehe Tabelle)

Wert	Bedeutung
0	- (Standardeinstellung)
1	PRIVATE (nur auf der ersten Seite)
2	PRIVATE (auf allen Seiten)

Hinweis:
Bei Pv 2 kann das .TP-Makro nicht mehr verwendet werden, da es durch das MM-Paket selbst aufgerufen wird.

9.16. Komplexbeispiel

Eingabestrom

```
 .SA 1
 .PH "''HEADER'\*(RE'"
 .OH "'SEITE \\\\nP'''"
 .EH "'''SEITE \\\\nP'"
 .PE "''FOOTER'''"
 .OF "''''Page \\\\nP"
 .EF "'Page\\\\nP'''"
 .PM BR
 .BS

 BEISPIEL fuer: - DISPLAYS {7}
                - Blattandgestaltung {9}
 .BE

Einfuehrung in die Bildschirmaufbereitung mit vi

Kapitel 1
 .H 1 Einleitung

1.1. Allgemeines

vi (visuell) ist ein bildschirmorientiertes interaktives
Textaufbereitungsprogramm, in dem der Bildschirm wie ein
Fenster in die aufzubereitende Datei wirkt. Veraenderungen
werden auf dem Bildschirm dargestellt, wodurch Modifizie-
rungen erleichtert werden. Die Regelmaessigkeit und mnemo-
nische Zuordnung der Befehle erleichtern den Befehlssatz
des traditionelleren, zeilenorientierten Aufbereitungspro-
grammes. ex steht mit vi zur Verfuegung und eine Umschal-
tung zwischen beiden ist leicht moeglich. vi kann an vielen
Bildschirmterminals eingesetzt werden. Neue Terminals wer-
den nach Aufbereitung einer das Terminal beschreibenden
Datei betrieben. Obwohl der Einsatz eines intelligenten
Terminals , mit dem man von einem lokalen Bildschirmgeraet
aus Zeilen und Zeichen einfuegen und loeschen kann, vor-
teilhaft ist, wird auch eine einwandfreie Funktion des
Aufbereitungsprogrammes an Terminals mit Uebertragungslei-
tungen geringer Bandbreite gewaehrleistet. Das Aufberei-
tungsprogramm optimiert die Reaktionszeit, indem ein klei-
neres Fenster und der Algorithmus fuer die Aktualisierung
des Bildschirms angewandt wird. Der Befehlssatz von vi kann
als einzeiliger Fenstereditor fuer Druckerterminals und
Speicherbildroehren benutzt werden.
 .DS 1 1 10 Dieses Dokument wurde in der Annahme
geschrieben, dass das verwendete System ein P8000 ist,
dass die Systemkonsole ein P8000-Terminal ist und dass die
Systemsoftware ein Super- satz von WEGA ist.
 .DE
```

Ausgabe

HEADER 10.108

SEITE 1

Einfuehrung in die Bildschirmaufbereitung mit vi

Kapitel 1

1. Einleitung

1.1. Allgemeines vi (visuell) ist ein bildschirmorientiertes interaktives Textaufbereitungsprogramm, in dem der Bildschirm wie ein Fenster in die aufzubereitende Datei wirkt. Veraenderungen werden auf dem Bildschirm dargestellt, wodurch Modifizierungen erleichtert werden. Die Regelmaessigkeit und mnemonische Zuordnung der Befehle erleichtern den Gebrauch des Befehlssatzes und unterstuetzen die Merkfaehigkeit. Der volle Befehlssatz des traditionelleren, zeilenorientierten Aufbereitungsprogrammes ex steht mit vi zur Verfuegung und eine Umschaltung zwischen beiden ist leicht moeglich. vi kann an vielen Bildschirmterminals eingesetzt werden. Neue Terminals werden nach Aufbereitung einer das Terminal beschreibenden Datei betrieben. Obwohl der Einsatz eines intelligenten Terminals, mit dem man von einem lokalen Bildschirmgeraet aus Zeilen und Zeichen einfuegen und loeschen kann, vorteilhaft ist, wird auch eine einwandfreie Funktion des Aufbereitungsprogrammes an Terminals mit Uebertragungsleitungen geringer Bandbreite gewaehrleistet. Das Aufbereitungsprogramm optimiert die Reaktionszeit, indem ein kleineres Fenster und der Algorithmus fuer die Aktualisierung des Bildschirms angewandt wird. Der Befehlssatz von vi kann als einzeiliger Fenstereditor fuer Druckerterminals und Speicherbildroehren benutzt werden.

> Dieses Dokument wurde in der Annahme geschrieben, dass das verwendete System ein P8000 ist, dass die Systemkonsole ein P8000-Terminal ist und dass die Systemsoftware ein Supersatz von WEGA ist.

1.2. Befehlsdarstellung

Fuer die Beschreibung von Befehlen werden folgende Darstellungen benutzt:

BEISPIEL fuer: -DISPLAYS
-Blattrandgestaltung

herein is meant solely for use by authorized employees and is not to be disclosed to others.

10. Inhaltsverzeichnis und Deckblatt

Durch Aufruf der Makros .TC (Table of Contents) und .CS (Cover Sheet) bietet das MM-Paket die Moeglichkeit des Anlegens eines Inhaltsverzeichnisses und wenn gewuenscht auch eines Deckblattes.

Hinweis:
Die Form des Deckblattes ist dem "Technischen Memorandum" und "Release-Paper" angepasst.
Fuer die Erstellung eines Deckblattes passend zum "Memorandum for File" siehe{6}.

Da die durch beide Makros ausgewiesenen Daten beim Anlegen eines Dokumentes erst automatisch "aufgebaut" werden, ist es sinnvoll, beide Makros nur am Ende des Dokumentes zu aktivieren, z.B. nach dem .SG-Makro {6.11.1} oder nach dem NS-Makro {6.11.2}. Die Reihenfolge ihres Aufrufes ist beliebig.

10.1. Das Anlegen des Inhaltsverzeichnisses

```
.TC [Niveau] [Abst] [Seitnr] [Tab]
    [Text 1] [Text 2] [Text 3] [Text 4]
```

Durch Aufruf des .TC-Makros werden alle die Ueberschriften im Inhaltsverzeichnis ausgewiesen, die entsprechend des Cl-Numberregisters als solche gekennzeichnet wurden {4.4}. Die nachfolgenden Argumente sind optional und haben folgende Bedeutung:

[Abst.] legt die Anzahl der Leerzeilen fest, die jeder Ueberschrift (=Eintragung im Inhaltsverzeichnis) bis zu der in [Niveau] angegebenen Niveauebene vorausgehen. Beide Parameter werden standardmaessig mit 1 geladen, d.h. dass allen Ueberschriften der ersten (hoechsten!) Niveauebene im Inhaltsverzeichnis eine Leerzeile vorangestellt wird (1/2 vertikaler Abstand).

Hinweis:
[Niveau] legt nicht fest, welche Ueberschriften im Inhaltsverzeichnis erscheinen. Dies wird nur durch das Cl-Register bestimmt.

Die Argumente [Seitnr] und [Tab] bestimmen, wo die zur Ueberschrift gehoerende Seitennummer steht und wie sie vom zugehoerigen Text getrennt wird. Es ist moeglich, die Seitennummer rechtsbuendig oder unmittelbar hinter dem Ueberschriftstext auszuweisen. Standardmaessig wird [Seitnr] mit 2 geladen, was bedeutet, dass die Seitennummern der Ueberschriften bis zur zweiten Niveauebene rechtsbuendig angeordnet werden. Ab der dritten Niveauebene erscheinen die zugehoerigen Seitennummern rechts neben der Eintragung, aber durch zwei Leerzeichen getrennt. Durch [Tab] wird angegeben, wie der Raum zwischen Ueberschrift und rechts-

buendiger Seitennummer behandelt wird. [Tab] = 0 (Standard-wert!) bedeutet, dass Eintragung und Seitennummer durch Leerzeichen getrennt sind, [Tab] = 1 bedeutet, dass der Leerraum mit Punkten (.....) gefuellt wird.
Durch Verwendung der [Text]-Argumente ist es moeglich, noch ueber dem Inhaltsverzeichnis einen eigenen zentriert ausgegebenen Text anzuordnen.
Das MM-Paket bietet dem Nutzer die Moeglichkeit, die Kopfgestaltung des Inhaltsverzeichnisses zu bestimmen. Dies geschieht durch die Definition der "Nutzer"-Makros .TX oder TY. Beide Makros sind argumentlos und werden automatisch aus dem .TC-Makro heraus aufgerufen. Wuenscht der Anwender diese Makros zu definieren, muss er allerdings eine Einschraenkung des .TC-Makros in Kauf nehmen, naemlich der Aufruf des .TC-Makros darf hoechstens die ersten vier Argumente beinhalten.
Der Unterschied zwischen beiden "Unter"-Makros (von denen immer nur einer definiert sein darf), besteht darin, dass bei Verwendung des .TY-Makros das standardmaessig erscheinende Wort "CONTENTS" unterdrueckt wird.

Beispiel:

```
 .de TX
 .ce 2
 spezielle Anwendung
 Winchest-Disk-Controller
 .sp 2
 .in + 10 n
 .Berlin: `\'3i'
 .in
 .sp
 ..
 .TC
```

Nach Aufruf des .TC-Makros erhaelt man als Ergebnis:

```
     Spezielle Anwendung
     Winchester-Disk-Controller

Berlin: _____________________________________________
                    CONTENTS
                         .
                         .
                         .
```

Wird .TY als ein leeres Makro definiert, wird ueber dem Inhaltsverzeichnis nur das Wort "CONTENTS" unterdrueckt.

```
 .de TY
 ..
```

Zur Anordnung der Ueberschriften innerhalb des Inhaltsverzeichnisses ist noch zu sagen, dass die Kennzeichnung der naechst niederen Ebene unter dem Text der vorangegangenen steht. Diese Art der Einrueckung kann aber durch Benutzung

der Ci-Zeichenkette abgeaendert werden. Durch diese Zeichenkette koennen, in Uebereinstimmung mit den maximal sieben moeglichen Ueberschriftsebenen, sieben Argumente angegeben werden, aber mindestens soviele wie durch das Cl-Register Ueberschriften ins Inhaltsverzeichnis ueberfuehrt werden.
Die Angabe der Argumente erfolgt in Inches:

Beispiel fuer Cl = 5

```
 .ds Ci .25i .5i .75i 1i 1i
oder
 .ds Ci 0 2n 4n 6n 8n
```

Bei der Erstellung des Inhaltsverzeichnisses sind auch zwei weitere Register von Bedeutung. Das Oc-Register gibt an, ob die Seiten des Inhaltsverzeichnisses mittels kleiner Buchstaben in der roemischen Schreibweise numeriert werden (Standardfall) oder ob keine Seitennumerierung stattfindet (Oc=1) und dafuer aber das P-Register auf 1 gesetzt wird. Durch Verwendung des .PF-Makros {9.10} kann der Nutzer die Plazierung der Seitennummer des Inhaltsverzeichnisses veraendern.
Der Wert des Cp-Registers bestimmt die seitenmaessige Anordnung der Uebersicht der im Dokument verwendeten Tabellen, Bildern usw. Nur wenn das Cp-Register zuvor auf 1 gesetzt wurde, erscheint diese Uebersicht mit auf der Seite des Inhaltsverzeichnisses, sonst erfolgt eine separate Ausgabe.

10.2. Das Deckblatt

```
.CS [Pages] [Other] [Total]
    [Figures] [Tables] [Refs]
```

Durch Aufruf des .CS-Makros erfolgt das automatische Generieren eines Deckblattes. Die Art und Weise, wie das Deckblatt erstellt wird, entspricht entweder dem "Technischen Memorandum" oder dem "Release-Paper". Fuer die Erstellung im "Memorandum_For_File" Stil siehe {6.4}. Die fuer diesen Makro benoetigten Informationen entstammen den Daten vor dem Aufruf des .MT-Makros {6.9}. Wird ein Deckblatt im "Technischen Memorandum-Stil erwuenscht, erscheinen die Angaben automatisch in der unteren linken Ecke des Deckblattes (die Anzahl der Textseiten, die Anzahl der anderen Seiten, die totale Seitenanzahl und die Anzahl der Referenzen). Durch Verwendung der entsprechenden .CS-Argumente koennen diese Werte jedoch veraendert werden. Fuer nicht angegebene Argumente erfolgt eine automatische Berechnung entsprechend der im Dokument verwendeten Seiten. Im "Release Paper" Stil werden alle .CS-Argumente ignoriert.

10.3. Komplexbeispiel

Eingabestrom

Die hier verkuerzt dargestellte Eingabe entspricht der vom Komplexbeispiel 4.

```
 .TL 7.5.87 1.7.80
 TL Titelzeile
 .AF "Confues Ancorp."
 .AU "Klaus Schmitt" KS Berlin ZFT WAE/Tel. 981
 .AT Themenleiter
 .AU "Hans Berger" HB Berlin KEAW LM/Tel. 2632
 .AT Mitarbeiter
 .TM 445566 1234567890 332211
 .AS 1
 Technical Memorandum
 .AE
 .OK "MT 0" bis "MT 3", "MT string"
 .MT
 .FC "Mit freundlichen Gruessen"
 .SG " " 1
 .bp
 .fi
 .ds HF 3 3 3 2
 .nr Hc 1
 .nr C1 3
 .nr Hb 3
 .nr Hi 0
 .nr Pi 6
 .nr Ps 0
 .nr Hy 1
 .nr Hu 1
 .HM 1 1 a I
 .SA 1
 .H 1 "HEAD 1"
 .
 .
 .
 .
 .TC 1 1 3
 .CS
```

Ausgabe

CONTENTS

Cover Sheet for Technical Memorandum

The information contained herein is for the use of employees and is not for publication (see GEI 13.9-3)

Title: TL Titelzeile Date: May 6, 1987

Other Keywords: MT 0 bis MT 3 ,"MT string" TM: 445566 1234567890 332211

Author(s)	Location	Extension	Charging Case: 7.5.87
Klaus Schmitt	Berlin	WAE/Tel.981	Filing Case: 1.7.80
Klaus Berger	Berlin	LM/Tel.2632	

ABSTRACT

Technical Memorandum

Pages Text: 5 Other: 0 Total: 5

No. Figures: 0 No. Tables: 0 No. Refs.: 2

E-1932-U(3-76)SEE REVERSE SIDE FOR DISTRIBUTION LIST

11. Verweise, Bezugnahmen, Referenzen

Das Anlegen von Verweisen, Refenrenzen u.ae. wird im MM-Paket durch zwei Makros unterstuetzt. Fuer das Erstellen von automatisch numerierten Referenzen wird die Zeichenkette *(Rf benutzt. Die Ausgabe aller Referenzen erfolgt normalerweise automatisch am Ende des Dokumentes (vor dem Inhaltsverzeichnis und dem Deckblatt) auf einer (oder mehreren) gesonderten Seite unter der Titelzeile "REFERENCES". Durch Aufruf des optionalen Makros .RP kann der Nutzer die Ausgabe aller bis dahin gemachten Referenzen an beliebiger Stelle innerhalb des Dokumentes erzwingen.

11.1. Referenzen mit automatischer Numerierung

Eine Referenz mit einer automatisch weiterzaehlenden Kennung wird durch das Anhaengen der Zeichenkette *(Rf unmittelbar hinter den Text, auf den sie sich bezieht, erstellt. Bei der Ausgabe wird dann diese Zeichenkette durch die in eckigen Klammern [...] erscheinende Referenznummer ersetzt. In Abhaengigkeit vom implementierten Druckertyp kann die Referenznummer auch eine halbe Zeile ueber dem entsprechenden Text erscheinen.

11.2. Eingrenzen des Referenztextes

Das Begrenzen des Textes erfolgt durch die Makros
.RS und .RF

Beispiel:
```
Textzeile auf die sich die Referenz bezieht.\*(Rf
      .RS [name]
      Referenz-Text
      .RF
```

11.3. Nachfolgende Referenzen

Das im .RS-Makro benutzte Argument verkoerpert den Namen einer Zeichenkette. Dieser Name wird der aktuellen Referenznummer zugewiesen.

Beispiel:
```
      .RS AA
      Referenz-Text
      .RF
```

Durch Aufruf dieser Zeichenkette *(AA ist es moeglich, sich auch spaeter mehrmals im Dokument auf die gleiche Referenz zu beziehen, so dass dafuer kein nochmaliges Definieren des Referenztextes notwendig ist.

11.4. Zusaetzliche Referenzseiten

.RP [arg 1] [arg 2]

Durch Aufruf dieses Makros kann der Nutzer zu jeder Zeit und beliebig oft die Ausgabe aller Referenztexte abfordern (z.B. sektionsweise). Die Bedeutung der beiden Argumente sind der nachfolgenden Tabelle zu entnehmen.

arg 1	Bedeutung
0	Reset fuer Referenzzaehler (Standardfall)
1	kein Reset fuer Referenzzaehler

arg 2	Bedeutung
0	Aufruf des argumentlosen .SK-Makro
1	kein Aufruf des .SK-Makro

Erfolgt kein Aufruf von .RP werden saemtliche Referenzen automatisch am Ende des Dokumentes auf einem Extrablatt unter der Ueberschrift "REFERENCES" ausgegeben. Diese standardmaessige Ueberschrift kann durch Umdefinieren der Zeichenkette Rp vom Nutzer geaendert werden.

Beispiel:
```
 .ds Rp "Referenzen und Verweise"
```

11.5. Komplexbeispiel

Eingabestrom

```
 .fi
 .SA 1
 .nr Hy 1
 .nr Hc 1
 .B
 .HU "Anwendung von Referenzen \*(Rf"
 .R
 .RS R1
Siehe Sektion {11} im MM-Handbuch
 .RF
Referenzen erscheinen oft in Form von Literaturhinweisen
und koennen im Unterschied zu Fussnoten \*(Rf
 .RS R2
Erstellung von Fussnoten siehe Sektion {8} im MM-Handbuch
 .RF
einen groesseren Textumfang haben, da ihre Ausgabe \*(R1
auf einem gesonderten Blatt erfolgt. Weiterhin ist es auch
moeglich, auf eine zu Anfang des Dokumentes gemachte Refe-
renz mehrfach zu verweisen \*(R1. Referenztexte muessen
ebenso wie die Texte von Fussnoten durch spezielle Beginn-
und Endemakros begrenzt werden \*(R2.
```

Ausgabe

Anwendung von Referenzen [1]

Referenzen erscheinen oft in Form von Literaturhinweisen und koennen im Unterschied zu Fussnoten [2] einen groesseren Textumfang haben, da ihre Ausgabe [1] auf einem gesonderten Blatt erfolgt. Weiterhin ist es auch moeglich, auf eine zu Anfang des Dokumentes gemachte Referenz mehrfach zu verweisen [1]. Referenztexte muessen ebenso wie die Texte von Fussnoten durch spezielle Beginn- und Endemakros begrenzt werden [2].

References

1. Siehe Sektion {11} im MM-Handbuch

2. Erstellung von Fussnoten siehe Sektion {8} im MM-Handbuch

12. Nicht zuzuordnende Einzelmakros

12.1. Einstellung des Schriftbildes

Es gibt drei Makros, durch deren Aufruf die Schriftart der Ausgabe fuer nroff und troff bestimmt wird.

```
.B [B-arg] [vorangegangene Schriftart] ...
.I [I-arg] [vorangegangene Schriftart] ...
.R
```

Die Wirkung der Makros bei nroff und troff ist aus der nachfolgenden Tabelle ersichtlich.

	B	I	R
nroff	Fettdruck	unterstreichen	normal
troff	Fettdruck	Kursivdruck	normal

Erfolgt ein Aufruf dieser Makros ohne Argumente, wird ausschliesslich der weitere Text in der entsprechenden Schriftart ausgegeben. Diese Einstellung bleibt solange aktiv, bis ein weiteres Umschalten erfolgt, bzw. bis ein Rueckschalten auf Normalschrift durch .R (Standardeinstellung) erzwungen wird.
Wird bei .B oder .I nur ein Argument ausgegeben, erfolgt die Ausgabe dieses Argumentes in der entsprechenden Schriftart. Anschliessend stellt sich automatisch der zuvor gueltige Schriftsatz ein. Bei Angabe von zwei bis maximal sechs Argumenten werden die jeweils ungeradzahligen Argumente in der dem Makro entsprechenden Schriftart ausgegeben, waehrend die Darstellung der geradzahligen Argumente in der zuvor eingestellten Schriftart lueckenlos angeschlossen wird (1/12 Abstand, wenn die erste Schriftart kursiv ist).

Beispiel:

```
.I Unterstreichung

Text

.I ungerade gerade ungerade gerade
```

Ausgabe

Unterstreichung Text ungerade gerade ungerade gerade

Es ist auch moeglich, diese Makros zu koppeln, so dass ein alternierendes Schriftbild entsteht. Folgende Paarungen sind moeglich:

```
.IB
.BI
.IR
.RI
.RB
```

.BR

Jedem Paar koennen maximal sechs Argumente folgen:

Hinweis:
Die moeglichen Schriftarten bei Ueberschriften sind {4.2.2.4} zu entnehmen.

Bei Benutzung von Ausgabegeraeten, die ein Unterstreichen nicht ermoeglichen, ist es sinnvoll,

.rm ul
.rm cu

einzufuegen.

12.2. Einstellung des rechten Randes

.SA [arg]

Der Aufruf des .SA-Makros bewirkt, dass die Ausrichtung des Textes nicht nur nach dem linkgen Rand, sondern auch einheitlich nach dem rechten Rand erfolgt. Das Ergebnis ist ein optisch sauberes Druckbild. Fuer die Ausrichtung des Textes werden zwei Flags benutzt, deren Stellung durch das Argument von .SA bestimmt wird. Die Flags werden als "aktuelles" Flag und als "Standard"-Flag bezeichnet. Durch .SA 0 werden beide Flags so eingestellt, dass keine Randausrichtung erfolgt, d.h. die Wirkung entspricht dem .na-Aufruf.
SA 1 bewirkt das Gegenteil, beide Flags werden so gestellt, dass der Text nach dem rechten Rand ausgerichtet wird, d.h. die Wirkung ist analog dem .ad-Aufruf.
Wird .SA ohne Argument benutzt, wird das "aktuelle" Flag entsprechend dem "Standard"-Flag gestellt. Die Wirkung dieses Aufrufes haengt dann also von der Standardeinstellung des "Standard"-Flags ab. (nroff: beide Flags stehen auf "Nichtausrichtung"; troff: beide Flags stehen auf Randausrichtung).

Generell kann gesagt werden, die Benutzung des .na-Aufrufes ist sinnvoll fuer die Abschaltung der Randeinstellung, wohingegen fuer die Einschaltung der .SA-Makros dem .ad-Aufruf vorgezogen wird. Auf diese Art ist es moeglich, eine Randausrichtung fuer den Rest des Textes durch Angabe von SA 0 oder .SA 1 einmal zu Beginn des Dokumentes ein- oder auszuschalten.

12.3. Erkennung der SCCS-Version

Die Zeichenkette 'RE' enthaelt die SCCS-Versionsnummer. Diese Nummer kann von Wichtigkeit sein, wenn im MM-Paket Fehler vermutet werden. Durch Verwendung des Numberregister -rDl {2.4} ist es moeglich, die Ausgabe der Zeichenkette 'RE' als Teil des Seitenkopfes staendig zu

erzwingen.

12.4. Zweispaltige Textausgabe

Durch das MM-Paket ist es auch moeglich, einen auszugebenen Text im Zwei-Spalten-Format darzustellen. Dies geschieht durch Angabe von

```
.2C
Text und Formatierungsanweisungen (mit Ausnahme
eines weiteren .2C-Aufrufes)
.1C
```

Durch Aufruf des .1C-Makros erfolgt ein Rueckschalten auf das normale Ein-Spalten-Format. Somit koennen .2C/.1C auch als Begrenzer fuer das Zwei-Spalten-Format betrachtet werden. Die in Sektion {9} beschriebene obere und untere Blattrandgestaltung bleibt unbeeinflusst. Anders verhaelt sich dagegen die Anordnung von, im Zwei-Spalten-Text enthaltenen, Fussnoten {8} und Displays {7}. Standardmaessig werden sie dem jeweiligen Spalten-Format angepasst, aber durch Aufruf des .WC-Makros ist auch eine abweichende Darstellung moeglich. Die Bedeutung der moeglichen .WC-Argumente ist der nachfolgenden Tabelle zu entnehmen.

Argument	Bedeutung
N	Standardeinstellung: -WF, -FF, -WD
WF	Darstellung der Fussnoten erfolgt ueber die gesamte Blattbreite (auch bei aktivem .2C)
-WF	Standardeinstellung: Darstellung der Fussnoten entspricht dem eingestellten Spalten-Format
FF	Darstellung der Fussnoten entsprechen der Anordnung der ersten Fussnote einer Seite
-FF	Standardeinstellung: Darstellung der Fussnoten ist abhaengig von WF/-WF
WD	Darstellung von Displays erfolgt ueber die gesamte Blattseite (auch bei aktivem .2C)
-WD	Standardeinstellung: Darstellung von Displays entspricht dem eingestellten Spalten-Format

Hinweis:
Erfolgt bei einem .WC-Aufruf die Angabe von sich widersprechenden Argumenten, wird das hintere als gueltig betrachtet.

Beispiel:

```
.WC WF -WF
```

hat die Wirkung von

```
.WC -WF.
```

Durch

```
.WC N
```

wird die Standardeinstellung zurueckerhalten.

12.5. Ueberschriften fuer das Zwei-Spalten-Format !*!

Hinweis:
Dieser Abschnitt betrifft nur Nutzer, die ihre Formatierungsmacros selbst definieren moechten.

Fuer Texte, die im Zwei-Spalten-Format ausgegeben wurden, ist es moeglich, Ueberschriften diesem Format anzupassen, oder aber ueber die gesamte Breite darzustellen. Dazu wird, wie in {9.11} das .TP-Makro durch Nutzereingriff veraendert.

Beispiel:
```
 .de TP
 .sp 2
 .tl \\nP'Ueberblick''
 .tl ''TITEL''
 .sp
 .nf
 .Za 16C 31R 34 50C 65R
links -- mitte -- rechts -- links -- mitte -- rechts
     (-- bedeutet TAB-Zeichen)
-- erste Spalte ------zweite Spalte
 .fi
 .sp 2
 ..
```

Durch das oben angefuehrte Beispiel wird ein zweizeiliger Seitenkopf ueber den Gesamttext und ein zweizeiliger Text ueber jeder Spalte erzeugt. Die TAB-Stops sind fuer eine Zeilenlaenge von 65-Zeichen angegeben.

12.6. Vertikaler Abstand

.SP [Zeilen]

Fuer die Erzeugung des vertikalen Zwischenraumes gibt es zwei unterschiedliche Moeglichkeiten:

- Durch Angabe der .sp-Anforderung bei nicht eingeschaltetem .ns-Mode (no space-Mode). Der .ns-Mode wird sinnvoller Weise am Ende des Seitenkopfes eingeschaltet, um unnoetigen Zeilenvorschub (durch .sp oder .bp-Anforderungen) zu Beginn einer neuen Seite zu vermeiden. Die Abschaltung erfolgt durch die .rs-Anforderung (restore spacing).

- Durch Benutzung des .SP-Makros wird verhindert, dass bei mehreren hintereinanderliegenden Makro-Aufrufen ein zu grosser vertikaler Zwischenraum entsteht, denn in solch einem Fall werden die im .SP-Aufruf angegebenen Argumente ignoriert und nur die Anzahl der .SP-Aufrufe kumuliert.

Beispiel:
```
 .SP 2
```

```
 .SP 3
 .SP
```
erzeugt einen Zwischenraum von 3 Zeilen. Aus diesem Grund benutzen viele Makros den .SP-Makro. So z.B. auch das .LE-Makro bei Angabe von .LE 1 {5.3.2}. Wird diesem Makro-Aufruf unmittelbar ein .P-Makro-Aufruf nachgestellt, erfolgt trotzdem nur die Ausgabe einer Leerzeile (1/2 vertikaler Abstand). Fuer die Benutzung des .SP-Makros gilt: Negative Argumente sind nicht erlaubt, wird kein Argument angegeben, erfolgt standardmaessig die Ausgabe von einer Leerzeile (ein vertikaler Abstand). Bei eingeschaltetem no-space-Mode werden die .SP-Makro-Aufrufe ignoriert.

12.7. Das Ueberspringen von Seiten

```
.SK [Seiten]
```

Durch Benutzung des .SK-Makros ist es moeglich Seiten zu ueberspringen, wobei aber die obere bzw. untere Blattrandgestaltung {9} nicht beeinflusst wird. Wurde fuer das Argument ein " ", 0 oder garnichts angegeben, erfolgt ein Vorruecken bis zum naechsten Blattanfang, wenn dieser nicht schon erreicht war. D.h. also durch .SK " "wird der nachfolgende Text immer an den Blattanfang positioniert, waehrend durch .SK 1 eine Leerseite mit dem entsprechenden Kopf- oder Fussteil erzeugt wird.

12.8. Das Erzwingen einer ungeraden Seitennummer

```
.OP
```

Durch Angabe des .OP-Makros wird gesichert, dass die Ausgabe des nachfolgenden Textes immer oben auf einer ungeraden Seitennummer erfolgt. Steht der aktuelle "Zeilenzaehler" schon oben auf einer ungeraden Seitennummer, geschieht keine "Papierbewegung". In allen anderen Faellen erfolgt mindestens der Vorschub um eine Seite.

12.9. Setzen der Punktgroesse und des vertikalen Abstandes bei troff

Die Standardeinstellungen der Punktgroesse in troff ist 10 (enthalten im Numberregister S {2.4}) und fuer den vertikalen Abstand 12 (d.h. 6 Zeilen pro Inch). Durch Aufruf des S-Makros koennen diese Werte veraendert werden

```
.S [Punktgroesse] [vertikaler Abstand].
```

Beide Argumente koennen durch die Angabe eines mnemonischen Codes gesetzt werden. Es bedeuten:

- D = Standardwert
- C = aktueller Wert
- P = vorangegangener Wert

Die Angabe der Argumente kann vorzeichenbehaftet erfolgen, was dann bedeutet, dass der augenblickliche Wert inkrementiert oder dekrementiert wird. Vorzeichenlose Argumente werden als Direktwert uebernommen.
Ein argumentloser .S-Makro-Aufruf erzwingt standardmaessig die Verwendung der vorangegangenen Werte (P). Wurde nur das erste Argument bei einem Aufruf angegeben, wird fuer das zweite Argument (vertikaler Abstand) der Standardwert (D) angenommen. Der Standardwert fuer den vertikalen Abstand ist immer 2p groesser als die eingestellte aktuelle Punktgroesse.

Hinweis:
Fussnoten {8} werden zwei Punktgroessen kleiner als der Gesamttext dargestellt. Die Trennung der einzelnen Fussnoten untereinander erfolgt mit einem vertikalen Abstand von drei Punktgroessen.

Die Angabe eines Nullargumentes (" ") impliziert den Einsatz des aktuellen Wertes (C).

Beispiele:
```
 .S       = .SPP
 .S" "n   = .SCn
 .Sn " "  = .SnC
 .SN      = .SnD
 .S" "    = .SCD
 .S" "" " = .SCC
 .Snn     = .Snn
```
(n ist ein numerischer Wert)

Werden fuer die Argumente Werte groesser 99 verwendet, wird fuer die Punktgroesse der Standardwert (D) 10 und fuer den vertikalen Abstand der Standardwert (D) + 2p angenommen.

Beispiele:
```
 .S 12 111 = .S 12 14
 .S 110    = .S 10 12
```

12.10. Erzeugen von Akzenten

Der nachfolgenden Tabelle kann entnommen werden, welche Zeichenkette zur Erzeugung von Akzents unter troff benutzt werden koennen:

	Eingabe	Ausgabe
Grave (Gravis)	a*`	`a
Acute (Akut)	a*'	a'
Circumflex (Zirkumflex)	a*^	a^
Tilde	n*~	n~
Cedilla	c*,	c,
Lower-case Umlaut	a*:	a'
Upper-case Umlaut	A*;	A'

12.11. Einfuegen von Text

.RD [Prompt] [D-name] [Z-name]

Durch Aufruf des .RD-Makros wird die momentane Ausgabe unterbrochen und Text von der Standardeingabe gelesen. Das Lesen der Standardeingabe wird bei Erkennen von zwei aufeinanderfolgenden Newlines bzw. Newline und CTRL-D beendet und die Ausgabe wird fortgesetzt. Es ist sinnvoll, den Text in no-fill-Mode (.nf) einzugeben.
Das erste Argument erscheint als Eingabeaufforderung auf dem Terminal. Wurde kein [Prompt] angegeben, fordert das Terminal mittels eines akustischen Signals zur Eingabe auf. Unter dem Namen des zweiten Argumentes (ein Diversion name) wird der gesamte eingegebene Text abgespeichert. Das dritte Argument (ein Zeichenkettenname) speichert die erste, dem Prompt folgende Zeile.

Beispiel:

```
 .RD Name aa bb
```

als Prompt erscheint auf dem Terminal

Name: (jetzt koennte nachstehende Eingabe erfolgen) K. May
Hauptstrasse 1,
Berlin 1100

Der Diversionsname aa enthaelt:

K. May
Hauptstrasse 1,
Berlin 1100

Die Zeichenkette "*(bb" enthaelt:
"K. May"

12.12. Komplexbeispiel

Eingabestrom

```
 .nr Hy 1
 .fi
 .SA 1
 .2C
 .HU P8000-Grundgeraet
Die Zentraleinheit des Programmier- und Entwicklungssystems
P8000 ist in einem Kompaktgehaeuse untergebracht. Eine
Kartenbaugruppenaufnahme dient innerhalb des P8000-Grundge-
raetes zur mechanischen Fixierung von zwei durch Flachband-
kabelstecker miteinander verbundenen Einzelleiterplatten
mit dem 8- und 16-Bit-Mikrorechnerteil.
 .P
Auf der Leiterplatte des 16-Bit-Mikrorechnerteils befinden
sich fuenf 64-polige Steckverbinder, die zur Aufnahme von
Speicherbaugruppen mit 64-KBit-DRAM-Speicherschaltkreisen
dienen. Auf jeder dieser einzeln steckbaren Speicherbau-
gruppen ist ein 256-KByte-DRAM-Speicherbereich mit Pari-
taetsbitfehlerueberwachung untergebracht.
 .P
Die nicht mit Speicherbaugruppen belegten 64-poligen Steck-
verbinder auf dem 16-Bit-Mikrorechnerteil koennen zur Auf-
nahme zusaetzlicher Ein-/Ausgabeerweiterungsbaugruppen ge-
nutzt werden.
 .P
Neben den Elektronikbaugruppen befinden sich im P8000-
Grundgeraet zwei 5 1/4 Zoll Floppy-Disk-Laufwerke mit einer
Speicherkapazitaet von jeweils bis zu 640 KByte und eine
kompakte Stromversorgungseinheit.
 .P
Das P8000-Grundgeraet ist standardmaessig mit acht seriel-
len und vier parallelen Interfaceschnittstellen ausgestat-
tet, die zur Ankopplung von Terminalarbeitsplaetzen, Hard-
Disk-Beistellgeraeten, Druckern, In-Circuit-Emulatoren,
EPROM-Programmiermodulen u.a.m. dienen koennen.
 .P
Im einzelnen haben die beiden Mikrorechnerteile folgenden
Aufbau.
 .P
8-Bit-Teil mit UA880 (4 MHz)
 .DL
 .LI
2 EPROM-Fassungen fuer 2716, 2732 oder 2764
 .LI
8 U264 (64KByte-DRAM)
 .LI
4 U214 (1Kx4-SRAM)
 .LI
2 UA856-SIO
 .LI
1 UA855-PIO
 .LI
1 UA858-DMA
```

```
 .LI
1 U8272-FDC
 .LE
 .P
16-Bit-Teil mit UB8001 (4 MHz)
 .DL
 .LI
4 EPROM-Fassungen fuer 2716, 2732 oder 2764
 .LI
4 U214 (1Kx4-SRAM)
 .LI
3 UB8010-MMU
 .LI
2 UA856-SIO
 .LI
1 UA855-PIO
 .LI
5 Steckverbinder fuer  Speicher- und  Ein-/Ausgabeerweite-
  rungskarten
 .LE
```

Ausgabe

P8000-Grundgeraet

Die Zentraleinheit des Programmier- und Entwicklungssystems P8000 ist in einem Kompaktgehaeuse untergebracht. Eine Kartenbaugruppenaufnahme dient innerhalb des P8000-Grundgeraetes zur mechanischen Fixierung von zwei durch Flachbandkabelstecker miteinander verbundenen Einzelleiterplatten mit dem 8- und 16-Bit-Mikrorechnerteil.

Auf der Leiterplatte des 16-Bit-Mikrorechnerteils befinden sich fuenf 64-polige Steckverbinder, die zur Aufnahme von Speicherbaugruppen mit 64-KBit-DRAM-Speicherschaltkreisen dienen. Auf jeder dieser einzeln steckbaren Speicherbaugruppen ist ein 256-KByte-DRAM-Speicherbereich mit Paritaetsbitfehlerueberwachung untergebracht.

Die nicht mit Speicherbaugruppen belegten 64-poligen Steckverbinder auf dem 16-Bit-Mikrorechnerteil koennen zur Aufnahme zusaetzlicher Ein-/Ausgabeerweiterungsbaugruppen genutzt werden.

Neben den Elektronikbaugruppen befinden sich im P8000-Grundgeraet zwei 5 1/4 Zoll Floppy-Disk-Laufwerke mit einer Speicherkapazitaet von jeweils bis zu 640 KByte und eine kompakte Stromversorgungseinheit.

Das P8000-Grundgeraet ist standardmaessig mit acht seriellen und vier parallelen Interfaceschnittstellen ausgestattet, die zur Ankopplung von Terminalarbeitsplaetzen, Hard-Disk-Beistellgeraeten, Druckern, In-Circuit-Emulatoren, EPROM-Programmiermodulen u.a.m. dienen koennen.

Im einzelnen haben die beiden Mikrorechnerteile folgenden Aufbau.

8-Bit-Teil mit UA880 (4 MHz)

- 2 EPROM-Fassungen fuer 2716, 2732 oder 2764
- 8 U264 (64KByte DRAM)
- 4 U214 (1Kx4-SRAM)
- 2 UA856-SIO
- 1 UA855-PIO
- 1 UA858-DMA
- 1 U8272-FDC

16-Bit-Teil mit UB8001 (4 MHz)

- 4 EPROM-Fassungen fuer 2716, 2732 oder 2764
- 4 U214 (1Kx4-SRAM)
- 3 UB8010-MMU
- 2 UA856-SIO
- 1 UA855-PIO
- 5 Steckverbinder fuer Speicher- und Ein-/Ausgabeerweiterungskarten

13. Fehler und Verschwinden einer Ausgabe

13.1. Handlungen bei Auftreten eines Fehlers

Wird durch einen Makro ein Fehler festgestellt, laufen folgende Handlungen ab:
- Ausgabe eines Breaks
- Untersuchen des Fehlerortes und Ausgabe des Ausgangspuffers des Formatierungsprogrammes
- Ausgabe einer Fehlermeldung, der zu entnehmen ist, welches Makro den Fehler entdeckte, Typ des Fehlers und die ungefaehre Zeilennummer in der aktuellen Eingabedatei (Fehlermeldungen siehe Anhang).
- Beenden des Programmes wenn das Register D {2.4} keinen positiven Wert hat. Hat das D-Register einen positiven Wert, wird die Abarbeitung selbst dann fortgesetzt, wenn sicher ist, dass der Programmlauf unsinnig ist.

Hinweis:
Die Fehlermeldung erscheint unmittelbar auf dem Terminal des Nutzers. Wird jedoch ein Ausgabefilter wie z.B. 300 (1) oder 450 (1) benutzt, kann es sein, dass die eigentliche Fehlermeldung durch "Vermischen" mit dem im Ausgabepuffer befindlichen Text verstuemmelt erscheint. Zu einer harmlosen "Broken-Pipe"-Nachricht kann es kommen, wenn tbl (1), eqn (1)/neqn (1) in Zusammenhang mit der -olist Option des Formatierungsprogrammes benutzt wird (letzte Dokumentseite wird nicht ausgedruckt).

13.2. Abhandenkommen der Ausgabe

Ein Verschwinden der Ausgabedatei tritt auf, bei fehlenden Begrenzern .DE oder .FE. Gluecklicherweise ueberpruefen die Makros, die diese Begrenzer benutzen, die Eingabedatei auf unerlaubte Verschachtelungen. Fehlen diese Endbegrenzer, wird das Programm rueckwaerts nach den entsprechenden .DF, DS und .FS durchsucht.
Das folgende Kommando

```
grep -n "^\.[EDFT] [EFNQS]" Dateien ...
```

druckt alle .DS, .DF, .DE, .FS, .FE, .TS, .TE, .EQ und .EN Makros aus, die in 'Dateien' gefunden wurden. Jedem von diesen Makros wird sein Dateiname und die Zeilennummer vorangestellt. Mit Hilfe dieser Angabe ist es dann moeglich, die illegale Verschachtelung ausfindig zu machen.

14. Erweiterung und Modifizierung der Makros !*!

14.1. Festlegungen ueber die Vergabe von Namen

Fuer die folgende Sektion gelten folgende Festlegungen fuer die Vereinbarung von Namen:

n	=	Zahl
a	=	Kleinbuchstabe
A	=	Grossbuchstabe
x	=	Zahl oder Buchstabe (ein beliebiges alphanumerisches Zeichen)
s	=	Sonderzeichen (ein beliebiges nichtalphanumerisches Zeichen)

Alle anderen Zeichen sind Literale.

Hinweis:
Da Namen von Anforderungen, Makros und Zeichenketten von den Formatierungsprogrammen in einer gemeinsamen internen Tabelle gespeichert werden, ist eine Doppelverwendung nicht erlaubt. Registernamen werden in einer gesonderten Tabelle abgespeichert.

14.1.1. Namen, die von Formatierungsprogrammen benutzt werden

Anforderungen: aa (gebraeuchlichste Form)
an (tritt gegenwaertig nur einmal auf: .c2)

Register aa (normal)
.x (normal)
.s (tritt gegenwaertig nur einmal auf: .$)
% (Seitennummer)

14.1.2. Namen, die vom MM-Paket benutzt werden

Makros AA (gebraeuchlichste Form, fuer den Nutzer zugaenglich)
A (seltenere Form, fuer den Nutzer zugaenglich)
)x (intern, konstant)
>x (intern, dynamisch)

Zeichenketten AA (gebraeuchlichste Form, fuer den Nutzer zugaenglich)
A (seltenere Form, fuer den Nutzer zugaenglich)
]x (intern, wird im allgemeinen zur Beschreibung von Funktionen verwendet)
}x intern, dynamische Verwendung)

Register Aa (gebraeuchlichste Form, fuer den Nutzer zugaenglich)

An (fuer den Nutzer zugaenglich)
A (durch den Nutzer erreichbar, aber das Setzen erfolgt in der Kommandozeile)
:x (meist interner Gebrauch, selten direkt erreichbar)
;x (intern, dynamisch, temporaer)

14.1.3. Namen, die von EQN/NEQN und TBL benutzt werden

Die Praeprozessoren fuer Gleichungen eqn (1) und neqn (1) benutzen fuer Register und Zeichenketten Namen der Form der Formen:

```
a- a+ a| nn #a ## #- #^ ^a T& TW
```

14.1.4. Namen, die von den Nutzern festgelegt werden koennen

Um von vornherein allen Problemen aus dem Weg zu gehen, sollten Nutzer Namen vergeben, die entweder aus einem einfachen Kleinbuchstaben oder aus einem Kleinbuchstaben, gefolgt von einem beliebigen Zeichen (mit Ausnahme eines weiteren Kleinbuchstabens), bestehen sollten. Die folgenden Beispiele zeigen einige Namensvereinbarungen:

Makros	: aA Aa
Zeichenketten	: a a), a], a} usw.
Register	: a aA

14.2. Erweiterungen

Die nachfolgenden Abschnitte erklaeren das Formatieren von Appendixuerberschriften und "haengenden" Erweiterungen.

14.2.1. Appendixueberschriften

Das aufgefuehrte Beispiel zeigt eine Moeglichkeit zur Erzeugung und Numerierung von Appendixes:

```
.nr Hu 1
.nr a0
.de aH
.nr a + 1
.nr P 0
.PH "''' Appendix \\ na - \\\\\\\\ nP'"
.SK
.HU "\\$1"
```

..

Nach der oben erwaehnten Initialisierung und Definition wird durch jeden Aufruf von ''.aH "title"'' eine neue Seite begonnen. Dabei wird ein Seitenkopf (Kopfgestaltung {9}) der Form 'Appendix a-n' erzeugt und eine unnumerierte Ueberschrift ausgegeben, die auf Wunsch fuer die spaetere Verwendung im Inhaltsverzeichnis weggespeichert wird. Sollen die Ueberschriften des Appendix' zentriert erscheinen, muss Hc=1 gesetzt sein {4.2.2.3}.

14.2.2. "Haengendes" Einruecken mit Tabs

Das folgende Beispiel veranschaulicht die Verwendung des "haengenden" Einrueckens der variablen Item-Lists (.VL) {5.3.3.6}. Zuerst wird ein durch den Nutzer definiertes Makro erstellt, das die vier Argumente, die das Kennzeichen bilden, akzeptiert. Jedes der Argumente ist durch ein Tab vom vorausgegangenen getrennt. Die Tab-Ladewerte werden spaeter definiert. Da das erste Argument mit einem Punkt oder Apostroph beginnen kann, wird durch die Zeichen "\&" gewaehrleistet, dass das Formatierungsprogramm solch ein Argument nicht mit einem Makro bzw. mit einer Anforderung verwechselt. Die beiden Zeichen "\&" werden vom Formatierungsprogramm als Zeichen mit einer "Null-Weite" verstanden und erscheinen nicht in der Ausgabe. Waehrend die beiden Zeichen "\t" als Tab interpretiert werden, dient "\c" zur Verkettung der Textzeile, die dem Makro folgt, mit der Textzeile, die durch das Makro erstellt wurde. Die Makrodefinition und ein Beispiel seiner Benutzung sind:

```
 .de aX
 .LI
 \&\\$1\t\\$2\t\\$3\t\\$4\t\c
 ..
  .
  .
  .
 .ta 9n 18n 27n 36n
 .VL 36
 .aX .nh off\-no
Keine Silbentrennung.
Die automatische Silbentrennung wurde abgeschaltet, mit Ausnahme von Worten, die Bindestriche enthalten (z.B. Eva-Maria)
 .aX .hy on \- no
Silbentrennung.
Die automatische Silbentrennung wird eingeschaltet.
 .aX .hc\[]c none none no
Der Indikator zur Markierung der Silbentrennung wird auf "c" gestellt oder geloescht. Waehrend der Textverarbeitung wird der Indikator unterdrueckt und erscheint nicht in der Ausgabe. Wird der Indikator einem Wort vorangestellt, wird dieses Wort von der Silbentrennung ausgenommen.
 .LE
```

Die resultierende Ausgabe ist:

.nh off - no	Keine Silbentrennung. Die automatische Silbentrennung wurde abgeschaltet, mit Ausnahme von Worten, die Bindestriche enthalten (z.B Eva-Maria)
.hy on - no	Silbentrennung. Die automatische Silbentrennung wird eingeschaltet.
.hc c none none no	Der Inidikator zur Markierung der Silbentrennung wird auf "c" gestellt oder geloescht. Waehrend der Textverarbeitung wird der Indikator unterdrueckt und erscheint nicht in der Ausgabe. Wird der Indikator einem Wort vorangestellt, wird dieses Wort von der Silbentrennung ausgenommen.

A N H A N G A

Definitionen von List-Makros !*!

Hinweis:
Dieser Anhang betrifft nur Nutzer, die ihre Formatierungsmakros selbst definieren moechten. Nachfolgend stehen die Definitionen der Listen-Initialisierungs-Makros {5.3.3}.

```
.de AL
.nr !D 0
.if !@\\S1@@ .if !@\\S1@1@ .if !@\\S1@a@ .if !@\\S1@A@
      \&.if !@\\S1@I@ .if !@\\S1@i@ .)D "AL:bad arg:\\S1"
.if \\n(.S<3 \{.ie \w@\\S2@=0 .)L \\n(Lin 0 2n 1 "\\S1"
.e1 .LB 0\\S2 0 2 1 "\\S1\}
.if \\n(.S>2 \{.ie \w@\\S2@=0 .)L \\n(Lin 0 2n 1"\\S1" 0 1
.e11 .LB 0\\S2 0 2 1 "\\" 0 1 \}
..
.de BL
.nr ;0 \\n(Pi
.if (\\n(.S>0)&(\w@\\S1@>0) .nr ;0 0\\S1
.ie \\n(.S<2 .LB \\n(;0  0 1 0 \\*(BU
 .e1 .LB \\n(;0 0 1 0 \\*(BU 0 1
.rr ;0
..
.de DL
.nr ;0 \\n(Pi
.if (\\n(.S>0)&(\w@\\S1>0) .nr ;0 0\\S1
.ie \\n(.S>2 .LB \\n(;0 0 1 0 \ (em
.e1 .LB\\n(;0 0 1 0 \(em 01
.rr ;0
..
.de ML
.if \\n(.S<1 .)D "ML:missing arg"
.nr ;0 \w@\\S1@\\S1@u/3ug\n(.su+1u\" get size in n's
.ie \\n(.S<2 .LB \\n(;0  0 1 0"\\S1"
.e1 .if \\n(.S=2 .LB 0\\S2 0 1 0"\\S1"
.if \\n(.S>2 \{.if !\w@\\S2@ .LB \\n(;0 0 1 0 "\\S1" 0 1
.       if \w@\\S2@ .LB 0\\S2 0 1 0 "\\S1" 0 1 \}
..
.de RL
.nr ;0 6
.if (\\n.S>0)&(\w@\\S1@>0).nr ;0 0\\S1
.ie \\n(.S<2 .LB \\n(; 0 0 2 4
.e1 .LB \\n(; 0 0 2 4 1 0 1
.TT ;0
..
.de VL
.if \\n(.$<1 .)D " VL : missing arg"
.ie \\n(.$<3 .LB 0\\$1 0\\$2 0 0
.el .LB 0\\$1 0\\$2 0 0 \& 1
..
```

Jedes dieser Makros kann auch anders definiert werden, z.B. wird gewuenscht, dass zwischen dem Bulletin-Zeichen und dem nachfolgenden Text zwei Leerzeichen angeordnet

werden, muss .BL vor Aufruf wie folgt definiert werden:

```
.de BL
.LB 3 0 2 0 \\\\*(Bu
..
```

A N H A N G B

Durch den Nutzer definierte Listenstrukturen !*!

Hinweis:
Dieser Anhang betrifft nur Nutzer, die ihre Formatierungsmakros selbst definieren moechten.

Fuer die Erstellung grosser Dokumente mit einer umfangreichen Listenstruktur ist es vorteilhaft, die entsprechenden Listen nur einmal zum Anfang zu definieren, anstatt zu jedem Listenbeginn. Ausserdem wird dadurch dem gesamten Dokument ein einheitliches Aussehen gegeben. So koennte ein allgemeines Listen-Initialisierungs-Makro z.B. in der Art definiert werden, dass er vom Verschachtelungsgrad zum Zeitpunkt seines Aufrufs abhaengt. Dabei wird vorausgesetzt, dass die Listenebenen 1 bis 5 folgendes Aussehen haben sollen:

A.
[1]
+
a)
+

Nachfolgend wird ein Makro (.al) definiert, durch das in jedem Fall eine neue Liste begonnen wird, deren Typ vom augenblicklichen Listenniveau bestimmt wird. Fuer das Verstehen dieser Makrodefinition ist es wichtig zu wissen, dass das MM-Paket das Number-Register :g zur Bestimmung der aktuellen Listenebene benutzt. Es g=0 ist momentan keine Liste aktiv. Durch jeden Listen-Initialisierungsaufruf (.LI) wird :g inkrementiert, durch jedes .LE wieder dekrementiert.

```
.de al
'\" Register g wird  fuer die zeitweilige Zwischenspeicherung
'\" von :g benutzt, bevor es geaendert wird.
.nr  g \\n(:g
.if \\ng=0 .AL A \" erzeuge A.
.if \\ng=1 .LB \\n(Li 0 1 4 \" erzeuge [1]
.if \\ng=2 .BL \" erzeuge ein Bullet
.if \\ng=3 .LB \\n(Li 0 2 2 \" erzeuge a)
.if \\ng=4 .ML +\" erzeuge +
```

Dieses Makro kann in Verbindung mit .LI/.LE anstelle des Makros .AL, .RL, .BL und .ML benutzt werden. So erzeugt z.B. die folgende Eingabe:

```
.al
.LI
erste Zeile
```

```
.al
.LI
zweite Zeile
.LE
.LI
dritte Zeile
.LE
```

die nachfolgende Ausgabe:

```
a. erste Zeile

    [1] zweite Zeile

B. dritte Zeile
```

Es ist jedoch auch moeglich, mit dem Listen-"Problem" aehnlich wie mit den Ueberschriften (.H) zu verfahren. Die Listen-Initialisierung kann ebenso wie .LI/.LE Teil eines einfachen Makros sein. Solch ein Makro (im nachfolgenden Beispiel .bl genannt) benoetigt ein Argument fuer die Nennung der entsprechenden Item-Ebene; es bestimmt das Listen-Niveau entweder zu Beginn einer neuen Liste oder durch Rueckstellen auf den vorangegangenen Wert. Dann folgt die Ausgabe eines .LI-Makro-Aufrufes fuer die Erstellung des Items:

```
.de bl
.ie \\n(.$ .nr g \\$1\" wenn ein Argument
'\" angegeben ist, handelt es sich um das Niveau
.el .nr g \\n(:g\'' falls kein Argument angegeben
'\' wurde, benutze das augenblickliche Niveau
.if \\ng-\\n(:g>1.)"** ILLEGAL SKIPPING OF LEVEL"
'\" Erhoehung des Niveaus um mehr als 1
.if \\ng>\\n(:g\{.aL\\ng-1\"if g > :g'
'\" beginne eine neue Liste
.   nr g \\n(:g\{\" und setze g auf den
'\" aktuellen Niveauwert
(.al aendert g)
.if \\n(:>\\ng .LC \\ng \" if :g > g
'\" Zurueckstellen um das Niveau zu korrigieren
'\" if :g=g, bleibe innerhalb der aktuellen Liste
.LI \" in allen Faellen Ausgabe eines Items
..
```

Um .bl zu aktivieren, muss die vorangegangene Definition des .aL-Makros geaendert werden, so dass man anstelle von :g den Wert von g erhaelt. Wird .bL ohne Argument aufgerufen, bleibt das gegenwaertige Listenniveau erhalten. Das LC-Makro des MM-Paketes loescht die Listendeskriptoren auf den Wert kleiner oder gleich seines Argumentes. So beinhaltet z.B. das .H-Makro den Aufruf ".LC 0". Soll der Text am Ende der Liste wieder zur Verfuegung stehen, muss der Aufruf von ".LC 0" eingefuegt werden, der die Liste komplett loescht. Das unten aufgefuehrte Beispiel veranschaulicht den relativ geringen Aufwand, der bei dieser Art

der Herangehensweise benoetigt wird. Die folgende Eingabe:

```
 .bL 1
 erste Zeile
 .bL 2
 zweite Zeile
 .bL 1
 dritte Zeile
 .bL
 vierte Zeile
 .LC 0
 fuenfte Zeile
```

erzeugt folgende Ausgabe:

```
A. erste Zeile
     [1] zweite Zeile
B. dritte Zeile
C. vierte Zeile
 fuenfte Zeile
```

A N H A N G C

Fehlermeldungen

C.1 Fehlermeldungen durch das MM-Paket

Jede durch das MM-Paket ausgegebene Fehlernachricht besteht aus einem konstanten und einem variablen Text. Der konstante Standardteil hat die Form:

```
ERROR : input Line n :
```

Der variable Teil beginnt mit dem Namen des Makros in dem der Fehler auftrat. Nachfolgend erfolgt ein Auflisten aller variablen Teile mit entsprechender Erklaerung:

Check TL, AU, AS, AE MT sequence	Die in {6.10} vorgeschriebene Aufrufreihenfolge wurde nicht beachtet.
AL: bad arg:value	Das im AL.-Makro benutzte Argument ist nicht 1, A, a, I oder i. Der falsche Wert wird unter 'value' ausgewiesen.
CS: cover sheet too long	Der Text fuer das Deckblatt ist laenger als eine Seite. Text oder Einrueckwert des Abstrakt-Makros muessen verringert werden.
DS: too many displays	Es haben sich mehr als 26 flexible Displays angesammelt und sind noch nicht ausgegeben worden.
DS: missing FE	Ein Display beginnt innerhalb einer Fussnote. Wahrscheinliche Ursache ist ein unterbliebener oder fehlerhafter .FE-Aufruf.
DS: missing DE	Ein .DS oder .DF erscheint innerhalb eines Displays, z.B. .DE fehlt oder ist falsch.
DE: no DS or DF active	Es erfolgte ein .DE-Aufruf ohne zugehoeriges .DS oder .DF.
FE: no FS	Es erfolgte ein .FE-Aufruf ohne zugehoeriges .FS
FS: missing FE	Ein .FS-Aufruf wurde nicht mit dem entsprechenden .FE

	abgeschlossen, z.B. innerhalb eines Fussnotentextes wurde eine zweite Fussnote eroeffnet.
FS: missing DE	Innerhalb eines Displays erscheint eine Fussnote, z.B. ein .DS- oder .DF-Aufruf wurde nicht mit .DE abgeschlossen.
H: bad arg:value	Das erste Argument eines .H-Makros muss eine Zahl zwischen 1 bis 7 sein. Der falsche Wert wird unter 'value' dargestellt.
H: missing FE	Innerhalb einer Fussnote erscheint der Aufruf eines .H- oder .HU-Makros.
H: missing DE	Innerhalb eines Displays erscheint der Aufruf eines .H- oder .HU-Makros.
H: missing arg	Das .H-Makro benoetigt mindestens ein Argument.
HU: missing arg	Das .HU-Makro benoetigt ein Argument.
LB: missing arg(s)	.LB benoetigt mindestens 4 Argumente.
LB: too many nested lists	Es wurde noch eine Liste eroeffnet, obwohl schon 6 Listen aktiv sind.
LE: mismatched	Es erschien ein .LE ohne vorangegangenes .LB oder andere Listen-Initialisierungsmakros {5.3.3}. Obgleich diese Auslassung zu keinem Fatalerror fuehrt, treten im Text gewisse "Probleme" auf.
LI: no lists active	Es erscheint ein .LI ohne einen vorangegangenen Listen-Initialisierungsmakro-Aufruf. Entweder wurde dieser Aufruf ausgelassen, oder durch dazwischenliegende .H- oder .HU-Makro-Aufrufe vom .LI getrennt.
ML: missing arg	.ML benoetigt mindestens ein

	Argument.
ND: missing arg	.ND benoetigt ein Argument
SA: bad arg: value	Wird fuer .SA ein Argument angegeben, stehen nur 0 oder 1 zur Auswahl. Der falsche Wert wird unter 'value' ausgewiesen.
SG: missing DE	Innerhalb eines Displays erscheint der Aufruf des .SG-Makros.
SG: missing FE	Innerhalb einer Fussnote erscheint der Aufruf des .SG-Makros.
SG: no authors	Dem .SG-Makro geht kein .AU-Makro voran.
VL: missing arg	.VL benoetigt mindestens ein Argument.

C.2 Fehlermeldungen durch das Formatierungsprogramm

Die meisten durch das Formatierungsprogramm auszugebenen Fehlermeldungen sind fuer den Nutzer verstaendlich. Jene Fehlermitteilungen, auf die der Anwender reagieren kann, sind nachfolgend alphabetisch geordnet aufgelistet. Alle anderen Fehlerausschriften sind systemabhaengig. Es werden ausgegeben:

Cannot do ev
- beim Setzen der Seitenbreite auf einen negativen oder extrem kleinen Wert
- beim Setzen der Seitenlaenge auf einen negativen oder extrem kleinen Wert
- bei wiederholter Abarbeitung eines Makropaketes (z.B. Angabe eines .so-Aufrufes, was schon in der Kommandozeile erfolgte)
- wenn fuer ein troff-Dokument, das umfangreicher als 10 Seiten ist, die -sl-Option gefordert wird.

Cannot execute filename
- die .!-Anforderung kann die genannte Datei nicht auffinden.

Cannot open filename
- wenn eine Datei aus der Liste von Dateien, die abgearbeitet werden soll, nicht eroeffnet werden kann.

Exception word list full
- zeigt an, dass zu viele Worte in der Liste fuer die Silbentrennung enthalten sind (.hu-Anforderung {3.4})

Line overflow
- Die erstellte Ausgabezeile ist zu lang fuer den Zeilen-Puffer des Formatierungsprogrammes. Der Teil, der zu lang ist, geht verloren. Siehe auch Fehlermeldung fuer zu lange Worte.

Non-existent font type
- es erfolgte eine Anforderung an einen unbekannten Schriftsatz.

Non-existent macro file
- das angeforderte Makropaket existiert nicht.

Non-existent terminal type
- Die Terminaloption bezieht sich auf einen unbekannten Terminaltyp.

Out of temp file space
- Der zusaetzliche Speicherraum fuer Makrodefinitionen usw. kann nicht zugewiesen werden. Diese Meldung erscheint oft wegen unabgeschlossener Fussnoten oder Displays (fehlendes .FE oder .DE!), unabgeschlossener Makrodefinitionen (fehlendes "..") oder zu grossen Inhaltsverzeichnisses.

Too many page numbers
- Die fuer die -o-Option des Formatierungsprogrammes angegebene Seitenanzahl ist zu gross.

Too many string/macro names
- Das Pool fuer die Namen von Zeichenketten und Makros ist voll. Nicht mehr benoetigte Zeichenketten und Makros koennen mit der .rm-Anforderung gestrichen werden.

Too many number registers
- Es gibt zu viele Numberregister. Unbenoetigte Numberregister koennen mit der .rr-Anforderung gestrichen werden.

Word overflow
- Das erzeugte Wort ist zu lang fuer den Wort-Puffer des Formatierungsprogrammes. Die die Laenge ueberschreitenden Zeichen werden gestrichen. Eine moegliche Ursache fuer "Line overflow" und "Word overflow" ist die fehlerhafte Benutzung von '\c' oder der .cu-Anforderung. Auch zu lange Gleichungen, erzeugt durch eqn(1)/neqn(1), fuehren zu dieser Fehlermeldung.

A N H A N G D

Ueberblick ueber alle Makros, Zeichenketten und Numberregister

D.1 Makro-Namen

Im folgenden Abschnitt sind alle durch das MM-Paket bereitgestellten Makronamen in alphabetischer Reihenfolge aufgefuehrt. Die erste Zeile enthaelt jeweils den Makronamen, eine Kurzbeschreibung und einen Verweis auf die entsprechende Sektion, in der das Makro ausfuehrlich beschrieben wird. Die zweite Zeile zeigt den Prototyp des Makro-Aufrufes.

Makros, die mit einem Stern gekennzeichnet sind, koennen im allgemeinen vom Nutzer nicht direkt angesprochen werden; es handelt sich hierbei um Makros, die von anderen Makros (z.B. Makros fuer die obere und untere Blattrandgestaltung) aufgerufen werden.

1C Ein-Spalten-Format {12.4}
.1C

2C Zwei-Spalten-Format {12.4}
.2C

AE Abstrakt Ende {6.4}
.AE

AF Abweichendes Format fuer die erste Seite {6.8}
.AF [Firmenname]

AL Automatisch erhoehte Liste, Start {5.3.3.1}
.AL [Typ] [Einrueckwert] [1]

AS Abstrakt Start {6.4}
.AS [arg] [Einrueckwert]

AT Autor(en)-Titel {6.2}
.AT [Titel] ...

AU Autor(en)-Angaben {6.2}
.AU name [Initialien] [Ort] [Abt.] [ext] [Raum]
[arg1] [arg2] [arg3]

AV Genehmigungsleiste {6.12}
.AV [name]

B Fettdruck {12.1}
.B [B-arg] [vorangegangene Schriftart] [B-arg]
[vorang. Schr.] [B-arg] [vorang. Schr.]

BE Fusstext-Ende {9.12}
.BE

BI Fettdruck/Kursivdruck {12.13}
    .BI [fett-arg] [kursiv-arg] [fett] [kursiv]
        [fett] [kursiv]

BL Bulletin-Liste, Start {5.3.3.2}
    .BL [Einrueckwert] [1]

BR Fettdruck/Normaldruck {12.1}
    .BR [fett-arg] [normal-arg] [fett] [normal]
         [fett] [normal]

BS Fusstext-Start {9.12}
    .BS

CS Deckblatt {10.2}
    .CS [Pages] [Other] [Total] [Figures]
        [Tables] [Refs]

DE Display-Ende {7.13}
    .DE

DF flexibles Display, Start {7.2}
    .DF [Format] [Fill] [Einrueckwert R]

DL Dash-Liste {5.3.3.3}
    .DL [Einrueckwert] [1]

DS festes Display {7.1}
    .DS [Format] [Fill] [Einrueckwert R]

EC Gleichungen {7.5}
    .EC [Titel] [Darst] [Flag]

EF Fusstext auf geradzahligen Seiten {9.6}
    .EF [arg]

EH Kopftext auf geradzahligen Seiten {9.3}
    .EH [arg]

EN Gleichungen-Ende {7.4}
    .EN

EQ Gleichungen-Start {7.4}
    .EQ [Name]

EX Ausstellungsstuecke {7.5}
    .EX [Titel] [Darst] [Flag]

FC Unterschriftsleiste (formeller Gruss) {6.11.1}
    .FC [Floskel]

FD Aeussere Form der Fussnoten {8.3}
    .FD [JW] [1]

FE Fussnoten-Ende {8.2}

```
     .FE

FG   Bilder {7.5}
     .FG [Titel] [Darst] [Flag]

FS   Fussnoten-Start {8.2}
     .FS [Name]

H    Numerierte Ueberschrift {4.2}
     .H Ebene [Ueberschrift] [Suffix]

HC   Silbentrennung {3.4}
     .HC [Trennungsindiaktor]

HM   Kennzeichen von Ueberschriften {4.2.2.5}
     .HM [arg1] ... [arg7]

HU   unnumerierte Ueberschrift {4.3}
     .HU Ueberschrift

HX*  Aenderung  der standardmaessigen Ueberschriftsdarstel-
     lung (vor Ausgabe der Ueberschrift) {4.6}
     .HX dlevel rlevel Ueberschrift

HY*  Aenderung der standardmaessigen  Ueberschriftsdarstel-
     lung (vor Ausgabe der Ueberschrift) {4.6}
     .HY dlevel rlevel Ueberschrift

HZ*  Aenderung  der standardmaessigen Ueberschriftsdarstel-
     lung nach Ausgabe der Ueberschrift) {4.6}
     .HZ dlevel rlevel Ueberschrift

I    Kursivdruck (Unterstreichung in nroff) {12.1}
     .I [I-arg] [vorangegangene Schriftart] [I-arg]
        [vorang. Schr.] [I-arg] [vorang. Schr.]

IB   Kursivdruck/Fettdruck {12.1}
     .IB [kursiv-arg] [fett-arg] [kursiv] [fett] [kursiv]
         [fett]

IR   Kursivdruck/Normaldruck {12.1}
     .IR [kursiv-arg] [normal-arg] [kursiv] [normal]
         [kursiv] [normal]

LB   Listen-Beginn {5.4}
     .LB Text-Einrueckwert Kennzeichen-Einrueckwert
         P Typ [Kennzeichen] [LI-Sp] [LB-Sp]

LC   Listen-Status-Loeschen {Anhang A}
     .LC [Listen-Ebene]

LE   Listen-Ende {5.3.2}
     .LE [1]

LI   Listen Item {5.3.1}
     .LI [Kennzeichnung] [1]
```

ML Liste mit selbstgewaehltem Kennzeichen {5.3.3.4}
    .ML Kennzeichen [Einrueckwert] [1]

MT Typen von Dokumenten {6.6}
    .MT [Typ] [Empfaenger]
    oder
    .MT [4] [1] {6.9}

ND Aenderung des Datums {6.7}
    .ND Datum

NE Bezeichnung-Ende {6.11.2}
    .NE

NS Bezeichnung-Start {6.11.2}
    .NS [arg]

nP Einruecken der ersten beiden Zeilen eines Absatzes {4.1}
    .nP

OF Fusstext auf ungeradzahligen Seiten {9.7}
    .OF [arg]

OH Kopftext auf ungeradzahligen Seiten {9.4}
    .OH [arg]

OK Andere Schluesselworte {6.5}
    .OK [Keyw] ...

OP Erzwingen einer ungeradzahligen Seitennummer {12.8}
    .OP

P   Absatz {4.1}
    .P [Typ]

PF Fusstext auf jeder Seite {9.5}
    .PF [arg]

PH Kopftext auf jeder Seite {9.2}
    .PH [arg]

PX* Aenderung der standardmaessigen Kopftextdarstellung {9.11}
    .PX

R   Normaldruck (Unterstreichung in nroff) {12.1}
    .R

RB Normaldruck/Fettdruck {12.1}
    .RB [normal-arg] [fett-arg] [normal] [fett]
        [normal] [fett]

RD Einfuegen von Text {12.11}
    .RD [Prompt] [D-name] [Z-name]

RF Referenztext-Ende {11.2}
    .RF

RI Normaldruck/Kursivdruck {12.1}
    .RI [normal-arg] [kursiv-arg] [normal] [kursiv]
        [normal] [kursiv]

RL Referenz-Liste, Start {5.3.3.5}
    .RL [Einrueckwert] [1]

RP zusaetzliche Referenzseiten
    .RP [arg 1] [arg 2]

RS Referenztext-Start {11.2}
    .RS [name]

S   Setzen der Punktgroesse und des vertikalen Abstandes
    bei troff {12.9}
    .S [Punktgroesse] [vertikaler Abstand]

SA Einstellung des rechten Randes {12.2}
    .SA [arg]

SG Unterschriftsleiste (Signatur) {6.11.1}
    .SG [arg] [1]

SK Ueberspringen von Seiten {12.7}
    .SK [Seiten]

SP vertikaler Abstand bei nroff {12.6}
    .SP [Zeilen]

TB Tabellen-Titel {7.5}
    .TB [Titel] [Darst] [Flag]

TC Inhaltsverzeichniserstellung {10.1}
    .TC [Niveau] [Abt] [Seitennr.] [Tab]
        [Text 1] [Text 2] [Text 3] [Text 4]

TE Tabellen-Ende {7.3}
    .TE

TH Tabellenkopf {7.3}
    .TH [N]

TL Titelzeile {6.1}
    .TL [arg 1] [arg 2]

TM Technical Memorandum Nummer {6.3}
    .TM [Nummer] ...

TP Aenderung der standardmaessigen Kopftextdarstellung
    {9.11}
    .TP

TS Tabellen-Start {7.3}
    .TS [H]

TX* Aenderung der standardmaessigen Kopfgestaltung des Inhaltsverzeichnisses {10.1}
    .TX

TY* Aenderung der standardmaessigen Kopfgestaltung des Inhaltsverzeichnisses (Unterdrueckung von 'CONTENTS' {10.13}

VL variable Item-Liste {5.3.3.6}
    .VL Text-Einrueckwert [Kennzeichen-Einrueckwert] [1]

VM Einstellung des oberen und unteren Blattrandes {9.13}
    .VM [Orand] [Urand]

WC Darstellung von Fussnoten und Darstellung im Zwei-Spalten-Format {12.4}
    .WC [Format]

D.2 Zeichenketten

Nachfolgend sind alle vom MM-Paket benutzten Zeichenketten in alphabetischer Reihenfolge aufgefuehrt. Jedem Zeichenkettenname folgt eine Kurzbeschreibung, ein Verweis auf die Sektion in der die Zeichenkette benutzt wird und der oder die Standardwerte.

BU Bulletin {3..7}

    nroff: +
    troff: +

Ci Einrueckwerte fuer die bis zu sieben Ueberschriften des Inhaltsverzeichnisses (Angabe erfolgt als Bezugsgroesse) {10.1}

F   Zur automatischen Numerierung von Fussnoten {8.1}
    nroff: \u\\n+(p\d
    troff: \v'-.4m'\\n+(:p\$0\v'.4m'

DT aktuelles Datum (falls nicht individuell veraendert)
    Monat, Tag, Jahr (z.B.May 17, 1987) {6.7}

EM Gedankenstrich, erzeugt ein em 'dash' fuer nroff und troff {3.8}

HF Schriftsatz fuer die sieben Ueberschriftsebenen {4.2.2.4}
    3 3 2 2 2 2 2 (Es bedeuten 1 und 2 Fettdruck
    3-7 Unterstreichung in nroff und Kursivdruck in troff)

HP Punktgroesse fuer die sieben Ueberschriftsebenen bei troff
    {4.2.2.4}

Le Titel: LIST OF EQUATIONS {7.6}

Lf Titel: LIST OF FIGURES {7.6}

Lt Titel: LIST OF TABLES {7.6}

Lx Titel: LIST OF EXHIBITS {7.6}

RE SCCS Versionsnummer {12.3}

Rf Zur automatischen Numemrierung von Referenzen {11.1}

Tm Trademark "TM"-Kennzeichnung {3.9}

Es koennen auch 'Akzent'-Zeichenketten benutzt werden {12.10}

D.3 Numberregister

In diesem Abschnitt erfolgt die Aufzaehlung der verwendeten Numberregister in alphabetischer Reihenfolge. Jedem Registernamen folgt eine Kurzbeschreibung, ein Sektionshinweis, der standardmaessige Ladewert und der zugelassene Wertbereich in der Schreibweise [m:n].
Regissternamen, die aus einem Buchstaben bestehen, koennen in der Kommandozeile gesetzt werden. Ein Stern (*) hinter einem Register verweist darauf, dass das entsprechende Register nur in der Kommandozeile oder vor dem Lesen der Makrodefinitionen durch das Formatierungsprogramm gesetzt werden muss {2.4/2.5}. Hinweise ueber das Setzen von Registern siehe {1.4}.

A* Benutzung von vorgedruckten Formblaettern und eines akustischen Signals {2.4}
    0, [0:2]

AU Unterdrueckung der Ausgabe von naeheren Angaben ueber den Autor {6.2}
    1, [0::1]

C* Typ der Kopie (Original, DRAFT usw.) {2.4}
    0 (Original), [0:3]

Cl Angabe bis zu welcher Ebene Ueberschriften im Inhaltsverzeichnis ausgegeben werden sollen {4.4}

Cp Angabe ueber seitenmaessige Darstellung der im Dokument verwendeten Tabellen, Bildern usw. {10.1}
    0 (gesonderte Darstellung), [0:1]

D Debugger-Flag {2.4}
    0, [0:1]

De Seitenvorschub fur flexible Displays {7.2}
    0, [0:1]

Df Formatangabe fur flexible Displays {2}
5, [0:5]

Ds Leerzeile vor und hinter einem festen Display {7.1}
1, [0:1]

Ec Zaehler fuer Gleichungen {7.5}
0, [0:1] bei jedem Aufruf des .EC-Makros erfolgt eine Erhoehung um 1

Ej Seitenvorschub vor Ueberschriften {4.2.2.1}
0 (kein Vorschub), [0:7]

Eq Anordnung von Gleichungsnamen {7.4}
0 (rechts), [0:1]

Ex Zaehler fuer Ausstellungsstuecke {7.5}
0, [0:?] bei jedem Aufruf des .EX-Makros erfolgt eine Erhoehung um 1

Fg Zaehler fuer Bilder {7.5}
0, [0:?] bei jedem Aufruf des .FG-Makros erfolgt eine Erhoehung um 1

Fs Abstand zwischen den Fussnoten {8.4}
1, [0:?]

H1-H7 Zaehler fuer die moeglichen sieben Ueberschriftsebenen {4.2.2.5}
0, [0:?] bei jedem Aufruf des .H- oder .HU-Makros wird der entsprechende Zaehler (in Abhaengigkeit vom Hu-Register) inkrementiert. H2 bis H7 werden bei jedem Aufruf der hoeheren Ueberschriftsebenen H1 automatisch auf 0 zurueckgesetzt

H6 Break nach Ueberschriften (nach .H und .HU) {4.2.2.2}
2, [0:7]

Hc Zentrieren von Ueberschriften {4.22.2.2}
0 (keine zentrierte Ueberschrift), [0:7]

Hi Einruecken von Ueberschriften {4.2.2.2}
1 (Einruecken wie Absaetze), [0:2]

Hs Leerzeilen vor Ueberschriften (nach .H und .HU) {4.2.2.2}
2 (Leerzeile nur nach .H1 und .H2), [0:7]

Hz Darstellungsart der Kennzeichen von Ueberschriften (fuer .H einfache oder verkettete Kennzeichnung) {4.2.2.5}
0 (verkettete Kennzeichnung z.B. 1.1.2), [0:1]

Hu Einstellung der Ebene, bis zu der Ueberschriften unnumeriert ausgegeben werden sollen (.Hu) {4.3}

2, [0:7]

Hy Silbentrennungsschalter {3.4}
0 (keine automatische Silbentrennung), [0:1]

L* Seitenlaenge {2.4}
66, [20:?] (11i, [Zi:?] fuer troff)
(fuer nroff erfolgt die Angabe in Direktwerten als Zeilenanzahl; fuer troff wird eine entsprechende Bezugsgroesse verwendet)

le Liste fuer Gleichungen {7.6}
0 (es wird keine Liste erstellt), [0:1]

Lf Liste fuer Bilder {7.6}
1 (es wird eine Liste erstellt), [0:1]

Li Einruecken der .Al-Liste {5.3.3.1}
6, [0:?]

Ls Abstand zwischen Listen-Items {5.3.3.1}
5 (Abstand zwischen allen Ebenen), [0:5]

Lt Liste fuer Tabellen {7.6}
1 (es wird eine Liste erstellt), [0:1]

Lx Liste fuer Ausstellungsstuecke {7.6}
1 (es wird eine Liste erstellt), [0:1]

N* Art der Numerierung {2.4}
0, [0:5]

Np Numerierungsart fuer Abschnitte {4.1}
0 (keine Numerierung), [0:1]

O* Seitenoffset {2.4}
.75i, [0:?] (0.5i, [0i:?] in .nr
:p-1 troff
(Angabe der Werte erfolgt analog L*)

Oc Art der Seitennumerierung fuer das Inhaltsverzeichnis {10.13}
0 (roemische Darstellung durch Kleinbuchstaben), [0:1]

Of Darstellung der Bildunterschriften {7.5}
0 (Punkt als Trennzeichen), [0:1]

P Seitennummer {2.4.}
0, [0:?]

Pi Einrueckwert fuer Absaetze {4.1}
5, [0:?]

Ps Abstand vor einem Absatz {4.1}
1 (eine Leerzeile zwischen den Abstaenden), [0:?]

Pt Anordnung der Absaetze {4.1}
    0 (alle Absaetze erscheinen linksbuendig), [0:2]

Pv "PRIVATE" Kopfdruck {9.15}
    0 (kein Ausdruck), [0:2]

S* Standardpunktgroesse fuer troff {2.4}
    10, [6:36]

Si Standardeinrueckwert fuer Displays {7.1}
    5, [0:?]

T* Typ des nroff Ausgabegeraetes {2.4}
    0, [0:2]

T6 Zaehler fuer Tabellen {7.5}
    0, [0:?] bei jedem Aufruf des
    .TB-Makros erfolgt eine Erhoehung um 1

U* Art der Unterstreichung (nroff) von Ueberschriften
    fuer .H und .HU {2.4}
    0 (durchgehende Unterstreichung), [0:1]

W* Seitenbreite (Zeilen- und Titellaenge) {2.4}
    6i, [10:1365] (6i, [Zi:7.54i]
    in .nr :p-1 troff
    (Angabe der Werte erfolgt analog L*

Debugger

adb

Vorwort

Adb ist das WEGA Standardtestprogramm. Durch adb wird es moeglich, Core-Files auszuwerten, wie sie bei einem Programmabbruch erzeugt werden, ausfuehrbare Files oder Core-Files in verschiedenen Formaten zu protokollieren, solche Files zu veraendern oder Programme mit Unterbrechungspunkten abzuarbeiten.

An Beispielen werden verschiedene formatierte Ausgaben, Testtechniken fuer C-Programme und die Auswertung der bei Programmabbruechen entstehenden Core-Files erlaeutert. Ausserdem werden weitere Anwendungsmoeglichkeiten von adb, wie z.B. das Auflisten von Filesysteminformationen, aufgezeigt. Dabei wird die Kenntnis der grundlegenden WEGA Kommandos sowie der Programmiersprache C vorausgesetzt.

Adb ist nicht auf spezielle Programmiersprachen beschraenkt. Die auf Assembler-Ebene ausgerichtete Benutzerschnittstelle von adb sorgt einerseits fuer universelle Anwendbarkeit, macht aber seine Anwendung relativ kompliziert.

Inhaltsverzeichnis Seite

1. Ueberblick

1.1. Aufruf von adb

Der adb - Aufruf hat folgenden Aufbau:

```
adb objfile corefile
```

wobei objfile eine Datei mit einem ausfuehrbaren Programm ist. Die Datei corefile enthaelt den beim Abbruch von objfile erzeugten Speicherabzug.

```
adb a.out core
```

Die Namen a.out und core werden standardmaessig fuer diese beiden Dateien angenommen, so dass der Aufruf

```
adb
```

ausreicht.

Der Filename minus (-) bedeutet, dass dieses Argument ausgelassen werden soll, so wie in

```
adb - core
```

1.2. File Adressen

Adb stellt Kommandos bereit durch die auf Speicherplatzinhalte in den angegebenen Files zugegriffen werden koennen. Durch das Kommando ? erfolgt ein Zugriff auf Speicherplaetze in objfile und durch / wird auf corefile zugegriffen. Die allgemeine Form dieser Kommandos ist:

```
address ? format
```

oder

```
address / format
```

1.3. Die aktuelle Adresse

Von adb wird eine aktuelle Adresse gefuehrt, die der aktuellen Zeile beim WEGA - Editor vergleichbar ist. Diese Adresse kann unter dem Symbol Punkt (.) abgefragt werden. Wird in einem adb - Kommando eine Adresse angegeben, so erhaelt Punkt (.) deren Wert.

```
.,10d
```

bewirkt die Ausgabe von 10 Dezimalzahlen beginnend bei der aktuellen Adresse. Punkt zeigt anschliessend auf die Adresse, deren Inhalt zuletzt ausgegeben wurde.

In Verbindung mit dem ? oder / Kommando kann die aktuelle Adresse durch Eingabe von newline erhoeht und durch Eingabe von ^ verringert werden.

Somit erhaelt durch

```
0126?i
```

Punkt den Wert oktal 126 und der U8000-Befehl, der auf dieser Adresse beginnt, wird ausgegeben.

Adressen koennen die Form von Ausdruecken haben. Bestandteile von Ausdruecken koennen ganze Dezimal-, Oktal- und Hexadezimalzahlen oder Symbole des zu testenden C-Programmes sein, die mit den folgenden Operatoren

+	Addition
-	Subtraktion
*	Multiplikation
%	Division (Integer)
&	UND (bitweise)
|	INKLUSIV ODER (bitweise)
#	Aufrundung
~	NAND

kombiniert werden koennen. Die gesamte Arithmetik in adb ist eine 32-bit Arithmetik. Bei der Benutzung symbolischer Adressen eines C-Programms kann sowohl name als auch _name verwendet werden. Beide Formen werden von adb akzeptiert.

1.4. Formate

Fuer die Arbeit mit adb stehen eine Reihe von Buchstaben und Zeichen zur Verfuegung, die zur Beschreibung des Ausgabeformats von Kommandos dienen. Wird ein adb - Kommando ohne Spezifikation des Ausgabeformats angegeben, so erfolgt die Ausgabe stets in dem zuletzt verwendeten Format. Die folgende Aufstellung gibt eine Uebersicht ueber die am haeufigsten auftretenden Formatkennzeichen:

b	ein Byte oktal
c	ein Byte als ein ASCII-Zeichen
o	ein Wort oktal
d	ein Wort dezimal
f	zwei Worte Gleitkomma
i	U8000-Befehl
s	eine durch 0 begrenzte Zeichenkette
a	der Wert von Punkt (.)
u	ein Wort als vorzeichenlose ganze Zahl
n	Ausgabe eines Zeilenwechsels
r	Ausgabe eines Leerzeichens
^	Verringern von Punkt (.) ohne Ausgabe

Formatkennzeichen sind auch verfuegbar fuer Doppelwoerter (z.B. D fuer 2 Wort dezimal und F fuer Gleitkomma mit doppelter Genauigkeit).

1.5. Allgemeines Kommandoformat

Die allgemeine Form eines adb - Kommandos ist:

```
address , count command_modifier
```

wodurch Punkt (.) auf address gesetzt und das Kommando command_modifier count - mal ausgefuehrt wird.

Die folgende Tabelle enthaelt eine Zusammenstellung wichtiger adb - Kommandos und deren Bedeutung:

Kommando	Bedeutung
?	Ausgabe des Inhaltes von objfile
/	Ausgabe des Inhaltes von corefile
=	Ausgabe des Wertes von Punkt (.)
:	Unterbrechungspunktsteuerung
$	Verschiedene Kommandos
;	Kommandoseparator
!	Aufruf von Shell

Mit den Kommandos $q oder $Q (oder CTRL D) wird die Arbeit mit adb beendet.

2. Test von C-Programmen

2.1. Auswertung eines Core-Files

Das Programm in Beispiel 1 hat einen Fehler (die Abbruchbedingung bei der Laufanweisung fehlt), der bei Ausfuehrung des Programms zu einem Programmgrammabbruch durch Speicherschutzverletzung fuehrt. Dabei wird ein Speicherabzug des Programms (corefile) zum Zeitpunkt des Programmabbruches erzeugt und im aktuellen Directory unter dem Namen core gespeichert.

In Beispiel 2 wird der Ablauf unter Steuerung von adb aufgezeigt. Adb wird durch

```
adb a.out core
```

aktiviert.

Durch das Kommando

```
$c
```

wird zunaechst ein C-Backtrace durch die bis zum Zeitpunkt des Programmes aufgerufenen Unterprogramme ausgegeben.

Das naechste Kommando

```
$C
```

liefert einen C-Backtrace und eine Interpretation aller lokalen Variablen.

Durch das Kommando

```
$r
```

erhaelt man die Registerinhalte einschliesslich des Programm-Counters sowie eine Interpretation des Befehls, auf welchen er zeigt.

Anschliessend erfolgt durch

```
$e
```

die Ausgabe der Werte aller externen Variablen.

Fuer jedes von adb behandelte File exisiert eine Map (Speicherzuordnung). Die Map fuer objfile kann durch ? und diejenige fuer corefile durch / erreicht werden. Es ist guenstig, beim Programmtest Befehle durch ? und Daten durch / abzufragen. Um Informationen ueber die Maps zu erhalten, existiert das Kommando

```
$m
```

Durch dieses Kommando werden beide Maps ausgegeben.

Offentsichtlich erfolgte der Programmabbruch im Programm in Beispiel 1 aufgrund einer fehlerhaften Abbruchbedingung von der Variablen charp (Zeiger auf eine Zeichenkette). Durch die Eingabe

```
*charp/s
```

wird die Variable charp wie ein Zeiger auf corefile verwendet. Die Ausgabe zeigt, dass die Adresse ausserhalb des Programmspeichers liegt.

Durch

```
.=o
```

wird die aktuelle Adresse (nicht ihr Inhalt) oktal ausgegeben. Die aktuelle Adresse kann also dazu benutzt werden, um sich die Stelle im Programm in Erinnerung zu bringen, auf die man im Test zuletzt bezug genommen hat.

Es ist auch gestattet, Adressen in Kommandos relativ zu Punkt (.) auszugeben. So wird zum Beispiel durch:

```
.-10/d
```

der Inhalt der Adresse mit dem Wert "aktuelle Adresse - 10" ausgegeben.

2.2. Geschachtelte Funktionsaufrufe

Zur Veranschaulichung dieser Problematik ist das Programm in Beispiel 3 gut geeignet. Es werden nach dem Aufruf von main nacheinander immer wieder die Funktionen f, g und h aufgerufen, bis es durch diese endlosen Funktionsaufrufe zu einem Stack-Ueberlauf kommt, der zum Programmabbruch und dem Abspeichern eines Core-File fuehrt. Zur Auswertung dieses Core-File wird durch (Beispiel 4)

```
adb
```

der Debugger aktiviert, der die Namen a.out und core standardmaessig fuer das ausfuehrbare File und das Core-File benutzt.

Das Kommando

```
$c
```

liefert eine Menge von Backtrace-Eintragungen von f, g und h, deren Ausgabe durch die DEL-Taste abgebrochen werden

kann (mit der DEL-Taste kann jederzeit der Abbruch eines laufenden adb-Kommandos erfolgen).

Adb wartet auf ein neues Kommando. Durch

```
,5$C
```

werden die letzten 5 Funktionsaufrufe mit lokalen Variablen protokolliert.

Fuer jede der Funktionen f, g und h gibt es einen Zaehler, der bei jedem Aufruf der Funktion um 1 erhoeht wird. Durch

```
fcnt/d
gcnt/d
hcnt/d
```

kann man den Stand dieser Zaehler fuer die einzelnen Funktionen erfahren.

Den Wert einer Variablen, zum Beispiel den von x im letzten Aufruf von h, kann man durch

```
h.x/d
```

abfragen. Diese Abfragemoeglichkeit existiert in dieser Form nur fuer die zuletzt gerufene Funktion. Alle anderen Werte koennen durch

```
$C
```

ausgegeben werden. Durch "address $c" kann das Stack-Backtrace an einer beliebigen Stelle begonnen werden.

2.3. Setzen von Unterbrechungspunkten

Das C-Programm in Beispiel 5 zeigt ein Programm, das Tabs in Leerzeichen umwandelt. Dieses Programm soll unter der Steuerung von adb abgearbeitet werden (Beispiel 6).

Adb wird durch:

```
adb a.out -
```

aufgerufen.

Durch folgendes Kommando ist es moeglich, Unterbrechungspunkte zu setzen:

```
address:b [request]
```

Die Kommandos

```
settab:b
open:b
read:b
tabpos:b
```

setzen Unterbrechungspunkte auf den Anfang dieser Funktionen. Da der C-Compiler einzelne Anweisungen im generierten Code nicht markiert, sind ohne Kenntnis des generierten Codes Unterbrechungspunkte nur am Funktionsanfang sinnvoll. Ausserdem ist zu beachten, dass Bibliotheksroutinen verwendet werden.

Um eine Uebersicht ueber alle gesetzten Unterbrechungspunkte zu erhalten, gibt es das Kommando:

```
$b
```

In der zugehoerigen Ausgabe ist ein count-Feld zu sehen. Ein Unterbrechungspunkt wird count-1 mal uebergangen, ehe es zum Stop kommt. Das command-Feld kann ein adb Kommando enthalten, das jedesmal, wenn der Unterbrechungspunkt passiert wird, zur Ausfuehrung kommt. In diesem Beispiel enthaelt das command-Feld kein Kommando.

Wenn man sich die ersten Befehle der Funktion settab ausgeben laesst, ist zu sehen, dass der oben festgelegte Unterbrechungspunkt genau auf den Befehl nach dem Aufruf der C-Routine gesetzt ist. Diese Befehle koennen durch

```
settab,5?ia
```

ausgegeben werden, wobei durch 5 die Anzahl der Befehle festgelegt wird. Eine andere Moeglichkeit waere:

```
settab,5?i
```

wodurch bei der Ausgabe nur der bei der Startadresse beginnende Befehl ausgegeben wird.

Durch das ? Kommando wird auf Adressen in a.out zugegriffen. Wenn durch ein adb Kommando mehrere Elemente abgefragt werden, so wird die aktuelle Adresse um die Anzahl der Byte erhoeht, die fuer die Erfuellung des Kommandos noetig sind. In diesem Beispiel werden 5 Befehle ausgegeben und die aktuelle Adresse wird um 18 (dezimal) erhoeht.

Soll das Programm unter der Steuerung von adb abgearbeitet werden, geschieht das durch

```
:r
```

Das Loeschen eines Unterbrechungspunktes, z.B. den auf den Anfang der Funktion settab gesetzten, geschieht durch:

```
settab:d
```

Durch die Eingabe von:

```
:c
```

kann die Programmabarbeitung nach dem Unterbrechungspunkt fortgesetzt werden. Nach dem naechsten Programmstop (in diesem Fall beim Unterbrechungspunkt am Anfang von open), koennen durch adb - Kommandos Teile des Speicherinhaltes ausgegeben werden, z.B. mit:

```
$C
```

im Stack-Backtrace oder durch

```
tabs/8x
```

drei Zeilen mit je 8 Elementen des Array-Tabs. Zu diesem Zeitpunkt (unmittelbar nach dem Aufruf von open) ist im C-Programm settab bereits abgearbeitet worden, wobei jedem achten Element von tabs der Wert 1 zugewiesen wurde.

2.4. Setzen von weiteren Unterbrechungspunkten

Zunaechst wird die Programmabarbeitung im Beispiel 6 durch

```
:c
```

fortgesetzt. Durch den Unterbrechungspunkt bei read erfolgt ein Programmstop.

Wird danach durch

```
:c
```

der Programmtest normal fortgesetzt, wird dies wie in allen vorangegangenen Faellen durch

```
a.out:running
```

quittiert.

Nachdem durch die Aufrufe von read das Wort "This" eingelesen wurde, folgt in der Eingabedatei ein Tabulatorzeichen und somit der Aufruf von tabpos, wo es zum Halt wegen des Unterbrechungspunktes kommt. Bei der Umwandlung des Tabulatorzeichens in Leerzeichen wuerde es oft zum Aufruf von tabpos kommen. Deshalb sollte dieser Unterbrechungspunkt geloescht werden. Dies geschieht durch:

```
tabpos:d
```

Die verbleibenden Unterbrechungspunkte sollen neu definiert

werden, um bei jedem Durchlauf ein Kommando zur Ausfuehrung zu bringen. Im ersten Fall geschieht dies durch:

```
settab:b settab,5?ia   +
```

was bei jedem Durchlauf dieses Punktes die Ausgabe der ersten 5 Befehle von settab zur Folge hat.

+ Bei Nutzung einer frueheren Version von adb muessen die Kommandos in folgender Form eingegeben werden:

```
settab:b   settab,5?ia;0
read,3:b   main.c?C;0
settab:b   settab,5?ia;0
```

Die Zeichenfolge ;0 setzt Punkt (.) zu 0 und stoppt am Haltepunkt. Die Forderung, dass zweimal ein Haltepunkt uebergangen wird und erst beim dritten Haltepunkt ein Programmstop erfolgt, erreicht man ueber das Kommando

```
read,3:b main.c?C;0
```

Der Inhalt der Variablen C in der Funktion main soll dadurch als ASCII-Zeichen ausgegeben werden.

Das Semikolon ; trennt verschiedene adb-Kommandos innerhalb einer Zeile.

Ein Haltepunkt kann ueberschrieben werden ohne vorher den alten Inhalt zu loeschen.

```
settab:b  settab,5?ia;*
```

Um eine Uebersicht ueber alle gesetzten Unterbrechungspunkte zu bekommen, wird

```
$b
```

ausgefuehrt.

Wie Beispiel 6 zeigt, wird bei Abarbeitung des Programms unter adb entsprechend der Unterbrechungspunkte die Abarbeitung unterbrochen, und es werden die jeweiligen Kommandos ausgefuehrt.

An dieser Stelle sei noch darauf hingewiesen, dass beim Kommando

```
:c
```

der Prozess mit dem Signal fortgesetzt wird, durch das er vorher unterbrochen wurde. Durch

```
:c 0
```

wird dieses Signal nicht uebergeben.

Ausserdem soll noch darauf hingewiesen werden, dass der Wert von Punkt (.) durch die Abarbeitung eines Programmes unter adb nicht veraendert wird (auch nicht durch Kommandos bei Unterbrechungspunkten).

Wenn man das Programm in Beipiel 3 unter der Ausnutzung eben erlaeuterter adb - Moeglichkeiten abarbeitet, laesst sich ohne Unterbrechung die Abarbeitung der Funktionen f, g und h verfolgen.

Wie in Beipiel 7 zu sehen, wird zunaechst

```
adb ex3 -
```

ausgefuehrt (ex3 enthaelt dann das ausfuehrbare Programm zum C-Programm in Beispiel 3).

Zunaechst werden folgende Unterbrechungspunkte gesetzt und der Programmtest gestartet:

```
h:b hcnt/d; h.hi/; h.hr/
g:b gcnt/d; g.gi/; g.gr/
f:b fcnt/d; f.fi/; f.fr/
:r
```

Jede der angegebenen Variablen soll als Dezimalzahl protokolliert werden. Dabei muss d in jeder Kommandofolge nur einmal angegeben werden und bleibt dann eingestellt. Bei der zugehoerigen Ausgabe nach Beispiel 7 wird zweierlei sichtbar. Zum einen wird die Richtigkeit der Kommandozeile erst beim dynamischen Programmtest ueberprueft. Das heisst, alle Fehler werden erst zur Laufzeit festgestellt, wobei es dann zum Programmstop beim entsprechenden Speicherplatz kommt.

Im folgenden etwas zur Behandlung von Registervariablen in adb: Adb benutzt die Symboltabelle, um Adressen von Variablen zu bestimmen. Registervariable, wie f.fr , haben Pointer zu nicht initialisierten Plaetzen im Stack und initiieren daher die Meldung "symbol not found".

Eine andere Moeglichkeit, sich die Daten in diesem Beispiel ausgeben zu lassen, ist die Verwendung der Variablen in den Prozeduraufrufen. Dies geschieht durch:

```
f:b fcnt/d; f.a/; f.b/; f.fi/
g:b gcnt/d; g.p/; g.q/; g.gi/
h:b hcnt/d; h.x/; h.y/; h.hi/
```

Danach kann durch:

```
:c
```

die Programmabarbeitung fortgesetzt werden.

Der Operator / wird anstelle von ? verwendet, was ein Lesen von Werten von corefile (hier der Testprozess unter adb) bewirkt. Wie in Beispiel 7 zu sehen ist, hat die Ausgabe stets dasselbe Format, z.B. wird fuer die Funktion f stets der Wert und der Name der externen Variablen fcnt ausgegeben, ausserdem die Adresse im Stack und die Werte von a, b und fi.

Wenn der Programmtest nicht unterbrochen wuerde, kaeme es zum Ausdruck vieler Seiten, ehe die Abarbeitung wegen Stack- Ueberlaufs abgebrochen wuerde. Durch die Anweisung:

```
f:b fcnt/d; f.a/"="d; f.b/"b="d; f.fi/"fi="d
```

kann die Ausgabe angenehmer gestaltet werden. Wie dies fuer alle Funktionen moeglich ist und welche Gestalt dann die Ausgabe hat, zeigt Beispiel 7.

2.5. Andere Moeglichkeiten bei der Arbeit mit Unterbrechungspunkten

- Die Uebergabe von Argumenten zu einem Programm sowie die Umlegung von Standard-Input und -Output ist durch:

  ```
  :r arg1 arg2 ... <infile >outfile
  ```

  moeglich. Dieses Kommando bricht den Testlauf ab und startet erneut a.out.

- Der Test im Einzelschritt ist durch

  ```
  :s
  ```

  moeglich. Dieses Kommando ermoeglicht auch den Programmstart. Es kommt dann zum Stop nach Ausfuehrung des ersten Befehls.

- Adb ermoeglicht es durch

  ```
  address:r
  ```

  ein Programm ab einer beliebigen Adresse abzuarbeiten.

- Im count - Feld kann angegeben werden, wieviele Unterbrechungspunkte uebergangen werden sollen, bevor es zum ersten Programmstop kommt:

,n:r
oder bei Programmfortsetzung
,n:c

- Ein Programm kann durch
  address:c
  an einer beliebigen Adresse fortgesetzt werden.
- Das Programm, das gerade getestet wird, kann durch
  :k
  abgebrochen werden.

3. Maps

WEGA unterstuetzt fuer ausfuehrbare Files verschiedene Formate. Diese werden benoetigt, um dem Lader anzuzeigen wie ein File zu laden ist.

Der am haeufigsten vorkommende Filetyp ist dabei die File mit der Magic-Number E707 (E707-Files), die zum Beispiel durch cc pgm.c erzeugt wird.

Files mit der Magic-Number E711 (E711-Files) werden vom Compiler durch cc -i pgm.c erzeugt.

Adb arbeitet mit diesen verschiedenen Formaten fuer objfile und realisiert den Zugriff zu den einzelnen Segmenten ueber die Maps (Beispiel 8).

Diese Maps koennen mit

```
$m
```

ausgegeben werden.

E707-Files Befehle und Daten sind vermischt. Dies macht es fuer adb unmoeglich, Daten von Befehlen zu unterscheiden und manche der ausgegebenen symbolischen Adressen koennen falsch sein (z.B. werden Datenadressen als eine Funktionsadresse + Verschiebung ausgegeben).

E711-Files Befehle und Daten sind getrennt. In diesem Fall werden die Daten durch einen eigenen Satz von Segmentregistern auf den Speicher abgebildet. Somit ist die Basisadresse des Datensegments ebenfalls relativ zu Null definiert. Da sich in diesen Fall die Adressen in objfile ueberlappen, ist es hier unumgaenglich fuer den Zugriff den Operator ?* zu verwenden (ueberlappen sich die Segmente nicht, wird auch bei ? ueber das zweite Map-Segment zugegriffen, wenn die Berechnung der Fileadresse in objfile aus der Kommandoadresse ueber des erste Map-Segment scheiterte). Das zugehoerige corefile enthaelt kein Textsegment. Der erste Teil der corfile verweist immer auf das Datensegment (bei E707-Files existiert praktisch nur das Datensegment, welches auch die Befehle enthaelt, weshalb bei diesem Filetyp auch der Textteil enthalten ist) und das zweite Mapsegment verweist immer auf den Stackbereich. Da sich diese beiden Teile nie ueberlappen koennen, genuegt im Normalfall der Zugriff durch /, um Adressen im Kommando auf Core-File-Adressen abzubilden. Nur wenn in der Map selbst Veraenderungen ausgefuehrt werden sollen, ist fuer den Bezug auf das zweite

Core-Map-Segment der Operator /* noetig.

Beispiel 9 zeigt die Ausgabe fuer Maps von E707- und E711-Files.

Dabei werden die Felder b, e und f fuer die Berechnung der Fileadresse aus der Kommandoadresse wie folgt genutzt:

f1 gibt die Laenge des File-Headers am Fileanfang an (020 Bytes fuer a.out und 02000 Bytes fuer core).

f2 gibt die Verschiebung vom Fileanfang bis zum Datenbereich an. Bei E707-Files mit gemischten Befehls- und Datensegmenten ist dieser Wert immer gleich der Header-Laenge. Bei E711-Files ist dieser Wert die Summe aus Header-Laenge und der Laenge des Textsegments.

b gibt stets die Anfangsadresse eines Segments an.

e gibt stets die Endadresse eines Segments an.

Ist in einem Kommando die Adresse A angegeben so wird die Fileadresse daraus wie folgt berechnet:

```
b1<A<e1 => file address = (A-b1)+f1
b2<A<e2 => file address = (A-b2)+f2
```

Der Zugiff auf Speicherplaetze kann unter Benutzung einiger durch adb zur Verfuegung gestellter Variablen (Ausgabe durch $v) geschehen:

```
b    Basisadresse des Datensegments
d    Laenge des Datensegments
s    Laenge des Stack-Segments
t    Laenge des Textsegments
m    Filetyp (E707 und E711)
```

In Beispiel 9 sind jeweils die Werte aller Variablen, die ungleich Null sind mitangegeben. Vom Nutzer kann auf diese Variablen wie folgt zugegriffen werden:

```
<b
```

im Adressfeld bewirkt die Verwendung des Wertes von b als Adresse. Eine Aenderung des Inhaltes der Variablen ist z.B. durch

```
02000>b
```

moeglich. Hier wurde b der Wert 02000 zugewiesen. Diese Variablen sind besonders dann sehr nuetzlich, wenn das untersuchte File ein ausfuehrbares oder eine corefile ist.

Zum Setzen dieser Variablen liest adb den Header von corefile und entnimmt diesem die entsprechenden Werte. Wenn das zweite File im adb - Aufruf kein corefile ist, werden die Werte dem Header von objfile entnommen.

4. Weitere Anwendungen fuer adb

Es ist moeglich bei adb die einzelnen Formate zu kombinieren um Ausgaben uebersichtlich zu gestalten. Im weiteren soll dies an einigen Beispielen erlaeutert werden.

4.1. Formatierte Ausgaben

Die Zeile

```
<b,-1/4o4^8Cn
```

bewirkt die Ausgabe von 4 Oktalzahlen (2 Byte Laenge) sowie danach die ASCII-Interpretation dieser 8 Byte aus corefile. Die einzelnen Bestandteile dieses Kommandos haben folgende Bedeutung:

<b	Die Basisadresse des Datensegments
<b,-1	Ausgabe von der Basisadresse an bis zum Fileende. Ein negativer Wert fuer count bewirkt die Wiederholung eines Kommandos bis zu einem Fehler (hier Fileende).

Das Format 4o4^8Cn setzt sich aus folgenden Bestandteilen zusammen:

4o	Ausgabe von 4 Worten als Oktalzahlen
4^	Zuruecksetzen der aktuellen Adresse um 4 Worte (auf den urspruenglichen Wert vor Beginn der Ausgabe)
8C	Ausgabe von 8 aufeinanderfolgenden ASCII-Zeichen unter Benutzung der Festlegung, dass Zahlen im Bereich 0 bis 037 als @ gefolgt vom entsprechenden Zeichen im Bereich 0140 bis 0177 ausgegeben werden; @ wird als @@ ausgegeben.
n	Ausgabe eines Zeilenwechsels

Durch das Kommando

```
<b,<d/4o4^8Cn
```

wird dieselbe Ausgabe wie oben realisiert, nur das die Ausgabe am Ende des Datensegments abgebrochen wird (<d liefert die Laenge des Datensegments in Byte).

Die Moeglichkeiten der formatierten Ausgabe von adb koennen nun mit der Eingabe vorbereiteter adb - Kommandofiles kombiniert werden, um haeufig wiederkehrende Ausgaben zu realisieren. Der adb - Aufruf hat dann folgende Gestalt:

```
adb a.out core < dump
```

Ein Beispiel fuer ein solches Kommandofile (dump) ist:

```
120$w
4095$s
$v
=3n
$m
=3n"C Stack Backtrace"
$C
=3n"C External Variables"
$e
=3n"Registers"
$r
0$s
=3n"Data Segment"
<b,-1/8ona
```

Die einzelnen Bestandteile dieser Kommandos haben folgende Bedeutung:

120$w	setzt die Breite der Ausgabe auf 120 Zeichen pro Zeile (Standard ist 80). Adb versucht Adressen symbolisch auszugeben und zwar in der Form: symbol + offset.
4095$s	stellt den Maximalwert fuer offset als Verschiebung zur naechstgelegene symbolischen Adresse auf 4095 (Standard ist 255).
=	ermoeglicht es, nachfolgende Zeichenketten unveraendert auszugeben. Das Kommando =3n"C Stack Backtrace" bewirkt die Ausgabe von 3 Leerzeilen mit anschliessender Ausgabe der Zeichenkette 'C Stack Backtrace'.
$v	veranlasst die Ausgabe aller adb - Variablen mit Werten ungleich Null (Beispiel 8).
0$s	setzt die maximale Verschiebung bei der Verwendung von Symbolen auf 0. Somit wird die symbolische Ausgabe unterdrueckt und durch eine Oktalausgabe ersetzt. Das ist nur bei der Ausgabe des Datensegments moeglich.
<b,-1/8ona	gibt das ganze File achtspaltig oktal aus. Beispiel 11 zeigt das Ergebnis einiger formatierter Ausgaben fuer das C - Programm in Beispiel 10.

4.2. Directory - Ausgabe

Eine andere Moeglichkeit zur Illustration der adb - Arbeit

zeigt Beispiel 12. Hier wird die Ausgabe von Directory - Inhalten, so wie sie im Directory abgespeichert sind, dokumentiert. Ein Directory - Eintrag besteht aus einer ganzen Zahl, der inumber und dem Filenamen (14 ASCII-Zeichen).

```
adb dir -
=n8t"Inum"8t"Name"
0,-1?u8t14cn
```

bewirkt die gewuenschte Ausgabe.

u	wird in diesem Beispiel die inumber als vorzeichenlose ganze Zahl ausgegeben
8t	veranlasst die notwendigen Tabulatorspruenge
14c	Ausgabe des Filenamens (14 ASCII-Zeichen)

4.3. Ilist Ausgabe

Auch der Inhalt der Ilist eines Filesystems (hier von /dev/src) kann durch adb ausgegeben werden.

```
adb /dev/src -
02000>b
?m<b
<b,-1?"flags"8ton"links,uid,gid"8t3dn",
  "size"8tDn"addr"8t20un"times"8t2YnY2na
```

?m<b	In diesem Beispiel wechselt die Basisadresse der Map zu 02000, bevor die Ilist Ausgabe erfolgt
2YnY	Ueber diesen Operator erfolgt die Ausgabe des Datums fuer die Erstellung und des letzten Zugriffs fuer diese Datei

Beispiel 12 zeigt Teile dieser Anwendung fuer eine Directory und ein Filesystem.

4.4. Konvertieren von Zahlen

Adb kann auch zur Zahlenkonvertierung benutzt werden. Zum Beispiel wird durch

```
072 = odx
```

folgendes ausgegeben:

```
072  58   %3a
```

Also der oktale, dezimale und hexadezimale Wert von 072. Das Format wird gespeichert, so das bei der Eingabe weiterer Zahlen ebenfalls die zugehoerigen Oktal-, Dezimal- und Hexadezimalwerte ausgegeben werden.

Werte von Zeichen koennen ebenso konvertiert werden:

```
'a' = crb
```

erzeugt

```
%0064
```

Adb Kann auch zur Berechnung von Ausdruecken benutzt werden. Allerdings muss davon abgeraten werden, da alle zweistelligen Operatoren dieselbe Prioritaet besitzen.

5. Aendern von Files

Durch adb koennen Files mit Hilfe des Write-Kommandos (w oder W) geaendert werden. Oft wird dieses Kommando zusammen mit dem Such-Kommando (l oder L) verwendet. Beide Kommandos haben folgenden Syntax:

```
address range  file designator  command  argument
```

address range — gibt den Adressbereich in der Datei an. Die Angabe des Adressbereichs kann erfolgen ohne Angaben, als eine Adresse, als Adressbereich oder als Punktadresse (.).

file designator — ? fuer a.out oder / fuer core

command — Write-Kommando (w oder W) oder Such-Kommando (l oder L)

argument — ein Ausdruck in der Form einer Dezimalzahl, Oktalzahl oder einer Zeichenkette

Durch w koennen 2 Byte und durch W 4 Byte geaendert werden und mit l kann ein Wort im Speicher mit dem Wert argument gesucht werden. Bei L wird ein Doppelwort mit diesem Wert gesucht.

Beispiel:

```
0,1000?l    suchen in der Datei von Adresse 0 bis 1000

  1000?l    suchen in der Datei von Adresse 1000 bis
            Dateiende

     ?l     suchen in der gesamten Datei
```

Um ein File modifizieren zu koennen, muss adb folgendermassen aufgerufen werden:

```
adb -w file1 file2
```

So kann z.B im Programm in Beispiel 10 durch

```
adb -w ex7 -
.?l 'Th'
.?W 'The '
```

das Wort 'Th' ist in 'The ' geaendert wurden. Durch .?l wird die suche bei Punkt begonnen und bei Uebereinstimmung der Werte abgebrochen. Durch .?W wird in die a.out geschrieben. Fuer E711-Files wird zum Schreiben ?* benutzt.

Haeufiger wird allerdings folgende Form benutzt:

```
?l 'Th'; ?s
```

wodurch beim ersten Auftreten von 'Th' die ganze Zeichenkette ausgegeben wird und die Adresse dieser Zeichenkette in Punkt (.) gespeichert ist.

Ein anderes Beispiel waere der Test eines C-Programms mit einem internen Flag. Durch adb kann dieses Flag gesetzt werden bevor das Programm abgearbeitet wird:

```
adb a.out -
:s arg1 arg2
flag/w 1
:c
```

:s der Prozess wird im Einzelschrittbetrieb fortgesetzt, oder wenn noch nicht vorhanden, gestartet.

w der Wert von flag wird im Adressraum des Unterprozesses geaendert.

6. Besonderheiten

1. Funktionsaufrufe und Argumente werden durch die C-Save-Routine auf den Stack gebracht, die am Anfang jeder Funktion aufgerufen wird. Werden Unterbrechungspunkte auf den Eintrittspunkt der Funktion gesetzt, erfolgt die Unterbrechung vor Aufruf dieser Routine. Damit ist der Stackinhalt so, als waere die Funktion noch nicht gerufen worden.

2. Bei Ausgaben von Adressen benutzt adb die Text- und Datensymbole von objfile (a.out). Dies fuehrt manchmal zu falschen Symbolen fuer Daten. Deshalb sollte bei Ausgaben von Daten immer / und bei Befehlen und Text ? verwendet werden.

3. In adb koennen die Werte der lokalen Variablen nicht ausgegeben werden.

7. Programmbeispiele

Beispiel 1:

```
char *charp = "this is a sentence"

main (argc,argv)
int     argc;
char    **argv;
{
        int     fd;
        char    cc;
        if (argc < 2)
        {
                printf ("Input file missing\n");
                exit (8);
        }
        if ((fd = open(argv[1],0)) == -1)
        {
                printf ("%s : not found\n", argv[1]);
                exit (8);
        }
        charp = "hello";
        printf ("debug 1 %s\n", charp);
        while (charp++)
                write (fd, *charp, 1);
}
```

Beispiel 2:

```
adb a.out core

ADB: P8000 1.6
? $c
no process
? $C
no process
? $r
r0      %0000
r1      %0000
r2      %0000
r3      %0000
r4      %0000
r5      %0000
r6      %0000
r7      %0000
r8      %0000
r9      %0000
r10     %0000
r11     %0000
r12     %0000
r13     %0000
r14     %0000
sp      %0000
fcw     %0000
pc      %0000
_main:          jr              _main+%7c
? $e
_environ:       %ffbc
_charp:         %1400
__iob:          %113c
__sobuf:        %0000
__lastbu:       %0f26
__sibuf:        %0000
nd:             %133c
_end            %0000
_deverr:        %0000
_errno:         %0009
? $m
? map           'a.out'
        b1 = %0         e1 = %f3a       f1 = %28
        b2 = %0         e2 = %f3a       f2 = %28
/ map           'core'
        b1 = 0          e1 = %1400      f1 = %400
        b2 = %fa00      e2 = %10000     f2 = %1800
? *charp/s _end+%c4:
data address not found
? charp/s
_charp:
? main.argc/d
Sorry, local variable names not implemented
? $q
```

Beispiel 3:

```
int  fcnt,gcnt,hcnt;
h(x,y)
{
        int hi; register int hr;
        hi = x+1;
        hr = x-y+1;
        hcnt++ ;
        f(hr,hi);
}

g(p,q)
{
        int gi; register int gr;
        gi = q-p;
        gr = q-p+1;
        gcnt++ ;
        h(gr,gi);
}

f(a,b)
{
        int fi; register int fr;
        fi = a+2*b;
        fr = a+b;
        fcnt++ ;
        g(fr,fi);
}

main()
{
        f(1,1);
}
```

Beispiel 4:

```
adb

ADB: P8000 1.6
:r
? .,$c
_f()
_g()
_h()
_f()
:
? ,5$C
Local variables not implemented
_h()
        stack frame:
                %02f6: %0000
                %02f8: %0000
                %02fa: %0000
                %02fc: %0000
                %02fe: %007a (return address)
-g()
        stack frame:
                %0300: %21b4
                %0302: %10db
                %0304: %10d9
                %0306: %10db
                %0308: %00ae (return address)
_f()
        stack frame:
                %030a: %10d9
                %030c: %0002
                %030e: %21b4
                %0310: %0002
                %0312: %0048 (return address)
_h()
        stack frame:
                %0314: %10d7
                %0316: %10d8
                %0318: %10d9
                %031a: %10d8
                %031c: %007a (return address)
_g
                %031e: %21b0
                %0320: %10d9
                %0322: %10d7
                %0324: %10d9
                %0326: %00ae (return address)
? fcnt/d
_fcnt:          2157
? gcnt/d
_gcnt:          2157
? hcnt/d
_hcnt:          2157
```

```
? h.x/d
sorry, local variable names not implemented
? $q
```

Beispiel 5:

```
#define MAXLINE         80
#define YES             1
#define NO              0
#define TABSP           8

char input[] = "data";
int  tabs[MAXLINE];

main()
{
        int fd;
        int col, *ptab;
        char c;
        ptab = tabs;
        settab (ptab);
        col =1;
        if ((fd = open(input, 0)) == -1
        {
                printf ("%s : not found\n", input);
                exit (8);
        }
        while (read (fd, &c, 1) > 0)
        {
                switch (c)
                {
                        case '\t':
                        while (tabpos (col) != Yes)
                        {
                        putchar ( ' ' );
                        col++;
                        }
                        break;
                        case '\n':
                        putchar ( '\n' );
                        col = 1;
                        break;
                        default:
                        putchar (c);
                        break;
                }
        }
}

tabpos (col)
int col;
{
        if (col > MAXLINE)
                return (YES);
        else
                return (NO);
}
```

```
settab (tabp)
int *tabp;
{
        int i;

        for (i=0; i<=MAXLINE; i++)
                (i % TABSP) ? (tabs[i]=NO : (tabs[i]=YES);
}
```

Beispiel 6:

```
adb a.out -

ADB: P8000 1.6
? settab:b
? open:b
? read:b
? tabpos:b
? $b
breakpoints
count           bkpt            command
1               _tabpos
1               _read
1               _open
1               _settab
? settab,5?ia
_settab:        jr              _settab+%48
_settab+%2:     clr             %0002(sp)
_settab+%6:     cp              %0002(sp),#%0050
_settab+%c      jr              gt,_settab+%44
_settab+%e      ld              r3,%0002(sp)
_settab+%12:
? settab,5?i
_settab:        jr              _settab+%48
                clr             %0002(sp)
                cp              %0002(sp),#%0050
                jr              gt,_settab+%44
                ld              r3,%0002(sp)
? :r
a.out: running
breakpoint      _settab:        jr      _settab+%48
? settab:d
? :c
a.out: running
breakpoint      _open           ld      r0,r7
? $C
_open()
                stack frame:
                        %ffb2: %0048 (return address)
_main()
                stack frame:
                        %ffb4: %0000
                        %ffb6: %0001
                        %ffb8: %0fd4
                        %ffba: %0000
                        %ffbc: %0022 (return address)
? tabs/8x
_tabs:  %0001 %0000 %0000 %0000 %0000 %0000 %0000 %0000
        %0001 %0000 %0000 %0000 %0000 %0000 %0000 %0000
        %0001 %0000 %0000 %0000 %0000 %0000 %0000 %0000
? :c
a.out: running
breakpoint      _read:          ld r0,r7
? :c
```

```
a.out: running
breakpoint      _read:          ld r0,r7
? tabpos:d
? settab:b settab,5?ia
? settab,5:b settab,5?ia; 0
? read,3:b tabs/8x
? $b
breakpoints
count           bkpt            command
3               _read           tabs/8x
1               _settab         settab,5?ia; 0
1               _open
? :c
a.out: running
T_tabs: %0001 %0000 %0000 %0000 %0000 %0000 %0000 %0000
h_tabs: %0001 %0000 %0000 %0000 %0000 %0000 %0000 %0000
i_tabs: %0001 %0000 %0000 %0000 %0000 %0000 %0000 %0000
breakpoint      read:           ld r0,r7
? $q
```

Beispiel 7:

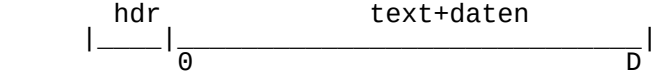
```
adb ex3 -

ADB: P8000 1.6
? h:b hcnt/d; h.hi/; h.hr/
? g:b gcnt/d; g.gi/; f.fr/
? :r
ex3: running
_gcnt:                  0
Sorry, local variable names not implemented
? f:b fcnt/d; f.a/"a = "d; f.h/"b = "d; f.fi/"fi = "d
? g:b gcnt/d; g.p/"p = "d; g.q/"q = "d; g.gi/"gi = "d
? h:b hcnt/d; h.x/"x = "d; h.y/"y = "d; h.hi/"hi = "d
? :r
ex3: running
_fcnt:                  0
Sorry, local variables names not implemented
? $q
```

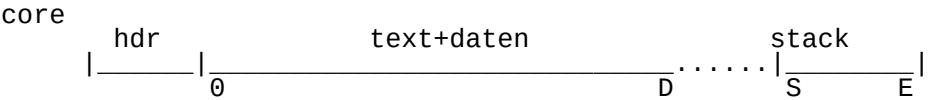
Beispiel 8:

E707 files

a.out

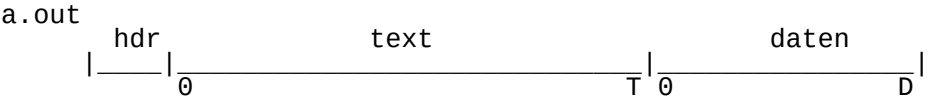


core

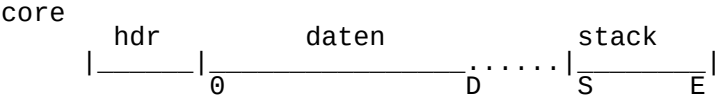


E711 files (getrennter Befehls- und Datenraum)

a.out



core



Die folgenden adb variablen werden gesetzt.

	E707	RM	E711
b Anfang des Datensegments	0	b	0
d Laenge des Datensegments	D	D-B	D
s Laenge des Stack-Segments	S	S	S
t Laenge des Textsegments	0	T	T

Beispiel 9:

```
adb mapE707 coreE707

ADB: P8000 1.6
? $m
? map   'mapE707'
                b1 = %0         e1 = %dc        f1 = %38
                b2 = %0         e2 = %dc        f2 = %38
/ map   'coreE707'
                b1 = %0         e1 = %100       f1 = %400
                b2 = %200       e2 = %1000      f2 = %500
? $v
variables
  address
e = %a4
  other
d = %100
m = %e707
s = %fe00
? $q
```

```
adb mapE711 coreE711

ADB: P8000 1.6
? $m
? map   'mapE711'
                b1 = %0         e1 = %100       f1 = %38
                b2 = %0         e2 = %0         f2 = %138
/ map   'coreE711'
                b1 = %0         e1 = %100       f1 = %400
                b2 = %200       e2 = %10000     f2 = %500
? $v
variables
  address
e = %a4
  other
d = %100
m = %e711
s = %fe00
t = %100
? $q
```

Beispiel 10:

```
char    str1[]  = "This is a character string";
int     one =     1;
int     number = 456;
long    lnum =  1234L;
char    str2[]  "This is the second character string";

main()
{
one = 2;
}
```

Beispiel 11:

```
adb mapE711 coreE711

ADB:P8000 1.6
? <b,-1/8oa
_str1:
  052150 064563 020151 071440 060440 061550 060562 060543

_str1+%10:
  072145 071040 071564 071151 067147 000000 000001 000710

_lnum:
  000000 002322 037640 000000 052150 064563 020151 071440

_str2+%8:
  072150 062440 071545 061557 067144 020143 064141 071141

_str2+%18:
  061564 062562 020163 072162 064556 063400 000000 177662

_environ+%2:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%12:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%22:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%32:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%42:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%52:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%62:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%72:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%82:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%92:
  000000 000000 000000 000000 000000 000000 000000 000000

_environ+%a2:
  000000 000000 000000 000000 000000 000000 000000 000000
? <b,20/4on^8Cn
```

```
_str1:          052150 064563 020151 071440 This is
                060440 061550 060562 060543 a charac
                072145 071040 071564 071151 ter stri
                067147 000000 000001 000710 ng@`@`@`@a@aH
                000000 002322 037640 000000 @`@`@dR? @`@`
                052150 064563 020151 071440 This is
                072150 062440 071545 061557 the seco
                067144 020143 064141 071141 nd chara
                061564 062562 020163 072162 cter str
                064556 063400 000000 177662 ing@`@`@`@2
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
                000000 000000 000000 000000 @`@`@`@`@`@`@`@`
? <b,20/4o4^8t8cna
_str1:          052150 064563 020151 071440 This is
_str1+%8:       060440 061550 060562 060543 a charac
_str1+%10:      072145 071040 071564 071151 ter stri
_str1+%18:      067174 000000 000001 000710 ngH
_lnum:          000000 002322 037640 000000 R?
_str2:          052150 064563 020151 071440 This is
_str2+%8:       072150 062440 071545 061557 the seco
_str2+%10:      067144 020143 064141 071141 nd chara
_str2+%18:      061564 062562 020163 072162 cter str
_str2+%20       064556 063400 000000 177662 ing2
_environ+%2:    000000 000000 000000 000000
_environ+%a:    000000 000000 000000 000000
_environ+%12:   000000 000000 000000 000000
_environ+%1a:   000000 000000 000000 000000
_environ+%22:   000000 000000 000000 000000
_environ+%2a:   000000 000000 000000 000000
_environ+%32:   000000 000000 000000 000000
_environ+%3a:   000000 000000 000000 000000
_environ+%42:   000000 000000 000000 000000
_environ+%4a:   000000 000000 000000 000000
_environ+%52:
? <b,10/2b8t^2cn
_str1:  %0054   %0068 Th
        %0069   %0073 is
        %0020   %0069 i
        %0073   %0020 s
        %0061   %0020 a
        %0063   %0068 ch
        %0061   %0072 ar
        %0061   %0063 ac
        %0074   %0065 te
        %0072   %0020 r
? $q
```

Beispiel 12:

```
adb dir -

ADB: P8000 1.6
? =nt"Inode"t"Name"
? 0,-l?ut14cn
                Inode   Name
%0000              2    .
                   2    ..
                 102    bin
                 101    usr
                 157    lib
                 164    dev
                 148    etc
                 197    pb.image
                 957    tmp
                 261    wega 3.0
? $q
```

```
adb /dev/src -

ADB: P8000 1.6
? ?m 0 %1000000 1024
? 0,-1?"flags"8ton"links,uid,gid"8t3dn"size"8tDn" \
  addr"8t20un"times"8t2Y2na
%0000:  flags   10000
links,uid,gid   0       0       0
size            0
addr    0       0       0       0       0       0       0
   0    0       0       0       0       0       0       0
   0    0       0       0       0
times   1981 Feb 12 13:50:17    1981 Feb 12 13:50:17
1981 Feb 12 13:50:17
%0040:  flags   040755
links,uid,gid   44      0       0
size            704
addr    3       9984    810     0       0       0       0
   0    0       0       0       0       0       0       0
   0    0       0       0       0
times   1981 Jul 17 16:58:42    1981 Jul 15 10:10:41
1981 Jul 15 10:10:41
%0080:  flags   100664
links,uid,gid   1       25      0
size            34
addr    52      12288   0       0       0       0       0
   0    0       0       0       0       0       0       0
   0    0       0       0       0
times   1981 Jul 16 17:06:34    1981 Jul 16 17:04:23
1981 Jul 16 17:04:23
```

8. Adb - Kommandozusammenfassung

Formatierte Ausgabe

? format	Ausgabe von a.out entsprechend format
/ format	Ausgabe von core entsprechend format
= format	Ausgabe des Wertes von Punkt (.)
?w expr	schreiben eines Ausdrucks in a.out
/w expr	schreiben eines Ausdrucks in core
?l expr	suchen eines Ausdrucks in a.out

Haltepunkte und Programmsteuerung

:b	setzen Haltepunkt
:c	Fortsetzung Programmabarbeitung
:d	loeschen Haltepunkt
:k	Abbruch Programmtestung
:r	Abarbeitung eines Programms unter adb
:s	Einzelschrittabarbeitung

Verschiedenartige Ausgaben

$b	aktuelle Haltepunkte
$c	Backtrace
$C	Backtrace und Ausgabe aller lokalen Variablen
$e	externe Variablen
$f	Gleitpunktregister
$m	Segment maps
$q $Q CTRL D	beenden von adb
$r	Register
$w	setzen Ausgabeformat
$<file	Kommandofile einlesen und abarbeiten
DEL	Abbruch eines laufenden adb-Kommandos

Aufruf von Shell

!	Aufruf von shell

Zuweisung von Variablen

>name	Punkt (.) oder Register

Formate

a	der Wert von Punkt (.)
b	ein Byte oktal
c	ein Byte als ASCII-Zeichen
d	ein Wort dezimal
f	zwei Wort Gleitkomma
i	U8000-Befehl
o	ein Wort oktal
n	Ausgabe Zeilenwechsel
r	Ausgabe Leerzeichen
s	durch 0 begrenzte Zeichenkette
nt	uebertragen der naechsten n Leerzeichen in Tab's
u	ein Wort als vorzeichenlose ganze Zahl
x	hexadezimal
Y	Datum
^	Dekrementieren von Punkt (.)

Ausdruckzusammenfassung

Ausdruck Komponenten

dezimal	256
oktal	0277
hexadezimal	%ff
Symbole	flag _main main.argc
Variable	<b
Register	<pc <r0

Dyadic Operatoren

+	Addition
-	Subtraktion
*	Multiplikation
%	ganzzahlige Division
&	bitweises UND
|	bitweises ODER (exklusiv)
#	Aufrundung

Monadic Operatoren

~	Komplement
*	zu Adresse
-	Negation

Das File-System-Testprogramm fsck

Inhaltsverzeichnis Seite

1. Einfuehrung

Nachdem das WEGA-System hochgefahren worden ist, wird automatisch das Datei-Testprogramm (fsck) geladen und abgearbeitet. Fsck ist ein interaktives Programm, das die redundanten strukturellen Informationen des WEGA-Dateisystems fuer Uebereinstimmungskontrollen benutzt. Bei Nichtuebereinstimmung erzeugt fsck eine Meldung, die der Nutzer entsprechend behandeln oder ignorieren kann. Durch diese Massnahme wird eine zuverlaessige Umgebung (environment) bei der Dateispeicherung auf dem Hard-Disk-Laufwerk gesichert.
Jede Dateiaktivitaet (Erzeugung, Aenderung oder Streichung) aktualisiert mindestens einen der fuenf Dateibloecke, die WEGA zur Ueberwachung der Dateien benutzt. Fsck ueberprueft den Inhalt der redundanten Felder dieser Bloecke und vergleicht die Informationen in den Bloecken mit denen in den Dateien. Wurde ein Fehler festgestellt, wird eine Nachricht an den Bediener ausgegeben. Die meisten Fehler erlauben dem Nutzer fsck fortzusetzen oder zu beenden. Ernsthafte Fehler, wie z.B. falsche Optionen, zwingen fsck zum Abbruch.

2. Aktualisierung des Dateisystems

2.1. Allgemeines

Immer wenn eine Datei erzeugt, veraendert oder geloescht wird, erfolgt durch das WEGA-System eine Aktualisierung des Superblocks, der Inodes, der indirekten Bloecke, der Datenbloecke (Directories und Dateien) und der Bloecke der Freiblockliste. Aktualisierungsanforderungen werden, um ein konsistentes (widerspruchfreies) Dateisystem zu erhalten, in einer festgelegten Reihenfolge durchgefuehrt. Die Kenntnis dieser Reihenfolge erleichtert spaeter das Beseitigen von aufgetretenen Fehlern.

2.2. Superblock

Der Superblock enthaelt Angaben ueber die Groesse des Dateisystems, die Groesse der Inode-Liste, Teile der Freiblock-Liste, die Anzahl der freien Bloecke, die Anzahl der freien Inodes und ein Teil der freien Inode-Liste. Das root-Dateisystem wird immer eingebunden (mount). Das Schreiben des Superblocks eines verbundenen Dateisystems erfolgt beim Loesen (umount) des Dateisystems oder bei Ausgabe eines 'sync'-Kommandos.

2.3. Inodes

Ein Inode enthaelt Angaben ueber den Typ des Inode (Directory, Daten, Spezial), die Anzahl der mit diesem Inode verbundenen Directoryeintragungen, die Liste der durch dieses Inode angeforderten Bloecke u. ae..
Das Schreiben von Inodes an das Dateisystem erfolgt nach dem Schreiben der Datei, auf die sich das Inode bezieht und beim Auftreten eines 'sync'-Kommandos.

2.4. Indirekte Bloecke

Es gibt drei Arten von indirekten Bloecken: einfach-indirekt, zweifach-indirekt und dreifach-indirekt. Ein einfach-indirekter Block enthaelt ein Verzeichnis von einigen durch das Inode belegten Blocknummern. Jede der 128 Eintragungen in einen indirekten Block entspricht einer Datenblocknummer. Ein zweifach-indirekter Block enthaelt ein Verzeichnis von einfach-indirekten Blocknummern und ein dreifach-indirekter Block enthaelt eine Liste von zweifach-indirekten Blocknummern. Nachdem die indirekten Bloecke durch das Operations-System modifiziert und in einer Warteschlange (queue) fuer das Rueckschreiben angeordnet wurden, erfolgt ihre Ausgabe an das Dateisystem. I/O-Anforderungen werden solange zurueckgestellt, bis WEGA den Puffer benoetigt oder ein 'sync'-Kommando auftritt.

2.5. Daten-Bloecke

Ein Daten-Block enthaelt Dateiinformationen oder Directoryeintragungen. Jede Directoryeintragung besteht aus einem Dateinamen und einer Inodenummer. Das Rueckschreiben der Daten-Bloecke und die Behandlung von I/O-Anforderungen erfolgt analog den indirekten Bloecken.

2.6. Freiblocklisten-Bloecke

Die Freiblocklisten-Bloecke beinhalten alle jene Bloecke, die nicht dem Superblock (ausser dem ersten Freiblocklisten-Block), den Inodes, den indirekten Bloecken oder den Daten-Bloecken zuzuordnen sind. Jeder Freiblocklisten-Block enthaelt einen Zaehler ueber die erfolgten Eintragungen in diesen Freiblocklisten-Block, einen Zeiger auf den naechsten Freiblocklisten-Block und eine Teilliste der freien Bloecke im Dateisystem. Das Rueckschreiben der Freiblocklisten-Bloecke und die Behandlung einer I/O-Anforderung erfolgt analog den indirekten Bloecken.

3. Das fehlerhafte Dateisystem

3.1. Allgemeines

Die Hauptursachen, die zur Entstehung eines fehlerhaften Dateisystems fuehren, sind ein falsches Abfahren (shutdown) des Systems und Hardwarefehler.

3.2. Fehlerhaftes Ab- und Hochfahren des Systems

Zum fehlerhaften Abfahren des Systems fuehren ein nicht eingegebenes 'sync' vor dem 'halt' der CPU, ein Schreibschutz eines eingebundenen Dateisystems und ein Umschalten eines eingebundenen Dateisystems in den off-line-Zustand.
Ein fehlerhaftes Hochfahren (startup) des Systems umfasst die Nichtueberpruefung des Dateisystems auf Widerspruchsfreiheit und die Nichtreparatur bei aufgetretener Fehlern.

3.3. Hardware-Fehler

Jedes Teil der Hardware kann zu beliebigen Zeiten Fehler verursachen. Die Fehler koennen von defekten (bad) Bloecken eines Hard-Disk-Laufwerks bis zur Nichtfunktion des Gesamtgeraets reichen.

4. Erkennung und Korrektur von Fehlern

4.1. Allgemeines

Unter einem ruhenden Dateisystem versteht man ein Dateisystem, das nicht mit dem root-Dateisystem verbunden ist bzw. in welchem nicht geschrieben wird. Ein solches ruhendes Dateisystem kann durch die Uebereinstimmungstests auf strukturelle Vollstaendigkeit ueberprueft werden. Dabei beziehen sich die Uebereinstimmungstests auf die redundanten Daten, d.h. auf einen Teil des Dateisystems. Ein ruhender Zustand waehrend des Dateitests ist wegen der "multipass"-Moeglichkeit von fsck wichtig. Fsck entdeckt und bringt jede Nichtuebereinstimmung zur Anzeige und bietet somit die Moeglichkeit fuer eine interaktive Sofortkorrektur. Dieser Abschnitt zeigt, wie Widersprueche entdeckt und Korrekturmassnahmen durchgefuehrt werden. Die Korrekturen beziehen sich auf die Superbloecke, Inodes, indirekten Bloecke, Datenbloecke mit Directoryeintragungen und freien Bloecke.

4.2. Superblock

Der Superblock ist fuer Fehler besonders anfaellig, da jede Aenderung in den Bloecken des Dateisystems oder der Inodes auch zu einer Modifizierung der Superblocks fuehrt. Fehler treten dann am haeufigsten auf, wenn der Rechner angehalten wird und das letzte, sich auf das Dateisystems beziehende Kommando, kein 'sync'-Kommando war. Die Ueberpruefung des Superblocks auf Widerspruchsfreiheit betrifft die Groesse des Dateisystems , die Groesse der Inode-Liste, die Freiblock-Liste, die Anzahl der freien Bloecke und die Anzahl der freien Inodes.

4.2.1. Die Groesse des Dateisystems und der Inode-Liste

Diese Groessen sind kritisch, da alle anderen Tests des Dateisystems von ihnen abhaengen und durch fsck nur ueberprueft werden koennen, wenn sie sich innerhalb bestimmter Grenzen bewegen. Das Dateisystem muss sowohl groesser als die Anzahl der durch den Superblock als auch groesser als die Anzahl der durch die Inode-Liste benutzten Bloecke sein. Die Anzahl der Inodes muss kleiner 65 535 sein.

4.2.2. Freiblockliste

Die Freiblockliste beginnt im Superblock und setzt sich durch die Freiblocklisten-Bloecke des Dateisystems fort. Jeder Freiblocklisten-Block wird dahingehend ueberprueft, dass weder eine Listenanzahl noch die Blocknummern ausserhalb des zulaessigen Bereiches liegen und dass Bloecke nicht mehrfach zugewiesen sind. Es wird geprueft,

ob alle Bloecke im Dateisystem vorhanden sind. Tritt innerhalb der Freiblockliste ein Fehler auf, wird sie durch fsck neu erstellt, ausgenommen sind alle Bloecke in der Liste der zugewiesenen Bloecke.
Fsck ueberprueft die Listenanzahl des ersten Freiblocklisten-Blocks auf einen Wert kleiner Null und groesser 50. Fsck testet, ob die Blocknummern aus der Freiblockliste kleiner als die Blocknummer des ersten Datenblock des Dateisystems sind. Anschliessend vergleicht fsck jede Blocknummer mit der Liste der schon zugewiesenen Bloecke. Ist der Zeiger des Freiblocklisten-Blocks ungleich Null, wird der naechste Freiblocklisten-Block eingelesen und der Prozess wird wiederholt. Als letztes wird geprueft, ob die Anzahl der durch die Freiblock-Listen belegten Bloecke plus der Anzahl der durch die Inodes benutzten Bloecke gleich der Totalanzahl aller Bloecke des Dateisystems ist.

4.2.3. Anzahl der freien Bloecke

Der Superblock enthaelt die Anzahl aller freien Bloecke innerhalb des Dateisystems. Fsck vergleicht diesen Wert mit der Anzahl der gefundenden freien Bloecke im Dateisystem und ersetzt bei Nichtuebereinstimmung diesen Wert durch den aktuellen.

4.2.4. Anzahl der freien Inodes

Der Superblock enthaelt die Anzahl der freien Inodes innerhalb des Dateisystems. Fsck vergleicht diesen Wert mit der Anzahl der gefundenen freien Inodes im Dateisystem und ersetzt bei Nichtuebereinstimmung diesen Wert durch den aktuellen.

4.3. Inodes

Eine grosse Anzahl aktiver Inodes erhoeht die Wahrscheinlichkeit des Auftretens von Fehlern. Durch fsck erfolgt eine Ueberpruefung der Liste der Inodes auf Widersprueche bezueglich Format und Typ, Verbindungszaehler, doppelter Bloecke, defekter Bloecke und Inode-Groesse.

4.3.1. Format und Typ

Jedes Inode enthaelt ein Wort (mode-word), das den Typ und den Zustand des Inodes kennzeichnet. Gueltige Inode-Typen sind "regular", "directory", "special Block" und "special character". Gueltige Inode-Zustaende sind "unallocated" (nicht zugewiesen), "allocated" (zugewiesen); keines von beiden (fehlerhaftes Formatieren: verursacht durch falsche Daten, die durch Fehler in der Hardware in die Inode-Liste

gerieten). Fsck kann dieses Inode loeschen.

4.3.2. Anzahl der Verbindungen

In jedem Inode befindet sich ein Zaehler, der die Anzahl aller Directoryeintraege enthaelt, die mit diesem Inode verbunden sind. Fsck ueberprueft diese Anzahl, indem es die Directories durchmustert. Dabei beginnt es im root-Directory und berechnet die tatsaechliche Anzahl aller Verbindungenn fuer jedes Inode. Ist die festgestellte Anzahl der Verbindungen gleich Null, aber die tatsaechliche Anzahl ungleich Null, erfolgt kein Directoryeintrag fuer das Inode. Fsck kann eine losgeloeste Datei mit dem "lost+found"-Directory verbinden.
Sind die gespeicherte und die tatsaechliche Anzahl der Verbindung zwar ungleich Null, aber nicht identisch, kann ein Directoryeintrag ohne Aktualisierung des Inodes hinzugefuegt oder geloescht werden. Fsck kann die gespeicherte Anzahl der Verbindungen durch die tatsaechliche ersetzen.

4.3.3. Doppelte Bloecke

Jedes Inode enthaelt eine Liste oder Zeiger auf Listen (indirekte Bloecke) von allen Bloecken, die durch das Inode beansprucht werden. Fsck vergleicht jede von einem Inode beanspruchte Blocknummer mit einer Liste schon zugewiesener Bloecke. Treten dabei irgendwelche Unstimmigkeiten auf, kann fsck beide Inodes, welche diese Blocknummer beanspruchen, loeschen. Wird eine Blocknummer schon durch ein anderes Inode belegt, wird diese Blocknummer zur Liste der doppelten Bloecke hinzugefuegt. Andererseits wird die Liste der zugewiesenen Bloecke durch die Einbeziehung dieser Blocknummer aktualisiert. Wenn schon doppelte Bloecke existieren, erfolgt durch fsck ein teilweises Durchmustern der Inodeliste, um den doppelten Block zu finden. Durch fsck wird untersucht, welche Dateien mit diesen Inodes verbunden sind. Dies ist notwendig, um festzulegen, welches Inode "veraltet" ist und geloescht werden sollte (das meist benutzte ist das Inode, mit dem juengsten Modifizierungsdatum). Dieser Fehler tritt auf, wenn ein Dateisystem benutzt wird, dessen Bloecke sowohl durch die Freiblockliste, als auch durch andere Teile des Dateisystems angefordert wurden. Das Auftreten von vielen doppelten Bloecken in einem Inode ist meist darauf zurueckzufuehren, dass ein indirekter Block nicht in das Dateisystem geschrieben wurde.

4.3.4. Defekte Bloecke

Jedes Inode enthaelt eine Liste oder einen Zeiger auf Listen, die die Nummern aller durch das Inode belegten Bloecke enthalten. Fsck ueberprueft, ob alle der durch

einen Inode belegten Blocknummern zwischen der ersten und der letzten Datenblocknummer des Dateisystems liegen. Eine Blocknummer ausserhalb dieses Bereiches wird als defekt (bad Block) bezeichnet. Das Auftreten vieler defekter Bloecke in einem Inode ist meist darauf zurueckzufuehren, dass ein indirekter Block nicht in das Dateisystem geschrieben wurde.

4.3.5. Groessentest

Jedes Inode enthaelt ein 32-bit (4-Byte) grosses Feld, das die Anzahl der Zeichen einer Datei enthaelt, die mit dem Inode verbunden sind. Fsck ueberprueft dieses Feld auf Fehler, wie z.B., ob ein Directory immer ein Vielfaches von 16 ist und dass die Anzahl der tatsaechlich benutzten Bloecke mit der durch das Inode angezeigten uebereinstimmt. Ein Directoryinode in einem WEGA-Dateisystem hat das Directorybit im Inode-Mode-Wort gesetzt. Die Groesse eines Directory muss immer ein Vielfaches von 16 sein, da jeder Directoryeintrag aus 16 Informationsbytes besteht; 2 Bytes fuer die Inodenummer und 14 Bytes fuer den Datei- oder Directorynamen. Bei einem fehlerhaften Aufbau eines Directory erfolgt durch fsck nur eine Warnung, keine Korrektur. Durch fsck erfolgt auch die Berechnung der Anzahl der Bloecke, die durch ein Inode belegt werden. Dazu dividiert fsck die in einem Inode angegebene Zeichenanzahl durch die Anzahl der Zeichen pro Block (512), rundet anschliessend auf und addiert fuer jeden indirekten Block einen Block hinzu. Entspricht die berechnete Anzahl nicht der tatsaechlichen Blockanzahl, warnt fsck vor einem moeglichen Dateigroessenfehler. Eine Korrektur erfolgt nicht, da WEGA keine Bloecke in eine Datei einfuegt.

4.4. Indirekte Bloecke

Da indirekte Bloecke zu einem Inode gehoeren, wirken sich auftretende Widersprueche in diesen Bloecken auf das Inode aus. Fsck testet, ob diese Bloecke nicht schon durch ein anderes Inode belegt sind und dass die Blocknummern nicht ausserhalb des Grenzen des Dateisystems liegen. Die schon unter 4.3.3 und 4.3.4 aufgefuehrten Prozeduren werden fuer alle indirekten Bloecke angewendet.

4.5. Datenbloecke

Es gibt zwei Arten von Datenbloecken, einfache Datenbloecke und Directorydatenbloecke. Einfache Datenbloecke enthalten in einer Datei gespeicherte Informationen und werden durch fsck nicht getestet. Directorydatenbloecke enthalten Directoryeintragungen und werden auf Widersprueche geprueft. Diese Pruefungen betreffen Directory-Inodenummern, die auf nicht zugewiesene Inodes zeigen; Directory-Inodenummern, die groesser als die

groesstmoegliche Inodenummer im Dateisystem sind; fehlerhafte Directory-Inodenummern fuer . (aktuelles Directory) und .. (Eltern-Directory) sowie Directories, die vom Dateisystem abgetrennt sind. Fsck streicht jede Inodenummer in einem Directory, das auf ein nicht zugewiesenes Inode weist. Dieser Fall tritt gewoehnlich auf, wenn ein Datenblock, der Directoryeintraege enthaelt, modifiziert und in das Dateisystem zurueckgeschrieben wurde und das Inode noch nicht zurueckgeschrieben wurde. Zeigt die Inodenummer eines Directory hinter das Ende der Inodeliste, entfernt fsck die Directoryeintragung. Dieser Zustand tritt auf, wenn fehlerhafte Daten in einem Directorydatenblock abgelegt wurden.
Die Directoryeintragung der Inodenummer fuer '.' muss die erste Eintragung im Directorydatenblock sein. Ihr Wert muss gleich der Inodenummer des Elterndirectory sein (oder der Inodenummer des Directorydatenblocks, wenn das Directory das root-Directory ist). Sind die Directory-Inodenummern falsch, werden sie von fsck durch die richtigen Werte ersetzt.
Fsck kontrolliert die allgemeinen Verbindungen des Dateisystems. Sind Directories nicht mit dem Dateisystem verbunden, wird durch fsck eine Rueckverbindunng in das Dateisystem in die "lost+found"-Directory vorgenommen. Dieser Zustand kann dann auftreten, wenn Inodes in das Dateisystem geschrieben wurden, ohne dass auch der zugehoerige Directory-Datenblock geschrieben wird.

4.6. Freiblocklisten-Bloecke

Freiblocklisten-Bloecke sind direkt dem Superblock zugeordnet und auftretende Widersprueche beeinflussen direkt den Superblock. Fsck sucht nach Werten von Zaehlern ausserhalb des erlaubten Bereiches, Blocknummern ausserhalb des Bereiches und Bloecken, die schon mit dem Dateisystem verbunden sind.
Abschnitt 4.2.2 enthaelt Moeglichkeiten der Erkennung und Korrektur von Widerspruechen, die bei den Freiblocklisten-Bloecken auftreten koennen.

5. Fehlerausschriften, die durch fsck erzeugt werden

5.1. Vereinbarungen

Fsck ist ein Testprogramm, das in mehreren Schritten abgearbeitet wird, wobei jeder Schritt eine andere Phase von fsck aufruft. Nach dem Initialisieren und Laden von Anfangswerten fuehrt fsck nacheinander fuer jedes Dateisystem verschiedene Tests durch. Es kontrolliert die Bloecke und Groessen, Pfadnamen, Verbindungszaehler, Referenzen und die Freiblock-Liste (die eventuell neu erstellt wird) und fuehrt in einigen Faellen Sofortkorrekturen durch. Wird dabei ein Widerspruch festgestellt, erfolgt durch fsck eine Ausschrift an den Operator. Eventuell wird ein Promptzeichen (Eingabeanforderung) ausgegeben, wenn vom Operator eine Antwort erwartet wird. Dieser Abschnitt erklaert die Bedeutung der Fehlerausschriften und die moeglichen Antworten. Die Auflistung der Fehlerausschriften erfolgt entsprechend der Phasen von fsck, in denen sie auftreten. Fehler, die in mehreren Phasen auftreten koennen, sind unter 5.2 (Initialisierung) angeordnet.

5.2. Initialisierung

Bevor fsck mit der Kontrolle beginnt, muessen bestimmte Tabellen angelegt und Dateien eroeffnet werden. In diesem Abschnitt erfolgt die Auflistung von moeglichen Fehlern, die sich bei der Initialisierung von Tabellen und der Eroeffung von Dateien, bei der Angabe der Optionen in der Kommandozeile, bei Speicheranforderungen, beim Status von Dateien, bei Dateisystemgroessentests und bei der Erstellung von Hilfs-Dateien ergeben koennen.

Fehler: C OPTION?
Ursache: C ist keine fuer fsck zugelassene Option; erlaubte Optionen sind -y, -n, -s, -S und -t.
Aktion: Fsck wird abgebrochen (fsck (1)).

Fehler: BAD -t OPTION
Ursache: Der -t Option folgt kein Dateiname.
Aktion: Fsck wird abgebrochen (fsck(1)).

Fehler: INVALID -s ARGUMENT, DEFAULTS ASSUMED
Ursache: Die Parameter der Option sind falsch angegeben.
Aktion: Fsck benutzt die Werte, die Bei der Anlage des Dateisystems benutzt worden sind (fsck(1)).

Fehler: INCOMPATIBLE OPTIONS: -n and -s
Ursache: Es ist nicht moeglich, die Freiblockliste zu aendern, ohne dabei das Dateisystem zu modifizieren.
Aktion: Fsck wird abgebrochen (fsck(1)).

Fehler: CAN'T GET MEMORY
Ursache: Die durch fsck erfolgte Speicheranforderung fuer die temporaeren Speichertabellen konnte nicht durchgefuehrt werden.
Aktion: Fsck wird abgebrochen.

Fehler: CAN'T OPEN CHECKLIST FILE: F
Ursache: Die Datei F des "root"-Dateisystems, die die Namen der zu ueberpruefenden Dateisysteme enthaelt (gewoehnlich /etc/checklist), kann nicht fuer einen Lesevorgang eroeffnet werden.
Aktion: Fsck wird abgebrochen. Ueberpruefe den Zugriffsmode von F.

Fehler: CAN'T STAT ROOT
Ursache: Die Anforderung von fsck zum Status des root-Directory / wurde fehlerhaft beantwortet. Dieser Fehler duerfte eigentlich nicht auftreten.
Aktion: fsck wird abgebrochen.

Fehler: CAN'T STAT F
Ursache: Die Anforderung von fsck zum Status des Dateisystems F wurde fehlerhaft beantwortet.
Aktion: Fsck ignoriert dieses Dateisystem und kontrolliert das naechste angegebene Dateisystem. Ueberpruefe Zugriffsmode von F.

Fehler: F IS NOT A BLOCK OR CHARACTER DEVICE
Ursache: Fsck erhielt einen falschen regulaeren Dateinamen, den er ignoriert und zur Kontrolle des naechsten Dateisystems uebergeht.
Aktion: Ueberpruefe Dateityp von F.

Fehler: CAN'T OPEN F
Ursache: Das Dateisystem F kann nicht fuer eine Leseoperation eroeffnet werden.
Aktion: Fsck ignoriert dieses Dateisystem und kontrolliert das naechste Dateisystem. Ueberpruefe Zugriffsmode von F.

Fehler: SIZE CHECK: fsize X isize Y
Ursache: Es werden fuer die Inode-Liste Y mehr Bloecke benutzt, als Bloecke im Dateisystem X vorhanden sind, oder es gibt mehr als 65535 Inodes im Dateisystem.
Aktion: Fsck ignoriert dieses Dateisystem und kontrolliert das naechste Dateisystem (siehe Abschnitt 4.2.1)

Fehler: CAN'T CREATE F
Ursache: Die Anforderung von fsck zum Erzeugen einer Hilfsdatei F konnte nicht durchgefuehrt werden.
Aktion: Fsck ignoriert dieses Dateisystem und geht ueber zur Kontrolle des naechsten Dateisystems. Ueberpruefe Zugriffsmode von F.

Fehler: CANNOT SEEK: BLK B (CONTINUE?)
Ursache: Die durch fsck ausgeloeste Anforderung zur Bewegung zum angegebenen Block B des Dateisystems konnte nicht durchgefuehrt werden.
Aktion: Moegliche Antworten auf die CONTINUE?-Anforderung sind:
YES: Es wird versucht, die Pruefung des Dateisystems fortzusetzen, was oft nicht funktioniert, da dieser Fehler die komplette Testung des Dateisystems verhindert. Um das Dateisystem doch zu pruefen, muss fsck noch einmal gestartet werden. Ist der Block Teil des virtuellen Speichers, wird der Test durch fsck beendet und es erfolgt die Ausschrift "FATAL I/O ERROR".
NO: Das Programm wird abgebrochen.

Fehler: CANNOT READ: BLK B (CONTINUE ?)
Ursache: Die durch fsck ausgeloeste Anforderung fuer das Lesen des Blocks B konnte nicht durchgefuehrt werden.
Aktion: Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Es wird versucht, den Kontrollauf des Dateisystems fortzusetzen, was oft nicht funktioniert, da dieser Fehler die weitere Testung des Dateisystems verhindert. Um das Dateisystem doch zu pruefen, muss fsck noch einmal gestartet werden. Ist der Block Teil des virtuellen Speichers, wird der Test durch fsck beendet und es erfolgt die Ausschrift "FATAL I/O ERROR".
NO: Das Programm wird abgebrochen.

Fehler: CANNOT WRITE: BLK B (CONTINUE ?)
Ursache: Die durch fsck ausgeloeste Anforderung fuer das Schreiben der Blocks B in das Dateisystem konnte nicht ausgefuehrt werden.
Aktion: Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Es wird versucht, den Kontrollauf des Dateisystems fortzusetzen, was oft nicht funktioniert, da dieser Fehler die weitere Testung des Dateisystems verhindert. Um das Dateisystem doch zu pruefen, muss fsck noch einmal gestartet werden. Ist der Block Teil des virtuellen Speichers, wird der Test durch fsck beendet und es erfolgt die Ausschrift "FATAL I/O ERROR".
NO: Das Programm wird abgebrochen.

5.3. Phase 1: Test der Bloecke und Groessen

Diese Phase beschaeftigt sich mit der Inodeliste. In diesem Abschnitt erfolgt die Auslistung von moeglichen

Fehlern, die sich aus der Ueberpruefung der Inodetypen, der "Zero-Link-Count"-Tabelle, der Inodeblocknummern fuer defekte und doppelte Bloecke, der Inodegroesse sowie des Inodeformats ergeben.

Fehler: UNKNOWN FILE TYPE I=I (CLEAR?)
Ursache: Das Modewort des Inodes I zeigt an, dass es sich nicht um ein "special inode", "regular inode" oder "directory inode" handelt (siehe Abschnitt 4.3.1).
Aktion: Moegliche Antworten auf die CLEAR?-Ausgabe sind:
YES: Fortsetzen des Programmes: Dieser Fehler verhindert einen vollstaendigen Test des Dateisystems. Um das Dateisystem doch zu testen, muss fsck noch einmal gestartet werden. Wird ein anderes Inode mit einem Verbindungszaehler von Null entdeckt, erfolgt die Wiederholung der Fehlerausschrift.
NO: Das Programm wird beendet.

Fehler: LINK COUNT TABLE OVERFLOW (CONTINUE?)
Ursache: Eine interne fsck-Tabelle, die die zugewiesenen Inodes mit einem Verbindungszaehl von Null enthaelt, hat keinen Platz mehr.
Aktion: Es muss ein neuer Compilerlauf fuer fsck mit einem groesseren Wert fuer MAXLNCNT gestartet werden. Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Fortsetzen des Programmes. Dieser Fehler verhindert einen vollstaendigen Test des Dateisystems. Um das Dateisystem doch zu testen, muss fsck noch einmal gestartet werden. Wird ein weiteres zugewiesenes Inode mit einem Verbindungszaehler von Null entdeckt, folgt die Wiederholung der Fehlerausschrift.
NO: Das Programm wird beendet.

Fehler: B BAD I=I
Ursache: Das Inode I enthaelt eine Blocknummer B, die kleiner als die Nummer des ersten oder groesser als die Nummer des letzten Datenblocks des Dateisystems ist. Hat das Inode I zu viele Blocknummern ausserhalb des Bereiches des Dateisystems, ruft dieser Fehler den EXCESSIVE BAD BLKS-Fehler in Phase 1 auf. Dieser Fehler aktiviert die BAD/DUP-Fehlerausschrift in den Phasen 2 und 4 (siehe Sektion 4.3.4).

Fehler: EXCESSIVE BAD BLKS I=I (CONTINUE?)
Ursache: Das Inode I enthaelt zu viele Bloecke mit einer Nummer ausserhalb des Dateisystems (siehe Abschnitt 4.3.4).
Aktion: Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Der Rest der Bloecke in diesem Inode wird ignoriert und die Ueberpruefung wird mit dem

naechsten Inode des Dateisystems fortgesetzt. Dieser Fehler erlaubt keinen vollstaendigen Test des Dateisystems. Um das gesamte Dateisystem zu pruefen, muss fsck noch einmal gestartet werden.
NO: Das Programm wird beendet.

Fehler: B DUP I=I
Ursache: Das Inode I enthaelt die Blocknummer B, die schon von einem anderen Inode belegt ist. Dieser Fehler produziert die EXCESSIVE DUP BLKS-Fehlerausschrift in Phase 1, falls das Inode I zu viele Blocknummern beinhaltet, die auch von anderen Inodes beansprucht werden. Dieser Fehler aktiviert immer die Phase 1B und die BAD/DUP-Fehlerausschrift in den Phasen 2 und 4 (siehe Abschnitt 4.3.3).

Fehler: EXCESSIVE DUP BLKS I=I (CONTINUE?)
Ursache: Das Inode I belegt eine mehr als zu tolerierende Anzahl von Bloecken (gewoehnlich 10), die schon von anderen Inodes belegt sind.
Aktion: Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Der Rest der Bloecke in diesem Inode wird ignoriert, die Ueberpruefung wird mit dem naechsten Inode des Dateisystems fortgesetzt. Dieser Fehler erlaubt keinen vollstaendigen Test des Dateisystems. Um das gesamte Dateisystem zu pruefen, muss fsck noch einmal gestartet werden.
NO: Das Programm wird beendet.

Fehler: DUP TABLE OVERFLOW (CONTINUE?)
Ursache: In einer internen fsck-Tabelle, die die doppelten Blocknummern enthaelt, ist kein Platz mehr.
Aktion: Es muss ein neuer Compilerlauf fuer fsck mit einem groesseren Wert fuer DUPTBLSIZE erfolgen. Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Fortsetzen des Programmes. Dieser Fehler verhindert einen vollstaendigen Test des Dateisystems. Um das Dateisystem doch zu testen, muss fsck noch einmal gestartet werden. Wird ein weiterer doppelter Block gefunden, erfolgt die Wiederholung der Fehlerausschrift.
NO: Das Programm wird beendet.

Fehler: POSSIBLE FILE SIZE ERROR I=I
Ursache: Die Groesse des Inode I stimmt nicht mit der aktuellen Anzahl der vom Inode benutzten Bloecke ueberein.
Diese Ausschrift ist nur eine Warnung (siehe Abschnitt 4.3.5).

Fehler: DIRECTORY MISALIGNED I=I

Ursache: Die Groesse eines Directory-Inodes entspricht nicht dem Vielfachen eines Directoryeintrages (16).
Diese Ausschrift ist nur eine Warnung (siehe Abschnitt 4.3.5).

Fehler: PARTIALLY ALLOCATED INODE I=I (CLEAR?)
Ursache: Das Inode I ist weder zugewiesen noch nicht zugewiesen (siehe Sektion 4.3.1). D.h. im Inode sind Unstimmigkeiten aufgetreten.
Aktion: Moegliche Antworten auf die CLEAR?-Ausschrift sind:
YES: Die Zuweisungen fuer das Inode I werden durch Nullsetzen seines Inhalts aufgehoben.
NO: Die Fehlerausschrift wird ignoriert.

5.4. Phase 1B: Wiederholtes Durchsuchen nach weiteren doppelten Bloecken

Wurde ein doppelter Block im Dateisystem gefunden, wird das System noch einmal nach dem Inode durchsucht, das auch diesen Block benutzt. In diesem Abschnitt ist die Fehlermoeglichkeit aufgefuehrt, die auftritt, wenn dieser doppelte Block gefunden wird.

Fehler: B DUP I=I
Ursache: Das Inode I enthaelt die Blocknummer B, die schon durch einen anderen Inode belegt wurde.
Dieser Fehler ruft die Fehlerausschrift BAD/DUP in Phase 2 hervor. Inodes mit sich ueberlagernden Bloecken koennen durch die Untersuchung dieses Fehlers und des DUP-Fehlers in Phase 1 bestimmt werden (siehe Abschnitt 4.3.3).

5.5. Phase 2: Ueberpruefung der Pfadnamen

Diese Phase streicht Directoryeintraege, die mit fehlerhaften Inodes aus den Phasen 1 und 1B verbunden sind. In diesem Abschnitt erfolgt die Auflistung aller Fehlerausschriften, die aus dem "root-Inodemode" und dem Status, den Directory- Inodezeigern innerhalb des Bereiches und den Directoryeintragungen, die auf die defekte Inodes zeigen, stammen.

Fehler: ROOT INODE UNALLOCATED. TERMINATING
Ursache: Das "root"-Inode (gewoehnlich Inodenummer 2) hat fehlerhaft gesetzte Modebits.
Aktion: Das Programm wird beendet (siehe Abschnitt 4.3.1).

Fehler: ROOT INODE NOT DIRECTORY (FIX?)

Ursache: Das "root"-Inode (gewoehnlich Inodenummer 2) ist kein Directoryinode (siehe Abschnitt 4.3.1).
Aktion: Moegliche Antworten auf die FIX?-Ausgabe sind:
YES: Deklariere das "root"-Inode als Directory. Sind die Datenbloecke der "root"-Inode keine Directorybloecke, wird eine lange Liste von Fehlerausschriften erzeugt.
NO: Das Programm wird beendet.

Fehler: DUPS/BAD IN ROOT INODE (CONTINUE?)
Ursache: Durch die Phasen 1 und 1B wurden doppelte Bloecke oder defekte Bloecke im "root"-Inode (gewoehnlich Inodennummer 2) des Dateisystems gefunden (siehe Abschnitte 4.3.3 und 4.3.4).
Aktion: Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Die DUPS/BAD-Fehlerausschrift im "root"-Inode wird ignoriert und es wird versucht, den Kontrollauf des Dateisystems fortzusetzen. Bei fehlerhaftem "root"-Inode erscheint eine grosse Anzahl anderer Fehlerausschriften.
NO: Das Programm wird beendet.

Fehler: I OUT OF RANGE I=I NAME=F (REMOVE?)
Ursache: In einem Directoryeintrag F steht die Inodenummer I, die groesser ist als das Ende der Inodeliste (siehe Abschnitt 4.5).
Aktion: Moegliche Antworten auf die REMOVE?-Ausgabe sind:
YES: Der Directoryeintrag F wird geloescht.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: UNALLOCATED I=I OWNER=O MODE=M SIZE=S MTIME=T NAME=F (REMOVE?)
Ursache: Ein Directoryeintrag F hat ein Inode I ohne Zuweisungsmodebits. Es werden Eigner O, Mode M, Groesse S, Aenderungszeit T und der Dateiname F ausgegeben (siehe Abschnitt 4.5).
Aktion: Moegliche Antworten auf die REMOVE?-Ausgabe sind:
YES: Der Directoryeintrag F wird geloescht.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: DUP/BAD I=I OWNER=O MODE=M SIZE=S MTIME=T DIR=F (REMOVE?)
Ursache: Durch die Phasen 1 und 1B wurden doppelte Bloecke oder defekte Bloecke gefunden, die mit dem Directoryeintrag F des Directoryinodes I verbunden sind. Es werden Eigner O, Mode M, Groesse S, Aenderungszeit T und der Directoryname F ausgegeben (siehe Abschnitt 4.3.3 und 4.3.4).
Aktion: Moegliche Antworten auf die REMOVE?-Ausgabe sind:
YES: Der Directoryeintrag F wird geloescht.
NO: Die Fehlerausschrift wird ignoriert.

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

Ursache: Durch die Phasen 1 und 1B wurden doppelte Bloecke oder defekte Bloecke gefunden, die mit dem Directoryeintrag F des Inodes I verbunden sind. Es werden Eigner O, Mode M, Groesse S, Aenderungszeit T und der Dateiname F ausgegeben (siehe Abschnitt 4.3.3 und 4.3.4)
Aktion: Moegliche Antworten auf die REMOVE?-Ausgabe sind:
YES: Der Directoryeintrag F wird geloescht.
NO: Die Fehlerausschrift wird ignoriert.

5.6. Phase 3: Ueberpruefung der Verbindungen

Die Phase ueberprueft die in Phase 2 festgestellten Directoryverbindungen. In diesem Abschnitt erfolgt die Auflistung aller Fehlerausschriften, die aus Directories stammen, auf die nicht verwiesen wird und die durch fehlende oder volle "lost+found"-Directories hervorgerufen werden.

Fehler: UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT?)
Ursache: Das Directoryinode I war bei der Ueberpruefung nicht mit einem Directoryeintrag verbunden. Es werden Eigner O, Mode M, Groesse S und Aenderungszeit T des Directoryinodes angegeben (siehe Abschnitt 4.5 und 4.3.2)
Aktion: Moegliche Antworten auf die RECONNECT?-Ausgabe sind:
YES: Das Directoryinode I wird mit dem Dateisystem durch Verbindung mit dem Directory fuer verlorengegangene Dateien (gewoehnlich "lost+found") gerettet. Falls dabei Probleme auftreten, erfolgt die Aktivierung des "lost+found"-Fehlers in der Phase 3. War die Verbindung erfolgreich, wird in Phase 3 die CONNECTED-Nachricht erzeugt.
NO: Die Fehlerausschrift wird ignoriert. Es wird immer die UNREF-Fehlerausschrift in Phase 4 erzeugt.

Fehler: SORRY. NO lost+found DIRECTORY
Ursache: Es gibt kein "lost+found"-Directory im "root"-Directory des Dateisystems.
Aktion: In diesem Fall ignoriert fsck die Anforderung zum Verbinden eines Directory mit "lost+found". Es wird immer die UNREFF-Fehlerausschrift in Phase 4 aktiviert. Man ueberpruefe den Zugriffsmode von "lost+found" (fsck (1)).

Fehler: SORRY. NO SPACE IN lost+found DIRECTORY
Ursache: Es ist kein Platz mehr vorhanden, um eine weitere Eintragung in das "lost+found"-Directory des "root"-Directory des Dateisystems aufzunehmen.

Aktion: Fsck ignoriert die Anforderung zum Verbinden eines Directory mit "lost+found". Es wird in jedem Fall die UNREFF-Fehlerausschrift in Phase 4 aktiviert. Ein Ausweg besteht im Streichen nicht mehr benoetigter "lost+found"-Eintraege oder im Vergroessern von "lost+found" (fsck(1)).

Fehler: DIR I=I1 CONNECTED. PARENT WAS I=I2
Diese Mitteilung ist nur als Hinweis zu verstehen, der anzeigt, dass ein Directoryinode I1 erfolgreich mit dem "lost+found"-Directory verbunden wurde. Das "Eltern"-Inode I2 des Directoryinode I1 wird durch die Inodenummer des "lost+found"-Directory ersetzt (siehe Abschnitt 4.3. und 4.3.2).

5.7. Phase 4: Ueberpruefung der Anzahl der Verbindungen

Die Phase ueberprueft die Anzahl der in Phase 2 und 3 festgestellten Verbindungen. In diesem Abschnitt erfolgt die Auflistung aller Fehlerausschriften resultierend aus Dateien, auf die nicht Bezug genommen wird, fehlendem oder vollem "lost+found"-Directory, falschen Verbindungszahlen fuer Dateien, Directories und Spezialdateien. Weiterhin werden Fehler angefuehrt aus Dateien und Directories, auf die nicht Bezug genommen wird, Fehler, die aus defekten und doppelten Bloecken und Directories herruehren sowie fehlerhafte Frei-Inode-Zaehler.

Fehler: UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (RECONNECT?)
Ursache: Das Inode I war zum Zeitpunkt der Dateisystemueberpruefung nicht mit einem Directoryeintrag verbunden.
Aktion: Es werden Eigner O, Mode M, Groesse S und die Aenderungszeit T des Inodes I ausgegeben (siehe Abschnitt 4.3.2). Moegliche Antworten auf die RECONNECT?-Ausgabe sind:
YES: Die Verbindung des Inodes I mit dem Dateisystem erfolgt durch Eintragung des Inodes in das Directory fuer verlorengegangene Dateien (gewoehnlich "lost+found"). Falls dabei Probleme auftreten, erfolgt die Aktivierung des "lost+found"-Fehlers in der Phase 4.
NO: Die Fehlerausschrift wird ignoriert. Es wird immer die CLEAR-Fehlerausschrift in Phase 4 erzeugt.

Fehler: SORRY. NO lost+found DIRECTORY
Ursache: Es gibt kein "lost+found"-Directory im "root"-Directory des Dateisystems.
Aktion: In diesem Fall ignoriert fsck die Anforderung zum Verbinden einer Datei mit "lost+found". Es wird immer die CLEAR-Fehlerausschrift in Phase 4 erzeugt. Man ueberpruefe den Zugriffsmode von

"lost+found".

Fehler: SORRY. NO SPACE IN lost+found DIRECTORY
Ursache: Es ist kein Platz mehr, um eine weitere Eintragung im "lost+found"-Directory des "root"-Directory des Dateisystems aufzunehmen.
Aktion: Fsck ignoriert die Anforderung zum Verbinden einer Datei mit "lost+found". Es wird in jedem Fall die CLEAR-Fehlerausschrift in Phase 4 erzeugt. Man ueberpruefe Groesse und Inhalt von "lost+found".

Fehler: (CLEAR?)
Ursache: Das im letzten Fehlerfall erwaehnte Inode kann nicht verbunden werden (siehe Abschnitt 4.3.2).
Aktion: Moegliche Antworten auf die CLEAR?-Ausgabe sind:
YES: Die Zuordnung fuer das zuletzt in der Fehlerausschrift erwaehnte Inode wird durch Setzen seines Inhalts auf Null aufgehoben.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: LINK COUNT FILE I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X (ADJUST?)
Ursache: Die Anzahl der Verbindungen fuer das Dateiinode I ist X, sollte aber Y sein.
Aktion: Es werden Eigner O, Mode M, Groesse S und Aenderungszeit T ausgegeben (siehe Abschnitt 4.3.2). Moegliche Antworten auf die ADJUST?-Ausgabe sind:
YES: Ersetze die Anzahl der Verbindungen des Inode I durch Y.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: LINK COUNT DIR I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X (ADJUST?)
Ursache: Die Anzahl der Verbindungen fuer das Directoryinode I ist X, sollte aber Y sein.
Aktion: Es werden Eigner O, Mode M, Groesse S und Aenderungszeit T ausgegeben (siehe Abschnitt 4.3.2). Moegliche Antworten auf die ADJUST?-Ausgabe sind:
YES: Ersetze die Anzahl der Verbindungen des Inode I durch Y.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: LINK COUNT F I=I OWNER=O MODE=M SIZE=S MTIME=T COUNT=X (ADJUST?)
Ursache: Die Anzahl der Verbindungen fuer das Inode I ist X, sollte aber Y sein.
Aktion: Es werden Eigner O, Mode M, Groesse S und Aenderungszeit T ausgegeben (siehe Abschnitt 4.3.2). Moegliche Antworten auf die ADJUST?-Ausgabe sind:
YES: Ersetze die Anzahl der Verbindungen des Inode I durch Y.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: UNREF FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)
Ursache: Das Dateiinode I war zum Zeitpunkt der Pruefung des Dateisystems nicht mit einem Directoryeintrag verbunden.
Aktion: Es werden Eigner O, Mode M, Groesse S und Aenderungszeit T des Inode I ausgegeben (siehe Abschnitt 4.3.2 und 4.5) Moegliche Anworten auf die CLEAR?-Ausgabe sind:
YES: Die Zuordnung des Inode I wird durch Nullsetzen seines Inhalts aufgehoben.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: UNREF DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)
Ursache: Das Directoryinode I war zum Zeitpunkt der Pruefung des Dateisystems nicht mit einem Directoryeintrag verbunden.
Aktion: Es werden Eigner O, Mode M, Groesse S und Aenderungszeit T des Inode I ausgegeben (siehe Abschnitt 4.3.2 und 4.5) Moegliche Anworten auf die CLEAR?-Ausgabe sind:
YES: Die Zuordnung des Inode I wird durch Nullsetzen seines Inhalts aufgehoben.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: BAD/DUP FILE I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)
Ursache: In Phase 1 oder 1B wurden doppelte oder defekte Bloecke gefunden, die mit dem Dateiinode verbunden sind.
Aktion: Es werden Eigner O, Mode M, Groesse S und Aenderungszeit T des Inode I ausgegeben (siehe Abschnitt 4.3.3 und 4.3.4) Moegliche Antworten auf die CLEAR?-Ausgabe sind:
YES: Die Zuordnung des Inode I wird durch Nullsetzen seines Inhalts aufgehoben.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: BAD/DUP DIR I=I OWNER=O MODE=M SIZE=S MTIME=T (CLEAR?)
Ursache: In Phase 1 oder 1B wurden doppelte oder defekte Bloecke gefunden, die mit dem Directoryinode verbunden sind.
Aktion: Es werden Eigner O, Mode M, Groesse S und Aenderungszeit T des Inode I ausgegeben (siehe Abschnitt 4.3.3 und 4.3.4). Moegliche Antworten auf die CLEAR?-Ausgabe sind:
YES: Die Zuordnung des Inode I wird durch Nullsetzen seines Inhalts aufgehoben.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: FREE INODE COUNT WRONG IN SUPERBLOCK (FIX?)
Ursache: Die tatsaechliche Anzahl der freien Inodes stimmt nicht mit der im Superblock des Dateisystems angegebenen Anzahl ueberein (siehe Abschnitt

4.2.4).
Aktion: Moegliche Antworten auf die FIX?-Ausgabe sind:
YES: Ersetze die Anzahl im Superblock durch die tatsaechliche Anzahl.
NO: Die Fehlerausschrift wird ignoriert.

5.8. Phase 5: Ueberpruefung der Freiblockliste

In diesem Abschnitt erfolgt die Auflistung aller Fehlermoeglichkeiten, die aus defekten Bloecken in der Freiblockliste, aus einer falschen Anzahl von freien Bloecken, aus doppelten Bloecken in der Freiblockliste, aus ungenutzten Bloecken des Dateisystems, die nicht in der Freiblockliste aufgefuehrt sind und aus einer falschen Gesamtanzahl von freien Bloecken resultiert.

Fehler: EXCESSIVE BAD BLKS IN FREE LIST (CONTINUE?)
Ursache: Die Freiblockliste enthaelt mehr als zulaessig viele (gewoehnlich 10) Bloecke mit einer Nummer kleiner als der erste Datenblock und groesser als der letzte Datenblock des Dateisystems (siehe Abschnitt 4.2.2 und 4.3.4).
Aktion: Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Der Rest der Freiblockliste wird ignoriert und die Ausfuehrung von fsck wird fortgesetzt. Dieser Fehler ruft in jedem Fall die "BAD BLKS IN FREE LIST"-Fehlerausschrift in Phase 5 hervor.
NO: Das Programm wird beendet.

Fehler: EXCESSIVE DUP BLKS IN FREE LIST (CONTINUE?)
Ursache: Die Freiblockliste enthaelt mehr als zulaessige viele (gewoehnlich 10) Bloecke, auf die durch Inodes oder durch fruehere Teile der Freiblockliste zugegriffen wird (siehe Abschnitt 4.2.2 und 4.3.3).
Aktion: Moegliche Antworten auf die CONTINUE?-Ausgabe sind:
YES: Der Rest der Freiblockliste wird ignoriert und die Ausfuehrung von fsck fortgesetzt. Dieser Fehler ruft in jedem Fall die "DUP BLKS IN FREE LIST"-Fehlerausschrift in Phase 5 hervor.
NO: Das Programm wird beendet.

Fehler: BAD FREEBLK COUNT
Ursache: Der Zaehler fuer die freien Bloecken in einer Freiblockliste ist groesser 50 oder kleiner 0. Dieser Fehler erzeugt in jedem Fall die "BAD FREE LIST"-Fehlerausschrift in Phase 5 (siehe Abschnitt 4.2.2).

Fehler: X BAD BLKS IN FREE LIST
Ursache: X Bloecke der Freiblockliste haben eine Blocknummer, die kleiner als der erste Datenblock oder groesser als der letzte Datenblock des

Dateisystems ist. Dieser Fehler erzeugt die "BAD FREE LIST"-Fehlerausschrift in Phase 5 (siehe Abschnitt 4.2.2 und 4.3.4).

Fehler: X DUP BLKS IN FREE LIST
Ursache: Es wurden in der Freiblockliste X Bloecke gefunden, auf die sich Inodes oder vorangegangene Teile der Freiblockliste beziehen. Dieser Fehler erzeugt immer die "BAD FREE LIST"-Fehlerausschrift in Phase 5 (siehe Abschnitt 4.2.2 und 4.3.3).

Fehler: X BLK(S) MISSING
Ursache: Es wurden X durch das Dateisystem nicht benutzte Bloecke nicht in der Freiblockliste gefunden. Dieser Fehler erzeugt immer die "BAD FREE LIST"-Fehlerausschrift in Phase 5 (siehe Abschnitt 4.2.2).

Fehler: FREE BLK COUNT WRONG IN SUPERBLOCK (FIX?)
Ursache: Die tatsaechliche Anzahl der freien Bloecke stimmt nicht mit der im Superblock gegebenen Anzahl ueberein (siehe Abschnitt 4.2.3).
Aktion: Moegliche Antworten auf die FIX?-Ausgabe sind:
YES: Ersetze die Anzahl im Superblock durch die tatsaechliche Anzahl.
NO: Die Fehlerausschrift wird ignoriert.

Fehler: BAD FREE LIST (SALVAGE?)
Ursache: Durch Phase 5 wurden in der Freiblockliste defekte oder doppelte Bloecke bzw. Bloecke, die vom Dateisystem vermisst werden, gefunden (siehe Abschnitte 4.2.2, 4.3.3 und 4.3.4).
Aktion: Moegliche Antworten auf die SALVAGE?-Ausgabe sind:
YES: Ersetze die augenblickliche Freiblockliste durch eine neue Freiblockliste. Die neue Freiblockliste wird so erstellt, dass die fuer das Positionieren der Disk benoetigte Wartezeit verringert wird.
NO: Die Fehlerausschrift wird ignoriert.

5.9. Phase 6: Retten der Freiblockliste

Diese Phase ueberprueft die Wiederherstellung der Freiblockliste. Es werden die Fehlermoeglichkeiten aufgefuehrt, die sich aus der Anzahl der zu ueberspringenden Bloecke und der Anzahl der Bloecke pro Zylinder ergeben koennen.

Fehler: DEFAULT FREE-BLOCK LIST SPACING ASSUMED
Diese Fehlermitteilung hat hinweisenden Charakter und zeigt an, dass die Anzahl der zu ueberspringenden Bloecke ungleich 1 ist oder dass die Anzahl der Bloecke pro Zylinder kleiner 1

oder groesser 500 ist. Fsck benutzt die Standardwerte von 1 fuer die Anzahl der zu ueberspringenden Bloecke und 72 fuer die Anzahl der Bloecke pro Zylinder (fsck (1)).

5.10. Korrektur

Wenn ein Dateisystem ueberprueft wird, werden dabei mehrere Korrektur-Massnahmen durchgefuehrt. In diesem Abschnitt sind alle Mitteilungen ueber das Dateisystem und den Aenderungsstatus des Dateisystems aufgefuehrt.

X FILES Y BLOCKS Z FREE
: Diese Ausschrift zeigt an, dass das ueberpruefte Dateisystem X Dateien enthaelt, die Y Bloecke benutzen und Z Bloecke im Dateisystem frei lassen.

***** BOOT WEGA (NO SYNC!) *****
: Durch diese Mitteilung wird angezeigt, dass ein verbundenes Dateisystem oder das Rootdateisystem durch fsck veraendert wurde. Falls durch WEGA der Bootprozess nicht sofort wiederholt wird, kann die durch fsck erfolgte Veraenderung durch speicherinternes Kopieren von Tabellen unwirksam werden.

***** FILE SYSTEM WAS MODIDIED *****
: Diese Mitteilung zeigt an, dass das aktuelle Dateisystem durch fsck modifiziert wurde. Wird dieses Dateisystem eingebunden oder handelt es sich um das Root-Dateisystem, muss fsck angehalten und WEGA neu geladen werden. Falls durch WEGA der Bootprozess nicht sofort wiederholt wird, kann die durch fsck erfolgte Veraenderung durch speicherinternes Kopieren von Tabellen unwirksam werden.

5.11. Liste der Nachrichten

Initialisierung

Phase 1: Test der Bloecke und Groessen

Phase 1B: Wiederholtes Durchsuchen nach weiteren doppelten Bloecken

Phase 2: Ueberpruefung der Pfadnamen

Phase 3: Ueberpruefung der Verbindungen

KOMBINAT VEB
ELEKTRO-APPARATE-WERKE
BERLIN-TREPTOW
»FRIEDRICH EBERT«

HEIM-ELECTRIC
EXPORT-IMPORT
Volkseigener Außenhandelsbetrieb
der Deutschen Demokratischen Republik

EAW-Automatisierungstechnik Export-Import

Storkower Straße 97
Berlin, DDR - 1055
Telefon 432010 · Telex 114158 heel dd

VEB ELEKTRO-APPARATE-WERKE BERLIN-TREPTOW
»FRIEDRICH EBERT«
Stammbetrieb des Kombinats EAW
DDR - 1193 Berlin, Hoffmannstraße 15-26
Fernruf: 2760
Fernschreiber: 0112263 eapparate bln
Drahtwort: eapparate bln