METAFONT für den *AMIGA*
Dokumentation zu Version 2.71

Andreas Scherer
Roland-Straße 16
52070 Aachen
Germany

scherer@genesis.informatik.rwth-aachen.de

26. November 1994

Dieser Text beschreibt meine Implementierung von METAFONT in der Version 2.71 für den Commodore Amiga. METAFONT ist der Name eines Systems zur Zeichensatzerzeugung von Prof. Donald E. Knuth von der Stanford Universität. Es liest Textdateien mit ‚Programmen', die das Aussehen von Buchstaben, graphischen Symbolen oder anderen Elementen festlegen, die für die Erstellung von qualitativ hochwertigen Dokumenten benötigt werden. Schriftsatzsysteme wie TEX verwenden die Informationen über die Zeichensatzabmessungen, die von METAFONT erzeugt werden, und Druckertreiber oder Bildschirmanzeiger verwenden die Pixelinformationen, wohin die „Tinte" gesetzt werden soll. Der Einsatz der zugehörigen Programme wird einigermaßen genau beschrieben, es wird aber keine Einführung in die METAFONT-Sprache selbst gegeben. Genaue Informationen über die Programmierung mit METAFONT finden sich im Literaturverzeichnis. (Abschnitt 6 dieses Textes.)

Inhaltsverzeichnis

1 Urheberrechtshinweise

Das METAFONT-Programm ist © 1984 für Donald E. Knuth. Alle Rechte sind vorbehalten. Der vollständige WEB-Quelltext ist veröffentlicht als [5]. Die zugehörigen Hilfsprogramme stammen von verschiedenen Autoren, unterliegen aber ausdrücklich nicht dem Urheberrecht und dürfen uneingeschränkt verwendet werden.

Diese Implementierung von METAFONT 2.71 für den Commodore Amiga ist zu FREEWARE erklärt. Alle Rechte sind vorbehalten. Sie dürfen das Paket verwenden, kopieren und weitergeben, und zwar in archivierter oder installierter Form, aber nicht mehr als eine geringe Bearbeitungsgebühr für Disketten und Versand verlangen.

Alle Programme dieses „Anwenderbausatzes“ verwenden das „EVPaths“ Paket. „EVPaths“ ist © 1994 für Giuseppe Ghibò. Alle Rechte sind vorbehalten. Der Einsatz von „EVPaths“ erfolgt mit freundlicher Genehmigung des Autors.

Obwohl beträchtliche Arbeit investiert wurde, das METAFONT-Programm fehlerfrei und zuverlässig zu machen, wird keinerlei Gewährleistung übernommen; der Autor und der Implementierer weisen jede Verantwortung für Schäden, die sich, direkt oder indirekt, aus der Benutzung oder dem Verhalten dieser Programme ergeben, zurück.

Diese Arbeit war „Liebesmühe“ und sowohl der Autor als auch der Implementierer hoffen, daß zukünftige Benutzer und Benutzerinnen daran ihre Freude haben.

Bitte senden Sie Anmerkungen, Vorschläge, Fehlermeldungen oder sonstige Zuschriften an eine der Adressen, die auf der ersten Seite angegeben sind. Dort können Sie auch ein „Entwicklerpaket“ erhalten, das den kompletten Quelltext mit allen Änderungen und Erweiterungen umfaßt. (Siehe Abschnitt 8 für nähere Einzelheiten.)

TEX ist ein Warenzeichen der American Mathematical Society.
METAFONT ist ein Warenzeichen von Addison-Wesley Publishing Company.
Commodore ist ein eingetragenes Warenzeichen von Commodore Electronics Limited.
Amiga, AmigaDOS, Amiga Kickstart und Amiga Workbench sind eingetragene Warenzeichen von Commodore-Amiga Incorporated.
SAS und SAS/C sind eingetragene Warenzeichen von SAS Institute Incorporated.
PostScript ist ein eingetragenes Warenzeichen von Adobe Systems Incorporated.

2 Systemvoraussetzungen

Die minimale Rechnerkonfiguration, mit der diese Implementierung von METAFONT getestet wurde, war ein Commodore Amiga 2000 mit dem handelsüblichen 68000 Prozessor von Motorola, 2 MB ChipRAM und 4 MB FastRAM, wobei mit den Standardeinstellungen tatsächlich nur etwa 1 MB benutzt wurde. Der Speicherbedarf kann sich in Abhängigkeit vom zu generierenden Zeichensatz und den gewählten Einstellungen in der METAFONT-Konfiguration erhöhen. (Siehe Abschnitt 4.2.) Das Betriebssystem bestand aus Kickstart 37.175 und Workbench 38.36. Alle in diesem Paket enthaltenen Programme können unter Version 1.3 des Betriebssystems ablaufen. Nur die Bildschirmanzeige von METAFONT benötigt Version 2, da sie erweiterte Graphikfunktionen einsetzt. Bei Verwendung der Version 1.3 ist diese Eigenschaft ausgeschaltet.

Es sind gesonderte Versionen der IniMF- und VirMF-Programme vorhanden, die für erweiterte Geräteausstattungen wie den Amiga 3000 oder Amiga 4000 oder Amigas mit Turbokarten gedacht sind. Diese beiden Programme verwenden Eigenschaften des 68020 Motorola Prozessors, die im 68000 Prozessor nicht vorhanden sind. Unter gleichen Bedingungen laufen diese Programmversionen etwa zwanzig bis dreißig Prozent schneller als die nicht-optimierten Versionen, auf Amigas ohne Prozessor mit den 32 Bit-Erweiterungen stürzen sie jedoch ab! Alle anderen Programme in diesem Paket sind so übersetzt, daß sie auf jeder handelsüblichen Amiga-Ausstattung laufen.

METAFONT ist ein großes System, gedacht für große Aufgaben. Die Erzeugung von `cmr10` für den 300 dpi DeskJet-Druckertreiber aus dem PasTEX-Paket benötigt auf einer 50 MHz schnellen 68030-Turbokarte mit mathematischem Koprozessor und ausreichend viel 32 Bit-RAM nur 52 Sekunden, auf einem Standard-Amiga mit 7 MHz und recht langsamem 16 Bit RAM dagegen 10 Minuten und 50 Sekunden, obwohl die Programme auf beiden Konfigurationen laufen. Ein ganz brutaler Test, bei deaktiviertem FastRAM (es blieben nur 2 MB ChipRAM, in denen alle notwendigen Programme und die Eingabedateien Platz fanden), benötigte 17 Minuten und 40 Sekunden.

3 Kompatibilität

Dies ist METAFONT 2.71 für den Commodore Amiga. Es stimmt völlig überein mit der originalen Implementierung von Donald E. Knuth und hat den TRAP-Test vom 25. Januar 1992 glänzend bestanden. Lediglich in `trapman.tex` als zulässig gekennzeichnete Abweichungen traten auf. `diff`-Dateien, die den letzten Lauf des TRAP-Testes dokumentieren, werden auf Anfrage vom Implementierer zugänglich gemacht. (Siehe Abschnitt 8 zu Einzelheiten.)

Weiter enthalten sind die Hilfsprogramme GFtoDVI Version 3.0 (Oktober 1989), GFtoPK Version 2.3 (Juli 1990), GFtype Version 3.1 (März 1991), MFT Version 2 (Oktober 1989), PKtoGF Version 1.1 (Oktober 1990) und PKtype Version 2.3 (November 1989) sowie VFtoVP und VPtoVF, beide in der Version 1.2 (September 1990).

4 Installation

4.1 Entpacken des Archives

Der „METAFONT-Anwenderbausatz“ ist in sich vollständig; alles was Sie zur Erzeugung wunderschöner Zeichensätze für TEX benötigen, finden Sie darin. Dieser Abschnitt beschreibt die verschiedenen Unterverzeichnisse und ihren Inhalt. Normalerweise sollte dieses Paket in Form eines `LHA`-Archives vorliegen, das eine arbeitsfähige Installation aller Programme und Skripten enthält. Leider gibt es kein ‚Installer‘-Skript, aber die Installation ist trotzdem sehr einfach. Zuerst müssen Sie die Dateien mit dem Kommando

```
> LHa x MetaFontV2.71[.lha] <Installationspfad oder -Device>
```

auspacken. (Sie benötigen dazu das `LHa`-Programm. Sollten Sie es noch nicht haben, holen Sie es sich von der Fish-Diskette 715 mit der aktuellen Version.[1]) Ein Hauptverzeichnis `MetaFont` wird angelegt, das das vollständige Archiv in einzelnen Unterverzeichnissen enthält, wie sie im folgenden beschrieben werden. Sie sollten dem Hauptverzeichnis den Namen „`MF:`“ mit „`Assign`“ zuweisen. Insgesamt benötigen Sie weniger als 3 MB Platz auf der Festplatte, um das komplette METAFONT-System zu installieren.

`MetaFont/bases`: Dieses Verzeichnis enthält die `base`-Dateien, die METAFONT zu Beginn eines Laufes sehr schnell laden kann. Hier finden Sie auch die Datei `plain.mf` (Version 2.71) und einige Installationsdateien für die Formate `plain` und `cm`. Neue `base`-Dateien sollten in diesem Verzeichnis angelegt werden.

`MetaFont/bin`: Der wichtigste Teil dieses Paketes sind die eigentlichen Programmdateien für METAFONT und die zugehörigen Hilfsprogramme. Sie sollten dem Betriebssystem behilflich sein, und die `PATH`-Umgebungsvariable um einen Verweis auf dieses Verzeichnis erweitern, damit die Programme aufgerufen werden können. Eine ausführlichere Beschreibung, wie die Programme aufzurufen sind, wird im nächsten Abschnitt 5 gegeben.

`MetaFont/config`: Die vorliegende Implementierung von METAFONT ist in der Lage, die Speicherbelegung beim Programmaufruf anzupassen, falls die benutzten Einstellungen zu klein für eine gestellte Aufgabe sind. In solch einem Fall bricht METAFONT unwiderruflich die Verarbeitung mit der Fehlermeldung `METAFONT capacity exceeded, sorry...` ab. Die Konfigurationsdatei `mfmemory.config` enthält Einträge der Form

```
set memmax              65530     % default 30000
set screenheight          200     % default  1024
set <internal size> <user value> % default <value>
```

Sie können die Einstellungen in beschränktem Maße verändern oder einzelne Zeilen aus dieser Datei löschen, so daß dann auf interne Werte zurückgegriffen wird. (Diese sind in der ‚default‘-Spalte vermerkt. Siehe Abschnitt 4.2 zu Einzelheiten über die Speicherkonfiguration.) Die Datei `modes` enthält eine Liste von Einstellungen, die mit der betriebsfertigen `base`-Datei dieses Benutzerbausatzes zusammenarbeiten.[2] Mehrere Drucker und der Standard-Previewer werden unterstützt. Sie können die Einträge an Ihre eigene Konfiguration anpassen.

`MetaFont/doc`: Zumindest dieses Verzeichnis kennen Sie schon, da Sie ja diesen Text lesen. Hier finden Sie die Dokumentation zu METAFONT 2.71 sowohl in deutscher als auch in englischer Sprache. Die LaTeX-Quellendateien müssen vorhanden sein, dagegen können die `dvi`-Ausgabedateien bei unzureichendem Diskettenplatz gelöscht sein. Diese können Sie mit Hilfe einer lauffähigen TeX-Installation

[1] Aber schicken Sie kein Geld an Stefan Boberg, denn es erfolgt keine Lieferung!

[2] Diese sind *nicht* verträglich mit den PasTeX-Einstellungen aus `pastex-modes.mf` aus dem CallMF Paket; sie beziehen sich stattdessen auf die Standardverteilung `modes.mf`.

aber neu erzeugen; es werden keine speziellen Makros oder ungewöhnlichen Zeichensätze benötigt.

`MetaFont/inputs`: Dieses Verzeichnis ist lediglich gedacht für METAFONT-Programmierer und Systemadministratoren. Es enthält Textdateien, die für die Erzeugung neuer oder veränderter `base`-Dateien benötigt werden. Hier sollten Sie zum Beispiel die vollständige Sammlung der Computer Modern Zeichensätze installieren, dem Standard für die meisten TEX-Anwender. Außerdem können Sie hier Ihre eigenen Zeichensatzprogramme hinzufügen.

`MetaFont/pool`: METAFONT (genau wie TEX) ist ein PASCAL-Programm und muß wegen der Einschränkungen dieser Programmiersprache eine eigene Methode zur Verwaltung von Zeichenketten anwenden. Die `pool`-Datei `mf.pool` enthält alle Texte und Meldungen, die METAFONT auf Ihren Bildschirm oder in die Logdatei ausgeben will. Sie wurde maschinell durch `TANGLE` bei der Übersetzung erzeugt und sollte nicht gelöscht oder in irgendeiner Weise verändert werden.

`MetaFont/rexx`: Dieses Verzeichnis enthält das Public Domain Paket ‚CallMF' (Version 1.0), bestehend aus drei ARexx-Skripten von Georg Heßmann, Martin Bokämper, Jörg Höhle und Ulrich Wisser. Es ist sehr sinnvoll in Zusammenhang mit Heßmann's PasTEX. Sowohl der Bildschirmanzeiger als auch der Druckertreiber versuchen CallMF aufzurufen, falls ein Zeichensatz in einer bestimmten Auflösung nicht vorhanden ist. Die ARexx-Skripten erzeugen den fehlenden Zeichensatz entweder automatisch oder in einem gesonderten Lauf.

Die Dokumentation zu CallMF [1] finden Sie in einem Unterverzeichnis von `rexx`, deshalb wird an dieser Stelle nicht näher auf die Installation und Verwendung von CallMF eingegangen. Lediglich auf Kompatibilitätsschwierigkeiten sei an dieser Stelle hingewiesen. Ich verwende derzeit ausschließlich Modi aus der Standardverteilung `modes.mf`, PasTEX geht aber von `pastex-modes.mf` aus. Leider stimmen die Bezeichnungen in diesen beiden Dateien nicht mit einander überein, so daß hier eine eigene Anpassung notwendig ist.

Empfohlen sei im Zusammenhang mit CallMF auch Version 5.58 des Programmes `DVIPS` von Tomas Rokicki in der exzellenten Amiga-Anpassung von Giuseppe Ghibò, die ebenfalls den Mechanismus mit den ARexx-Skripten verwendet.

4.2 Steuerung der Konfigurationsgrößen

Ursprünglich handelt es sich bei METAFONT (und TEX) um ein PASCAL-Programm, das nicht in der Lage ist, seine Speicherkonfiguration zur Laufzeit anzulegen. Um diese zu verändern, müssen Sie normalerweise den Quelltext von METAFONT mit geänderten internen Vorgabewerten neu übersetzen. Doch mit dieser Implementierung brauchen Sie nicht mehr die Meldung „`ask a wizard to enlarge me`“ fürchten, sondern können stattdessen selbst ‚zaubern'! Sie wurde nämlich mit Hilfe des `Web2C`-Sprachwandlers aus der UNIX-Verteilung durchgeführt und somit ist es möglich, die zusätzliche Eigenschaft zur Kontrolle der internen Variablen einzubauen.

Die Programme IniMF und VirMF verwenden zur Einrichtung ihrer Konfiguration die Datei `mfmemory.config`, die in einzelnen Zeilen Anweisungen der Form

```
set memmax       100000
set screenheight    400
set scalefactor       2
```

enthalten sollte. Die Beispieldatei im `config`-Verzeichnis ‚`set`‘zt die in meiner gegenwärtigen Standardkonfiguration verwendeten Werte, die etwas höher sind als die internen Vorgabewerte. Nicht alle Einstellungen müssen in dieser Datei angegeben werden. Beachten Sie aber, daß Sie auch nicht alle Einstellungen bis zum Maximum hochdrehen können – dazu müßten Sie mehr als 8 Gigabyte freien Speicher haben![3]

Wenn Sie eine Standard-`base`-Datei mit IniMF erzeugen, so ist jede weitere Änderung der Konfiguration `mfmemory.config` zulässig für VirMF in Verbindung mit dieser speziellen base-Datei, außer Änderungen am Wert `memtop`. Dies führt zur METAFONT-Meldung „`(Fatal base file error; I'm stymied)`“, was anzeigt, daß diese `base`-Datei von einem IniMF mit einer anderen Konfiguration stammt.

Hier ist eine vollständige Liste aller Parameter, die in der Konfigurationsdatei zulässig sind, ihre Vorgabewerte und einige Regeln für ihre kleinsten und größten Werte:

Bezeichnung	Vorgabe	Grenzwert	Zweck
`memmax`	30000	`maxhalfword`	Größe des Hauptspeichers
`maxinternal`	100	≤ 259774	Anzahl der Internas
`bufsize`	500	`maxhalfword`	Eingabezeichen
`errorline`	72	`maxhalfword`	Länge der Fehlerzeile
`halferrorline`	42	$<$`errline`-15	Erste Fehlerzeile
`maxprintline`	79	≥ 60	Länge der Ausgabezeile
`screenwidth`	768	4095	Breite der Bildschirmanzeige
`screenheight`	1024	4095	Höhe der Bildschirmanzeige
`stacksize`	30	300	Anzahl der Eingabequellen
`maxstrings`	2000	`maxhalfword`	Anzahl der Zeichenketten
`poolsize`	32000	`maxhalfword`	und der enthaltenen Zeichen
`movesize`	5000	10000	Speicher für Oktantenschritte
`maxwiggle`	300	1000	Gerundete Punkte pro Kreis
`gfbufsize`	100	4096	Ausgabepuffergröße
`pathsize`	300	1000	Knotenzahl in einem Pfad
`bistacksize`	785	785	Teilungsstack
`headersize`	25	100	Worte in TFM-Köpfen
`ligtablesize`	5000	32510	Ligaturen/Einrückungen
`maxkerns`	500	1000	Unterschiedliche Abstände
`maxfontdimen`	50	100	Anzahl von ‚fontdimen‘sionen
`memtop`	30000	`memmax`	Höchste Speicherstelle
`scalefactor`	1	10	Anzeige ‚SF‘-fach verkleinert
`maxinopen`	6	20	Anzahl der Ausgabedateien
`paramsize`	150	1000	Makro-Parameter

[3] Sollten Sie soviel haben, dann glaube ich Ihnen nicht!

Einige dieser Werte werden vor ihrer Verwendung noch mit 4 oder 8 multipliziert, so daß die tatsächlich benötigte Bytezahl für eine spezielle Einstellung nicht so einfach zu bestimmen ist. Solange Sie jedoch keine Probleme mit den Vorgabewerten oder dieser Standardkonfiguration haben, empfehle ich Ihnen, sie so zu lassen. In den meisten Fällen von ungenügender Speichergröße muß nur `memmax` erhöht werden, und dies wurde tatsächlich bis zum fürchterlich hohen Wert 750.000 ausprobiert, und zwar auf meinem Amiga 2000 mit 8 MB 32 Bit FastRAM.

Sie können die Größe der Bildschirmanzeige durch Veränderung der beiden Werte `screenwidth`, `screenheight` und `scalefactor` beeinflussen. Die Bedeutung der ersten beiden dürfte klar sein, nur sei angemerkt, daß `screenheight` den Wert für die *interlaced* Darstellung bezeichnet, obwohl non-interlaced Bildschirmmodi automatisch erkannt werden und entsprechend nur halb soviele Pixelzeilen angezeigt werden. Außerdem wird durch die beiden Werte nicht die Fenstergröße sondern die zugehörigen Werte `innerwidth` und `innerheight` eingestellt. Der `scalefactor` ist ein Reduktionsfaktor, das heißt die Anzeige wird sowohl in der Höhe als auch in der Breite um diesen Faktor verkleinert dargestellt. Um sinnvoll mit dieser Einstellung arbeiten zu können, enthält die mitgelieferte `plain.base` die internen Vorgaben

```
screen_rows:=4095; screen_cols:=4095; % infinity
```

wie sie in der Druckermodusdatei `modes.mf` festgelegt sind. Bei der Bildschirmanzeige verwendet METAFONT den jeweils kleineren Wert von `screenheight` und `screen_rows` beziehungsweise `screenwidth` und `screen_cols`, so daß mit diesen Maximalwerten nur die in der Konfigurationsdatei angegebenen Werte berücksichtigt werden.

Im Unterschied zu älteren Versionen von VirMF ist es jetzt nicht mehr notwendig, das „Online Display“ explizit durch Setzen der Umgebungsvariablen `MFWTERM` auf den (kleingeschriebenen) Wert `"amiterm"` zu aktivieren. Vorteilhafterweise sollte `MFWTERM` gar nicht gesetzt sein. Allerdings kann durch Angabe eines abweichenden Wertes die Bildschirmanzeige auch in den Modi `proof` und `smoke` unterdrückt werden.

4.3 Verwendung der Umgebungsvariablen

METAFONT kennt und verwendet zahlreiche Umgebungsvariablen bei der Suche nach den unterschiedlichen Eingabedateien, für den Aufruf des Systemeditors als Antwort auf das ‚`E`‘-Kommando im Fehlerfall und zur Einstellung der Bildschirmanzeige. IniMF und VirMF verwenden einen Algorithmus zur rekursiven Suche nach `base`-Kommandodateien, der Konfigurationsdatei `mfmemory.config`, den Zeichensatzbeschreibungsprogrammen und der Textdatei `mf.pool`. Auch die zugehörigen Hilfsprogramme kennen diese und weitere Umgebungsvariablen und verwenden sie entsprechend. In der alten Version kam es an einigen Stellen zu unerwünschten Überschneidungen in der Verwendung von Umgebungsvariablen, die mit dieser neuen Version beseitigt sind.

Sie können die Umgebungsvariablen wie üblich entweder mit dem Systemaufruf

```
SetEnv <Variable> <Ersetzung>
```

setzen oder verändern, um die bei Ihnen gültige Konfiguration zu berücksichtigen, oder Sie können die internen Standardwerte beibehalten. Hier ist eine Liste der Suchpfad-

Umgebungen, ihrer Vorgabewerte und dessen, was METAFONT in den jeweiligen Verzeichnissen zu finden hofft.

Variable	Vorgabewert	Dateien
MFBASES	"MF:bases"	*.base Kommandodateien
MFINPUTS	"MF:inputs/*,"	
	"MF:mfinputs/*,"	
	"MF:inputs/amsfonts/sources/*,"	
	"MF:mfinputs/amsfonts/sources/*,"	
	"MF:inputs/old-german/*,"	
	"MF:mfinputs/old-german/*"	*.mf Programmdateien
MFPOOL	"MF:,MF:pool"	mf.pool Textverzeichnis
MFCONFIG	"MF:config"	mfmemory.config

Zusätzlich dazu wurden auch für die Hilfsprogramme einige Umgebungsvariable eingerichtet, die insbesondere die Unterscheidung zwischen den verschiedenen Arten von Zeichensatzkodierungen ermöglichen.

Variable	Vorgabewert	Dateien
GFFONTS	"TEX:gf/*,"	
	"TEX:fonts/gf/*,"	
	"TEX:texfonts/gf/*"	*.*gf Dateien
PKFONTS	"TEX:pk/*,"	
	"TEX:fonts/pk/*,"	
	"TEX:texfonts/pk/*,"	
	"TEX:pk/amiga/*,"	
	"TEX:fonts/pk/amiga/*,"	
	"TEX:texfonts/pk/amiga/*,"	
	"TEX:pk/deskjet/*,"	
	"TEX:fonts/pk/deskjet/*,"	
	"TEX:texfonts/pk/deskjet/*,"	
	"TEX:pk/cx/*,"	
	"TEX:fonts/pk/cx/*,"	
	"TEX:textfonts/pk/cx/*,"	
	"TEX:pk/ljfour/*,"	
	"TEX:fonts/pk/ljfour/*,"	
	"TEX:texfonts/pk/ljfour/*"	*.*pk Dateien
VFFONTS	"TEX:vf,"	
	"TEX:fonts/vf,"	
	"TEX:texfonts/vf"	*.vf Dateien
PLFONTS	"TEX:pl,"	
	"TEX:fonts/pl,"	
	"TEX:texfonts/pl"	*.[v]pl Dateien

Alle Programme in der neuen Version suchen ohne Berücksichtigung dieser Pfade immer zuerst im aktuellen Verzeichnis nach den jeweiligen Dateien. Dadurch konnte

insbesondere das Problem mit der Notation für das „aktuelle“ Verzeichnis beseitigt werden. Zwar ist weiterhin die leere Zeichenkette und zusätzlich der Punkt `"."` erlaubt, jedoch bringt dies keinerlei Gewinn; eine Datei wird stets von der ersten Stelle ihres Auftretens im Gesamtsuchpfad aus benutzt.

Die umständliche Angabe von mehrfach geschachtelten Unterverzeichnissen ist nicht mehr notwendig. An jeden beliebigen Eintrag in einer Umgebungsvariablen kann jetzt ein Stern `*` angehängt werden, wodurch sowohl in diesem Pfad als auch in jedem direkten Unterpfad nach der erforderlichen Datei gesucht wird (siehe den obigen Eintrag für `MFINPUTS`). Beliebig tiefe Suche im Verzeichnisbaum ist durch Anhängen eines Doppelsterns `**` möglich, kann aber bei stark verzweigten Verzeichnissen beim Aufruf der Programme etwas Zeit kosten.

Eine weitere positive Eigenschaft der neuen Pfadsuchroutinen ist, daß Sie die obigen Vorgabewerte nicht in die Umgebungsvariablen aufnehmen müssen. Wenn Ihre Konfiguration die Verwendung anderer oder zusätzlicher Pfade erfordert, genügt die Angabe der fehlenden Einträge in die entsprechende Umgebungsvariable. Sämtliche internen Vorgaben werden dann einfach daran angehängt, also im Suchprozeß zum Schluß auf jeden Fall berücksichtigt.

An dieser Stelle möchte ich mich bei Giuseppe Ghibò bedanken, der mir freundlicherweise den Quelltext seines `EVPATHS`-Paketes zur Verfügung gestellt hat. Ohne diese konstruktive Mitwirkung würde es die elegante Implementierung der rekursiven Pfadsuche für METAFONT nicht geben. Insgesamt verdankt die neue Version ihre Existenz und den erweiterten Umfang dem hartnäckigen Fordern aus Italien.

Tritt bei der Verarbeitung ein Fehler auf oder unterbrechen Sie METAFONT bei seiner Arbeit, so meldet sich METAFONT mit dem Fragezeichen als Eingabemarke. Geben Sie in solch einem Fall ‚`e`‘ (oder ‚`E`‘) ein, so wird dadurch Ihr Texteditor aufgerufen. Welcher das in Ihrem Fall ist, wird durch die Umgebungsvariable `MFEDIT` gesteuert, die ein Aufrufmuster enthält, um Ihren bevorzugten Systemeditor in der gerade bearbeiteten Datei an der fehlerhaften Stelle zu positionieren. Dies ist nicht notwendigerweise die Datei, mit der Sie METAFONT aufgerufen haben. Geben Sie als Wert von `MFEDIT` einen beliebigen Programmaufruf oder ein beliebiges Skript an, der/das Ihren Editor in der Datei ‚`%s`‘ und der Zeile ‚`%d`‘ positioniert. Die Standardeinstellung lautet folgendermaßen und startet den mit dem Betriebssystem gelieferten Editor `MEmacs`:

```
EDITOR "MEmacs goto %d %s"
```

Im `rexx`-Verzeichnis finden Sie einige ARexx-Skripten zur Verwendung des Cygnus Ed Professional Editors. Wenn Sie diese verwenden wollen, müssen Sie einfach der Umgebungsvariablen `MFEDIT` die Zeichenkette

```
"rx MF:rexx/MFEdit.rexx %s %d"
```

zuweisen. Das zweite Skript `NameStruc` müssen Sie in das `REXX:`-Verzeichnis kopieren, da es direkt aus `MFEdit.rexx` aufgerufen wird.

Wie schon in der vorangegangenen Version, können Generierungsläufe für die virtuellen Gerätetreiber `proof` und `smoke` am Bildschirm mitverfolgt werden, so wie es im ‚METAFONTbook‘ von Don Knuth [4] beschrieben ist. Da dies aber einige erweiterte Bibliotheksroutinen verwendet, die in älteren Versionen des Betriebssystems nicht

vorhanden sind, funktioniert dies nur ab AmigaDOS 2.0 und höher. Die explizite Aktivierung durch Setzen der Umgebungsvariablen `MFWTERM` auf den Wert ‚`amiterm`' ist nicht mehr notwendig, da dies zu Mißverständnissen führte.

5 Verwendung der Programme

5.1 Erzeugung neuer base-Dateien mit IniMF

Obwohl Sie eine betriebsfertige `plain.base`-Datei in diesem Paket finden, kann es sein, daß Sie zusätzliche `base`-Dateien erzeugen wollen, die andere METAFONT-Kommandos als die Standardeinstellung aus dem ‚METAFONTbook' [4] enthalten. Zu diesem Zweck gibt es das Programm IniMF, das in der Lage ist, solche neuen binären `base`-Dateien zu erzeugen, die dann sehr schnell nachgeladen werden können. Wenn Sie IniMF vom CLI (command line input, die Shell) ohne Argumente aufrufen, meldet es sich mit zwei Sternchen, die anzeigen, daß „METAFONT begierig darauf ist, Großes zu leisten" und Sie „bitte einen Dateinamen eingeben" sollen.

```
> IniMF
This is METAFONT, C Version 2.71 (INIMF)
**
```

Wenn Sie dem hier gegebenen Vorschlag folgen, ist es ganz einfach, den nächsten Schritt zu tun. Beantworten Sie die METAFONT-Eingabeaufforderung mit dem Namen eines „Initialisierungsprogrammes" wie zum Beispiel des mitgelieferten `plain.inimf` oder `cm.inimf` und der Rest wird völlig automatisch ausgeführt, so daß schnell eine neue `base`-Datei `plain.base` oder `cm.base` entsteht. Oder Sie geben eine Folge von `input`-Kommandos ein, um mehrere Dateien in die `base` einzubinden. Das letzte Kommando sollte `dump` lauten; dies weist IniMF an, die verdichtete Kommandodatei zu erzeugen und seinen Bearbeitungslauf zu beenden. Alternativ können Sie auch folgenden Einzeiler eingeben

```
> IniMF plain.inimf
```

was zum gleichen Ergebnis wie oben führt.

Wenn Sie, wie hier vorgeschlagen, `plain.inimf` oder `cm.inimf` verwenden, wird die Standardverteilung `modes.mf` (aktuelle Version 2.1) von „Karl Berry and the METAFONT community" in der `base`-Datei eingebaut. Diese Datei enthält Beschreibungen für eine Vielzahl von Ausgabegeräten, wie verschiedenen Matrix-, Laser- oder Tintenstrahldrucker, Fotosatzmaschinen und Bildschirmanzeigern. Es gibt einen kleinen Nachteil, wenn `modes.mf` komplett eingebunden wird, denn dadurch wird die `base`-Datei erheblich größer. Es steht Ihnen aber frei, nur diejenigen Modi zu kopieren und in einer Datei `local.mf` abzulegen, die Sie für Ihre eigene Konfiguration benötigen. Dann sollten Sie in `plain.inimf` die Zeile „`input modes`" durch „`input local`" ersetzen und wie bereits beschrieben mit Hilfe von IniMF die `base`-Datei neu erzeugen.

PasTEXniker, die das CallMF-Paket verwenden wollen, können an dieser Stelle einige Probleme bekommen, denn Georg Heßmann hat sich eigene Modusbezeichnungen

ausgedacht, die *nicht* mit denen aus `modes.mf` zusammenpassen. Diese Änderungen sind in der Datei `pastex-modes.mf` im Verzeichnis `MF:rexx/callmf/inputs` gesammelt. Sie können diese Modi einsetzen, müssen dazu aber die Einträge in `modes` anpassen und die `base`-Datei neu erzeugen, um sie zu verwenden.

5.2 Erzeugung neuer Zeichensatzgrößen mit VirMF

Die Produktionsversion von METAFONT heißt ‚VirMF' und unterscheidet sich in einigen Punkten von IniMF. Das Programm ist dazu gedacht, in hoher Geschwindigkeit Zeichensätze zu erzeugen und kann deshalb keine `base`-Dateien schreiben, diese aber zu Beginn eines Laufes schnell einlesen. Genau wie IniMF können Sie VirMF auf zweierlei Weise aufrufen. Geben Sie lediglich seinen Namen ein (und gesetzt, das Betriebssystem weiß, wo es das Programm zu suchen hat), so meldet sich METAFONT für einen interaktiven Dialog.

```
> VirMF
This is METAFONT, C Version 2.71
**
```

Sie können diese Aufforderung mit jedem gültigen Kommando beantworten und dabei die ‚plain' `base` voraussetzen, wie sie im ‚METAFONTbook' [4] beschrieben wird.

Als leichten Einstieg versuchen Sie doch einfach folgendes. (Zitiert von Seite 31 des ‚METAFONTbook' [4]). Geben Sie ‚`\relax`' ein—also Rückstrich, `r`, `e`, `l`, `a`, `x`—und drücken Sie ⟨return⟩ (oder was auch immer auf Ihrer Tastatur für „Zeilenende" steht). METAFONT ist ganz begierig darauf loszulegen, bereit einen großen Zeichensatz zu erzeugen; Sie aber sagen ihm, es erst einmal ruhig anzugehen, denn dies wird ein ganz vergnüglicher Spaziergang. Der Rückstrich bedeutet, daß METAFONT nicht eine Datei einlesen soll, sondern direkt Anweisungen von der Tastatur entgegen zu nehmen; der Befehl ‚`relax`' bedeutet „tue nichts".

Die Maschine wird jetzt mit einem einzelnen Sternchen antworten: ‚`*`'. Dies bedeutet, daß sie jetzt bereit ist, Befehle (und nicht den Namen einer Datei) entgegen zu nehmen. Nur so zum Spaß geben Sie jetzt folgendes ein:

```
drawdot (35,70); showit;
```

und ⟨return⟩—nicht vergessen, die Strichpunkte mit dem anderen Zeug einzugeben. Ein mehr oder weniger runder Klecks sollte jetzt auf Ihrem Bildschirm erscheinen! Und Sie sollten ein weiteres Sternchen als Eingabeaufforderung sehen. Tippen Sie jetzt

```
drawdot (65,70); showit;
```

und ⟨return⟩, um einen zweiten Klecks zu erhalten. (Im folgenden werden wir die Notwendigkeit, nach jeder Eingabezeile ⟨return⟩ zu drücken, nicht mehr erwähnen.) Zum Abschluß tippen Sie

```
draw (20,40)..(50,25)..(80,40); showit;
shipit; end.
```

Dies malt eine Kurve durch drei gegebene Punkte, zeigt das Gesamtergebnis an, schreibt es gleichzeitig in eine Ausgabedatei und beendet das Programm. METAFONT sollte noch mit ‚[0]‘ antworten, was soviel bedeutet, daß es einen Buchstaben mit der Nummer Null in dem gerade verarbeiteten „Zeichensatz“ geschrieben hat; und es sollte Ihnen mitteilen, daß es eine Ausgabedatei mit dem Namen `mfput.2602gf` erzeugt hat.

Alternativ können Sie auch ein oder mehrere Argumente beim Aufruf von VirMF angeben. Das erste Argument kann der Name einer abweichenden `base`-Datei sein, was durch ein vorangestelltes `&`-Zeichen (wie bei `&plain`) festgelegt wird, und es dürfen mehrere Kommandos in Form einer Zeichenkette folgen, die nach dem Einlesen der `base`-Datei ausgeführt werden. Zum Beispiel geben Sie folgendes ein, wenn Sie einen Zeichensatz in einer bestimmten Auflösung erzeugen wollen:

```
> VirMF &plain "\mode:=amiga; mag:=magstep2; input cmr10"
```

Sie müssen die Anführungszeichen in dieser Form eingeben, denn die Strichpunkte haben eine besondere Bedeutung für das Betriebssystem des Amiga. Dieses Kommando wird den normalen Textzeichensatz `cmr10` passend für den Bildschirmanzeiger der Auflösung 100 dpi in zweifacher Vergrößerung erzeugen, so daß der Zeichensatz die Bezeichnung `cmr10.144gf` erhalten wird und METAFONT folgenden Text auf Ihren Bildschirm (und in die Logdatei) schreiben wird.

```
This is METAFONT, C Version 2.71
(cmr10.mf (cmbase.mf) (roman.mf (romanu.mf [65] [66] [67] [68]
[69] [70] [71] [72] [73] [74] [75][76] [77] [78] [79] [80] [81]
[82] [83] [84] [85] [86] [87] [88] [89] [90]) (romanl.mf [97]
[98] [99] [100] [101] [102] [103] [104] [105] [106] [107] [108]
[109] [110] [111] [112] [113] [114] [115] [116] [117] [118]
[119] [120] [121] [122]) (greeku.mf [0] [1] [2] [3] [4] [5] [6]
[7] [8] [9] [10]) (romand.mf [48] [49] [50] [51] [52] [53] [54]
[55] [56] [57]) (romanp.mf [36] [38] [63] [62]) (romspl.mf [16]
[17] [25] [26] [27] [28]) (romspu.mf [29] [30] [31]) (punct.mf
[33] [60] [35] [37] [39] [40] [41] [42] [43] [44] [46] [47] [58]
[59] [61] [64] [91] [93] [96])  (accent.mf [18] [19] [20] [21]
[22] [23] [24] [32] [94] [95] [125] [126] [127]) (romlig.mf [11]
[12] [13] [14] [15]) (comlig.mf [34] [45] [92] [123] [124]) ) )
Font metrics written on cmr10.tfm.
Output written on cmr10.144gf (128 characters, 7276 bytes).
Transcript written on cmr10.log.
```

Wenn Sie sich schon etwas mit der METAFONT-Sprache auskennen, wissen Sie sicher, was an dieser Stelle noch möglich ist; falls nicht, rate ich Ihnen, sich das ‚METAFONTbook‘ [4] oder den zweiten Band von Kopka [10] zu kaufen.

5.3 Umwandlung zwischen den gf- und pk-Formaten

METAFONT (und TEX) ist ein höchst portables Programmiersystem, das es für eine Vielzahl von Rechnern gibt, angefangen bei kleinen Systemen wie dem Amiga bis

hin zu echten „Zahlenfressern“. Auch seine binäre Ausgabe ist unabhängig von einer speziellen technischen Ausstattung. Wie in dem Beispiellauf aus Abschnitt 5.2 gezeigt, erzeugt METAFONT normalerweise drei verschiedene Ausgabedateien.

Schriftsatzsysteme wie TEX kümmern sich nicht darum, wo am Ende die Tinte hinkommt. Sie wollen lediglich Informationen darüber, wie groß die einzelnen Buchstaben sind und wo in einem bestimmten Zeichensatz sie sich befinden. Diese Informationen werden in eine ‚tfm‘-Datei geschrieben (im Beispiel war das `cmr10.tfm`) und enthalten die „(TEX) font metric“ Information.

Druckertreiber und Bildschirmanzeiger, die „device independent files“ verarbeiten, wie sie von TEX oder anderen Programmen erzeugt werden, brauchen dagegen Informationen über das tatsächliche Aussehen der verwendeten Zeichen. METAFONT schreibt diese Pixelinformation für jeden Buchstaben eines Zeichensatzes im sogenannten „generic font“ Format, das heißt es schreibt eine ‚gf‘-Datei (`cmr10.144gf` in obigem Beispiel). Die Dateiendung bezeichnet dabei die spezifische Auflösung des erzeugten Zeichensatzes in „Punkten pro Zoll“, so daß Sie zwischen Zeichensätzen für den Bildschirmanzeiger und den Druckertreiber unterscheiden können. Das „generic font“ Format ist an dieser Stelle sehr gut geeignet und einige Gerätetreiber können es sogar direkt verarbeiten.

In den meisten Fällen aber wird Ihr TEX-System mit etwas ganz anderem arbeiten, der sogenannten ‚pk‘-Darstellung der Pixelinformation. ‚pk‘-Dateien enthalten die gleiche Information wie die zugehörigen ‚gf‘-Dateien, nur ist diese stark komprimiert, so daß ‚pk‘-Dateien viel kleiner sind. Die Umwandlung zwischen diesen beiden Formaten der Pixelinformation geschieht mit Hilfe der beiden Programme GFtoPK und PKtoGF. Diese werden folgendermaßen aufgerufen

```
> GFtoPK [-v] gffile [pkfile]
> PKtoGF [-v] pkfile [gffile]
```

Die ‚-v‘-Option sagt den Programmen, etwas „wortreicher“ zu sein, ansonsten würden Sie überhaupt keine Ausgabe am Bildschirm sehen. Das erste Argument bezeichnet die Eingabedatei (dabei ist die Endung wichtig) im jeweiligen Ausgangsformat. Das optionale zweite Argument bezeichnet die Ausgabedatei im jeweils anderen Format. Geben Sie dieses zweite Argument nicht an, so wird die Endung der Eingabedatei auf standardisierte Weise in die Endung der Ausgabedatei umgewandelt. Der Aufruf

```
> GFtoPK cmr10.144gf
```

wird `cmr10.144gf` nach `cmr10.144pk` umwandeln. Sie können für das zweite Argument auch einen vollständigen Pfadnamen voranstellen, so daß dann die Ausgabedatei gleich in das richtige Verzeichnis geschrieben wird.

5.4 Probedrucke der Zeichensätze mit GFtoDVI

METAFONT kennt zwei spezielle Modi genannt `proof` und `smoke`, die zwei virtuellen Gerätetreibern entsprechen mit einer Auflösung von 2602 dpi. Das Programm GFtoDVI kann formatierte Ausdrucke dieser Zeichensätze erzeugen. Geben Sie einfach ein

```
> GFtoDVI
```

und GFtoDVI wird Sie nach einem `gf`-Dateinamen fragen. Sie können einige interne Einstellungen ändern, indem Sie einen Schrägstrich an den Dateinamen anhängen. Dann wird Sie GFtoDVI zur Ersetzung spezieller Zeichensätze auffordern, meistens zur Einstellung eines geräteabhängigen ‚gray' oder ‚black' Zeichensatzes. (Siehe hierzu das ‚METAFONTbook' [4].) Wie die meisten anderen Programme, kann auch GFtoDVI Aufrufargumente direkt aus der Kommandozeile übernehmen, so daß Sie auch einfach

```
> GFtoDVI cmr10.2602gf/
```

eingeben können und anschließend weitere Eingaben angefordert werden.

5.5 Ausdruck von Zeichensatzprogrammen mit MFT

Eine andere Besonderheit dieser Implementierung ist das Programm MFT, das gewöhnliche METAFONT Programmquellen in TEX Eingabedateien umwandelt, um sie anschließend formatiert ausdrucken zu können. Das einzige Aufrufargument ist der Name einer ‚`mf`'-Datei, die Teile eines METAFONT-Programmes enthält. Sie können die Arbeitsweise von MFT dadurch beeinflussen, daß Sie ein anderes Formates als `plain.mft` angeben. Es macht sicher Spaß, mit anderen Einstellungen herumzuspielen.

```
> MFT file[.mf] [-cs] [change[.ch]] [style[.mft]]
```

5.6 Analyse von gf- und pk-Dateien

Mehr zu Testzwecken gedacht sind die folgenden beiden Programme. Sie liefern genaue Informationen über den Inhalt der Pixeldateien im ‚generischen' und im ‚komprimierten' Format in Textform.

```
> GFtype [-m] [-i] gffile
> PKtype pkfile
```

Die beiden Optionen für GFtype bedeuten „Mnemonische Ausgabe" und „pIxel Ausgabe". Ihre Angabe wird weitere Informationen zur Textausgabe hinzufügen, besonders die Darstellung der einzelnen Buchstaben in der Form von ASCII ‚Bildern' für die eingehende Kontrolle dessen, was METAFONT produziert.

5.7 Virtuelle Zeichensätze

Neu in die Verteilung aufgenommen habe ich Konvertierungsprogramme zur Erzeugung von „virtuellen Zeichensätzen". Nicht zuletzt dank `DVIPS` von Tomas Rokicki und geeigneten Ausgabetreibern für PostScript kann die Beschäftigung mit dieser Form der Zeichenmanipulation interessant sein.

```
> VFtoVP vfmfile tfmfile vplfile
> VPtoVF vplfile vfmfile tfmfile
```

Zu beachten ist hier lediglich, das „VF“ eigentlich nur für „Grand Wizards“ gedacht sind. Hier sollte man schon sehr genau wissen, was man vorhat. (Ich weiß das leider noch nicht, so daß ich hier nicht mehr dazu sagen will.)

6 Empfohlene Lektüre

Hier ist eine kurze Liste mit Büchern zu METAFONT und seinen Kollegen, die eingehende Informationen über die Programmierung mit dieser Sprache enthalten. Außerdem finden Sie darin viel Interessantes über die Geschichte von TEX und METAFONT seit 1977, Schriftsatz mit Computern im allgemeinen sowie über das literarische Programmieren, das bei der Entwicklung von „Computers & Typesetting“ verwendet wurde.

Literaturverzeichnis

[1] Georg Heßmann u. a., ARexx Unterstützung für METAFONT, Dokumentation zur Version 1.0, Beschreibung des ‚CallMF‘-Paketes von Georg Heßmann, Martin Bokämper, Jörg Höhle und Ulrich Wisser zur Verwendung mit PasTEX.

[2] Donald E. Knuth, Computers & Typesetting A: The TEXbook, Addison-Wesley Publishing Company, 19. Auflage, Juni 1990. *Das* Referenzhandbuch für alle Benutzer des TEX-Systems. Es beschreibt die TEX-Sprache auf vielschichtige Weise.

[3] Donald E. Knuth, Computers & Typesetting B: TEX: The Program, Addison-Wesley Publishing Company, 4. Auflage, Februar 1991. Dieser 600 Seiten starke Schmöker enthält den ausführlich dokumentierten Quelltext für das TEX-Programm, geschrieben in `WEB`.

[4] Donald E. Knuth, Computers & Typesetting C: The METAFONTbook, Addison-Wesley Publishing Company, 4. Auflage, Juni 1990. *Das* Referenzhandbuch für alle Benutzer des METAFONT-Systems. Es beschreibt die METAFONT-Sprache auf vielschichtige Weise.

[5] Donald E. Knuth, Computers & Typesetting D: METAFONT: The Program, Addison-Wesley Publishing Company, 4. Auflage, Februar 1992. Dieser 566 Seiten starke Schmöker enthält den ausführlich dokumentierten Quelltext für das METAFONT-Programm, geschrieben in `WEB`.

[6] Donald E. Knuth, Computers & Typesetting E: Computer Modern Typefaces, Addison-Wesley Publishing Company, 1. Auflage, Mai 1986. Dieses Buch enthält Beispiele für alle Buchstaben und Zeichen der Computer Modern Zeichensatzfamilie, die das Erscheinungsbild aller TEXnischen Schriftstücke prägt.

[7] Donald E. Knuth, Virtual Fonts: More Fun for Grand Wizards. *TUGboat* **11**, 1990, Heft 1, Seite 13–23.

[8] Donald E. Knuth, Literate Programming, Center for the Study of Language and Information, 1. Auflage, 1992. Dies ist eine Zusammenstellung von Knuthschen Texten im Zusamenhang mit dem „literarischen Programmieren“, wie es in höchster Vollendung bei den TEX- und METAFONT-Programmen angewandt wurde. Außerdem enthält es eine Liste aller Fehler von TEX 82 (bis einschließlich den 20. September 1991).

[9] Helmut Kopka, LATEX–Eine Einführung, Addison-Wesley Publishing Company, 4. Auflage, 1992. Dies ist der erste von zwei Bänden für deutschsprachige LATEX-Benutzer. Offenbar ist es beliebter als das Referenzhandbuch von Leslie Lamport. 1994 ist eine völlig überarbeitete Ausgabe der Bücher von Helmut Kopka in einer dreibändigen Reihe erschienen.

[10] Helmut Kopka, LATEX–Erweiterungsmöglichkeiten mit einer Einführung in METAFONT, Addison-Wesley Publishing Company, 3. Auflage, 1992. Band 2 mit einer Einführung in die Programmierung mit METAFONT und das literarische Programmieren mit `WEB` und `CWEB`.

7 Geschichte

Diese Amiga-Umsetzung von METAFONT 2.71 basiert zum großen Teil auf dem kompletten TEX- und METAFONT-System von Edmund Mergl, wie es auf den Fish-Disketten 611–616 veröffentlicht wurde. Ausgehend von METAFONT 2.7 aus dieser Implementierung habe ich viele Änderungen und Erweiterungen vorgenommen. Der größte Unterschied zwischen der Version von Edmund Mergl und meiner eigenen ist, daß nun METAFONT für sich allein stehend konzipiert ist, mit völlig selbständigem Quelltext und Hilfsmitteln. (Siehe Abschnitt 8.)

Von Beginn an habe ich auf dem Amiga mit dem SAS/C-Entwicklungssystem in den Versionen 6.3 und 6.51 gearbeitet. Dieser großartige Compilers ist völlig kompatibel mit dem ANSI-Standard und hat eine ganze Reihe von Auslassungen bezüglich Stil und Sprache bei dem Versuch festgestellt, den Quelltext neu zu übersetzen. Eine große Aufräumaktion für die Zusatzquellen in C, insbesondere die Einführung der vollständigen Prototypen und die Korrektur einiger fehlender Casts, war deshalb notwendig.

Sobald die selbsterzeugte Version 2.7 sauber lief, wurde sie mit der bekannten Amiga-Umsetzung von Stefan Becker von Fish-Diskette 486 verglichen und erwies sich als derart langsamer, daß Verbesserungen notwendig waren, um mit diesem System zu wetteifern. Der kritische Kern der „inneren Schleife“ von METAFONT wurde in Assembler für den Motorola 68020 umgeschrieben, wofür ich an dieser Stelle meinem Bruder danken möchte. Dies ergab eine Beschleunigung des Laufzeitverhaltens um etwa zwanzig bis dreißig Prozent verglichen mit der nichtoptimierten Version und um etwa fünf bis acht Prozent gegenüber der optimierten Version von Stefan Becker.

Der glänzend bestandene TRAP-Test bewies die Korrektheit der gemachten Änderungen dieser Implementierung, wobei die Testroutine von Don Knuth tatsächlich den letzten bösen Fehler meiner eigenen Umsetzung fand (der ausgerechnet etwas mit der Variablen ‚`be_careful`‘ zu tun hatte). Jedoch zeigte der Augenschein, daß sowohl mein

eigenes METAFONT als auch das von Stefan Becker Probleme mit der Bildschirmanzeige hatten. Weder die Implementierung von Stefan Becker noch die von Edmund Mergl arbeiteten so, wie im ‚METAFONTbook' [4] beschrieben. Besonders die Testroutine `6test.mf` für die Computer Modern Zeichensätze machte dies deutlich. (Ein Beispiel für diesen Test findet man auf Seite 192 von [4].) Mit einer geänderten Version von `amiga.c`, die gleichzeitig einige fortgeschrittene Funktionen für die Ausgabe verwendet und deshalb auch die Version 2 des Amiga Betriebssystems benötigt, lief es besser.

Unter Anwendung einiger Ideen von David Crooke aus seiner Amiga-Umsetzung von TEX 3.141, baute ich eine geänderte Pfadsuche sowie die Möglichkeit der Speicherkonfiguration in METAFONT ein. (Siehe Abschnitt 4.2 zu Einzelheiten darüber.)

Im September 1993 erfolgte dann der erste Schritt in Richtung Version 2.71 mit den Fehlerbeschreibungen des Addison-Wesley-Verlages. Alle Änderungen, die die neue Version betrafen, wurden in die Change-Datei eingebaut.

Schließlich erhielt ich im Oktober 1993 die offizielle Version 2.71[4], die das alte System vollständig ersetzt. Sie enthält ein neues `plain.mf` `base`-Programm und einen völlig neuen TRAP-Test. Die letzte wichtige Änderung dieser Implementation war die Bereitstellung einer wirklich GROSSEN Version von METAFONT (und von TEX 3.1415), indem `max_quarterword` auf 32.767 (32 Kilo Viertelworte) und `max_halfword` auf 1.073.741.823 (1 Giga Halbworte) erweitert wurde, so daß nun wirklich niemand mehr an die maximal mögliche Einstellung stoßen wird (Sie bräuchten schon mehr als 8 Giga Byte freien Speicher!). Dabei scheint jeder zu glauben, daß ‚262.142' als maximaler Wert von `memmax` festgelegt sei, nur weil es so in [3] und [5] angegeben ist. Dieser Wert gilt jedoch nur für 36 Bit Worte. Die vorliegende C-Implementation arbeitet aber auf der Basis von „64 Bit Worten", das heißt ein Halbwort ist als **long int** definiert, so daß die viel größeren Werte verwendet werden können.

7.1 Erweiterungen

Die wesentlichste Neuerung an diesem METAFONT-Paket betrifft die Pfadsuche. Dank der großzügigen Unterstützung durch Giuseppe Ghibò konnte ich auf äußerst einfache und elegante Weise die rekursive Suche nach Dateien in Unterverzeichnissen installieren, ein Vorhaben, das bisher am Umfang der neuen UNIX-Verteilung scheiterte. Um Mißverständnisse zu vermeiden, wurde das „aktuelle" Verzeichnis aus der Rekursion entfernt. Dateien im aktuellen Verzeichnis werden grundsätzlich zuerst verwendet. Wie üblich werden die einzelnen Pfade in den Umgebungsvariablen durch Kommata getrennt. Durch Anhängen eines Sterns ‚`*`' an einen solchen Pfad wird dieser zusammen mit allen direkten Unterverzeichnissen durchsucht. Doppelte Sterne ‚`**`' bewirken die beliebig tiefe Suche im Verzeichnisbaum, was allerdings etwas Zeit beim Start der Programme beanspruchen kann. Für die verschiedenen Dateiarten sind interne Vorgabewerte definiert, die METAFONT und seine Hilfsprogramme automatisch erkennen, ohne daß sie in den externen Umgebungsvariablen wiederholt werden müßten.

Der leicht zu handhabende Quelltext ermöglichte gleichzeitig die deutliche Vereinfachung der kompletten Entwicklungskonfiguration für TEX, METAFONT und *alle*

[4] vom Stuttgarter ftp-Server `ftp.uni-stuttgart.de (129.69.8.13)`. Dieser wurde mittlerweile durch den DANTE-Server `ftp.dante.de (129.206.100.192)` ersetzt.

zugehörigen Hilfsprogramme. Insbesondere ist jetzt auch die Erstellung und Verwendung von „virtuellen“ Zeichensätzen möglich, also die Verknüpfung von Zeichen aus beliebigen Definitionsdateien unter einem neuen Namen. In diesem Zusammenhang sei jedem ernsthaften TEX-Anwender auf dem Amiga die Verwendung des Programmes `DVIPS` von Tomas Rokicki in der wirklich hervorragenden Umsetzung für den Amiga von Giuseppe Ghibò empfohlen, die in Version 5.58 auch das CallMF-Paket unterstützt.

Neben den vier neuen Umgebungsvariablen zur Suche nach Zeichensatzdateien (siehe Abschnitt 4.3) hat sich auch die Verwendung der Variablen `MFWTERM` entscheidend geändert. Nachdem ich in der ersten Dokumentation einen fehlerhaften Eintrag angegeben hatte, der wahrscheinlich bei vielen Benutzern die Verwendung des „Online Display“ verhinderte, habe ich mich in dieser neuen Version dazu entschieden, auf die externe Aktivierung der Bildschirmanzeige zu verzichten.

7.2 Danksagung

Seit Veröffentlichung der ersten Version meiner METAFONT-Implementierung habe ich mehrere Zuschriften mit Hinweisen erhalten, die zu einer Verbesserung dieser Dokumentation geführt haben. Teilweise wurden die beschriebenen Schwierigkeiten in der Version vom Mai 1994 für die PasTEX-Verteilung auf der „Meeting Pearls“-CD beseitigt, wenn auch im Anschluß daran noch eine Anfrage bezüglich des „Online Display“ erfolgte. Glücklicherweise handelte es sich in keinem Fall um Fehler in den Programmen, sondern lediglich um Ungenauigkeiten und Auslassungen in der Dokumentation. Für die Zuschriften danke ich Stefan Sell, Markus Eiblmeier und Dieter Sach.

Besonderer Dank gilt Georg Heßmann, der meine METAFONT-Implementierung für „PasTEX on CD“ verwendete, und mir einige Erweiterungen abverlangte, die die Qualität der Programme wesentlich steigerte. Insbesondere ist ihm zu verdanken, daß IniMF und VirMF „resident“ sind.

Den größten Einfluß auf dieses erweiterte Programmpaket hatte jedoch Giuseppe Ghibò, der mich dazu drängte, seine hervorragende Umsetzung der rekursiven Pfadsuche zu installieren. Ohne sein Paket `EVPATHS` hätte ich mich nie an diese Aufgabe gemacht. Daneben ermöglichte mir sein einfach zu installierender und zu verwendender Quelltext aber auch den Einbau der zusätzlichen Programme VFtoVP und VPtoVF sowie die Trennung der unterschiedlichen Dateiformate für die Zeichensätze durch zusätzliche Umgebungsvariablen.

8 Quelltext

Bei diesem Paket handelt es sich um den „METAFONT Benutzerbausatz“, der keinerlei Quelltext für die eigentlichen Programme enthält. Auf Anfrage ist beim Implementierer ein „Entwicklerbausatz“ erhältlich. Wenn Sie den haben wollen, brauchen Sie nur einen Rückumschlag mit 10 Deutschen Mark (oder 10 US Dollar) an die Adresse zu schicken, die auf der ersten Seite dieser Dokumentation angegeben ist, und Sie werden die neueste Version mit allen notwendigen Hilfsmitteln für die Neuübersetzung und

alle zukünftigen Fehlerbeseitigungen auf Diskette erhalten. Je nach Interesse kann ich auch Version 3.1415 von TEX zur Verfügung stellen.

Im Zeitalter der elektronischen Medien haben an dieser Stelle all diejenigen deutliche Vorteile, die über einen Zugang zum Internet verfügen. Da ich mir dann den Aufwand für die Diskettenversion und die schriftliche Beantwortung sparen kann, ist der Quelltext kostenlos erhältlich, wenn Sie mir eine elektronische Adresse mitteilen, an die ich den Quelltext per EMail oder besser `ftp` schicken kann. Dies soll auf keinen Fall eine Herabsetzung oder gar Ausnutzung der „normalen“ Benutzer sein, es handelt sich lediglich um eine erhebliche Arbeitserleichterung.