# PasTEX3.14

# AMIGA Implementation

**Dokumentation**

**vom**
**25. August 1991**

von
Georg Heßmann

# Inhaltsverzeichnis:

# 1. Copyright

Da TEX selbst nicht von mir stammt und die Anpassungen/Änderungen daran nur minimal sind, ist diese Version Public Domain. Jeder kann diese Version kopieren und weitergeben.

Man beachte jedoch, daß einzelne Teile der mitgelieferten Programme (vor allem die Treiber ShowDVI und DVIprint) bzw. Makro-Pakete ein Copyright besitzen!

Diese Anleitung "`tex.tex`" ist ebenfalls Copyright Georg Heßmann und darf nicht kommerziell vervielfältigt werden. Man darf sie sich ausdrucken und auch kopieren, man darf sie aber **nicht** für Geld weitergeben.

An dieser Stelle möchte ich mich sehr bei Bernd Raichle bedanken, der den UNIX TEX Sourcecode stark verbessert hat und mir den Source zur Verfügung gestellt hat! Von Ihm stammen auch einige Kapitel diese Anleitung. Dank auch an David Göhler für die Hilfe bei dieser Anleitung.

# 2. Allgemeines über TEX

TEX wurde erfunden, um mit dem Computer qualitativ hochwertige Drucke erstellen zu können. Ziel war es, Bücher und Texte aller Art vom Computer so aufbereiten zu lassen, daß sie schön aussehen. Dies bezieht sich aber nicht in erster Linie auf die Auflösung des Druckers. TEX setzt Texte so, daß sie ästhetisch schön sind.

Diese Satzkunst beherrschten früher nur Setzer. Normalerweise braucht man eine gute Ausbildung und ein gehöriges Maß an Erfahrung, um gutaussehende Texte fabrizieren zu können. So ist ein Text beispielsweise angenehm zu lesen, wenn man nicht zu viele Schriftarten auf einer Seite verwendet, weil sich das Gehirn sonst nicht richtig auf den Text konzentrieren kann. Genauso sind Leerzeilen häufig angebracht, um einen Text zu strukturieren. Dies sind nur einfache Grundregeln. Setzer kennen hunderte davon, und häufig sehen von Profis produzierte Texte einfach besser aus als andere. Manchmal weiß man gar nicht so richtig warum, aber sie sehen einfach besser aus. Setzer waren darüber hinaus auch in der Lage, Fehler des Schreibers zu korrigieren – was manchmal auch zu Verschlimmbesserungen geführt hat.

Wenn man nun als Privatmann mit wenig Zeit und nicht mit diesen speziellen Kenntnissen ausgerüstet trotzdem *schöne* Texte drucken möchte, sollte das Programm einem viele Kleinigkeiten abnehmen. TEX ist in dieser Hinsicht sehr freundlich und nimmt einem – wenn man es nicht ausdrücklich anders haben will – fast alles ab. So kann man sich auf den Inhalt konzentrieren.

## 2.1. Mensch und Maschine

"Computer sind doof" höre ich es noch von Kinderstimmen sagen. Und wie so häufig haben sie recht. Rechner sind recht beschränkt. Man muß ihnen alles haarklein vorgeben, damit sie das tun, was sie tun sollen. Da sie aber nicht in der Lage sind, die menschliche Stimme zu verstehen (geschweige denn interpretieren zu können, was man gesagt hat), muß man sich auf die Ebene der Maschine begeben und mit der Tastatur dem Rechner befehlen, was er zu tun hat.

Im Zeitalter der Mäuse und Grafikoberflächen gerät die Tastatur zunehmend in Vergessenheit, beim Tippen von Texten ist sie (zum Glück) noch unentbehrlich. Da man die Texte sowieso über die Tastatur in den Computer kriegen muß, ist es sinnvoll auch die Anweisungen, welche Texte wie und wo auf einer Seite unterzubringen sind, in den Text einzubauen.

Genauso funktioniert TEX. Man schreibt einen Text, wie man es mit einer Schreibmaschine auch tun würde. Doch jedesmal, wenn man etwas Besonderes veranstalten möchte – wie zum Beispiel ein Wort fettzudrucken oder in Schrägschrift zu setzen – muß man dies dem Computer mitteilen. Dies geschieht durch Anweisungen, die man direkt davor in den Text einbaut. Wie sollte der Computer sonst auch mitbekommen, daß ein bestimmtes Wort in fetter Schrift zu drucken ist?

Der Computer weiß zwar, wie mit dem Text umzugehen ist, für den Drucker sind die Buchstaben auf dem Bildschirm oder auf der Festplatte wiederum chinesisch. Also muß man den Text so umwandeln, daß dieser daraus einen schön aussehenden Text macht. Hierzu braucht man nun das TEX-Programm. Die Umwandlung ist in zwei Stufen aufgeteilt. Die erste Stufe erledigt das Programm `virtex` (steht für "virgin" TEX). Es vollführt die Umwandlungen, die für jeden Drucker gleich sind. Als Erzeugnis liefert es eine Datei, die mit dem Kürzel `.dvi` endet. Die zweite Stufe übernimmt der Druckertreiber `DVIprint`, der die speziellen Umwandlungen für den angeschlossenen Drucker vornimmt. Er nimmt die Datei mit der Endung `.dvi` und verschickt das Ergebnis seiner Umwandlung an den Drucker.

"Soll das etwa heißen, daß ich jeden Text erst zweimal übersetzen (umwandeln) und auch noch ausdrucken muß, bevor ich mitkriegen kann, wie er aussieht?".

Jain. Ja, weil die beiden Umwandlungen immer sein müssen. Ohne sie kann man nicht sehen, was aus dem Text-Anweisungen-Gemisch geworden ist. Nein, weil es eine Möglichkeit gibt, die Texte so umwandeln zu lassen, daß man sie sich auf dem Bildschirm ansehen kann.

Hierzu gibt es ein Programm namens `ShowDVI`, daß jede Datei, die ein Kürzel `.dvi` hat, umsetzt und auf dem Bildschirm anzeigt. Die zweite Stufe läuft also anders. Das `ShowDVI` druckt sozusagen nicht auf dem Drucker aus, sondern auf dem Bildschirm. Ohne Papier verschwenden zu müssen, kann man also vorab nachschauen, ob der Text was geworden ist. Wenn er einem gefällt, wirft man das Programm `DVIPrint` an, um den Text zu drucken. Gilt es Änderungen anzubringen, muß man eben beide Stufen noch einmal durchlaufen.

**Zusammenfassung** Um mit TEX einen Text zu erzeugen, muß man

- einen Text tippen und mit Anweisungen mischen
- diesen von dem Programm `virtex` in eine Datei umwandeln, die mit dem Kürzel `.dvi` endet
- diesen von dem Programm `ShowDVI` auf den Bildschirm bringen und entscheiden, ob er gefällt
- wenn er nicht gefällt, wieder von vorn anfangen
- wenn er gefällt, von `DVIprint` über den Drucker ausgeben lassen.

Manchem mag dieses Verfahren nicht ganz zeitgemäß erscheinen. Ich wage aber zu behaupten, daß ich mit dieser Methode deutlich schneller als mit einem sogenannten DTP-Programm bin und daß die Ausdrucke ästhetisch schöner sind.

## 2.2. Die Teile von TEX

TEX besteht – neben den Makrofiles – aus zwei Programmen: `initex` und `virtex`.

Mit `initex` wird ein Format-File (Endung '`fmt`') erzeugt. Dazu initialisiert dieses Programm alles von Grund auf, liest Makros und Trenntabellen ein und schreibt seinen Speicherinhalt in ein relativ kompaktes Format, dem Format-File. Dieser Vorgang wird üblicherweise als "Dumpen" eines Formates bezeichnet.

`virtex`, eine "virgin" Version von `initex`, kann dieses Format-File nun relativ schnell laden und mit dem Formatieren eines Textes beginnen. Wegen der nicht mehr notwendigen Initialisierung benötigt `virtex` weniger Speicher als `initex`.

Wozu nun dieses Format-File ?

TEX (genauer plainTEX) und LaTEX selbst sind keine eigenen Programme, sondern nur in ein Format-File gedumpte Makropakete, die man mit `virtex` laden kann. In einer Shell kann man dazu *aliases* definieren, z.B.

```
alias tex    virtex &plain
alias latex  virtex &lplain
```

## 2.3. Hardwarevoraussetzungen

TEX selbst läuft mit mindestens 1 MB Speicher, nur zum Erzeugen eines Format-Files mit `initex` schadet ein halbes MB mehr auf keinen Fall.

Für die Big-Version von TEX sollten mindestens 1,5 MB (zum Dumpen entsprechend mehr) zur Verfügung stehen.

TEX nur mit einem bzw. auch mit zwei Diskettenlaufwerken betreiben zu wollen, funktioniert zwar, grenzt aber hart an Masochismus. Empfehlenswert ist auf jeden Fall eine Festplatte mit ca. 5–10 MB Platz (je nachdem, wieviele Fonts, Makro- und Format-Files man gleichzeitig auf der Platte haben will).

# 3. Besonderheiten der Implementierung

Die aktuelle Implementierung baut auf der C-Version von TEX aus dem Unix Web2C-Paket auf[1] und wurde noch um einige Features erweitert.

- Konfigurierbarkeit
- Suchpfadangabe mittels Environment-Variablen oder in einem Konfigurationsfile
- Unterstützung von CodePages zum Mapping bei unterschiedlicher Zeichensatzbelegung im Amiga und in TEX

[1] Diese Version ist also nicht von Hand nach C übersetzt und eingetippt.

– inkl. "Statistics"[2]

## 3.1. Kommandozeilen-Optionen

In der Kommandozeile können neben dem zu formatierenden File noch einige Optionen mit angegeben werden.

Die Syntax des Aufrufes sieht damit folgendermaßen aus:

`initex [`*Optionen*`] [&`*fmt*`[.fmt]] [`*file*`[.tex]]`

bzw.

`virtex [`*Optionen*`] [&`*fmt*`[.fmt]] [`*file*`[.tex]]`

Als Optionen sind zulässig:

`-c`*dir-liste*
: Hiermit kann man eine Liste an Directories, getrennt durch Komma, angeben, in denen das Konfigurationsfile `tex.cnf` gesucht wird. Die Pfad-Liste muß direkt hinter dem '`c`' stehen. Eine weitere Möglichkeit wäre das Setzen der Environment-Variable `TEXCONFIG` auf *dir-liste*.

`-b` `\batchmode`

`-n` `\nonstopmode`

`-s` `\scrollmode`

`-e` `\errorstopmode` (Default)
: Es kann nur eine dieser Optionen gleichzeitig angegeben werden. Hiermit kann der *Interaction-Mode* entsprechend den TEX-Kommandos `\batchmode`, `\nonstopmode`, `\scrollmode` bzw. `\errorstopmode` gesetzt werden.

`-l`*language-nummer*
: Ab Version 3.0 kann man Trenntabellen für bis zu 256 Sprachen gleichzeitig laden. Die Umschaltung zwischen diesen Tabellen geschieht durch den neuen TEX-Counter `\language`. Hat man in Position 0 die englischen und in Position 1 die deutschen Trennungen geladen, so kann mittels `-l1` auf die deutsche Trennung umgeschaltet werden, ohne eine Änderung des zu formatierenden Files. Gibt man eine Zahl $> 255$ oder $< 0$ an, so entspricht dies `\language=0`, ist für die angegebene Nummer keine Tabelle geladen, so wird die Trennung ausgeschaltet[3].

`-d` Debugflag, damit werden einige mehr oder weniger aussagekräftige Informationen zusätzlich mit ausgegeben.

Die Optionen müssen jeweils einzeln für sich, durch ein Leerzeichen voneinander getrennt, angegeben werden. Zulässig ist z.B. `-s -e`, unzulässig dagegen `-se`.

## 3.2. Environment-Variablen

Um vernünftig mit TEX arbeiten zu können, werden neben dem Programm

[2] TEX kann man auch ohne die sogenannten "Statistics" compilieren, dabei erhält man eine Version die um ca. 1% schneller läuft. Jedoch sind dann die `\tracing..`-Befehle nicht mehr verfügbar.

[3] Die Style-Option `german.sty` (Version 2.3c) schaltet z.B. auf Position 1 um. Sind hierfür keine Trennmuster geladen, trennt TEX nicht mehr.

selbst, noch weitere Files benötigt. Wo diese Files gesucht werden, kann mittels Environment-Variablen[4] angegeben werden (oder im Konfigurationsfile).

`TEXINPUTS`
: Wo soll TEX nach den Makro-, Style-Files und dem zu formatierenden Text suchen? Hier sollte man das aktuelle Directory (anzugeben als '`.`') nicht in der Liste vergessen. (Files mit Endung '`tex`', '`sty`')
(*Default:* "`.,tex:macros`")[5]

`TEXFORMATS`
: Wie schon erwähnt, benötigt man zum Formatieren eines Textes ein Format-File, in dem z.B. das LATEX-Makropaket enthalten ist. (Files mit Endung '`fmt`')
(*Default:* "`tex:formats`")

`TEXFORMAT`
: Falls beim Start von TEX keine Formatfile angegeben wurde, wird der Inhalt der Variable als Formatfile verwendet. Ist die Variable nicht gesetzt, wird als default `plain` verwendet. Diese Variable ist vor allem beim Start von Workbench sehr nützlich, da dort keine Kommandline Optionen übergeben werden können.
(*Default: "keiner"*)

`TEXFONTS`
: TEX benötigt außerdem noch Informationen über die verwendeten Fonts, die in den 'TEX Font Metric'-Files zu finden sind. (Files mit Endung '`tfm`')
(*Default:* "`.,tex:fonts`")

`TEXPOOL`
: `initex` liest zu Beginn alle in TEX verwendeten Strings (z.B. Fehlermeldungen) aus einem "Poolfile". (File '`tex.pool`')
(*Default:* "`.,tex:`")

`EDITOR`
: Falls in der Eigabe für `initex` oder `virtex` ein Fehler ist (und man sich im interaktiven Modus ist) kann man mit '`e`' einen Editor starten. In der Variable kann man eintragen, welcher Editor aufgerufen werden soll. Für `%s` wird der Filename und für `%d` wird die Zeilennummer des Fehlers eingetragen.
(*Default:* "`ed %s -i`")

`TEXREXX`
: Diese Variable bestimmt, ob der Editor als System-Kommando oder als ARexx Script gestartet werden soll. Wenn die Variable nicht belegt ist, so wird der Editor als System-Kommando gestartet. Ist die Variable mit einem beliebigen String belegt, so wird der Editor nicht direkt gestartet, sondern es wird eine ARexx-Script aufgerufen. Ist `TEXREXX` nun mit dem String "`edit`" belegt, dann wird nicht erst darauf gewartet, daß der Benutzer mittels dem `e` Kommando den Editor startet, sondern er wird sofort nach Auftreten des ersten Fehlers gestartet.

---

[4] Es werden die Standard ENV: Variablen verwendet

[5] Diese Default-Werte sind direkt in das TEX-Programm eincompiliert, können aber auch teilweise über das Konfig-File geändert werden.

`REXXEDITOR`
: Diese Variable gibt an, welche ARexx Script aufgerufen werden soll, falls das '`e`' Kommando eingegeben wird und `TEXREXX` gesetzt ist. Normalerweise ist diese Script in `rexx:` und hat die Extension `.rexx`.
(*Default:* "`texedit %s %d`")

`TEXCONFIG`
: Suchpfad für das Konfigurationsfile `tex.cnf`. Ist ebenso durch die '`-c`'-Option beim Aufruf möglich. (File '`tex.cnf`') Diese Variable wird auch von den Treibern ShowDVI und DVIprint verwendet. Dort darf sie allerdings keine Liste von Pfaden sein, sondern nur ein einzelner Pfad.
(*Default:* "`tex:config`")

## 3.3. Konfigurationsfile "tex.cnf"

Der Nachteil der meisten Implementierungen ist die fest vorgegebene Größe der einzelnen Arrays in TEX. Erscheinen nun neue Makro-Pakete, wie z.B. das neue *Font Selection Scheme* von LATEX, so erreicht man meist eine der Grenzen der vielen Arrays in TEX (beim genannten ist dies der *String Pool*) und der Benutzer erhält die nette Meldung: "you can ask a wizard to enlarge me".

Mit dem Konfigurationsfile `tex.cnf` kann der Benutzer nun selbst zu einem TEX-Wizard werden. Falls einmal die Meldung von oben erscheinen sollte, ändert man einfach die Größe des zu kleinen Arrays.

Im Konfigurationsfile werden vier Typen von Informationen abgelegt:
(1) Suchpfade
(2) Voreinstellungen
(3) CodePage-Information
(4) Größe der TEX-Arrays

Die Information wird als Schlüsselwort-/Werte-Paar angegeben, dabei muß das Schlüsselword am Zeilenanfang beginnen und mit Leerzeichen oder Tabs vom Wert getrennt werden.

Nicht erkannte Schlüsselworte werden ohne Meldung ignoriert, dies kann man nutzen, um Kommentare in das File einzufügen. Besser ist jedoch die Verwendung eines Prozentzeichens '`%`' als Kommentarbeginn. Der Rest der Zeile nach dem Prozentzeichen wird ignoriert.

Das Konfigurationsfile wird bis zum Fileende gelesen, jedoch kann man durch ein '`#`' als erstes Zeichen einer Zeile das Lesen vorzeitig abbrechen. Dies ist z.B. sinnvoll, wenn danach noch ein längerer Kommentar folgt.

### 3.3.1. Suchpfade

Statt der Angabe von Suchpfaden über die Environment-Variablen können diese auch hier angegeben werden. Als Schlüsselwörter sind bekannt: `TEXINPUTS`, `TEXFORMATS`, `TEXFONTS` und `TEXPOOL`. Diese müssen groß geschrieben werden.

Der Wert der entsprechenden Environment-Variable überschreibt dabei die Angabe im Definitionsfile.

### 3.3.2. Default Sprache

Die Zeile

```
language    197
```

setzt den TEX-Counter `\language` auf den gegebenen Wert, hier 197. Dies entspricht '`\language=197` ' zu Beginn des zu formatierenden Files. Hat man mit der `-l`-Option über die Kommandozeile einen Wert übergeben, so wird der Wert im Konfigurationsfile ignoriert.

### 3.3.3. TEX-Arrays

Die Original TEX-Source ist in der Sprache Pascal ohne Erweiterungen, wie dies z.B. Turbo-Pascal bietet, geschrieben worden. Dadurch konnten Portierungen auf andere Systeme und Sprachen einfach und ohne große Änderungen durchgeführt werden. Der Nachteil: die Größe aller Array ist zum Compilierzeitpunkt statisch fest vorgegeben.

Diese Implementierung erlaubt es, die Größe der meisten Arrays, die TEX intern verwendet, erst zum Startzeitpunkt zu setzen. Welche Größen gesetzt werden können und welche Bedeutung diese Arrays haben, wird in einem nachfolgenden Abschnitt behandelt.

Für nicht im Konfigurationsfile angegebene Werte werden Defaultwerte verwendet, bei mehrmaligem Setzen wird der letzte Wert genommen.

Mit der `-d` Option kann man sich die Werte anzeigen lassen.

### 3.3.4. CodePage

Die CodePage-Definition beginnt mit dem Schlüsselwort `codepage` zu Beginn einer Zeile, beendet wird sie durch zwei Kleinerzeichen `<<`. Werden mehrere Definitionen in einem Konfigurationsfile angegeben, so werden alle als eine einzige Definition angesehen.

Im Kapitel 5 wird dies noch genau erklärt.

### 3.3.5. Ein Beispiel-Konfigurationsfile

So könnte das Konfigurationsfile `tex.cnf` aussehen:

```
% tex.cnf Beispiel eines Konfigurationsfile fuer TeX
%
% zuerst mal alle moeglichen Environment-Variablen setzen ...
%
TEXINPUTS       .,TeX:macros,Work:tex/macros,Work:tex/texte
TEXFORMATS      .,TeX:formats,Work:tex/formats
TEXFONTS        .,TeX:fonts
% nur fuer IniTeX:
TEXPOOL         .,TeX:,Work:tex/pool
%
create-info     on
%info-file-name TeX:config/dvi.info
% ... und jetzt noch einige Werte umsetzen
```

```
%
stringvacancies 10000
maxstrings       6000
triesize        16000
itriesize       19000    % triesize (for IniTeX only)
memmax          33000
memtop          33000
#
^-- Markiert das Ende von TeX.cnf: '#' am Zeilenanfang

Hier kann nun noch beliebig viel Kommentar-Text kommen.
```

Die Zeile `create-info` gibt an, ob TEX ein Info-File für das erzeugte DVI-File anlegen soll. Falls schon ein Info-File existiert, wird es natürlich nicht überschrieben. Mit `info-file-name` kann man optional einen anderen Namen/Pfad des Info Files angeben, das als Vorlage verwendet wird.

## 3.4. TEX – ARexx Schnittstelle

### 3.4.1. Editor Aufruf

Wie im normalen Unix[6]-TEX kann man auch von PasTEX aus einen Editor starten. Hat TEX einen Fehler gefunden, so kann man mittels des '`e`' Kommandos einen Editor aufrufen.

Wie in Kapitel 3.2 schon geschrieben, kann man mit Setzen der Variable `TEXREXX` bestimmen, daß der Editor nicht mittels eines System-Kommandos, sondern über eine ARexx Script aufgerufen werden soll. Der Weg über eine Script wurde gewählt, um eine größtmögliche Flexibilität zu erhalten.

### 3.4.2. ARexx Scripts

Es gibt eine Reihe von ARexx Scripts, die die Zusammenarbeit von TEX – Editor – ShowDVI – DVIprint vereinfachen. Zu diesen Scripten existiert derzeit leider nur die englische Anleitung `TeXRexx.tex`.

Hier ein kurzer Überblick über die wichtigsten Scripten:

`TeX-Server.rexx` Dies ist die Hauptscript. Sie muß immer im Hintergrund laufen. Sie kann unter 2.0 auch von ShowDVI aus aufgerufen werden.

`Start_TeX.*` Dies sind Scripten, die die Übersetzung eines TEXFiles starten. Es gibt diese Script für verschiedene Editoren und für ShowDVI.

`Quit_TeX.*` Diese Scripten senden der `TeX-Server` Script den Befehl, sich zu beenden.

`*toFront.*` Eine Sammlung von Scripten, die die verschiedenen Screens nach vorne holt.

`TeXedit.rexx` Dies ist eine wichtige Script. Sie wird von TEX aus aufgerufen, wenn ein Fehler aufgetreten ist. Diese Script wiederum ruft die

[6] Unix ist ein registrated Trademark from AT&T...

| | |
|---|---|
| | verschiedenen Editoren mit den fehlerhaften TEX-File auf und positioniert den Cursor an die beanstandete Zeile. |
| `NextError.ced` | Wurden in TEX mehrere Fehler gefunden, so kann man mit dieser Script von einem zum nächsten Fehler springen. |
| `StartDVIprint.*` | Diese Scripten rufen DVIprint auf. |
| `CallMF` | Dies ist eine Sammlung von Scripten, die von den Treibern aufgerufen werden, falls Fonts nicht gefunden werden. |

# 4. Arrays in TEX

Im Konfigurationsfile können den folgenden Parametern Werte zugewiesen werden:

`memmax`
`memtop`

`memmax` bzw. `memtop` geben die Größe des wichtigsten Arrays in TEX an. In diesem "main memory" wird z.B. die komplette Seite aufgebaut und Makrodefinitionen abgelegt.
`initex` ignoriert den Wert von `memmax` und setzt `memmax` = `memtop`. Der Wert von `memtop` soll beim Dumpen mit `initex` möglichst groß gewählt werden, dieser wird im Format-File mitabgelegt.
Für `virtex` und `initex` muß gelten: `memtop` $\leq$ `memmax` $\leq$ 65534. Man kann also für `virtex` ein größeres "main memory" als für `initex` verwenden.
Speicherbedarf: 4 Bytes $*$ `memmax` bzw. `memtop`

`triesize`
`itriesize`
`trieopsize`

Alle Trenntabellen, die von `initex` eingelesen werden, werden sehr kompakt in einem "Trie" gehalten. `initex` benötigt jedoch viel Speicher für diesen Trie, da er erst vollständig aufgebaut werden muß, bevor dieser komprimiert werden kann[7]. Durch den sehr großen Speicherbedarf des Tries in `initex`, muß bei wenig Speicher `membot` auf einen Wert $<$ 65534 gesetzt werden. Da der komprimierte Trie für `virtex` sehr viel weniger Platz benötigt, kann meist `memmax` auf 65534 gesetzt werden, so daß man zum Formatieren den maximal verfügbaren 'main memory' zur Verfügung hat. Die Größe des Trie's wird durch `triesize` (für `virtex`) bzw. `itriesize` (für `initex`) angegeben, beide müssen kleiner als 65536 sein. Die Anzahl an "trie operands" ist `trieopsize` ($\leq$ 32767).
Speicherbedarf:

| | `(i)triesize` | `trieopsize` |
|---|---|---|
| `initex` | 15 Bytes | 10 Bytes |
| `virtex` | 5 Bytes | 3 Bytes |

`fontmax`
`fontmemsize`

[7] Dies ist der Hauptunterschied von `initex` und `virtex`

`fontmax` gibt die Anzahl der maximal geladenen Fonts an (muß $\leq 255$ sein). Die Fontinformation aus den `tfm`-Files wird in ein Array der Größe `fontmemsize` geladen. `fontmemsize` kann dabei beliebig groß sein.
Speicherbedarf: 79 Bytes $*$ `fontmax` + 4 Bytes $*$ `fontmemsize`

`maxstrings`
`poolsize`
`stringvacancies`

In einem Array namens "String Pool" werden alle Strings, darunter fallen auch alle TEX-Makros und -Primitive, wie z.B. `\relax`, abgelegt. Die Größe dieses Arrays wird durch `poosize` bestimmt, die Anzahl der Strings durch `maxstrings`. `stringvacancies` gibt nun an, wieviel Platz im String Pool nach dem Laden eines Format-Files noch mindestens für vom Benutzer definierte Strings vorhanden sein muß. Alle drei Parameter können unbegrenzt größ gewählt werden.
Speicherbedarf: 1 Byte $*$ `poolsize` + 4 Byte $*$ `maxstrings`

`bufssize`
`maxinopen`

Jede eingegebene Zeile (von einem File oder von der Tastatur) wird in einem Array der Größe `bufsize` abgelegt. `bufsize` muß kleiner als 65536 sein. `maxinopen` gibt die Anzahl der gleichzeitig offenen Files an (muß kleiner als 128 sein).
Speicherbedarf: 1 Byte $*$ `bufsize` + 8 Byte $*$ `maxinopen`

`maxprintline`
`errorline`
`halferrorline`

`maxprintline` gibt die maximale Länge der Log-Ausgabe auf den Bildschirm bzw. ins Log-File an (üblicherweise = 79, muß $\geq 60$ sein). `errorline` gibt die maximale Länge der Kontextinformation bei einem Fehler an (muß $\geq 45$ sein), `halferrorline` ist die Länge der ersten Zeile dieses Kontextes ($30 \leq$ `halferrorline` $\leq$ errorline $- 15$).
Speicherbedarf: 1 Byte $*$ `errorline`

`savesize`
`stacksize`
`dvibufsize`

Zum Sichern von Werten innerhalb von Gruppen (lokale Zuweisungen, `\aftergroup`) wird Platz der Größe `savesize` verwendet. `stacksize` gibt die Anzahl der "Input-Sources" an (File, Makros, Tokenlists, ...). Der Buffer zum Schreiben des `dvi`-Files ist `dvibufsize` Bytes groß[8].
Speicherbedarf: 2 Bytes $*$ `savesize` + 10 Bytes $*$ `stacksize` + 1 Byte $*$ `dvibufsize`

Welche Werte sollen man nun den einzelnen Parametern zuweisen? Dazu zieht man

[8] `dvibufsize` muß durch 8 teilbar sein!

einfach das Log-File zu Rate, das beim Dumpen eines Format-Files erzeugt wurde[9].

Beim Dumpen von TEX (German `plain.fmt`) erhält man folgendes Log-File:

```
This is a PD-Version of Pas-TeX (made Jan 26 1991 [br]/[hes])
This is TeX, C Version 3.1 (INITEX)  28 JAN 1991 02:51
**plain \input amiga \input /doc/nice \dump
(plain.tex Preloading the plain format: codes, registers,
             ... [lines deleted] ...
Beginning to dump on file plain.fmt
 (format=plain 91.1.28)
2121 strings of total length 28932
7874 memory locations dumped; current usage is 118&7748
1063 multiletter control sequences
\font\nullfont=nullfont
\font\footfont=cmr10
             ... [lines deleted] ...
16011 words of font info for 54 preloaded fonts
0 hyphenation exceptions
Hyphenation trie of length 9980&11780 has 281 ops out of 500
  281 for language 0
No pages of output.
```

Daraus erkennt man untere Grenzen für einige wichtige Parameter: `maxstrings` $>$ 2121, `poolsize` $>$ 28932, `memmax` $>$ 7874, `fontmax` $>$ 54, `fontmemsize` $>$ 16011. Die tatsächlich verwendeten Werte sollten größer gewählt werden: `memmax` solte, wenn möglich, auf 65534 gesetzt werde, die anderen Parameter auf mindestens 10% größere Werte.

Der Trie mit den Trenntabellen kann, nachdem er komprimiert wurde, nicht mehr grösser werden. Daher kann man `triesize` auf einen Wert $\geq$ 9980 und `triopsize` auf $\geq$ 281 setzen. (Zum Dumpen dieses Formates muß für `initex` der Parameter `trieopsize` $>$ 11780 gesetzt werden.)

Alle anderen Parameter sollte man unverändert lassen und erst nachdem die Fehlermeldung `TeX capacity exceeded` auftritt, sollte der entsprechende Parameter vergrößert werden.

## 4.1. Unterschied TEX–BigTEX

Was ist nun dieses ominöse BigTEX, daß schon einmal in der Anleitung angesprochen wurde?

Dies ist praktisch identisch mit dem normalen TEX, nur daß die Elemente manche interne Arrays vergrößert wurden. Daher können nun manche Arrays größer gemacht werden, als dies im "kleinen" TEX möglich ist.

Diese Arrays sind:

[9] Aus diesem Grund sollte man diese Log-Files nicht löschen.

| Array | TEX | BigTEX |
|---|---|---|
| `memmax/memtop` | 65532 | 524284 |
| `maxstrings` | 65536 | *beliebig* |
| `bufsize` | 65536 | *beliebig* |

Damit man mit dem normalen TEX und dem BigTEX parallel arbeiten kann, verwendet BigTEX nicht das File `TeX.cnf` sondern `BigTeX.cnf`. Dadurch kann man für die beiden TEX-Versionen verschiedene Arraygrößen, sowie verschiedene Suchpfade definieren. Vor allem den `TEXFORMATS` Pfad sollte man in `BigTeX.cnf` anders als in `TeX.cnf` definieren, da die Formatfiles der beiden Versionen nicht untereinander kompatibel sind.

Einen Preis muß man natürlich auch für diese gewonnene Flexibilität zahlen. Der Speicherverbrauch nimmt erheblich zu. Nicht nur weil Sie jetzt größere Arrays allocieren können, sondern auch da die einzelnen Arrayelemente nun mehr Speicherplatz benötigen. Bei `memmax` sind dies etwa um den Faktor 1.5 mehr Bytes.

Wozu aber beötigt man so große Arrays?

Eigentlich nur für wenige Specialfälle. Der wohl am häufigsten auftretende dürfte das Makropaket *pictex* sein. Dieses erstellt Graphiken nur aus Buchstaben. Man kann sich denken, daß da schnell eine ganze Zahl von einzelnen Zeichen zusammenkommen, bis ein Bild fertig erstellt ist. Um mit diesen Makropacket vernünftig arbeiten zu können benötigt man schon ein BigTEX. Um aber mit BigTEX vernünftig arbeiten zu können benötigt man schon wenigstens 3MB an Hauptspeicher.

# 5. CodePage — Was ist das ?

TEX verwendet eine für alle Implementierungen standardisierte Kodierung der einzelnen Zeichen, ASCII. Diese wurde jedoch nur festgelegt für Zeichen von `0x20` (Space) bis `0x7f` (Delete) und einige andere, wie `^^@` (NUL), `^^M` (CR), `^^J` (LF), `^^L` (FF) und `^^I` (Tab). (Wie man sieht, nur 7bit-Zeichen und einige Control-Zeichen.)

Existiert nun auch für Sonderzeichen, wie z.B. Umlaute, eine Festlegung auf die 8bit-Zeichen (Zeichen $\geq 128$), wird auf jeden Fall eine Umwandlung der Kodierung im Amiga auf die in TEX verwendete Kodierung erforderlich. Diese kann man nun in die Implementierung "hart" eincodieren oder man läßt dem Benutzer die Freiheit, diese in einer `Codepage` angeben zu können, so daß Änderungen sehr einfach werden.

Die Codepage ist rechnerspezifisch, d.h. tauscht man TEX-Files zwischen zwei Rechnern mit unterschiedlicher Zeichensatz-Belegung aus, so muß man auch die Codepage ändern. Nur die intern von TEX verwendete Kodierung ist zwischen allen Rechnern gleich[10].

[10] Dies kann man ausnutzen, um TEX-Files rechnerunabhängig zu machen: Man verwendet einfach die Notation `^^xx` (xx ist eine Hex-Zahl in Kleinbuchstaben),

## 5.1. Funktionsweise

TEX verwendet intern zwei Arrays `xord[]` und `xchr[]` zur Umwandeln bei der Eingabe bzw. bei der Ausgabe. Beim Lesen wird jedes Zeichen `s` durch das Zeichen `t:=xord[s]` ersetzt, beim Schreiben wird das Zeichen `t` anschliessend wieder ersetzt durch `s:=xchr[t]`. Im Endeffekt wird das Zeichen `s` des Rechners in TEX durch das Zeichen `t` dargestellt.

Sonderzeichen (d.h. Control-Zeichen und 8bit-Zeichen) werden üblicherweise durch `^^xx` dargestellt (`xx` ist dabei die Zeichenposition im TEX-Zeichensatz in Hexnotation), dies hat den Vorteil, daß dieses Zeichen rechnerunabhängig wieder korrekt eingelesen werden kann. Als Benutzer ist es jedoch etwas sinnvoller, darstellbare Sonderzeichen auch als solche anzuzeigen, deshalb verwendet TEX ein weiteres Array, in dem die Darstellungsweise abgelegt wird[11].

In der Codepage-Definition hat man nun die Möglichkeit, den Default-Inhalt dieser Arrays zu verändern.

## 5.2. Syntax

Die Codepage-Definitionen werden im Konfigurationsfile `tex.cnf` angegeben, dabei kann man die Definition auch in mehrere Teile angeben. Jeder Teil wird durch `codepage` zu Beginn der Zeile eingeleitet, anschliessend folgt pro Zeile eine Anweisung. Die Definition wird durch `<<` abgeschlossen.

Folgende Anweisungen sind möglich:

(1) `x = y`
Das Zeichen `x` wird beim Lesen umgewandelt in `y`, geschrieben wird `y` nicht mehr in der Notation `^^yy`, sondern als `x`.
*Beispiel:* (`"a` wäre Amiga-Zeichen (ein Buchstabe!) für Umlaut a, `^^80` sei TEX-Codierung für den Umlaut a)
Nach

```
"a=^^80
```

wird jedes `"a` sofort in 128 umgewandelt, außerdem gibt TEX für Zeichen 128 nicht mehr `^^80` aus, sondern `"a`.
(xord[x]=y; xchr[y]=x; printable(y)=true;)

(2) `x > y`
Das Zeichen `x` wird beim Lesen umgewandelt in `y`. Hier kann `y` aus dem vollständigen Bereich `^^00-^^ff` gewählt werden.
*Beispiel:*
Nach

```
"a>a
```

wird `"a` als `a` gelesen.
(xord[x]=y;)

---

`xx` ist dabei nicht(!) die Kodierung des Zeichens im Zeichensatz des Rechners, sondern die von TEX verwendete.

[11] Dieses weitere Array entspricht den ersten 256 Strings des *String Pools*.

(3) `< y`
Das Zeichen `y` wird nicht mehr in der Notation `^^yy` ausgegeben, sondern als *ein* Zeichen `y`. Dabei findet evtl. noch bei der Ausgabe eine Umwandlung statt, falls eine Anweisung `x = y` in der Codepage-Definition angegeben wird.
(printable(y)=true;)
*Default:* alle Zeichen $< 32$ oder $> 126$ werden in `^^y` Notation ausgegeben.
(printable(y)=false; for y in [0...32, 126...255])

(4) `<| y`
Negation zu (3). `y` wird danach als `^^yy` ausgegeben.
*Beispiel:*
Nach

```
"a=^^80
```

wird das Zeichen 128 als ein Zeichen `"a` ausgegeben. Soll dies unterbleiben, danach noch angeben:

```
<|^^80
```

(printable(y)=false;)

Die einzelnen Anweisungen werden zeilenweise abgearbeitet, dabei ist die Reihenfolge der Anweisungen wichtig.

Allgemeine Anmerkung:

Gibt man in TEX Zeichen in `^^`-Notation an, so werden diese nicht mehr gemappt, d.h. mit `^^`-Notation muß der Zeichencode in TEX und nicht der im jeweiligen Rechner verwendeten Code angegeben werden.

Gemappt werden nur direkt eingegebene Zeichen. Mapping findet beim Lesen in `input_line()` statt, außerdem beim Schreiben mit `print_char()` und beim Umwandeln von Filenamen.

## 5.3. Einschränkungen

Die Codepage-Definition ist nur in `initex` sinnvoll, da die Arrays in das Format-File mit abgelegt werden. D.h., verwendet man mehrere Codepages, so muß für jede ein neues Format-File erzeugt werden. (Diese Einschränkung soll in einer späteren Version wegfallen.)

In der ersten Zeile, die `virtex` liest, findet noch keine Umsetzung statt, da zu diesem Zeitpunkt das Format-File noch nicht gelesen wurde.

# 6. Installation

Bevor man die im vorigen Kapitel beschrieben Programme benutzen kann, müssen diese so vorbereitet werden, daß sie wie gewünscht funktionieren. Diese Vorbereitung nennt man *Installation*.

Sie besteht darin, Programme und Daten zu kopieren und die Datei `S:Startup-Sequence` so zu erweitern, daß nach dem Anschalten und erfolgreichem Starten

des Rechners sofort mit TEX gearbeitet werden kann. Wie TEX zu installieren ist, kann im folgenden oder auch in der Datei "`README`" im Verzeichnis `PasTeX1.3_1` nachgelesen werden.

## 6.1. Automatische Installation

Für die Installation von TEX gibt es ein Programm (eigentlich eine Script-Datei) mit Namen "`InstallPasTeX`", welches – einmal gestartet – alles nötige kopiert und einfügt. Die Script stellt dann noch ein paar Fragen,was alles kopiert werden soll. Darauf kann man mit '`y`' für Ja, oder '`n`' (oder RETURN) für Nein antworten.

Die Script "`InstallPasTeX`" wird wie folgt aufgerufen:

`execute PasTeX1.3_1:InstallPasTeX` *Verzeichnis*

Für *Verzeichnis* müssen Sie den Namen eines Verzeichnisses auf Ihrer Festplatte angeben, in welchem TEX installiert werden soll. Je nachdem ob Sie auch BigTEX installiert haben möchten und mit wievielen verschiedenen Formatfiles Sie arbeiten möchten werden zwischen 3MB und 5MB Plattenplatz benötigt.

**Wichtig:** Fast alle Einstellungen, die bei der Installation gemacht werden, kann man nach eigenem Gutdünkem verändern, nur eine sollte man nicht ändern: Es empfiehlt sich dringend, das Assign `TeX:` auf dem Verzeichnis, in welchem alle Programme und Daten stehen, zu belassen. Die automatische Installation nimmt diese Einstellung vor. Man sollte erst dann versuchen sie zu ändern, wenn man ein kleiner PasTEX-GURU ist. Denn es ist auch möglich ohne dieses Assign zu arbeiten. Dann müssen aber an einigen Stellen Voreinstellungen geändert werden.

Sollten wider Erwarten bei der automatischen Installation nicht zu überwindende Probleme auftreten, lesen Sie bitte im Kapitel *Fehler, Probleme und Lösungen* nach, woran dies liegen könnte. Falls Sie das auch nicht weiterhilft, so versuchen Sie einen anderen Benutzer von PasTEX zu finden, den Sie fragen können. Im allerschlimmsten Fall können Sie mir auch schreiben.

## 6.2. Unterverzeichnisse

Die Unterverzeichnisse von `TeX:` haben ganz bestimmte Bedeutungen. So erwarten Programme wie `virtex`, `DVIprint` und `ShowDVI` Fontangaben, Makrodateien und Fonts normalerweise (per default) in ganz bestimmten Verzeichnissen.

Im folgenden soll aufgelistet werden, welche Dateien in welchen Verzeichnissen stehen (sollten). Diese Zuordnung sollte man nicht ohne triftige Gründe ändern. Denn sie ist quasi einheitlich auf vielen Rechner in der vorgegebenen Form zu finden und macht es Ihnen und anderen leicht, sich zurechtzufinden, wenn Sie auf einem fremden System sind.

Hier kommt die Liste:

`TeX:bin` hier befinden sich die ausführbaren Programme. Diese können auch in einem anderen Verzeichnis ste-

hen, Hauptsache das Verzeichnis mit diesen Programmen liegt im Pfad. Es sind

`virtex` –
`initex` – Die TEX-Programme
`input` – Eingabe-Befehl für das InstallTeX-Script

Wenn Sie unseren Previewer und Druckertreiber benutzen, sollten Sie diese ebenfalls in diese Verzeichnisse kopieren. Auf Wunsch übernimmt dies die automatische Installation.

`TeX:bigbin` hier sind die BigTEX Programme `initex` und `virtex`. Das sie genauso heißen, wie die normalen TEX Programme sind sie in ein extra Verzeichns untergebracht.

`TeX:fonts` beinhaltet die `.tfm`-Dateien. In diesen Datei steht für jeden Font und jeden Buchstaben eines Fonts, wie groß er ist. Diese Größenangaben braucht `virtex` zum Übersetzen der Texte.

`TeX:macros` hier sucht `virtex` normalerweise nach TEX-Texten, die er im aktuellen Verzeichnis nicht finden kann. Für LaTEX-Benutzer: Hier sollten die Style-Files stehen.

`TeX:formats` TEX an sich, hat einen sehr kleinen und unpraktischen Befehlssatz. Um damit gut umgehen zu können, gibt es für TEX verschiedene Makropakete. Zu den am meist gebrauchten gehören `plain`-TEX (normalerweise nur als TEX bezeichnet) und `lplain` (stets LaTEX- genannt). Die Namen dieser Dateien enden stets auf `.fmt`. Die Makropakete stehen in diesem Verzeichnis.

`TeX:bigformats` in diesem Verzeichnis sucht BigTEX nach seinen Formatfiles.

`TeX:fontlib` wird vom TEX-Compiler nicht gebraucht. In diesem Verzeichnis suchen Previewer und Druckertreiber nach Fontlibaries[12].

`TeX:pk` wird vom TEX-Compiler nicht gebraucht. In diesem Verzeichnis suchen Previewer und Druckertreiber nach Pixel-Fonts.

`TeX:config` hier wird normalerweise das Konfigurationsfile `tex.cnf` gesucht. Auch die Treiber suchen dort ihre Definitionsfiles.

`TeX:doc` hier finden nach der Installation alle Beschreibungen als `TeX`- und Text-Dateien.

[12] Siehe Beschreibung zum `flib`-Kommando

`TeX:doc.englisch` falls Sie dieses Directory auch haben installieren lassen, befindet sich dort die (leicht veraltete) Englische Anleitung. Allerdings befindet sich dort auch eine Anleitung, zu der *noch* kein deutsches Äquivalent gibt. Dies ist die "`TeXRexx.tex`" Anleitung zu den ARexx Scripts. Von daher ist dieses Verzeichniss also auch nicht ganz unwichtig.

## 6.3. Pfade

TEX benötigt eigentlich keine speziellen Pfade. Es ist aber günstig die Programme wie `virtex`, `ShowDVI`, und `DVIprint` nicht nach `C:`, sondern nach `TeX:bin` zu kopieren und auf dieses Verzeichnis einen Pfad zu legen. Dann fällt es Ihnen viel leichter alte Versionen durch neuere zu ersetzen, bei Platzproblemen TEX auf Disketten zu sichern und zu löschen und so weiter.

In der `S:Startup-Sequence` sollte deswegen ein Eintrag wie

```
path TeX:bin add
```

stehen.

Statt die vielen zusätzlichen Einträge, die für TEX nötig sind, in die `S:Startup-Sequence` aufzunehmen, können Sie diese auch in eine Script-Datei schreiben und ausführen, sobald Sie TEX benutzen wollen. Auf meinem Rechner gibt es beispielsweise die Datei `init-tex`, die automatisch die Anweisungen in `S:Shell-Startup` ausführt und die zusätzlichen, für TEX nötigen Schritte unternimmt.

Wer möchte, kann so eine spezielle Shell starten,

```
newshell NEWCON:0/11/640/200/TeX s:init-tex
```

in der alle nötigen Einstellungen schon gemacht sind. Wenn man dann noch in der Datei `S:Shell-Startup` den Eintrag

```
alias inittex newshell NEWCON:0/11/640/200/TeX s:init-tex
```

stehen hat, reicht es im CLI `inittex` einzugeben, um eine typische TEX-Shell zu starten.

# 7. Ein Probedurchlauf

In diesem Kapitel exerziere ich mit Ihnen einen Probelauf durch, damit sie sehen, wie man mit TEX Texte übersetzt und wie einfach das geht. Ich habe diese Kapitel in vier Abschnitte gegliedert:

1. Einrichten eines Verzeichnisses
2. Übersetzen mit `virtex`
3. Anschauen mit `ShowDVI`

4. Drucken mit `DVIprint`

## 7.1. Einrichten eines Verzeichnisses

Wechseln Sie in das Verzeichnis, in welchem Sie ein weiteres Unterverzeichnis einrichten wollen. Bei mir gibt es ein Verzeichnis `TeX:texte` in welchem alle weiteren Unterverzeichnisse stehen. Also richten Sie sich am besten ein solches Verzeichnis ein und tippen Sie dann

```
cd TeX:texte
makedir testdir
cd testdir
```

Jetzt ist es an der Zeit, mit einem Editor einen Text einzugeben. Dieser sollte mit dem Kürzel `.tex` enden, sonst wird ihn `virtex` nicht finden:

```
ed testdatei.tex
```

Ein Beispiel für einen kurzen, aber berühmten Text:

```
\hrule
\vskip 1in
\centerline{\bf A SHORT STORY}
\vskip 6pt
\centerline{\sl by A. U. Thor}
\vskip .5cm
Once upon a time, in a distant
  galaxy called \"O\"o\c c,
there lived a computer
named R.~J. Drofnats.

Mr.~Drofnats---or ``R. J.,'' as
he preferred to be called---
was happiest when he was at work
typesetting beautiful documents.
\vskip 1in
\hrule
\vfill\eject
\bye
```

Wenn Sie diesen Text abgetippt und abgespeichert haben, können Sie an die Übersetzung gehen.

## 7.2. Übersetzen mit “virtex”

Bevor Sie den Text übersetzen können, sollten Sie sich vergewissern, daß sich alle Programme und Daten in den richtigen Verzeichnissen befinden und die Environment-Variablen korrekt gesetzt sind. Zumindest die Variable `TEXFORMATS`

sollte gesetzt sein und als Eintrag den Namen des Verzeichnisses `TeX:formats` enthalten. In diesem Verzeichnis muß eine Datei mit Namen `plain.fmt` stehen. Wenn Sie lieber die deutsche Version von TEX verwenden (das Makropaket heißt `plaing.fmt`), dann ersetzen die `plain` durch `plaing` im folgenden Aufruf.

Der Aufruf von `virtex` sieht so aus:

```
virtex &plain testdatei
```

Nach der Eingabe sollte sie folgende Ausgaben auf dem Bildschirm zu sehen bekommen:

```
This is a PD-Version of Pas-TeX (made Jan 22 1991 [br]/[hes])
This is TeX, C Version 3.1
(testdatei.tex [1] )
Output written on testdatei.dvi (1 page, 672 bytes).
Transcript written on testdatei.log.
```

`virtex` hat eine Ausgabedatei mit Namen `testdatei.dvi` und eine Protokolldatei mit Namen `testdatei.log` erzeugt.

## 7.3. Anschauen mit ShowDVI

Den Inhalt dieser Datei können Sie sich nun mit einem sogenannten *Previewer* anschauen. *To view* ist englisch und heißt anschauen. Mit einem Previewer kann man sich den Ausdruck *vorher* anschauen. Das Programm hierzu heißt `ShowDVI` und befindet sich auf einer der PreviewDisks.

Bitte beachten Sie, daß sie zum Anschauen des Textes in `testdatei.dvi` mehr benötigen, als auf den Disketten `TeXDisk1` und `TeXDisk2` geliefert bekommen. Hierzu benötigen Sie die Disketten PreviewDisks 1, 2 und 3, die Sie bei uns beziehen können. Wichtig ist, daß die Font-Libraries im Verzeichnis `TeX:fontlib` stehen oder daß Sie die benötigten Fonts an einen anderen, für den Previewer bekannten[13], Ort kopiert haben.

Der Aufruf sieht wie folgt aus:

```
ShowDVI testdatei
```

Danach sollte eine neue Screen im Interlaced-Modus erscheinen und das Ergebnis anzeigen. Sie können `ShowDVI` wieder verlassen, in dem Sie `CTRL-C` drücken.

## 7.4. Drucken mit DVIprint

Das Drucken erfolgt in genau der gleichen Weise wie das Anzeigen des Textes mit `ShowDVI`. Bitte beachten Sie auch hierbei, daß Sie zum Drucken mehr benötigen, als auf den beiden Disketten `TeXDisk1` und `TeXDisk2` zu finden ist.

[13] Welche dies sind, steht in der Anleitung zu ShowDVI

Da sich Drucker verschiedener Hersteller manchmal deutlich unterscheiden, muß man dem Druckertreiber mitteilen, welcher Drucker angeschlossen ist. Er kann 9-Nadel-Drucker, 24-Nadel-Drucker (NEC-P6- kompatibel) und HP-Drucker bedienen. Welcher Drucker angeschlossen ist, kann man ihm mit der `-d`-Option mitteilen.

Folgende Aufrufe sind zum Beispiel möglich:

Für einen 9-Nadel-Drucker:

```
DVIprint -d 5 testdatei
```

Für einen 24-Nadel-Drucker:

```
DVIprint -d 1 testdatei
```

Für einen HP-Drucker:

```
DVIprint -d 3 testdatei
```

Danach sollten Sie einen wirklich *schönen* Ausdruck in den Händen halten.

# 8. Fehler, Probleme und Lösungen

Natürlich kann immer etwas schief gehen. Und meist geht etwas schief. Da Sie kein Roboter sind, machen auch Sie genauso wie ich Fehler. Programme reagieren auf so etwas meist recht unfein und stellen ihre Arbeit ein.

TEX reagiert auf Fehler mit ausführlichen Fehlermeldungen, die zuerst sehr verwirrend sind. Diese hier zu erklären, wäre zu aufwendig. Bitte schauen Sie sich dazu in einem Buch die entsprechenden Seiten an; die im folgenden Kapitel aufgeführten Bücher enthalten für die Fehlerbehandlung genug Informationen.

Wichtig für die Fehlersuche ist, daß Sie sich vergewissern, daß sie alle für die Installation wichtigen Assigns, Pfade, Environment-Variablen wirklich gesetzt oder angelegt haben. Bitte überprüfen Sie im Fehlerfall alles sehr sorgfältig. Es kommt darauf an, alles wirklich am Rechner nachzusehen, und nicht nur kurz im Kopf zu überschlagen, ob man dies oder jenes schon getan hat. Ein Aufruf von `dir` oder `assign` ist oft sehr aufschlußreich. Auch sollte man die Programme unbedingt unter einer *Standardumgebung*, ausprobieren. Falls einmal nichts geht, versuchen Sie das ganze nocheinmal, nachdem Sie von einer orginal Commodore Workbench-Disk gebootet haben. Ohne irgendwelche anderen Utilities und Shells zu verwenden.

Wer scheinbar unüberwindbare Probleme hat, der kann sich auch an mich (Georg Heßmann) wenden. Hierzu müssen aber folgende Regeln unbedingt eingehalten werden:

Schicken Sie uns einen Brief (nicht telefonieren, daß bringt meist gar nichts), in dem Sie uns das Problem genau erläutern. Drucken Sie ein Listing des Textes, bei dem Fehler auftritt, geben Sie uns einen Ausdruck Ihres Platteninhaltsverzeichnisses,

machen Sie einen Ausdruck aller Belegungen der Environmentvariablen. Wenn möglich schicken Sie uns eine Diskette mit dem Text, bei dem der oder die Fehler auftreten.

Und ganz wichtig: Ohne einen ausreichend frankierten, an Sie adressierten Briefumschlag können wir Ihnen keine Antwort schicken und werden dies auch nicht tun. Dies ist keine Böswilligkeit, sondern nur finanziell notwendig. Wir sind Studenten mit einem sehr geringen Einkommen und können diese Software nur zum Selbstkostenpreis abgeben, wenn wir dadurch keine Verluste erleiden.

Hier die Adressen:

Heimatadresse:

Georg Heßmann
Oberer Markt 7
8712 Volkach

Studienadresse:

Georg Heßmann
Ingling 17
4784 Schardenberg
Österreich

## 8.1. Fragen und Antworten

Hier ein Versuch ein paar Fragen schon im vorraus zu beantworten.

Frage: Kennt dieses TEX *virtuelle Fonts*?

Antwort: Nein, TEX selbst hat keine Ahnung von virtuellen Fonts. Er liest von Fonts nur die Information im zum Font gehörenden tfm-File (TEX Font Metric), dieses existiert sowohl für "normale", als auch für virtuelle Fonts.

Frage: Ich kann `tex` nicht finden?

Antwort: `tex` ist `virtex` mit einem zu ladenden Format (z.B. plainTEX oder LATEX).

Frage: Was ist ein *preloaded format*?

Antwort: Auf einigen Betriebssystemen kann man `virtex`, nachdem es das Format-File geladen hat, abbrechen und ein sogenanntes *Core*-File erzeugen. Aus diesem Core-File und dem ausführbaren File `virtex` kann man ein weiteres ausführbares File erzeugen, indem jedoch das Format-File schon geladen ist. Der Vorteil, ein meist schnelleres Laden von `virtex` und Formatfile, wird durch den Nachteil, dem Platzbedarf des neuen ausführbaren Files, der sehr viel größer als der von `virtex` ist, wieder ausgeglichen.

Frage: Ich erhalte beim Laden des Format-Files den Fehler `(Fatal format file error; I'm stymied)`.

Antwort: Alle anderen Filetypen (`tfm`, `dvi`, `pk`, ...) sind zwischen verschiedenen Implementierungen – auch auf unterschiedlichen Rechner – voll austauschbar. Das Format-File nimmt darin eine Sonderstellung ein, da es möglichst schnell zu laden sein sollte und eine Abbildung der internen TEX-Arrays darstellt. Dadurch kann es nicht mehr so einfach austauschen. Abhilfe: man erzeugt mit `initex` ein neues Format-File.

# 9. Literatur

Ohne Buch geht es nicht. Wenn Sie TEX teilweise auch sehr preiswert bekommen, so kommen Sie nicht umhin, sich ein Buch zu kaufen, was Sie in TEX einführt. TEX ist so mächtig wie eine Programmiersprache, kennt mehr Anweisungen als jedes DTP-Programm, meckert in vielfältiger Weise und fabriziert hin und wieder Dinge, die man nicht verstehen kann. Hier hilft nur ein Buch.

Da es auch recht gute deutsche Literatur gibt, möchte ich Ihnen zwei Werke empfehlen, die zu kaufen sich lohnt.

*Einführung in TEX*, Norbert Schwarz, Addison Wesley, 1988, ISBN 3-925118-97-7

*LaTEX – Eine Einführung*, Helmut Kopka, Addison Wesley, 1988, ISBN 3-89319-136-4

Wenn Sie auch des Englischen kundig sind, gibt es nur eine Wahl: das Buch zu TEX vom Erfinder selbst.

*The TEXbook*, Donald E. Knuth, Addison Wesley, Reading, 1988.